Desarrollo de un sistema software para el cálculo de grados día en una plantación de brócoli con el fin de optimizar el ciclo productivo valiéndose de estaciones meteorológicas inalámbricas

Tapia Vargas, Cristian Xavier y Tamayo Romo, Franklin David

Departamento de Eléctrica y Electrónica

Carrera de Ingeniería en Software

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

M. Sc. Escobar Sánchez, Milton Eduardo

Latacunga, 5 de marzo de 2021

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA DE SOFTWARE

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"Desarrollo de un sistema software para el cálculo de grados día en una plantación de brócoli con el fin de optimizar el ciclo productivo valiéndose de estaciones meteorológicas inalámbricas"** fue realizado por los señores **Tapia Vargas, Cristian Xavier** y **Tamayo Romo, Franklin David** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Latacunga, marzo del 2021

Firmado electrónicamente por:
MILTON EDUARDO
ESCOBAR SANCHEZ

……

Escobar Sánchez, Milton Eduardo Mgs.

C. C.: 1710557545

URKUND

Urkund Analysis Result

Firmado electrónicamente por:
MILTON EDUARDO ESCOBAR SANCHEZ

Analysed Document: TESIS_Final_V1.docx (D97046658)
Submitted: 3/3/2021 6:07:00 AM
Submitted By: meesacobar1@espe.edu.ec
Significance: 1 %

Sources included in the report:

https://bdigital.uexternado.edu.co/bitstream/001/1844/1/ABCBA-spa-2019-Propuesta_metodologica_para_realizar_la_especificacion_de_requerimientos_y_control_de_cambios_de_los_sistemas

Instances where selected sources appear:

1

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA DE SOFTWARE

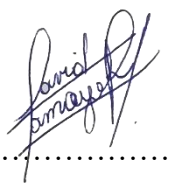
RESPONSABILIDAD DE AUTORÍA

Nosotros, **Tapia Vargas, Cristian Xavier**, con cédula de ciudadanía n° 0503297079 y **Tamayo Romo, Franklin David** con cédula de ciudadanía n° 1804778023, declaramos que el contenido, ideas y criterios del trabajo de titulación: **Desarrollo de un sistema software para el cálculo de grados día en una plantación de brócoli con el fin de optimizar el ciclo productivo valiéndose de estaciones meteorológicas inalámbricas** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Latacunga, marzo del 2021

……

Tapia Vargas, Cristian Xavier
C.C.: 0503297079



……

Tamayo Romo, Franklin David
C.C.: 1804778023

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA DE SOFTWARE

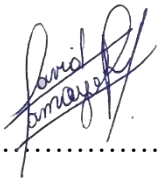
AUTORIZACIÓN

Nosotros, **Tapia Vargas, Cristian Xavier**, con cédula de ciudadanía n° 0503297079 y **Tamayo Romo, Franklin David** con cédula de ciudadanía n° 1804778023, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Desarrollo de un sistema software para el cálculo de grados día en una plantación de brócoli con el fin de optimizar el ciclo productivo valiéndose de estaciones meteorológicas inalámbricas** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Latacunga, marzo del 2021

……

Tapia Vargas, Cristian Xavier
C.C.: 0503297079



……

Tamayo Romo, Franklin David
C.C.: 1804778023

DEDICATORIA

Una vez culminada esta etapa de estudios, quiero dedicar este logro a todas las personas que me han acompañado a lo largo de esta lucha, ya que por ellas no me he rendido y no lo haré jamás.

A mi madre Sra. Luz Angélica Vargas, por todo el apoyo incondicional, amor y lecciones que a lo largo de mi vida me han convertido en el hombre que ahora soy.

A mi padre Ing. Amilcar Tapia, por todos los grandes valores que me ha inculcado, gracias a eso en este momento me he convertido en lo que siempre anhele.

A mi hermano Dr. Wilmer Tapia quien no me dejó vagar en la obscuridad, gracias a su ejemplo y consejos logré cumplir hoy mi meta.

A mi sobrino Mateo, quien con su llegada al mundo inspiró en mí el sentimiento de lucha para lograr todos mis cometidos.

A mis abuelos, mis ángeles de la guarda, que lamentablemente no me acompañaron hasta este punto, pero donde quiera que se encuentren espero llenarlos de orgullo.

A mi primo el Sr. Cristian Jácome, por ser como un hermano para mí e incentivarme en cada momento de mi vida para cumplir el objetivo de llegar a ser un profesional.

Sr. Cristian Tapia

DEDICATORIA

A mi madre Lcda. Anita Romo, que siempre me ha guiado y apoyado en mi vida. Dándome mucho cariño y ofreciéndome lo mejor de ella, por todo el esfuerzo que ha realizado para que pueda alcanzar todas mis metas.

A mi padre Ing. Franklin Tamayo, por enseñarme los valores fundamentales de la vida, dándome un buen ejemplo de cómo ser una persona de bien, apoyándome incondicionalmente para cumplir mis metas y objetivos de vida.

A mis hermanas Lcda. María Isabel Tamayo y Dra. Gabriela Fernanda Tamayo, por todo el cariño, y ejemplo que me dan a través de sus enseñanzas de vida, para poder ser un hombre con propósito y valores de vida.

A mis sobrinitas Isabella, Macarena y Fabiana, por ser una luz en mi camino, transmitiéndome felicidad en su máximo esplendor.

Les dedico a ustedes, mi familia, quienes han estado ahí para mí en todo momento de mi vida y han sabido expresar su amor incondicional.

Sr. David Tamayo

AGRADECIMIENTO

Al finalizar este trabajo quiero agradecer a toda mi familia la cual ha estado presente a lo largo de mi vida, en las buenas y malas situaciones, sabiéndome aconsejar e impulsar para llegar a la consecución de mis sueños.

A la Universidad de las Fuerzas Armadas ESPE, por otorgarme la oportunidad de alcanzar mi meta académica además de inculcar en mi diversos valores y experiencias las cuales son de vital importancia tanto en el desarrollo de mi vida profesional como personal.

A mi director de tesis Mgs. Milton Escobar, quien a ha sido además de un profesor, un amigo. Gracias por inculcar en nosotros el valor del esfuerzo y la dedicación, llevando al límite nuestras capacidades con la finalidad de conocer de lo que estamos hechos.

A la Ing. Ximena López, quien siempre ha estado apoyando nuestro desarrollo tanto académico como profesional y ayudándonos en cualquier imprevisto presentado.

"Un poco más de persistencia, un poco más de esfuerzo, y lo que parecía irremediablemente un fracaso puede convertirse en un éxito glorioso"

Elbert Hubbard

Sr. Cristian Tapia

AGRADECIMIENTO

Un agradecimiento profundo a Dios por darme las oportunidades necesarias para lograr esta meta tan anhelada, permitiendo que todos los caminos me hayan llevado a este momento.

Gracias a mis padres Lcda. Anita Romo e Ing. Franklin Tamayo por siempre confiar en mí y darme el apoyo necesario para llegar a este punto de mi vida, diciéndome siempre lo siguiente: “siempre da lo mejor de ti y supérate lo que más puedas”, no les defraudare nunca amados padres.

A mis hermanas porque siempre han estado conmigo, me han apoyado y comprendido en cada paso que doy, cuidándome desde que era un bebe como su hermanito menor.

A mi director de tesis Mgs. Milton Escobar, quien nos ha enseñado el verdadero valor del esfuerzo llevando al límite la determinación para llegar a un objetivo planteado, gracias por su paciencia y su tiempo dejando sus mejores enseñanzas y los mejores consejos como amigo.

A mi alma mater, que me ha brindado todo el conocimiento que poseo, abriéndome una infinidad de puertas para el mundo real.

“Todos los triunfos nacen cuando nos atrevemos a comenzar”

Eugene Fitch Ware

Sr. David Tamayo

Tabla de contenidos

Carátula.....1

Certificación.....2

Urkund.....3

Responsabilidad de autoría.....4

Autorización.....5

Dedicatoria.....6

Dedicatoria.....7

Agradecimiento 8

Agradecimiento 9

Índice de contenidos.....10

Índice de tablas 14

Índice de figuras.....15

Índice de ecuaciones 18

Resumen.....19

Abstract.....20

Generalidades.....21

Planteamiento y Formulación del Problema. 22

***Preguntas de investigación*.**.....25

Justificación e Importancia. 26

Objetivos 27

Objetivo General.....27

***Objetivos Específicos*.** 27

Hipótesis.....28

Variables de la investigación. 28

Marco Teórico.....29

Fundamentación teórica del brócoli. 30

***El brócoli en Ecuador*.....30**

***Zona de investigación y sus climas relevantes*. 31**

***Importancia del Brócoli.*.....32**

***Tiempo térmico del brócoli*.....33**

***Grados día y su influencia en cultivos.*.....33**

Modelo para el cálculo de grados días.....33

Software.....34

***Características del Software*.....35**

***Tipos de Software*. 35**

Desarrollo del software.....36

***Fases del desarrollo del Sistema de Software*. 37**

***Modelos o ciclos de vida del Software*. 41**

***Calidad del Software*. 45**

Características de la Calidad del Software. 45

Lenguajes de programación.....46

***Tipos de lenguajes para programación*. 46**

***Algoritmos*. 48**

Frameworks. 48

***NodeJS*.....49**

***Angular*. 50**

***Flutter*.....51**

Editores de código fuente. 52

Interfaz de programación de aplicaciones meteorológicas. 53

Arquitectura básica del Software.....55

***Modelos para el diseño de la arquitectura de software.*.....55**

Componentes básicos que conforman una arquitectura **.....57**

Base de datos**.....57**

Características básicas de una base de datos. 57

Tipos de base de datos. 58

Mongo DB **.....61**

Back-end**.....61**

Front-end**. 62**

Comunicación entre componentes**.....62**

Internet de las cosas (IoT). 63

Metodología.....64

Metodología tradicional**. 64**

Metodología Ágil**. 65**

Metodología de Programación Extrema (XP).....65

Metodología SCRUM.....67

Diferencias entre XP y SCRUM**.....74**

Aplicación de SCRUM como metodología en el desarrollo de aplicaciones móviles y web **.....75**

Inteligencia Artificial. 76

Aprendizaje de máquina (Machine Learning). **.....76**

Modelos para el aprendizaje de máquina**. 79**

Modelo Autorregresivo Integrado de Media Móvil (ARIMA) 81

Métricas de Evaluación de algoritmos de regresión 82

Metodología Box-Jenkins para la implementación de modelos predictivos utilizando ARIMA 87

Análisis, diseño y desarrollo del sistema de software para el cálculo de grados días 89

Etapa 1. Levantamiento de los requisitos de software.....89

***Requisitos funcionales*. 92**

***Requisitos no funcionales*. 92**

***Requisitos de interfaz*. 93**

Etapa 2. Diseño de la arquitectura del software. 97

***Vista de escenarios*. 98**

***Vista lógica*. 100**

***Vista física*. 104**

***Vista de desarrollo*. 107**

Etapa 3. Desarrollo del sistema de cálculo de grados días (aplicación móvil y web). 108

***Configuración de entorno de trabajo.* 109**

***Elaboración de las tareas de desarrollo* 113**

***Desarrollo del modelo de predicción.* 126**

Etapa 4. Pruebas y métricas del sistema de cálculo de grados días (aplicación móvil y web). 129

Validación del sistema de software para el cálculo de grados días 139

Validación de grados días y calidad del brócoli 139

Validación comparativa de cultivos.....145

Conclusiones. 147

Recomendaciones.....148

Biliografía.....150

Anexos...…….161

Índice de tablas

Tabla 1 *Problemas, soluciones, enfoques y evidencias identificadas* 24

Tabla 2 *Plantilla Equipo SCRUM* 68

Tabla 3 *Plantilla historia de usuario* 69

Tabla 4 *Plantilla de product backlog* 71

Tabla 5 *Plantilla de sprint backlog* 72

Tabla 6 *Diferencia del uso de los parámetros de gestión utilizados en proyectos XP y SCRUM.* 74

Tabla 7 *Proceso de corrección de los casos de prueba* 132

Tabla 8 *Incidentes presentados en varias tecnologías* 134

Tabla 9 *Comparación del método manual con el sistema de grados días* 141

Tabla 10 *Análisis de muestras* 144

Tabla 11 *Comparativo entre diferentes técnicas de cosecha* 145

Índice de figuras

Figura 1 *Estructura de la investigación realizada* 30

Figura 2 *Fases del proceso de desarrollo del sistema de software* 38

Figura 3 *Modelo en cascada o convencional* 42

Figura 4 *Modelo evolutivo* 43

Figura 5 *Modelo incremental* 44

Figura 6 *Modelo en espiral* 45

Figura 7 *Framework* 49

Figura 8 *Arquitectura NodeJS* 50

Figura 9 *Ciclos de requerimiento y respuesta en Angular* 51

Figura 10 *Arquitectura de Flutter* 52

Figura 11 *Funcionamiento básico de una API* 54

Figura 12 *Relación de tablas* 59

Figura 13 *Base de datos no relacional* 60

Figura 14 *Comunicación entre el front-end y el back-end* 63

Figura 15 *Modelo XP* 66

Figura 16 *Ciclo de vida de SCRUM* 67

Figura 17 *Proceso para elegir un tipo de métricas de evaluación para un modelo de regresión.* 83

Figura 18 *Metodología de Box-Jenkins* 88

Figura 19 *Etapas para la creación del sistema de software propuesto en esta investigación.* 89

Figura 20 *Levantamiento de requisitos* 90

Figura 21 *Especificación de requisitos software* 91

Figura 22. *Requisitos de interfaz* 94

Figura 23 *Maquetación Móvil* 95

Figura 24 *Maquetación Web* 96
Figura 25 *Diseño de la arquitectura software* 97
Figura 26 Casos de uso 100
Figura 27 *Diagrama de clases* 101
Figura 28 *Colecciones de MongoDB* 102
Figura 29 *Diagrama de secuencia* 104
Figura 30. *Vista física* 105
Figura 31 *Diagrama de paquetes* 107
Figura 32 *Desarrollo del sistema de cálculo de grados día* 108
Figura 33 *Configuración de entorno de trabajo* 109
Figura 34 *Conexiones entre componentes* 111
Figura 35 *Elaboración de las tareas de desarrollo* 113
Figura 36 *Desarrollo de las historias de usuario* 116
Figura 37 *Resultado sprint 1 web* 117
Figura 38 *Resultado sprint 1 móvil* 117
Figura 39 *BurnDown chart gestión usuarios* 118
Figura 40 *Resultado sprint 2 web* 119
Figura 41 Resultado sprint 2 móvil 119
Figura 42 *BurnDown chart Gestionar haciendas* 120
Figura 43 *Resultado de sprint 3 web* 121
Figura 44 *Resultado de sprint 3 móvil* 121
Figura 45 *BurnDown chart Gestión hectárea* 122
Figura 46 Resultado sprint 4 web 123
Figura 47 *Resultado sprint 4 móvil* 123
Figura 48 *BurnDown chart gestión cultivos* 124
Figura 49 *Resultado sprint 5 gestión control web* 125

Figura 50 *Resultado sprint 5 gestión control móvil* 125

Figura 51 *Burndown chart gestión control* 126

Figura 52 *Librerías utilizadas en Python* 127

Figura 53 *Implementación de ARIMA* 128

Figura 54 *Datos de temperatura* 129

Figura 55 *Pruebas y métricas efectuadas al sistema de cálculo de grados días* 130

Figura 56 *Modelo propuesto para Pruebas Funcionales de Software.* 131

Figura 57 *Funcionalidades, casos de prueba e incidentes del sistema de cálculo de grados día (aplicación móvil y web)* 134

Figura 58 *Incidentes según su funcionalidad del sistema de cálculo de grados día (aplicación móvil y web)* 135

Figura 59 *Nivel de criticidad de los incidentes del sistema de cálculo de grados día (aplicación móvil y web)* 136

Figura 60 Datos de predicción de prueba vs datos reales.....137

Figura 61 *Diagramas de dispersión y porcentajes de error resultantes de las métricas de evaluación (Error Absoluto Medio y Error Cuadrático Medio)*.....138

Figura 62 *Etapas de validación del sistema de software* 139

Figura 63 *Plantación de brócoli en Lasso* 140

Figura 64 *Días de desarrollo y cosecha de los grupos de brócoli*.....142

Figura 65 *Grados días de los grupos de brócoli* 143

Figura 66 *Comparativa de muestras* 144

Índice de ecuaciones

Ecuación 1 *Modelo de cálculo de grados día.* 34

Ecuación 2 *Fórmula general de ARIMA* 82

Ecuación 3 *Fórmula error cuadrático*.....85

Ecuación 4 Fórmula error cuadrático medio 85

Ecuación 5 *Fórmula error absoluto medio* 86

Ecuación 6 *Fórmula* R^2.....87

Ecuación 7 *Fórmula baseline* 87

RESUMEN

La presente investigación está orientada al desarrollo de un sistema de software (móvil y web) que ayude a mejorar el tiempo de cosecha de las plantaciones de brócoli mediante el cálculo de grados día, enfocándose en apoyar a las personas encargadas del control de calidad del cultivo de esta hortaliza. El desarrollo de este proyecto se conforma de 4 etapas primordiales: La primera etapa consta del establecimiento de los requerimientos de software realizados a partir del análisis de las necesidades manifestadas por los expertos agrónomos. En la segunda etapa se lleva a cabo el diseño de la arquitectura del sistema en base a la metodología 4+1 la cual hace uso de diagramas UML (Lenguaje Unificado de Modelado) para un mejor entendimiento de su estructura. La tercera etapa cosiste en el desarrollo y codificación del proyecto de software según lo establecido por la metodología ágil SCRUM, así como también en la implementación de un modelo de regresión (ARIMA) instaurado según los parámetros de la metodología Box-Jenkins. Finalmente, en la cuarta etapa se ejecutan las pruebas de integración al sistema con el fin de examinar si este cumple con todos los requerimientos planteados en la primera fase, de igual manera se efectúa una valoración de eficacia de la predicción brindada por este software haciendo uso de métricas de evaluación para algoritmos de regresión. Obteniéndose como resultado final el desarrollo de un sistema software móvil y web para el cálculo de grados día, sistematizando el procesamiento de información de los cultivos de brócoli de una manera manual a un procedimiento inteligente. Optimizando el tiempo de desarrollo de brócoli y con esto su ciclo productivo en un 29% reduciéndose sus días de avance de 130 a 93 en promedio. Así como un incremento de calidad en un 9.35% debido a su reducción de grados días de 1025 al 910.

Palabras clave:

- **GRADOS DÍA**
- **MÉTRICAS**
- **ARIMA**
- **BOX-JENKINS**
- **PRUEBAS DE INTEGRACIÓN**

ABSTRACT

This research is oriented to the development of a software system (mobile and web) that helps to improve the harvesting time of broccoli plantations through the calculation of degree days, focusing on supporting the people in charge of quality control of the cultivation of this vegetable. The development of this project consists of 4 main stages: The first stage consists of establishing the software requirements based on the analysis of the needs expressed by the agronomist experts. The second stage involves the design of the system architecture based on the 4+1 methodology, which uses UML (Unified Modeling Language) diagrams for a better understanding of its structure. The third stage consists of the development and coding of the software project according to the agile SCRUM methodology, as well as the implementation of a regression model (ARIMA) established according to the parameters of the Box-Jenkins methodology. Finally, in the fourth stage, the system integration tests are carried out in order to examine whether it meets all the requirements set out in the first phase, and also to evaluate the effectiveness of the prediction provided by this software using evaluation metrics for regression algorithms. The final result is the development of a mobile and web software system for the calculation of degree days, systematizing the processing of broccoli crop information from a manual way to an intelligent procedure. Optimizing the broccoli development time and thus its productive cycle by 29%, reducing its days of advance from 130 to 93 on average. As well as an increase in quality by 9.35% due to its reduction in grade days from 1025 to 910.

Key words:

- **DEGREES DAY**
- **METRIC**
- **ARIMA**
- **BOX-JENKINS**
- **INTEGRATION TESTS**

1. Generalidades

Según (FAO, 2016), en su boletín estadístico periódico revela que, para el año 2050 existirá un incremento de 2570 a 9700 millones de personas que dependerán de la agricultura a nivel mundial. Lo cual obliga a cambiar las directrices a futuro. Buscándose una forma sostenible de satisfacer la demanda de alimentos mediante modificaciones importantes en el sistema agrícola, a fin de garantizar los medios de vida y la seguridad alimentaria (Organización de las Naciones Unidas para Alimentación y Agricultura, 2018).

En Ecuador, país donde una de las principales actividades económicas es la agricultura, en la actualidad el 11% de la tierra tiene cultivos permanentes y el 18% se lo utiliza en pasto. Esta actividad garantiza el derecho que tienen los seres humanos a una alimentación sana; la demanda de productos agrícolas va en aumento y esta a su vez debe responder a las necesidades de la población, por lo que los retos y desafíos que presenta este sector es complejo, ya que su producción debe responder a niveles de calidad (FAO, 2017).

Este estado en el cual la producción del brócoli ha generado un impulso a la economía de la población nacional, dicha actividad ha creado importantes fuentes de trabajo dando un empleo directo a 8053 personas; según el censo Estadístico Agropecuario Nacional del 2015 (Zavala , 2015). Situándose sus empresas en la región sierra, las cuales exportan su producción a otros países. Incrementándose su demanda mundial debido a que existen estudios científicos que demuestran que este vegetal previene y controla el cáncer (ProEcuador, 2016).

Las empresas más representativas en el país que se dedican a la producción de brócoli son las siguientes: 1) Ecofroz - Pro congelados S.A. y 2) Provefrut - Nova Alimentos S.A ubicadas en la provincia de Pichincha y Cotopaxi respectivamente. Además, existen otras compañías dedicadas a la industrialización del brócoli que no han logrado superar las dificultades técnicas que poseen ya sea por falta de capital, infraestructura o tecnología. Sobresaliendo aquellas industrias con procesos sólidamente implementados que son orientados a un mejoramiento de la calidad.

Permitiéndoles competir en el mercado nacional e internacional en la comercialización de brócoli (Guamán, 2017).

La empresa donde se realizará la presente investigación es ECOFROZ, S.A. compañía de tipo agroindustrial; productora de vegetales y frutas congeladas, la misma que se encuentra ubicada en el sector del Valle de Machachi, a 65 km de la ciudad Quito, con un clima medio que oscila entre 10° y 18° centígrados. Actualmente existen cinco líneas de fabricación: Habichuelas, col, mango, romanesco, espinaca y brócoli. Su producción es exportada a los siguientes países: Estados Unidos, Europa y Japón (Zavala , 2015).

La presente investigación está enfocada en dar solución al inconveniente que existe actualmente al momento de calcular de manera manual la fecha de cosecha del cultivo de brócoli. Requiriendo monitorear las temperaturas máximas y mínimas diarias de manera automática, siendo de mucha importancia esta información al momento de suministrar los datos de entrada al algoritmo de predicción que se basa en el método de grados día. Lo cual beneficiara al desarrollo de un cultivo hortícola de mayor calidad y con esto la optimización del ciclo productivo del brócoli (Guamán, 2017).

1.1. Planteamiento y Formulación del Problema.

Uno de los productos más importantes dentro de la exportación hortícola del Ecuador, es el brócoli actualmente existen pérdidas debido a la ineficacia en su cosecha superando el 50% de su producción, razón por la cual la agricultura moderna se ha enfocado en estudios normalizados de nutrientes, fertilización y control de cultivos (Chávez, 2012).

De acuerdo con (Almeida Granja, 2012) el procesamiento de brócoli conlleva el uso de habilidades manuales que deben ser alcanzadas por el personal que realiza su cosecha. A fin de evitar rechazos en el cultivo, facilitando la continuidad y fluidez del proceso de producción del brócoli. Dejando a un lado el tiempo de cosecha óptimo lo cual reduce las propiedades nutricionales, obteniendo como resultado un producto no apto para su exportación.

Una de las causas de la ineficacia en la producción de cultivos de brócoli, es el proceso de cosecha, debido a que el control de la temperatura es realizado de manera

empírica sin considerar detalles externos e internos. Lo cual, incide directamente en el nivel de producción del mismo, en cuanto a calidad y cantidad (Borja, 2015).

Según (Toledo, 2003) otro de los inconvenientes en la producción de cultivos de brócoli es su exposición al sol, lo cual afecta su parte comestible (inflorescencia) debido a que esta tiende a calentarse varios grados, sobrepasando el cálculo óptimo de temperaturas, por esta razón es recomendable cosechar a tiempo el producto y así evitar pérdidas de calidad.

Además, cabe recalcar que los agricultores especializados en la producción de brócoli, calculan el tiempo de cosecha empíricamente, revisando de manera manual todos los parámetros de la planta e incluso desconociendo la relevancia de las temperaturas, siendo inexactos en el tiempo de cosecha (Toledo, 2003).

En la actualidad el sector que se dedica al agro busca mejorar su calidad de producción incluyendo herramientas tecnológicas que ayuden al monitoreo de sus cultivos, automatizando así la toma de decisiones para poder cumplir con estándares internacionales (Schejtman, 1998).

Las empresas dedicadas al cultivo del brócoli, requieren de un producto mejor posicionado mundialmente, lo cual se ve reflejado en las utilidades que obtienen, la apertura de nuevos mercados de exportación y mayor ingreso económico al país (Guamán, 2017).

De acuerdo a lo establecido en los párrafos anteriores, se puede concluir que las plantaciones de brócoli alargan su periodo de cultivo influyendo en su cosecha y su calidad. Debido a los inconvenientes que causa el clima, humedad u horas de luminosidad (Vigliola, 2015). Requiriéndose dar solución a todo esto mediante el desarrollo de un sistema software para el cálculo de grados día, monitoreándose los factores antes mencionados.

De una manera general se describe los diferentes problemas, soluciones, enfoques y evidencias que se desarrollaran en la presente investigación como se detallar en la Tabla 1.

¿De qué manera el desarrollo de un sistema software para el cálculo de grados día en una plantación de brócoli optimizará el ciclo productivo valiéndose de estaciones meteorológicas inalámbricas?

Tabla 1

Problemas, soluciones, enfoques y evidencias identificadas

Problema	Solución	Enfoque	Evidencia
No existe un sistema software para calcular grados días en plantaciones de brócoli	Diseño e implementación de un sistema software para el cálculo de grados día en una plantación de brócoli	Estrategia de procesos	Funcionamiento de un sistema software para el cálculo de grados día en una plantación de brócoli (Carta de validación de expertos agrónomos).
Ciclo productivo del brócoli demasiado largo	Implementación de un algoritmo de predicción de temperaturas usando una API meteorológica a través del uso del en cuenta el internet de las cosas.	Estrategia de competencias	Optimización del ciclo productivo en una plantación de brócoli (Algoritmo de predicción ARIMA – validación a través de la métrica error cuadrado
Control del estado semanal del cultivo	Implementación de formularios semanales que permitan el control del crecimiento del brócoli	Estrategia de seguimiento y control	Reportes del control del cultivo

Nota. En esta tabla se muestra los problemas soluciones y enfoques evidenciados para el desarrollo del proyecto

- Construir el marco teórico que permita realizar un estudio sobre la situación actual de la producción de brócoli en la Empresa Ecofroz S.A. durante el cuarto trimestre del año 2019, desarrollando un sistema software basado en el sistema lenguaje de procesamiento de datos, frameworks móviles, middlewares y frameworks web,

también se utilizará los textos planos que proporcionaran las estaciones meteorológicas inalámbricas.

- Desarrollar sistema software, mediante el análisis de requisitos de la aplicación basado en la norma IEEE 830, diseñar y la aplicación correspondiente.
- Implementar el sistema software para el cálculo de grados día en una plantación de brócoli.
- Validar el sistema software para el cálculo de grados día en una plantación de brócoli.
- Establecer una arquitectura implementando componentes de software que mejor se adapten al desarrollo de la solución informática y al entorno del internet de las cosas; considerando aspectos como la seguridad, transaccionalidad, interoperabilidad, escalabilidad y licenciamiento.

1.1.1. Preguntas de investigación.

¿Cuál es el fundamento teórico en el cual se basará la realización de la presente investigación enfocándose en la producción actual de brócoli en la empresa ECOFROZ?

¿Cuáles son los requerimientos necesarios para diseñar e implementar el sistema software propuesto en la presente investigación basándose en la norma IEEE 830?

¿Cómo se puede implementar un sistema software para el cálculo de grados día en una plantación de brócoli?

¿De qué manera se puede validar el sistema software para el cálculo de grados día a implementarse en una plantación de brócoli?

¿Qué componentes de software se requieren para la implementación de una arquitectura que se adapte al desarrollo de la solución informática centrada en el control de grados día y al entorno del internet de las cosas, considerando aspectos que generen un software de calidad?

1.2. Justificación e Importancia.

La industrialización de la agricultura se ha dado a conocer con el nombre de "agroindustria[1]", los estudios de sistemas agrícolas documentan la historia del control del agua, formas de organización sociopolítica, la tecnología de riego, el mercado, el mejoramiento de las plantas, y la intensificación del uso del suelo y los tipos de asentamientos humanos (Pérez J. , 2013).

Al no existir un sistema de cálculo de grados día para el control del cultivo la eficacia nutricional del brócoli se vería afectada, disminuyendo su calidad e incluso llegando a detener su exportación, generando pérdidas para la economía de la empresa y la sociedad (Toledo, 2003).

Otro de los inconvenientes detectados es el desconocimiento de las anomalías meteorológicas (factores climáticos) lo que incide en que la parte comestible del brócoli (inflorescencia) suele exceder sus grados térmicos óptimos, causando así una disminución nutritiva en el producto, inclusive, haciéndolo no apto para el mercado mundial (Soria, 2016).

A través de esta solución informática se beneficiará toda la empresa consultora Asis – Agro y las diferentes plantaciones de brócoli de Ecofroz, ya que en los cultivos de este producto alrededor del país, aún no se ha realizado ningún estudio que ayude a la recopilación de temperaturas para el cálculo de grados día, llegando a alcanzar mejores resultados en comparación a sus competidores.

El presente proyecto de investigación, está orientada a optimizar el ciclo productivo del brócoli realizando cálculos de grados día por medio de una API meteorológica de código abierto la misma que suministrara datos históricos de temperatura máxima y mínima del medio ambiente. Dicha información será utilizada como entradas para el desarrollo del algoritmo de predicción de temperaturas, el cual utilizará como apoyo el internet de las cosas, suministrando al personal encargado una fecha optima de

[1] La agroindustria es un concepto que se ha ido modificando con el tiempo, en su concepción genérica constituye el aprovechamiento de las materias primas producidas por la actividad agrícola para ser transformadas en productos terminados de consumo humano o animal, con fines de industrialización.

cosecha que será visualizada en dispositivos móviles y navegadores web, además de permitir llevar un seguimiento semanal de la evolución del cultivo.

Por este motivo, cabe resaltar que este tema de investigación tiene su cierta originalidad, debido a que, indagando en distintos tipos de material de consulta, poco o nada de información se puede encontrar acerca de este proceso.

1.3. Objetivos

1.3.1. Objetivo General

- Desarrollar un sistema software para el cálculo de grados día en una plantación de brócoli con el fin de optimizar el ciclo productivo valiéndose de estaciones meteorológicas inalámbricas, en la Empresa Ecofroz S.A. en transcurso del cuarto trimestre del año 2019.

1.3.2. Objetivos Específicos.

- Construir el marco teórico que permita realizar un estudio sobre la situación actual de la producción de brócoli en la Empresa Ecofroz S.A. durante el cuarto trimestre del año 2019, desarrollando un sistema software basado en el sistema lenguaje de procesamiento de datos, frameworks móviles, middlewares y frameworks web, también se utilizará los textos planos que proporcionaran las estaciones meteorológicas inalámbricas.
- Desarrollar sistema software, mediante el análisis de requisitos de la aplicación basado en la norma IEEE 830, diseñar y la aplicación correspondiente.
- Implementar el sistema software para el cálculo de grados día en una plantación de brócoli.
- Validar el sistema software para el cálculo de grados día en una plantación de brócoli.
- Establecer una arquitectura implementando componentes de software que mejor se adapten al desarrollo de la solución informática y al entorno del internet de las cosas; considerando aspectos como la seguridad, transaccionalidad, interoperabilidad, escalabilidad y licenciamiento.

1.4. Hipótesis

¿Si se desarrolla un sistema software para el cálculo de grados día, entonces se optimiza el ciclo productivo del brócoli?

1.5. Variables de la investigación.

- Variable Independiente: desarrollo de un sistema móvil y web
- Variable Dependiente: optimización del ciclo productivo con el uso del cálculo de grados día.
- Indicadores:
    - Monitoreo de temperaturas diarias a través de una API de control meteorológico de código abierto que provee datos climáticos.
    - Predicción de temperaturas según datos climatológicos históricos por medio del algoritmo ARIMA para el cálculo de grados día.
    - Seguimiento de avance del cultivo mediante reportes semanales de crecimiento del brócoli.

2. Marco Teórico

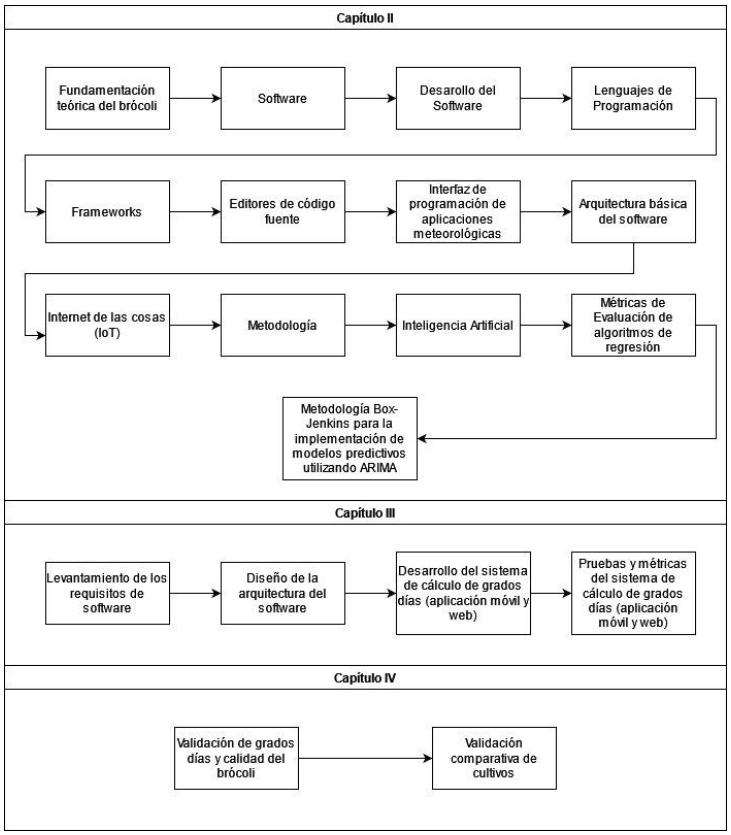
Este capítulo presenta en detalle el contenido teórico de esta investigación, como se muestra en la Figura 1. A continuación, se listan sus partes: 1) Conceptualización del brócoli, 2) Generalización del Software, 3) Fundamentación del Desarrollo del Software, 4) Inducción a los Lenguajes de Programación, 5) Marcos de Trabajo Frameworks, 6) Abstracción de la Arquitectura Software, 7) Metodologías de Desarrollo de Proyectos y 8) Modelo de Predicción.

En el capítulo III (Análisis, diseño y desarrollo del sistema de software para el cálculo de grados días) se especifica el diseño y desarrollo del sistema propuesto en la siguiente investigación, como se puede visualizar en la Figura 1. detallándose a continuación sus fases: 1) levantamiento de requisitos, 2) desarrollo de la metodología SCRUM, 3) aplicación de la arquitectura y 4) Desarrollo del Modelo de Predicción.

Finalmente, se muestra en la Figura **1**. lo correspondiente al capítulo IV (Validación del sistema de software para el cálculo de grados días) donde se realiza verificación y validación del sistema propuesto en esta investigación a continuación se presentan sus etapas: 1) pruebas de integración y 2) métricas del algoritmo de predicción.

Figura 1

Estructura de la investigación realizada



Nota. En esta figura se muestra toda la estructura de la investigación realizada

2.1. Fundamentación teórica del brócoli.

2.1.1. El brócoli en Ecuador

La producción de brócoli en Ecuador comenzó en 1990 desarrrollandose de manera robusta e invariable a lo largo del tiempo, el afianzamiento de este producto fue descisivo a partir del año 2000 haciendo de esta hortaliza el fruto secundario no habitual de exportación, detrás de las rosas, poscicionando al país como el noveno fabricante mundial de brócoli fresco (APROFEL, 2016).

Este posicionamiento mundial en parte es gracias a la creación de la CORPEI en 1997 la cual impulsa la exportación de productos ecuatorianos no petroleros, y por las construcciones viales y portuarias del país, el brócoli puede ser exportado ágilmente (Ordóñez, 2010).

Aprovechando estos beneficios y tomando en cuenta el territorio de la Sierra ecuatoriana el cuál reúne incomparables ventajas geográficas para la obtención de un buen brócoli, con características como su relación directa a los rayos del día, dando una luminosidad única en el planeta la cual pintarrajea este fruto de un tono verde agudo en conjunción al sembrado en elevación que limita la aparición de plagas haciendo que el florecimiento tenga una aglutinación óptima y gracias a un clima estable en el transcurso del año, se puede producir frecuentemente, a capacidad de tres cosechas anuales (CESA, 2015).

2.1.2. Zona de investigación y sus climas relevantes.

Lasso, lugar donde se realizó la investigación, pertenece a la Sierra ecuatoriana, concretamente asentada en la Parroquia de SAN LORENZO DE TANICUCHI, que por su localidad geográfica se encuentra ubicada en la porción céntrica del callejón interandino, entre las cordilleras céntrica y occidental, a una elevación media aproximada de 2.981 metros sobre el nivel del mar (m.s.n.m.); conformada por una región llana que mantiene una inclinación moderada, con una característica que presenta ciertas ondulaciones con pendientes menores al 8.0% (Riera Tubón, 2004).

Por lo cercanía de este territorio con el volcán Cotopaxi, el tiempo que sobresale es mayormente frio, con poca predisposición a lluvias en abundancia en el transcurso del año, existiendo en la región poblada tiempos que oscilan entre los 14˚c a 22˚c con una media de 18˚c; es de suma importancia subrayar que el período de verano se presenta con vientos fuertes (Pourrut, 1983). Además, existe una influencia de los siguientes tipos de climas:

A) Ecuatorial de Alta Montaña

En los sitios montañosos el clima disminuye conforme la altura, mientras que a su vez aumentan las precipitaciones, al menos hasta una altura determinada. Esta zona presenta un clima invernal negativo y unos veranos positivos, si bien el clima medio anual se establece entorno a los 6 ºC; la fluctuación térmica está debajo a los 18º y las precipitaciones son más abundantes en verano, superando los 1.000 milímetros anuales

en la región alta. Este tiempo de alta montaña es el que predomina en la Sierra andina (Barros López, 2010).

B) Clima mesotérmico semihúmedo

Otro de los climas que forman parte del ecosistema ecuatoriano y que tiene mayor relevancia en el sector agrario, es el llamado mesotérmico semihúmedo. Las lluvias anuales son de 500 a 2.000 milímetros, tiene dos estaciones lluviosas que oscilan entre febrero a mayo y octubre a noviembre. Es el clima que en su mayoría se encuentra ubicado en la Sierra. El clima medio oscila entre 12 y 20º C (Pourrut, 1983).

C) Clima mesotérmico seco

Su precipitación anual es de menos de 500 milímetros. Este clima seco y moderadamente caluroso está presente en el fondo del valle entre los callejones andinos. Las temperaturas y la flora de este ambiente son las mismas que las del tiempo mesotérmico semihúmedo (Pourrut, 1983).

2.1.3. Importancia del Brócoli.

La principal caracterización de esta hortaliza es su composición rica en vitamina C, al igual que posee propiedades medicinales por su agudo contenido en hierro, es considerablemente recomendada para regular la diabetes, actúa beneficiosamente en los riñones, vesículas e intestinos, a más de fortificar las defensas del cuerpo. Recientes estudios científicos atribuyen al Brócoli beneficios anticancerígenos que previenen problemas prostáticos (Vigliola, 2015).

Además, los beneficios del brócoli para la salud son amplios, no obstante, cabe resaltar que también es uno de los productos más importantes para la economía nacional, ya que después de las rosas, el brócoli es uno de productos más exportados. Reconociendo a Ecuador como uno de los mejores productores de esta hortaliza a nivel mundial (Casseres, 1984).

2.1.4. Tiempo térmico del brócoli.

La temperatura del brócoli es uno de los elementos más importantes en el progreso de los cultivos del brócoli, midiéndose a través de la suma de temperaturas diarias. Requiriéndose para esta operación de un clima base de 4 grados centígrados (°C). A esta técnica se la denomina cálculo de grados días (Daubenmire, 2000).

Adicionalmente el clima soportado por esta hortaliza oscila en el rango de 4 a 30° C, pudiendo sobrevivir a heladas débiles (-3 a 0 ° C) solamente en estado juvenil (0 a 2 semanas). Además, para que una pella (conjunto de tallos) de alta calidad logre formarse necesita de una temperatura que fluctúa entre 14 y 16° C, caso contrario será de menor calidad (Francescangeli, Stoppani , & Martí , 2014).

2.1.5. Grados día y su influencia en cultivos.

Los grados día es la medida de calentamiento (medidos en grados centígrados o Kelvin), requeridos para conseguir un clima confortable para un cultivo en general, determinando así el pronóstico de fecha para la cosecha (Rodríguez M. , 2014).

Estos son importantes para evaluar las necesidades energéticas del cultivo, teniendo en cuenta que para ello se necesita de tres factores importantes: 1) La temperatura máxima, 2) La temperatura mínima y 3) La temperatura umbral (Santos, 2010).

El uso del cálculo de grados día ha demostrado que el tiempo de cosecha de un cultivo es fluctuante, ya sea por las características del suelo o causas medioambientales, concluyendo con una mayor o menor duración en el ciclo de desarrollo de la planta (Maqueira López, 2016).

a. Modelo para el cálculo de grados días.

El modelo que se utiliza para la cosecha de un producto agrícola en general es el cálculo de grados días. En la Ecuación 1 se observa que su valor se calcula mediante la sumatoria de la media entre la temperatura máxima diaria (T° max) y temperatura mínima diaria (T° min) menos la temperatura umbral (T° umbral). Además, debe

considerarse que la temperatura umbral no debe ser inferior a la sumatoria de la temperatura máxima diaria (T° max) y temperatura mínima diaria (T° min) (Arnold, 1959).

Ecuación 1

Modelo de cálculo de grados día.

$$DG: \sum (T^{\underline{o}}\, máx. + T^{\underline{o}}\, mín)\ /2 - \ T^{\underline{o}}\, umbral$$

Nota. En esta ecuación se muestra la fórmula para el cálculo de grados días

Otro aspecto importante de dicha temperatura es tomar en consideración que en el caso de brócoli debe ser de 4 grados centígrados(°C). Estos factores meteorológicos (ambiente climatológico) definidos anteriormente, deben acumular 909 grados para realizar correctamente su cosecha (Ayala, 2010).

2.2. Software.

El software es la porción no física que hace alusión a un sistema o conjunción de sistemas de cómputación que incluye datos, reglamentos y explicaciones para poder comunicarse con el computador haciendo viable su trabajo (CISET, 2019)

Sin Software, los ordenadores serían inútiles. Este es desarrollado mediante el uso de distintos lenguajes de programación que consisten en símbolos, reglas semánticas y sintácticas que definen el significado de sus elementos y enunciados (CISET, 2019). Estas instrucciones se ejecutan en programas informáticos que proporcionan las características y funciones deseadas.

También se pueden definir como estructuras de datos que permiten a los programas procesar la información correctamente. El software es el elemento de un sistema lógico y no de uno físico. Por tanto, tiene características que difieren considerablemente de las del hardware (Pressman, 1988).

2.2.1. Características del Software.

Uno de los aspectos más importantes a tomarse en cuenta es que no todos los tipos de software cumplen con todas las características definidas para un sistema, ya que priorizaran una cualidad más que otra. Dependiendo de su importancia de acuerdo a su funcionalidad. Según (Calero, 2010) las principales características del software son:

a) Flexibilidad. Capacidad de poder quitar, agregar o modificar funcionalidades sin perder enfoque alguno o averiar el sistema, adaptándolo así a posibles requisitos cambiantes.
b) Portabilidad. Cualidad enfocada a la ejecución del sistema en varias plataformas por medio de la reutilización y adaptación del código fuente sin la necesidad de crear código nuevo.
c) Corrección. Habilidad para poder detectar errores en fases tempranas del desarrollo del software, ahorrando así tiempo y dinero, brindando así un software más seguro para el usuario.
d) Seguridad. Capacidad del sistema para bloquear ataques maliciosos, resistirlos y seguir funcionando con normalidad ante estos, haciéndolo confiable para el almacenamiento de datos sensibles.
e) Usabilidad. Característica que tiene el sistema para ser efectivo, atractivo y comprensible para el usuario, utilizando entornos gráficos llamativos y funcionales a la vez.

2.2.2. Tipos de Software.

Existen infinidad de maneras de clasificar los tipos de software una de estas es agruparlos de acuerdo a su movilidad (Laplana Martín, 2019). Estos además varían dependiendo de la plataforma en la que se ejecutan, permitiendo al usuario mantener un contacto directo con la interfaz, estas pueden ser de tipo Web y Móvil los cuales se explican a continuación (Clasificación Del Software., 2017) :

a) Sistema web.

Un sistema web es una aplicación informática la cual es accedida desde diversos navegadores vía web por una red como intranet o internet, siendo su ventaja principal él no necesitar ningún tipo de instalación y poder correrla desde diferentes entornos ya sea Linux, Windows, Mac, etc. (Alegsa, 2020).

Estás aplicaciones están relacionadas con el almacenamiento en la nube, ya que guardan permanentemente toda la información necesaria para su interacción con el usuario en servidores web, ayudando a la consulta de datos en cualquier momento que sean requeridos (Mamani, 2017).

b) Aplicación Móvil.

Las aplicaciones móviles o también denominadas app (acortamiento del inglés application) son programas informáticos los cuales se diferencian de otros, porque estos corren en un entorno móvil y no en un entorno fijo, haciendo de su uso más versátil y eficiente en cualquier lugar donde el usuario se encuentre. (Benítez, 2017).

Este tipo de software realiza su compilación de una manera individual dependiendo de la plataforma en que se utilice. Ejecutándose la mayoría de estos programas en teléfonos inteligentes(smartphones) tabletas y otros dispositivos móviles. Facilitando a las personas su uso y ejecución en diversas tareas (profesional, ocio, educativas, etc.) (Moreno, 2016). Lo cual se ve reflejado en su fácil acceso e instalación, utilizándose diversas plataformas de descarga como por ejemplo Google Play y App Store (Martelo, 2014).

2.3. Desarrollo del software.

El desarrollo de software es un tema que se enfoca en la construcción organizada de aplicaciones informáticas por medio de procesos, este se basa en la descripción de un problema que requiere automatizar una necesidad (Canós, 2012).

Dicha automatización requiere la intervención de varias personas detallándose cada una de estas a continuación (Sicilia, 2008):

1) El cliente. - persona que tiene la necesidad de resolver algún problema de su empresa por medio del uso de programas informáticos.
2) El Analista del Sistema. - personal encargado de analizar la problemática del cliente, convirtiendo dichos inconvenientes en requisitos los cuales deben ser cumplidos al momento del desarrollo del sistema propuesto.
3) El personal de desarrollo. - es el personal técnico que analiza los requisitos obtenidos por el analista del sistema, convirtiendo dichas funcionalidades en un sistema óptimo para el uso del cliente por medio del uso de lenguajes de programación que estén enfocados en la arquitectura propuesta para su desarrollo.

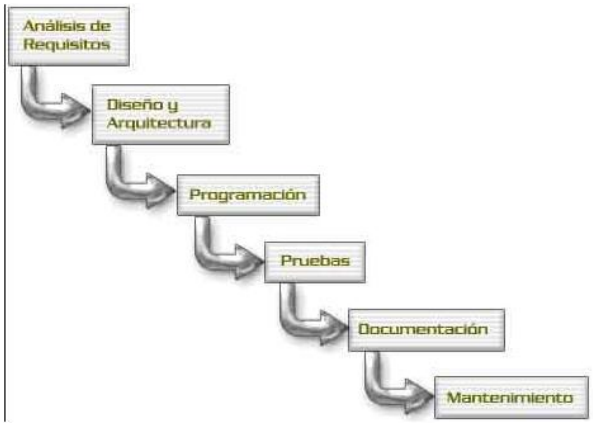
2.3.1. Fases del desarrollo del Sistema de Software.

Partes más importantes en la implementación del software, ya que determina el orden en el que debe realizarse el desarrollo del sistema. Estás permiten una adecuada ejecución del proyecto, optimizando tiempos y entregando un producto de calidad (Canós, 2012).

En la Figura 2, se observan las etapas más comúnmente utilizadas y recomendadas en el proceso de desarrollo de software estándar. Estas etapas son: 1) Análisis de Requisitos, 2) Diseño de la arquitectura, 3) Desarrollo, 4) Pruebas, 5) Documentación y 6) Mantenimiento.

Figura 2

Fases del proceso de desarrollo del sistema de software



Nota. En esta figura se muestra las fases que toma el proceso para elaborar un sistema de software. Tomado de (Campderrich , 2014)

A continuación, se describe cada una de las fases mencionadas anteriormente requeridas para el desarrollo del software:

1. **Análisis de requisitos. -** Fase en la que se extraen los requerimientos (funcionalidades a implementar en el sistema) de un producto, en función de las necesidades del cliente; el resultado de esta actividad se ve reflejado en el documento ERS (Especificación de Requerimientos Software). Dichas necesidades deben ser estructuradas de acuerdo a estándares que guían la planificación y ejecución del proceso de desarrollo de software, apoyándose en la elaboración y organización de la documentación. De acuerdo con (Toro, 2001) los más importantes son los que se detallan a continuación:
   - **ISO 29148. -** Estándar que define varias guías para la elaboración de procesos relacionados con la ingeniera de requisitos y brinda servicios de soporte a lo largo del ciclo de vida del producto.
   - **ISO 15288. -** Estándar que establece un conjunto de procesos que sirven para describir a detalle toda la elaboración del proyecto, además aporta una seria de indicadores encaminados a la planificación de cada fase de elaboración del software.
   - **ISO 24766. -** Estándar que proporciona orientación sobre las capacidades optimas con las que deberían contribuir las herramientas que ayudan a la documentación de requerimientos.

- **IEEE Std 830. -** Estándar que define guías para describir los requerimientos software e de manera sencilla, dividiéndolos en: requisitos funcionales, los cuales definen las funciones que el sistema deberá realizar y requisitos no funcionales que especifican el tiempo de respuesta, plataforma en la que se ejecutará, tipo de seguridad, usabilidad, confiabilidad y su disponibilidad.

2. **Diseño de la arquitectura. -** Fase donde se realizan una serie de diagramas (de componentes y UML) basados en el análisis de requisitos (fase 1 - desarrollo de software) que describen la funcionalidad y tecnologías a implementar en el software de manera precisa. Cabe destacar que UML es el sistema de modelado de arquitectura software más utilizado debido a su facilidad de aplicación, optimización de tiempo, alta reutilización y minimización de costos. Además, es de fácil entendimiento para el usuario. Según (Almenara, 1992) los diagramas que se utilizan para detallar la arquitectura del software son los siguientes:
   - **De casos de uso. -** Diagrama mediante el cual se representa la interacción de las funcionalidades obtenidas en el levantamiento de requerimientos (fase 1 - análisis de requisitos) permite visualizar la relación existente entre usuarios u otros sistemas. Además, detalla la comunicación de todos los componentes que forman parte del software para que así todo el personal involucrado o cliente pueda comprender de mejor manera su funcionalidad (requisitos funcionales).
   - **De clases. -** Diagrama que representa la relación de varios conjuntos de datos denominados clases. Cada clase está conformada de atributos (datos o variables), métodos (operaciones entre variables) y visibilidad (publica, protegida o privada). Este diagrama tiene como objetivo visualizar la forma en que se comportará la información manipulada por el sistema.
   - **De secuencias. -** Diagrama que permite visualizar eventos del sistema en orden cronológico. Este sirve para describir de modo gráfico la forma en que los actores (usuarios o funcionalidades) intercambian información en una secuencia determinada.
   - **De despliegue. -** Diagrama que muestra la interconexión de cada nodo del sistema (front-end, back-end, base de datos), los cuales en la mayoría de ocasiones están estructurado por hardware y software. El objetivo de este diagrama es analizar y dar a conocer las tecnologías (lenguajes de programación, frameworks) que se implementarán en el proyecto.

- **De paquetes. -** Diagrama que representa la secuencia de interacción del usuario con los elementos lógicos que conforman el sistema, a estos se los llama paquetes. Su propósito es dar a conocer el orden mediante el cual el software es accedido. Cada paquete se lo especifica por la función que cumple como, por ejemplo: capa de autenticación, presentación, servicio web, negocios o datos.

3. **Desarrollo. -** Fase donde se construye el software mediante el uso de lenguajes de programación, de acuerdo a los establecido en el documento obtenido en la etapa de análisis de requisitos (fase 1 – desarrollo de software) y en la etapa de diseño de la arquitectura (fase 2 - diseño de la arquitectura) (Pressman, 1988).

4. **Pruebas. -** Fase en la cual se verifica y valida el software construido de acuerdo a las funcionalidades descritas en el análisis de requerimientos (fase 1 – desarrollo de software). Así como también se evalúa el comportamiento de las tecnologías que se establecieron en el diseño de la arquitectura (fase 2 - diseño de la arquitectura) y se examina los tiempos de respuesta y la usabilidad del sistema. Según (Sommerville, 2005) Hay varios tipos de pruebas, siendo las más destacadas las siguientes:
   - **Pruebas unitarias. -** Evaluaciones que aíslan una parte del código fuente para verificar si este funciona correctamente. Estás se las realiza durante la fase de programación (fase 4 - Desarrollo). Este tipo de pruebas se ejecutan de manera automática, empleando herramientas informáticas especializadas.
   - **Pruebas de integración. -** Verifican si el sistema desarrollado cumple con todo lo descrito en la fase de análisis de requisitos (fase 1 - desarrollo de software). De esta manera también se corrobora la calidad del software acorde con las necesidades del cliente, su funcionalidad, eficiencia y que tan rápido el usuario aprende a utilizarlo.
   - **Pruebas exploratorias. -** Realizadas por una persona denominada “tester” la cual comprueba cómo responde el software, que tan usable es, su grado de eficiencia y si cumple o no con las necesidades del cliente. Esta sirve para evaluar cada operación que realiza el sistema, corroborando que no posea error alguno.
   - **Pruebas de carga. -** Etapa donde se evalúa el comportamiento del sistema en escenarios poco favorables como son: ingreso simultaneo de usuarios,

transacciones realizadas en el mismo momento y tiempos de respuesta a peticiones de datos requeridos por el usuario.

5. **Documentación. -** Fase donde se detalla por escrito todo el desarrollo del software incluyendo manuales técnicos, manuales de usuario, código fuente, códigos de error y todo lo que facilite su comprensión. Esto se lo realiza para que el usuario tenga una guía de aprendizaje u otros programadores puedan entender y modificar algún componente del sistema. Cada guía realizada en esta etapa puede hallarse de forma electrónica o física (Sommerville, 2005).

6. **Mantenimiento. -** Fase donde se modifica el software después de ser entregado al cliente. Esto se lo realiza para poder optimizar el sistema, mejorar su rendimiento, corregir errores o adaptarlo a nuevos requerimientos que el usuario final necesita añadir al sistema inicial (García F. R., 2002).

2.3.2. Modelos o ciclos de vida del Software.

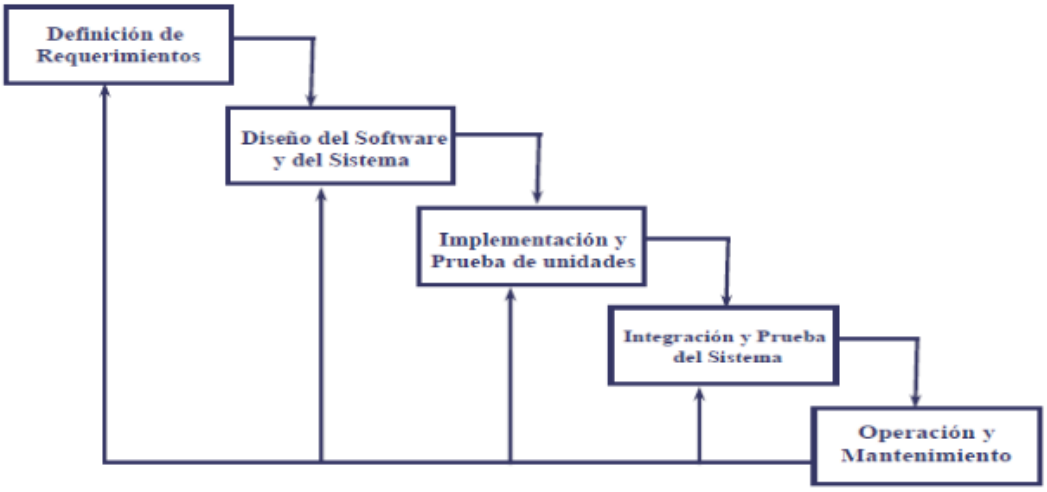
Procesos por los cuáles debe atravesar el desarrollo del proyecto para generar un sistema acorde a los requerimientos del usuario (Gutierrez, 2011)Se los utiliza para poder ordenar el avance del proyecto generando guías, descripción de errores, detalle de funcionalidades, etc. Estos procesos pueden requerir varias tareas que varían de acuerdo al modelo a escoger (Sommerville, 2005).

A continuación, se listan los modelos más conocidos:

1. **Modelo en cascada. -** En la Figura 3 se puede visualizar uno de las primeras representaciones que se propuso en el desarrollo del software. Siendo su característica principal que cada una de sus cinco etapas se ejecuta en orden secuencial por lo que no se puede comenzar una etapa sin terminar la anterior, este modelo necesita iniciar con la especificación de requisitos de software para posteriormente dividir estos en partes más entendibles y así continuar con las siguientes 4 fases (Diseño del software y del sistema - Implementación y pruebas de unidades - Integración y pruebas del sistema) hasta llegar a la última etapa operación y mantenimiento (Gutierrez, 2011).

Figura 3

Modelo en cascada o convencional



Nota. En esta figura se muestra las fases que conforman el modelo en cascada. Tomado de (Gutierrez, 2011)

2. **Modelo evolutivo. -** En la
3.
4.

5. **Figura 4** se puede observar las fases que conforman este modelo, iniciando con la descripción informal del problema, seguido de las actividades necesarias para el avance del proyecto de desarrollo de software como son la especificación de requisitos, desarrollo y validación, las cuales generan varias versiones del sistema (inicial, intermedias y final), necesitando mejorarlo hasta obtener una versión final que cumpla con todas las necesidades del cliente (Bernd & Allen, 2014).

Figura 4

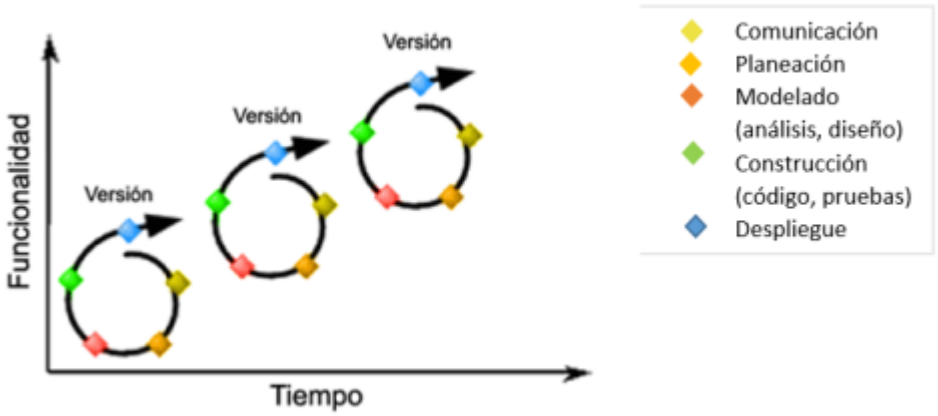
Modelo evolutivo

Nota. En esta figura se muestra el proceso por el que pasa el desarrollo del sistema de software según el modelo evolutivo. Tomado de (Bernd & Allen, 2014)

6. **Modelo incremental. -** En la Figura 5 se observa la forma en que este modelo realiza el desarrollo de software, iniciando por la fase de comunicación en la cual se detalla la problemática del cliente, luego se procede con la planeación donde se establece las funcionalidades que tendrá el software, seguido del modelado, en esta se analiza las tecnologías que se implementarán junto con el diseño de las interfaces de cara al usuario y como interactuarán entre sí, a continuación se pasa a la construcción del proyecto para finalizar con su despliegue. Todas estas fases se repiten por cada funcionalidad establecida, entregando así varias versiones del sistema hasta llegar a cumplir con todas (Pressman, 1988) (Alava, 2015).

Figura 5

Modelo incremental

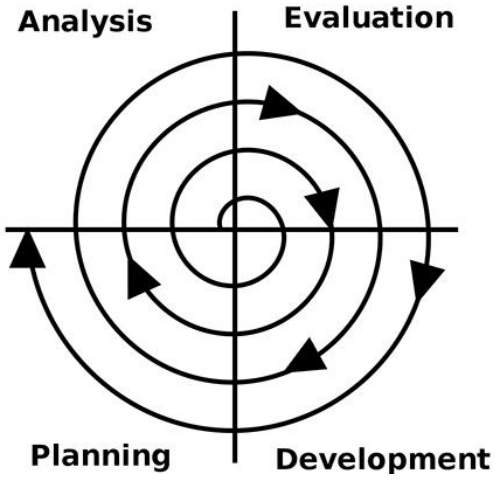


Nota. En esta figura se muestra las fases e iteraciones por la que atraviesa el desarrollo de software en el modelo incremental. Tomado de (Alava, 2015).

7. **Modelo en espiral. -** En la Figura 6 se puede observar una espiral conformada por 4 fases, comenzando por la etapa de análisis (Analysis) en la cual se evalúa los requerimientos del cliente, a continuación, se realiza la etapa de evaluación (Evaluation) donde se realizará la codificación del sistema, seguido de la etapa de desarrollo (Development) donde se codificará el proyecto para terminar con la planeación (Planning) donde se examinará si el sistema es apto para el cliente, de no serlo la espiral descrita sigue su curso presentando prototipos hasta llegar a una versión final (Dutoit, 2014)

Figura 6

Modelo en espiral



Nota. En esta figura se muestra las 4 fases que conforman el modelo en espiral. Tomado de (Bernd & Allen, 2014)

2.3.3. Calidad del Software.

La calidad ayuda a definir el grado de madurez de un producto, cumpliendo con estándares y requisitos mínimos, priorizando los requerimientos del software. Es la congruencia que existe entre los requerimientos y el rendimiento del software. (Roger., 2010).

Además, está conformada de varias características que son evaluadas en el transcurso del desarrollo del software. Siendo las más importantes: 1) Fiabilidad, 2) Eficiencia y 3) Funcionalidad (Callejas-Cuervo, 2017).

a. Características de la Calidad del Software.

Conjunto de cualidades que pueden ser experimentadas al manipular cualquier tipo de software, manifestándose así su nivel de calidad. Estas son de gran importancia ya que permiten evaluar el sistema antes de poder ser entregado al cliente, entre las más importantes se encuentran las siguientes (Calero, 2010):

- **Fiabilidad. -** Capacidad del software para operar libre de fallos por largos periodos de tiempo.

- **Funcionalidad. -** Habilidad del sistema para cumplir funciones que satisfacen las necesidades del usuario.
- **Eficiencia. -** Cualidad del producto desarrollado para proporcionar un desempeño óptimo con respecto a la cantidad de recursos utilizados.
- **Facilidad de mantenimiento. -** Capacidad para adaptar nuevas funcionalidades o editar algunas existentes del sistema producido.

2.4. Lenguajes de programación.

Las computadoras tienen la facultad de realizar procesos; estos pueden ser diseñados de forma tal que generen una tarea específica generada a través de lenguajes de programación. La funcionalidad de estos lenguajes es variada desde crear un programa que ayude al control de un acto lógico y físico de un computador o máquina, dar a entender algoritmos con claridad o ayudar en la comunicación humana. Para ser considerado un lenguaje de programación debe existir semántica y reglas sintácticas que define sus estructuras y el significado de sus propias expresiones y elementos (Olarte, 2018).

2.4.1. Tipos de lenguajes para programación.

Dentro del mundo de la programación existen 3 tipos de lenguajes, los cuáles son de mayor o menor comprensión para el ser humano, siendo principalmente utilizados para el desarrollo de sistemas software los lenguajes de alto nivel. A continuación, se detallan estos tipos de lenguajes:

1. **De máquina. -** Denominado también de bajo nivel, es un conjunto de dígitos binarios o bits que la máquina o computadora lee e interpreta. Para un humano es imposible comprender este lenguaje ya que son miles de números procesados al instante (Olarte, 2018).
2. **Ensamblador. -** Aquel que es más comprensible para los humanos, consiste en un conjunto de varias instrucciones a ejecutar por un microprocesador. Para que este pueda ser interpretado, un programa basado en ensamblador transforma el

código escrito a lenguaje de bajo nivel el cuál como se indicó en el párrafo anterior, puede ser comprendido por un computador (Olarte, 2018).

3. **De alto nivel. -** Permite la implementación e interpretación de algoritmos que pueden ser fácilmente leídos por un humano. Este tipo de lenguaje es el que más se usa entre los programadores, ya que su eficiencia y eficacia en cuanto al procesamiento de datos lo hace viable para su utilización en proyectos informáticos (Olarte, 2018).

a. Lenguajes más utilizados de alto nivel:

Después de explicar los tipos de lenguajes de programación en los párrafos anteriores, se puede describir en detalle los lenguajes de alto nivel más utilizados. A continuación, se realiza una corta descripción de sus características principal:

1. **JavaScript. -** Lenguaje de programación que permite la implementación de tareas complejas en páginas web, como la creación de contenido actualizable, animación de imágenes, control multimedia, procesamiento de operaciones matemáticas, entre otras (Solis, 2016).
2. **TypeScript. -** Lenguaje de programación Basado en JavaScript, tiene la característica de ser fácilmente escalable, haciendo ideal su aplicación en proyectos de complejidad media -alta. Siendo su principal característica la transformación de todo su código a JavaScript al momento de su compilación, es decir transforma todo su código a JavaScript nativo. A demás cabe destacar que el framework más utilizado en el desarrollo de sistemas web denominado Angular está conformado (Fernández, 2016).
3. **Python. -** Lenguaje de programación orientado a la implementación de código legible, permitiendo ser usable en múltiples plataformas y paradigmas de codificación. Su curva de aprendizaje de gran relevancia indica su facilidad de entendimiento. Tiene una licencia de código abierto por OSI, siendo este de libre distribución. Su variedad de librerías, funciones y tipos de datos simplifican el desarrollar tareas de código escaso (Van Rossum, 1991).
4. **Dart. -** Lenguaje de programación mayormente utilizado en proyectos de desarrollo móvil debido a su compatibilidad con el uso del framework Flutter (Dimas, 2018).

2.4.2. Algoritmos.

Sucesión de instrucciones secuenciales, con la finalidad de dar solución a diferentes decisiones o necesidades llevando a cabo procesos establecidos, hay que subrayar que los algoritmos no tienen relación con los lenguajes de programación, dado que un diagrama de flujo o algoritmo puede representarse en múltiples lenguajes de programación, es decir esto es un paso previo a la programación (López, 2018).

Según (Winston, 2005) los algoritmos están conformados de varias características detallándose a continuación sus principales:

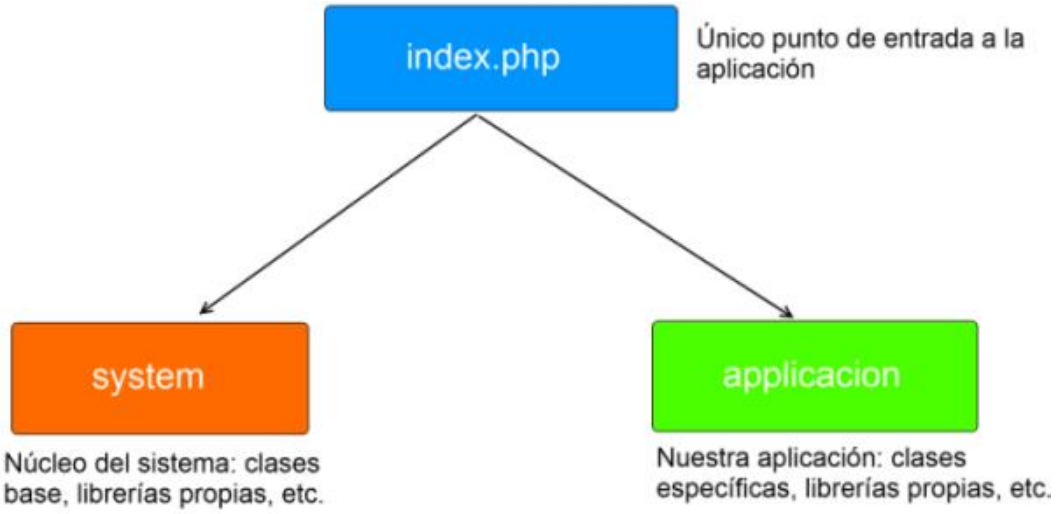
- **Concretos. -** Debe llegar a ofrecer exactamente lo que un requisito le asigne para satisfacerlo.
- **Secuenciales. -** Estos tienen un orden específico de ejecución, por este motivo deben ejecutarse uno a la vez.
- **Finitos. -** Al ser finito este debe tener un límite, pero pueden existir una cantidad infinita de ellos.
- **Precisos. -** Deben ser interpretado de manera puntual, dejando de lado la ambigüedad en cuanto a su análisis.
- **Ordenados. -** Se deben formar de manera secuencial y precisa para que al momento de su revisión tenga sentido.
- **Definidos. -** Siempre que se le dé un valor de entrada, este debe generar el resultado deseado sin ninguna variación.

2.5. Frameworks.

En la Figura 7 se observa la estructura básica de un framework, este tiene un punto de entrada que interactúa directamente con el usuario denominado index.php, el que a su vez se conecta a application y system. System es el núcleo del sistema el cual contiene todo lo básico para que el software pueda funcionar y application aloja toda la programación implementada por el desarrollador, así como código externo (Gutiérrez, 2017).

Figura 7

Framework



Nota. En esta figura se muestra la estructura básica de un framework. Tomado de (GITBook, 2018)

Los frameworks son variados y pueden ocupar diferentes lenguajes de programación, está característica los hace muy versátiles debido a la integración con diferentes herramientas, mejorando la implementación con las interfaces con las que interactúa el usuario y la lógica con la que se desarrolla el software (Gutiérrez, 2017).

2.5.1. NodeJS.

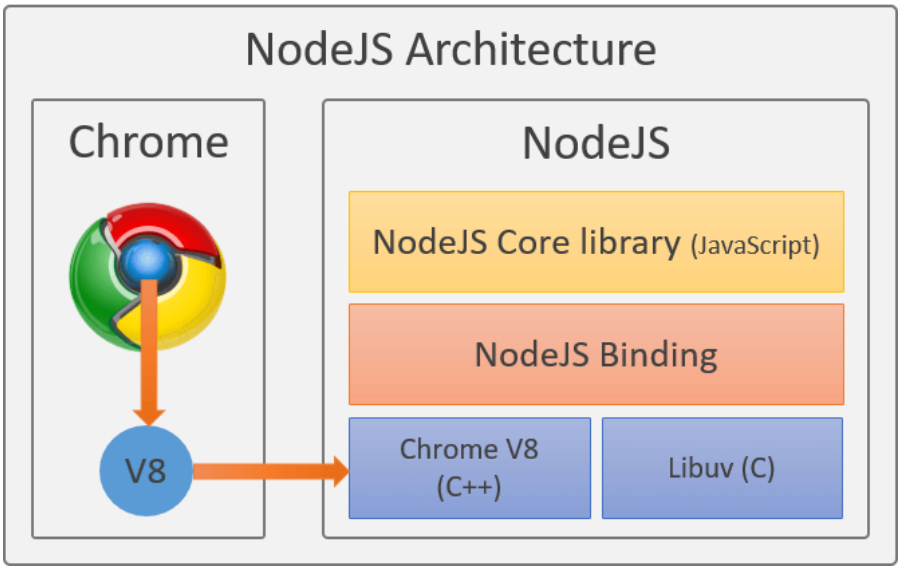
Framework basado en JavaScript, este se ejecuta del lado del servidor en tiempo real, caracterizándose principalmente en poseer todo lo necesario para leer un programa de JavaScript. Es de código libre, ligero y totalmente escalable lo que lo hace ideal para usarse en proyectos software de mediana a gran escala (Lucas, 2019).

En la Figura 8, se puede observar su arquitectura, este framework es accedido por medio de cualquier navegador, en este caso Google Chrome que se conecta de manera asíncrona al motor chrome v8 y a la librería libuv los cuáles están alojados virtualmente en la nube y son el núcleo de esta estructura informática. En un nivel superior se puede observar a NodeJS Binding, el cuál es un puente de comunicación entre los elementos que conforman el núcleo y la capa superior denominada NodeJS Core library que está constituida por el lenguaje de programación JavaScript y permite

la implementación de la lógica necesaria para el funcionamiento del servidor web a implementar en el sistema (Js, 2016).

Figura 8

Arquitectura NodeJS



Nota. En esta figura se muestra la arquitectura del framework NodeJS. Tomado de (Js, 2016)

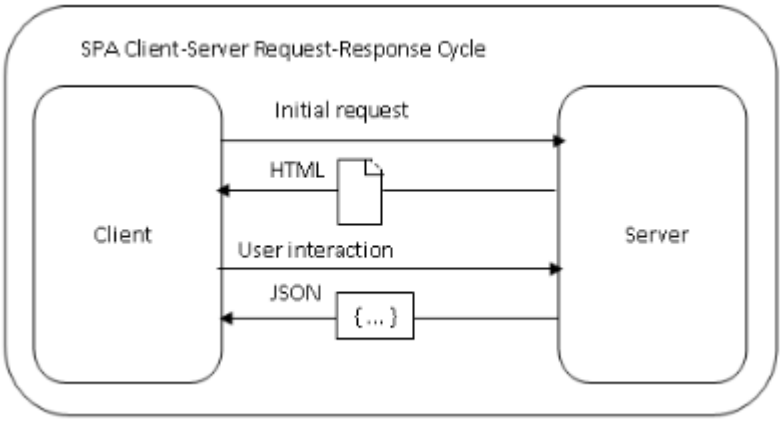
2.5.2. Angular.

Framework que facilita la visualización de interfaces en el desarrollo de aplicaciones, es decir permite al cliente observar una sola página sin tener que recargar el sistema como lo hace un sistema web habitual, haciendo su uso más eficaz y fluido (Careuno A. , 2018).

En la Figura 9, se muestra la interacción del servidor web (Server) y Angular (Client) el cuál envía un requerimiento inicial (initial request) retornando datos a mostrar en formato HTML. Una vez mostrados estos datos al cliente, este hace una consulta de información (User interaction) devolviendo un reporte en formato JSON (Jadhav, 2015).

Figura 9

Ciclos de requerimiento y respuesta en Angular



Nota. En esta figura se muestra los ciclos y respuestas que permiten el funcionamiento del framework Angular. Tomado de (Jadhav, 2015).

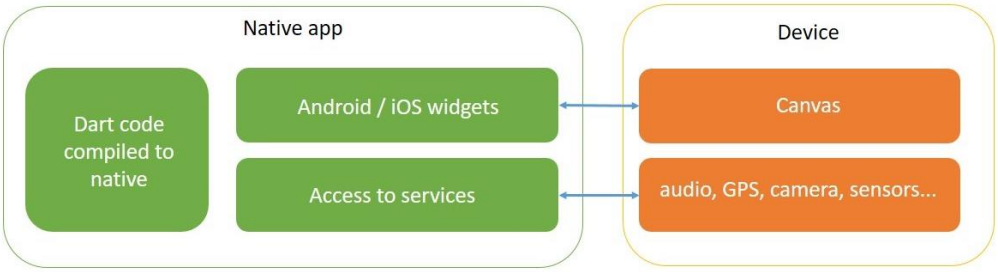
A demás cabe destacar que angular posee una gran compatibilidad con el framework de lado del servidor NodeJS ya que ambos están conformados por el lenguaje JavaScript, esto hace su comunicación más clara y estable soportando varios escenarios complejos como accesos múltiples de usuarios o peticiones simultaneas. (Careuno, 2018).

2.5.3. Flutter.

Marco de trabajo utilizado en el desarrollo de aplicaciones móviles multiplataforma, (movil framework). En la Figura 10 se puede observar la arquitectura de este framework comenzando por la compilación del código realizado en Dart a lenguaje nativo para que pueda ser ejecutado en android o iOS (Android/iOS widgets) mostrando su interfaz (canvas) en el dispositivo. De igual manera se puede acceder desde la aplicación a los servicios del smartphone como pueden ser su GPS, audio, cámara u otros sensores (Viejo, 2020).

Figura 10

Arquitectura de Flutter



Nota. En esta figura se muestra los componentes que forman parte de la arquitectura de Flutter. Tomado de (Catalunya, 2019)

2.6. Editores de código fuente.

Sistemas de software especializados en el desarrollo de aplicaciones informáticas. Estas son herramientas fundamentales para los programadores ya que dichos entornos tienen la capacidad de editar el código fuente, así como también integrar varias herramientas que hacen posible el funcionamiento de proyectos con cierto grado de complejidad (Payne, 2012). En el mundo computacional existen varios editores de código fuente de gran utilidad, a continuación, se detallan los más sobresalientes:

a) **Sublime Text. -** Aplicación de código privativo (de pago) que tiene la particularidad de ser ligera y versátil, ya que es capaz de ejecutar y editar la mayoría de formatos existentes (php, css, html, etc.). Una de sus desventajas es que tiene un diseño no muy amigable con el usuario, lo que hace que tenga una curva de aprendizaje mayor (Haughee, 2013).
b) **Visual Studio Code. -** Aplicación de código abierto (gratuita) que se caracteriza por ser robusta y de rápida respuesta al manejar varias tareas de gran intensidad. Cabe mencionar que cuenta con soporte para Git el cual sirve para controlar las versiones de avance de cualquier sistema de software, así como también con 'IntelliSense' que ofrece autocompletado de código e información sobre funciones (tareas que realiza el sistema) (Bree, 2016).

c) **Atom. -** Aplicación de código abierto (gratuito) que cuenta con una gran confiabilidad aún que carece de rendimiento y estabilidad, dichos problemas se los está solucionando poco a poco con el pasar del tiempo. Una de sus características más sobresalientes es su fácil integración con Git, llevando así un control de versiones aceptable en diferentes proyectos informáticos (Luna, 2017).

2.7. Interfaz de programación de aplicaciones meteorológicas.

La interfaz de programación de aplicaciones meteorológicas es un acceso universal a datos climatológicos (meteorological APIs) los cuáles pueden ser obtenidos de diferentes fuentes de información como satélites, estaciones meteorológicas inalámbricas, sensores meteorológicos, etc. Su uso se ha vuelto muy popular y de gran utilidad conforme la tecnología ha ido avanzando, logrando así que cualquier persona pueda consultar pronósticos del clima desde su smartphone en cualquier parte del mundo (Mansutti Rosón, 2018).

En la Figura 11 se puede observar el funcionamiento básico de una API, esta consta de varios actores que son (ISARQ, 2018):

1) Datos. - Conjunto de información recolectada por diversos sensores que pueden estar o no especializados en la meteorología. Dichos datos se encuentran almacenados en el servidor o base de datos de alguna empresa, listos para ser consultados externamente.

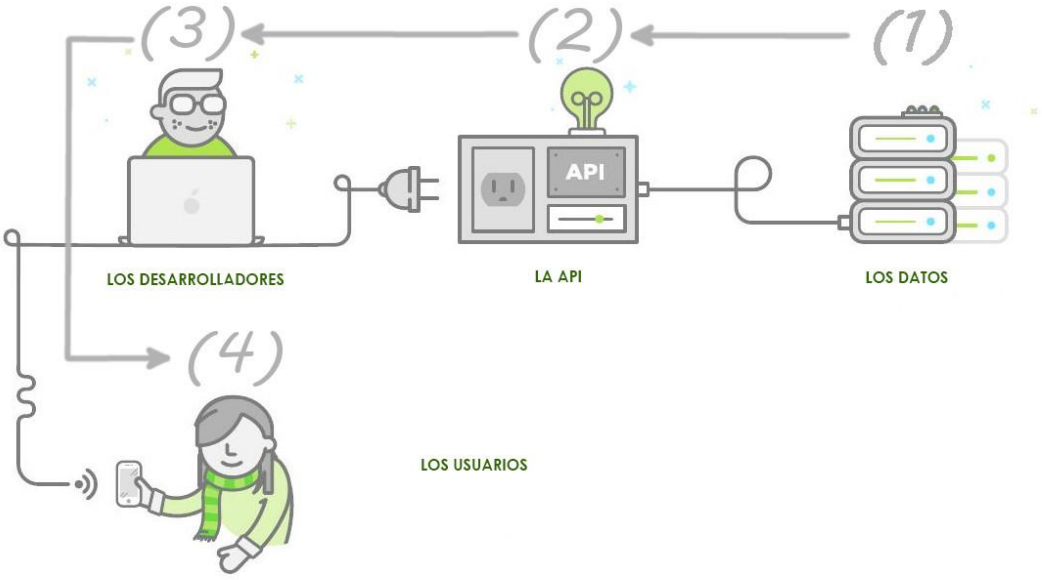
2) API. - Interfaz que proporciona acceso universal a los datos (actor 1) que se necesiten consultar, es decir, este actor actúa como una puerta de entrada a la información requerida por los desarrolladores (actor 3). La API es de gran utilidad para compartir únicamente información concreta que se desee distribuir omitiendo datos sensibles.

3) Desarrolladores. - Son las personas que acceden a la información (actor 1) a través de la API (actor 2) para poder desarrollar sistemas basados en esta, de igual manera permite aprovechar componentes (software) para poder reutilizarlos optimizando así su código fuente.

4) Usuarios. - Son los individuos que acceden a los sistemas de software creados por los desarrolladores (actor 3) para hacer uso de los servicios que les brindan sin ningún problema y desde múltiples dispositivos (smartphones, tabletas, ordenadores, etc.).

Figura 11

Funcionamiento básico de una API



Nota. En esta figura se muestra la interacción de los actores para que una API pueda funcionar. Tomado de (ISARQ, 2018)

Además, cabe recalcar que existen varias APIs meteorológicas diferenciándose cada una de estas según la forma en que obtienen sus datos, a continuación, se detallan las más importantes:

a) **Climacell MicroWeather. -** Interfaz de programación de aplicaciones que recaba información de varios sensores virtuales como vehículos inteligentes, drones o señales inalámbricas ofreciendo así datos meteorológicos mundiales minuto a minuto en tiempo real. Cabe recalcar que sus datos no son muy fiables debido a que existen limitaciones en el hardware ya que este no está enfocado únicamente en el aspecto climatológico es decir no son especializados en una sola área (Gino, 2020).

b) **OpenWeather Map.-** Interfaz de programación de aplicaciones que obtiene información a través de cuarenta mil estaciones meteorológicas inalámbricas distribuidas por todo el mundo, ofreciendo datos climatológicos de alta precisión en tiempo real, así como también informes históricos mundiales lo que beneficia a múltiples áreas industriales (agroindustria, aeronáutica, etc.) debido a que estas empresas requieren llevar un control de las anomalías climatológicas (lluvia, tormentas, tornados, etc.) para la toma de decisiones (Rahmat, 2018).

c) **Weatherbit. -** Interfaz de programación de aplicaciones que adquiere información meteorológica a través de varios sitios web u otras API´s para de esta manera proporcionar datos climatológicos de precisión media. Una de las grandes desventajas es que el manejo de la versión gratuita no brinda información climatológica eficaz y duradera. Razón por la cual es recomendable utilizar la API OpenWeather Map ya que esta permite obtener una consulta más precisa y depurada de la información climatológica (Scott, 2019).

2.8. Arquitectura básica del Software.

Organización de los componentes que se usan en el desarrollo del software, según su interacción y coordinación, los cuales permiten llegar a cumplir con diversas funcionalidades deseadas. La arquitectura es un puente entre los requerimientos y el código (Reynoso, 2004).

Para definir está arquitectura se debe tomar en cuenta las decisiones de diseño junto con los requerimientos que permitan satisfacer las necesidades del cliente. Sus componentes principales o básicos son: base de datos, front-end y back-end (Romero, 2006).

2.8.1. Modelos para el diseño de la arquitectura de software.

Guías que se utilizan para analizar y diseñar la arquitectura con la que un software podrá funcionar. Estos son necesarios para establecer los pasos a seguir con el fin de seleccionar los componentes óptimos que se requieren al desarrollar e implementar un sistema. Así como también plasmar de una manera gráfica su

funcionamiento facilitando su entendimiento y con esto una forma más sencilla de programar o codificar sus funcionalidades (Kruchten, 1995).

Existen varios modelos para el diseño de la arquitectura de software, estos son utilizados de acuerdo a las necesidades del proyecto que se vaya a desarrollar, destacándose dos de mayor relevancia, los cuáles son:

1. **C4. -** Modelo enfocado en representar de manera gráfica todos los componentes que forman parte del software, utilizando contenedores estáticos que relacionan el código fuente y las personas que manipulan el sistema. Presentando una desventaja al momento de utilizar metodologías de desarrollo de software que emplean diagramas UML, ya que estos usan únicamente diagramas jerárquicos (Vivanco, 2019).
2. **4+1. -** Modelo orientado al uso de diagramas UML (Lenguaje de modelado unificado), los cuales permiten la extracción de información a través de la construcción de vistas que describen toda la arquitectura del software, basadas en las funcionalidades recopiladas en la etapa del levantamiento de requerimientos. De acuerdo con (Kruchten, 1995) las vistas que se requieren utilizar son las siguientes:

    - **Vista de escenarios. -** Representa las funcionalidades más significativas del sistema y su interacción entre sí, con la finalidad de observar su comunicación y la forma con la que establecen un enlace con el cliente. Esta vista se basa en el diagrama de casos de uso.
    - **Vista Lógica. -** Representa la interconexión de varios conjuntos de datos agrupados de manera lógica al igual que la forma en la que el sistema intercambia esta información de forma secuencial y ordenada, siendo de gran ayuda para analizar el comportamiento del software cuando es utilizado por el usuario final. Esta vista está basada en los diagramas de clases y de secuencias.
    - **Vista Física. -** Representa la conexión existente de la tecnología que conforma el sistema (frameworks, servidor web, base de datos), visualizando el comportamiento de cada módulo y su relación de acuerdo

a las funciones que este realizará. Esta vista se basa en el diagrama de despliegue.

- **Vista de desarrollo. -** Representa la agrupación de funcionalidades que cumple el sistema (paquetes). Sirve para detallar de manera secuencial la forma en la que el cliente hace uso del sistema. Esta vista se basa en el diagrama de paquetes.

2.8.2. Componentes básicos que conforman una arquitectura

El correcto funcionamiento del software a desarrollarse por parte del personal técnico informático debe estar constituido de diferentes componentes los cuáles forman una arquitectura (Stevens, 2002). Cada uno de estos cumplen una función importante que se detalla en los siguientes subtemas a tratar.

2.8.3. Base de datos.

Continuidad de los datos organizados e interrelacionados, los cuales son recopilados y utilizados por sistemas de información de alguna empresa o negocio (Raffino, 2020). Estos nacieron de la necesidad de almacenar información para preservarla con el tiempo y evitar su deterioro permitiendo así utilizarla cuando se la necesite (Korth, 1993).

Se puede aplicar este componente en diversos campos como la banca, líneas aéreas, universidades, telecomunicaciones, finanzas, ventas, producción, recursos humanos, entre otros. El uso de bases de datos se ha vuelto fundamental en prácticamente todas las empresas gracias a su facilidad de manipulación indirecta por parte de sus usuarios (Silberschatz, 2002).

a. Características básicas de una base de datos.

Existen varias cualidades que dan a conocer la eficiencia de las bases de datos en general. A continuación, se listan y se describen de manera general sus características más importantes (Ullman, 1999):

- **Independencia física. -** Capacidad de realizar modificaciones en sus componentes físicos sin la necesidad de reescribir la aplicación que manipula el cliente.

- **Independencia lógica. -** Habilidad de modificar la forma en que se relaciona la información sin tener que realizar modificación alguna en el software que consulta su información.
- **Redundancia mínima. -** Facultad de no registrar el mismo tipo de información dos veces, optimizando así la capacidad de almacenamiento y la tolerancia a fallos en el software.
- **Respaldo y recuperación. -** Capacidad de resguardar una copia de seguridad de la información almacenada para restablecerla cuando sea necesaria, sin perder propiedad alguna.
- **Integridad de los datos. -** Posibilita la consulta de datos de forma coherente, integra y precisa sin que estos pierdan algún atributo. Esto es esencial ya que en la mayoría de veces se almacena información delicada.
- **Búsquedas complejas optimizadas. -** Mejora en el tiempo de respuesta al momento de consultar información requerida por el usuario, accediendo así a los datos de forma rápida, segura y eficaz.
- **Seguridad de auditoría y acceso. -** Habilidad de control de acceso a datos según el tipo de usuario que los requiera. Si son requeridos por algún ente no autorizado, estos simplemente no se mostrarán.
- **Ingreso por medio de lenguajes de programación. -** Facultad de poder automatizar consultas de información por medio de la codificación en diversos lenguajes de programación.
- **Accesos recurrentes. -** Cualidad para poder mantenerse estable ante varios requerimientos de información por parte del usuario, como pueden ser peticiones recurrentes o acceso de personal al mismo tiempo.

b. Tipos de base de datos.

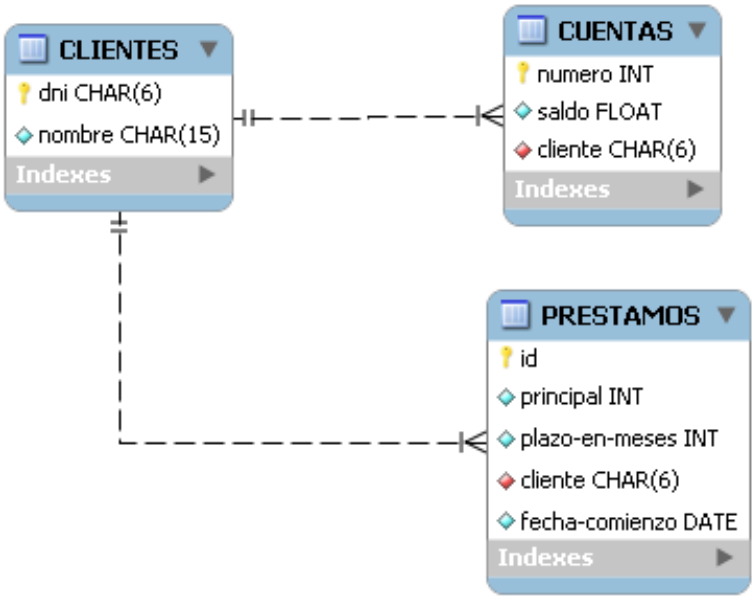
Las bases de datos han sido creadas gracias a las diferentes necesidades de la evolución informática, cada una de ellas tiene su propia manera de estructurarse (Coronel, 2011). Los tipos de bases de datos son A) relacionales y B) no relacionales las cuáles son mencionadas a continuación:

A) Base de datos relacional. - es un depósito de datos con relaciones predefinidas mediante un identificador único de cada tabla. Este identificador permite el acceso a la información para poder editarla, modificarla o eliminarla (Cabello, 2010).

En la Figura 12 se puede observar la relación que existe entre cada tabla (conjunto de información) que se implementa en este tipo de base de datos. La tabla cliente tiene como atributos dni (identificador único) y nombre, esta se relaciona de la forma en que un cliente tiene varias cuentas y préstamos. La tabla cuentas se conforma de número (identificador único), saldo y cliente (relación con la tabla clientes) y la tabla prestamos tiene como datos el id (identificador único), principal, plazo-en-meses, cliente (relación con la tabla clientes) y fecha-comienzo (Sánchez, 2004).

Figura 12

Relación de tablas



Nota. En esta figura se muestra la conexión de las tablas que forman parte de una base de datos relacional. Tomado de (Sicilia, 2008).

B) Base de datos no relacionales. - Son las colecciones de información almacenada en archivos, estas están diseñadas para modelos de datos con una estructura que no guarda relación entre sí. Son reconocidas por su funcionalidad y su rendimiento a gran escala (Lafuente, 2018).

En la Figura 13 se puede observar que en este tipo de base de datos no existe relación alguna, se conforma únicamente por herencia de atributos. La colección cultivo está compuesta por el _id (identificador único), fecha, estado y hectárea mientras que la colección predecir meses tiene como atributos el _id (identificador único), tempMax, tempMin, actualizado, fecha_act y cultivo (herencia con la colección de cultivo) (Valbuena, 2014).

Figura 13

Base de datos no relacional

Nota. En esta figura se muestra cómo se confirman las colecciones de una base de datos no relacionada. Tomado de (Valbuena, 2014)

En la actualidad los volúmenes de datos en las aplicaciones son mayores a los que existían hace apenas algunos años, por ello estas bases de datos son óptimas para el manejo de gran cantidad de información gracias a su funcionamiento a bajas latencias. Una de las más sobresalientes es Mongo DB ya que es considerada de gran

compatibilidad para entornos web y móviles debido a que estos requieren bases de datos adaptables, escalables y altamente funcionales (Rendón, 2019).

2.8.4. Mongo DB

Posee colecciones de documentos del tipo JSON (una colección de documentos se asemeja a una tabla en SQL). Esta permite cambiar ágilmente las relaciones entre la información almacenada cuando las aplicaciones evolucionan. Ofrece flexibilidad operativa y fiabilidad haciendo posible ejecutarla desde proyectos básicos a sumamente complejos (Sistel, 2017).

De acuerdo con (Mongodb, 2020) las ventajas que presenta Mongo DB frente a las bases de datos relacionales son las siguientes:

- **Velocidad. -** Realiza muchas operaciones en periodos de tiempo mucho más cortos dando un buen rendimiento en escritura y lectura.
- **Volumen. -** Gracias a su amplia capacidad de memoria, es posible administrar grandes cantidades de volúmenes de datos sin problema.
- **Variabilidad. -** Crea campos dinámicamente en el registro, en lugar de almacenar diferentes sectores en cada documento, debido a que son flexibles en términos en el manejo de información.

2.8.5. Back-end.

Parte del sistema que contiene las operaciones lógicas necesarias para solucionar todos los problemas planteados por el cliente. Este genera una conexión entre el front-end y la base de datos para así consultar datos de forma rápida, eficaz y segura (Gálvez Neculman, 2019).

Según (Reyes, 2015) el back-end está conformado por tres pilares fundamentales:

1. **El servidor. -** programa informático que recibe las peticiones generadas por el cliente para así ser procesadas y dar solución a sus requerimientos. Esta tiene una conexión directa con la base de datos para así consultar de manera más eficaz la información requerida por alguna petición.

2. **La aplicación. -** Rutina que se ejecuta en el servidor mediante el protocolo HTTP. Esta contiene la lógica de repuesta que se pone en marcha por medio de la red. Esta hace posible el intercambio de datos con el front-end.
3. **La base de datos. -** Conjunto de datos organizados de tal forma que puedan ser de fácil acceso, gestionables, o actualizables en cualquier momento. En esta se encuentra toda la información necesaria para que el software pueda funcionar.

Cabe destacar que para el caso de MongoDB (base de datos no relacional) la información despachada por el servidor hacia el cliente debe tener un protocolo de datos, para este caso es el formato JSON el cuál es un estándar de datos liviano que ayuda al intercambio de datos más rápida y eficaz. Además, cabe destacar que se basa en el lenguaje de programación JavaScript (Eich, 2020).

2.8.6. Front-end.

Componente de la arquitectura del sistema en desarrollo, responsable de interactuar directamente con el usuario, teniendo como aspecto importante la usabilidad (Murcia Pérez, 2013). Este es accedido por medio de interfaz gráfica o comandos. También se la nombre como capa de presentación (Reyes, 2015).

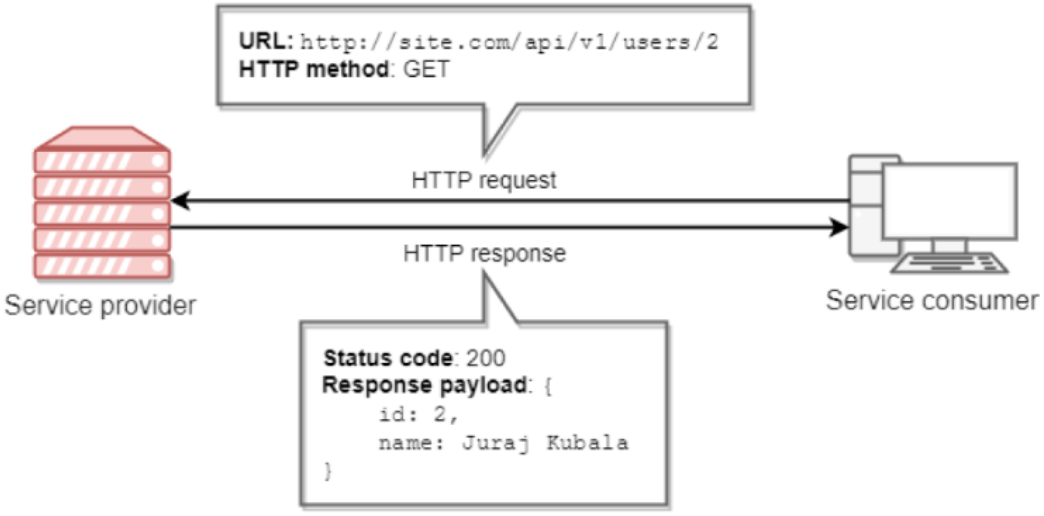
La interacción entre esta capa y el usuario se denomina peticiones. Estas son enviadas por medio de la URL del navegador web o gestos en dispositivos móviles o de escritorio. Dichas peticiones se receptan por el servidor para cumplir con el pedido deseado, el protocolo usado para está intercomunicación es HTTP (Schacherbauer, 2001).

2.8.7. Comunicación entre componentes.

En la Figura 14 se puede observar cómo se interconecta cada componente que forma parte del software. El front-end (Service consumer) realiza una petición (HTTP request) al back-end (Service provider) para la consulta de datos almacenados sobre un usuario en particular, este a su vez emite una respuesta (HTTP response) mostrando los datos requeridos (Valencia Altamirano, 2018).

Figura 14

Comunicación entre el front-end y el back-end



Nota. En esta figura se muestra la comunicación entre los módulos que forman parte del proyecto. Tomado de (Valencia Altamirano, 2018).

2.9. Internet de las cosas (IoT).

Conjunto de dispositivos de baja capacidad de procesamiento relacionados entre sí que cumplen con una única tarea. Estos están conectados a internet para enviar o recibir datos de interés para el funcionamiento de un sistema (García G. A., 2018).

Generalmente estos artefactos IoT se utilizan para automatizar tareas en un hogar o empresa como el encendido de luces de forma automática, video vigilancia, impresión 3D, climatización, entre otros. Esta arquitectura se conforma por tres elementos. De acuerdo a (Alcaraz, 2014) estos son:

1. **Los dispositivos. -** Elementos comunes que se conoce en el día a día como, relojes, autos, luces, cafeteras, televisiones, etc. Estos deben poseer una característica primordial como es la comunicación entre dispositivos, la cual puede ser por medio de sensores, chips, antenas o internet.

2. **Sistema de control. -** Esencial para procesar los datos capturados por los dispositivos, enviándolos al software que requiere está información y a su vez gestiona la incorporación de nuevas conexiones.
3. **La red. -** Es la vía de comunicación entre los dispositivos y el sistema de control. Esta puede ser implementada por medio de tecnologías como datos móviles, Wi-Fi o Bluetooth.

2.10. Metodología.

De acuerdo con (Maida, 2015) es una conjunción de métodos y técnicas que abordan de una forma abierta e igualitaria todas las actividades del desarrollo de un proyecto. Es un proceso de software completamente detallado.

Todas las metodologías están basadas en combinaciones de los modelos generales de procesos, estas definen roles, actividades y artefactos, integrando recomendaciones técnicas y prácticas (Marquès, 1995). Las metodologías se dividen en dos clases que son tradicional y ágil, estas son detalladas en los siguientes subtemas.

2.10.1. Metodología tradicional.

Método organizado de creación de sistemas informáticos que define estructuras secuenciales al momento de la elaboración del proyecto. Originalmente creado para poner fin al caos del desarrollo de software. Cabe destacar que dicha metodología es muy rígida siendo su principal desventaja no poder modificar en ningún momento un nuevo requerimiento o imprevisto dejándolo de lado (Letelier, 2006).

Su meta principal es cumplir con todas las tareas planificadas en los períodos de tiempo establecidos, requiriéndose analizar todos los objetivos necesarios para la a elaboración del proyecto antes de que este se empiece a ejecutar (Figueroa, 2008).

Su desventaja es ser compleja y nada flexible al momento de implementarse en un proyecto de software, haciendo que su costo sea realmente alto por las siguientes razones: 1) No proponer soluciones viables a problemas que surgen a lo largo del desarrollo y 2) Falta de detalle en la documentación que especifica la forma de implementar esta metodología (Letelier, 2006).

2.10.2. Metodología Ágil.

La creación de este método nació debido a la necesidad de solucionar problemas existentes en el desarrollo de software, basándose en dos principios: 1) Adaptación rápida y 2) el cambio ágil de decisiones. (Figueroa, 2008).

Esta metodología entrega un sistema de software comprendido de varias versiones utilizables, denominadas incrementos. Dichos incrementos no consideran prioritariamente la documentación, planificación y control (Mendes Calo, 2010).

Además, cabe destacar las principales ideas que se consideran y se recomiendan al momento del desarrollo de un proyecto de software utilizando esta metodología son (Maida, 2015):

- Interacción frecuente entre el software y el cliente.
- Publicar un texto de software que funcione y no escribir algo que no va a funcionar en el futuro.
- Dialogo continuo con el cliente el cual debe ser constante, para entender todas las funcionalidades que tendrá el software.
- Mayor enfoque a modificaciones o imprevistos que surgen a lo largo del desarrollo.

Diferenciándose principalmente este tipo de metodologías con las metodologías tradicionales en los siguientes aspectos (Mendes Calo, 2010): a) Solución eficaz a errores que surgen en el desarrollo del software y b) Optimización en los costos del proyecto.

Existen varias metodologías ágiles, cada una de estas con sus correspondientes ventajas según las necesidades que se requieran. Detallándose a continuación las 2 más representativas o importantes:

a. Metodología de Programación Extrema (XP).

Metodología que tiene como objetivo el cumplimiento de tareas en un tiempo especificado, así como la colaboración entre el equipo, el aprendizaje de los

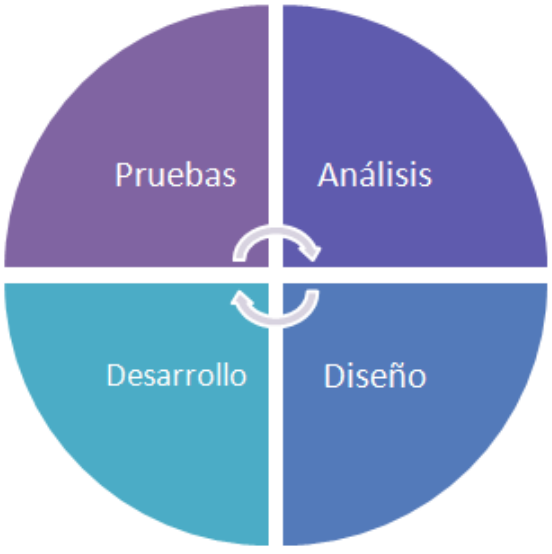
programadores y la comunicación de estos con el cliente, generando así un ambiente de trabajo ideal (Mendes Calo, 2010).

Según (Olarte, 2018) está es una metodología en la cual se debe aplicar buenas prácticas de desarrollo software y reglas, siendo muy útil en donde los requisitos funcionales no son claros o son imprecisos.

En la Figura 15 se observa cada una de la fase por las que pasa un proyecto de desarrollo de software basado en la metodología XP. Iniciando por la etapa de análisis donde se extraen las historias de usuario seguido de la fase de diseño en la cual se analiza la arquitectura que tendrá el software, así como las tecnologías que lo conformaran, luego se procede por la fase de desarrollo, donde se realiza la codificación del sistema en parejas y como última etapa, se ejecutan las pruebas. Evaluándose si el software cumple con los estándares de calidad y si es óptimo para el cliente. Cada uno de estas fases puede ser repetida en caso de surgir algún inconveniente durante el avance del proyecto (Letelier, 2006).

Figura 15

Modelo XP



Nota. En esta figura se muestra las 4 fases que conforman el modelo XP. Tomado de (Grau, 2016).

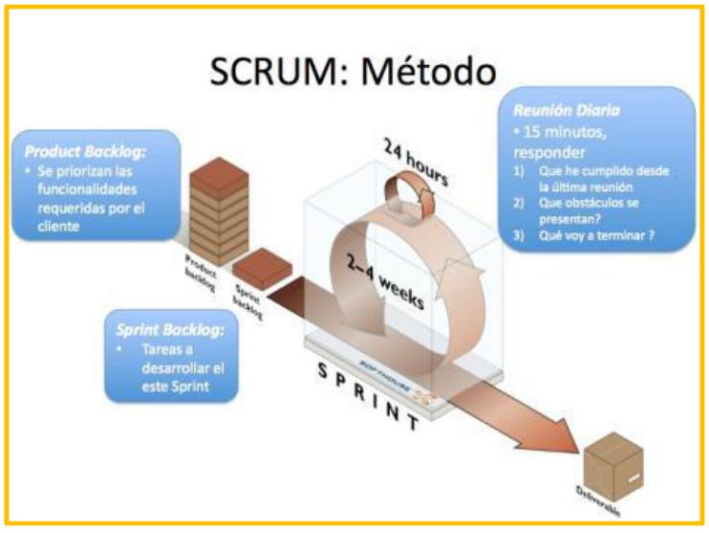
b. Metodología SCRUM.

Utilizada en la implementación de proyectos de media y alta complejidad, tiene como objetivo la entrega de versiones funcionales del software en cortos períodos de tiempo. Está metodología se basa en tres pilares fundamentales, que son: 1) la transparencia, 2) la adaptación y 3) la inspección del sistema. Permitiendo validaciones continuas por parte del cliente dentro de su empresa para una posterior corrección de la empresa desarrolladora (Schwaber, 2013).

En la Figura 16 se observa todos los procesos por los que pasa el ciclo de vida de esta metodología, comenzando por detallar las necesidades del cliente (Product backlog), estás se las divide en varios conjuntos de tareas que se realizarán en un periodo de 2 a 4 semanas (Sprint Backlog), teniendo en cuenta que cada tarea descrita en el sprint se la ejecutará en un tiempo máximo de 24 horas. Al finalizar cada sprint se efectuarán reuniones diarias de máximo 15 minutos, donde se tratan 3 puntos que son: 1) ¿Que he cumplido desde la última reunión?, ¿Que obstáculos se presentan?, ¿Qué voy a terminar? Para concluir se entrega una parte funcional del proyecto desarrollado (Incremento). Este ciclo de vida se repite por cada sprint que se haya analizado, entregando así una versión del producto hasta completar cada uno de estos (Bahit, 2012).

Figura 16

Ciclo de vida de SCRUM



Nota. En esta figura se muestra los pasos para desarrollar un sistema de software según la metodología SCRUM. Tomado de (Presma, 2012)

Los sprint se los realiza de forma prioritaria, comenzando con el que tiene mayor relevancia para el proyecto, gracias a esto se reducen los errores que puedan aparecer en el futuro. Se puede decir que la metodología SCRUM es aplicable para proyectos con un entorno realmente complejo, ya que sus aspectos más importantes son la flexibilidad, productividad y complejidad (Schwaber, 2013)

SCRUM tiene varios artefactos que permiten que el progreso del proyecto sea sencillo de entender tanto para los desarrolladores como para el cliente. Cada uno de estos instrumentos tienen como propósito especificar la tarea que cumple cada persona, su forma de avance y el tiempo en que la deberá realizar (Trigás Gallego, 2013). A continuación, se detallan cada uno de estos artefactos.

1) Artefacto 1 (Equipo del proyecto).

Artefacto utilizado para determinar el equipo de trabajo, facilitando el registro de datos del personal involucrado en el proyecto. Está basado en una plantilla que contiene los siguientes campos: el nombre del individuo, rol que ejercerá (desarrollador, product owner, SCRUM master), su profesión y las responsabilidades o las tareas asignadas (Colla, 2012), dichos parámetros se los puede observar en la Tabla 2.

Tabla 2

Plantilla Equipo SCRUM

EQUIPO SCRUM
Nombre:
Rol:
Categoría profesional:
Responsabilidades:

Nota. En esta tabla se muestra las personas que conforman el equipo de desarrollo según la metodología SCRUM. Tomado de (Colla, 2012)

2) Artefacto 2 (Estructura de Historia de usuario).

Artefacto denominado historia de usuario que se basan en el documento de especificación funcional, y en los casos de uso. Esta información es descrita por el cliente a manera de explicaciones cortas de lo que el sistema deberá efectuar al ser

concluido (Suaza, 2015). Dichas explicaciones deben estar bien detalladas para que los desarrolladores puedan construir más fácilmente el software evitando inconsistencias o fallos (Joskowicz, 2008).

La estructura de este artefacto debe ajustarse a una plantilla que sirve para especificar cada tarea que deberá cumplirse, este documento se conforma por los siguientes campos: identificador, usuario, nombre de la historia, prioridad en el negocio, riesgo en desarrollo, puntos de estimación, interacción asignada, programador responsable, descripción y validación, como se muestra en la Tabla 3.

Tabla 3

Plantilla historia de usuario

Historia del usuario	
Identificador:	Usuario:
Nombre de la historia:	
Prioridad en el negocio:	Riesgo de desarrollo
Puntos estimación:	Iteración Asignada
Programador responsable:	
Descripción:	
Validación:	

Nota. En esta tabla se muestra cada historia de usuario que conforma el presente proyecto (Suaza, 2015)

A continuación se detallan de una manera general todos los campos de este artefacto mencionados anteriormente (Izaurralde, 2013):

Identificador. - Forma de nombrar una historia de usuario, pudiendo especificarse utilizando palabras o letras con las que se las pueda diferenciar del resto, con el fin de hacer más fácil su búsqueda o referenciarla.

Usuario. - Cliente final que desea implementar la funcionalidad descrita en la tarea, dentro del sistema.

Nombre de la Historia. - Nombre de la tarea que se llevará acabo.

Prioridad en el negocio. - Importancia que tiene el desarrollo de una funcionalidad en el sistema de acuerdo a lo que el cliente considera lo más prioritario es decir lo antes posible. Considerándose estas prioridades como: alta, media o baja.

Riesgos en desarrollo. - Posibilidad de que pueda existir un contratiempo al momento que un desarrollador codifique una funcionalidad dentro del proyecto, causando algún fallo o alterando los tiempos establecidos lo cual conllevaría un aumento de costos e incluso una suspensión definitiva del desarrollo del sistema. Existiendo denominaciones para este riesgo como: alta, media o baja.

Puntos de estimación. - Tiempo estimado en el que se debe codificar la funcionalidad de un proyecto informático. Dicha estimación es muy complicada ya que el equipo de desarrollo debe analizar múltiples factores, así como contemplar la experiencia adquirida en proyectos similares realizados anteriormente.

Iteración asignada. - Orden de desarrollo que se le asigna a una tarea o tareas (sprint) las cuales están contenidas en una historia de usuario. Como por ejemplo si en una historia de usuario se requieren realizar las siguientes tareas de desarrollo creación de login, CRUD (*Create, Read, Update and Delete*) de usuarios y CRUD (*Create, Read, Update and Delete*) de productos. A estas se le asigna un ordenamiento que se lo puede representar como: 1, 2, 3, etc.

Programador responsable. - Nombre del técnico programador responsable del desarrollo de las funcionalidades que se encuentran especificadas dentro de la historia de usuario.

Descripción. - Breve explicación de las tareas que deberán desarrollarse, así como también los datos necesarios para ejecutar dicha actividad.

Validación. - Indicación que muestra si las funcionalidades implementadas han pasado evaluaciones que indican que están correctamente desarrolladas. Si está tarea cumple con todos los estándares se indicará un estado de aceptada, caso contrario se especificará con el estado rechazada.

3) Artefacto 3 (Product Backlog).

Artefacto que permite realizar una enumeración ordenada y detallada de todas las historias de usuario que contiene el producto software. Dicho artefacto es la única fuente de requisitos para cualquier cambio a realizarse en el software. Únicamente el dueño del producto (product owner) puede realizar las modificaciones que sean necesarias, ya que es el responsable del contenido, disponibilidad y disposición de esta lista (Schwaber, 2013).

La plantilla utilizada para registrar la estructura de la lista del producto (product backlog) contiene los siguientes parámetros: inicio, fin, jornada, historias pendientes, horas pendientes y un listado de historia de usuario, donde se especifican los siguientes campos: id, estimación, estado y número de sprint asignado (Izaurralde, 2013) como se puede observar en la Tabla 4.

Tabla 4

Plantilla de product backlog

		Proyecto		
		Inicio	Fin	Jornada
		Historias pendientes		
		Horas pendientes		
ID	Historia de usuario	Estimación	Estado	Sprint #

Nota. En esta tabla se muestra el análisis del tiempo que tomara cada tarea según su prioridad. Tomado de (Izaurralde, 2013)

A continuación, se mencionan los parámetros de la Tabla 4 (Izaurralde, 2013):

Inicio. - Fecha de inicio del proyecto.

Fin: Fecha final del proyecto.

Jornada. - Promedio de horas diarias en las que se va a desarrollar cada historia de usuario.

Historia de usuario. - Nombre de la historia de usuario analizado según las tareas que se desarrollaran.

Estimación. - Número de horas planificadas que tomará el desarrollo completo de la historia de usuario según su ID.

Estado. - Se muestra el avance de cada historia de usuario de la siguiente manera: pendiente, en curso, finalizado.

Sprint. - El número de iteraciones en la cual se va a desarrollar la historia de usuario. El sprint no debe durar más de 20 días (4 semanas).

4) Artefacto 4 (Estructura de sprint backlog).

Artefacto que reúne todos los elementos seleccionados para el Sprint, adicionando un campo para registrar el avance del proyecto junto con un indicador de cumplimiento de objetivos que fueron analizados en el artefacto 2 (Historias de usuario) (Schwaber, 2013).

La plantilla necesaria para registrar el sprint backlog está conformada de varios parámetros, siendo estos: inicio, fin, jornada, id, categoría, tareas, responsables del sprint, horas estimadas y estado de la tarea (Dimes, 2015). Esto se lo puede visualizar en la Tabla 5.

Tabla 5

Plantilla de sprint backlog

Proyecto		
Inicio	Fin	Jornada
Tareas pendientes		
Horas pendientes		

ID	Categoría	Tareas	Responsable	Estimado (horas)	estado	Historia de usuario

Nota. En esta tabla se muestra todas las tareas a realizar junto con su planificación. Tomado de (Dimes, 2015)

A continuación, se describe brevemente cada uno de los parámetros del sprint backlog:

Inicio: Fecha de inicio del proyecto.

Fin: Fecha final del proyecto.

Jornada: Promedio de horas en las que se cumplió las tareas asignadas.

ID. - Código que se asigna a la tarea del sprint seleccionado.

Categoría. - Describe el tipo de tarea a cumplir considerando actividades de diseño o programación.

Tarea. - Detalla brevemente las funcionalidades que el equipo de desarrollo realizará, describiendo las mismas de manera concreta o general. Esto se efectúa para que los desarrolladores involucrados puedan comprender cada una de estas de mejor manera.

Responsable. - Es la persona encargada del desarrollo de la historia de usuario.

Estimado. - Número de horas en la que se estima que se terminará la tarea seleccionada.

Estado. - Muestra el avance de cada historia de usuario de la siguiente manera: pendiente, en curso, finalizado.

2.10.3. Diferencias entre XP y SCRUM.

Según (Colla, 2012) una parte importante para diferenciar el uso de la metodología XP o SCRUM es a través de un análisis enfocado en el tipo de proyecto a desarrollar, equipo (personal involucrado), diseño o arquitectura que se requiera implementar. Lo cual permitirá escoger entre estas dos metodologías, sacando el mayor provecho al momento de ser utilizadas en el desarrollo del proyecto propuesto en esta investigación. A continuación, se realiza una comparativa de cada uno de los parámetros de gestión más importantes que utiliza cada metodología. Permitiendo diferenciar su aplicación en diferentes ámbitos como se puede observar en la Tabla 6.

Tabla 6

Diferencia del uso de los parámetros de gestión utilizados en proyectos XP y SCRUM.

Parámetros de gestión de proyecto	SCRUM	XP
Especificación de tareas	Las tareas a cumplir se establecen en un tiempo de entrega de 2 a 4 semanas, estas se conocen como sprint	Cada tarea a cumplir se entrega en un tiempo establecido de 1 a 3 semanas.
Finalización de tareas	Al finalizar un sprint este es evaluado por el dueño del producto. Si dicho sprint es de su conformidad no se lo vuelve a tocar en ningún otro momento.	Las tareas terminadas se las entrega al cliente. Estás pueden ser modificadas en cualquier momento del desarrollo del proyecto.
Gestión del personal de desarrollo	Cada persona involucrada en el proyecto trabaja de forma individual	El personal de desarrollo realiza programación en parejas.
Cambios que pueden realizar los miembros del equipo.	Cualquier miembro del equipo puede cambiar el orden de elaboración de las tareas.	El personal del proyecto sigue estrictamente la prioridad de ejecución de tareas indicadas por el cliente.
Tipo de enfoque	Está más enfocada en la administración de proyecto.	Esta más centrada en la creación del sistema y su programación.

Nota. En esta tabla se muestra una comparación entre la metodología XP y SCRUM

En conclusión, después de haber analizado estás metodologías, se puede decir que SCRUM es la más óptima para aplicar a proyectos que cuentan con un limitado personal de desarrollo. De igual forma es necesario destacar su rápida respuesta a decisiones que surgen a lo largo del ciclo de vida del desarrollo del software, su documentación fácil de entender gracias al uso de artefactos y su tipo de enfoque.

2.10.4. Aplicación de SCRUM como metodología en el desarrollo de aplicaciones móviles y web

SCRUM como método de trabajo ágil es ideal para el desarrollo de aplicaciones tanto móviles como web, ya está diseñado para realizar cualquier tipo de sistema software permitiendo la elaboración de trabajos de mediana y gran complejidad posibilitando escalarlos o dividirlos en módulos presentables para el cliente, hasta concluir completamente con su desarrollo (S.L., 2015).

Además, el objetivo principal de dicho marco de trabajo (SCRUM) es asignar tareas que permitan la organización del personal y de sus actividades de una forma ordenada (Calvo, 2019). Existen algunos trabajos de investigación que aplican está metodología (SCRUM) tanto para el desarrollo móvil como para el web a continuación se indican los más relevantes de acuerdo a su título:

a) <<Diseño de aplicación para buscar ubicación de Farmacias de turno utilizando la Metodología SCRUM desarrollada en la Escuela Superior Politécnica Del Litoral>> (Moreno Hermenejildo, 2017). Este sistema está enfocado al desarrollo móvil y fue utilizado para informar la existencia de farmacias cercanas a la ubicación del cliente también indica los servicios que posee la farmacia haciendo posible hacer un pedido a domicilio.

b) <<Aplicación de SCRUM para crear sistema de estandarización de planes de trabajo en sistema web 2.0 para el CNE a través de la reutilización de software, utilizando herramientas de software libre de la Escuela Politécnica Nacional>> (Méndez Camacho, 2016).- Este software se orientó al desarrollo web y fue utilizado para la gestión de procesos electorales y publicación del plan de trabajo por parte de los

candidatos y la valoración de planes de trabajo individuales en escala de uno a cinco por parte de la ciudadanía.

c) <<Método ágil SCRUM, aplicado a la implantación de un sistema informático para el proceso de recolección masiva de información con tecnología móvil de la Escuela Politécnica del Ejercito>> (Manuel, 2012). - Esta aplicación móvil ayuda a la recolección de datos en procesos operativos para la empresa Electica de Quito siendo el factor fundamental enviar el consumo mensual mediante la aplicación para poder calcular su planilla.

2.11. Inteligencia Artificial.

Tratando de comprender la forma en que las personas perciben, entienden, manipulan o predicen acciones dentro del entorno que las rodea, el ser humano ha intentado simular todos estos atributos creando entes tecnológicos que puedan desenvolverse por sí mismos, dando así origen a la inteligencia artificial (Norvig, 2004).

Además, se considera una de las ramas de la Informática, con fuertes raíces en otras áreas como la lógica y las ciencias cognitivas, basándose en la realización de tareas computarizadas recurrentes de gran volumen, permitiendo lograr resultados de gran fiabilidad ya que estos procesos conllevan de mucho esfuerzo para el ser humano. Para este tipo de automatización, las personas son parte esencial ya que estas realizan las configuraciones necesarias para el sistema pueda funcionar (Goonight, 2018).

2.11.1. Aprendizaje de máquina (Machine Learning).

Proceso de aprendizaje automático que se realiza a través de la búsqueda de patrones dentro de los datos los cuáles son suministrados por el ser humano, permitiendo así adaptar funcionalidades, clasificar o realizar un pronóstico de información según su necesidad o requerimiento (Aluja, 2001).

En la actualidad cabe destacar que la obtención de información que requieren los algoritmos de aprendizaje máquina (machine learning) son obtenidos de distintas maneras siendo una de estas a través del uso de API’s (Interfaz de Programación de Aplicaciones). Cabe destacar que estos datos son muy importantes para realizar el entrenamiento y pruebas correspondientes. Un ejemplo práctico del uso de dichas API’s

son los factores climatológicos (temperatura, humedad, luminosidad) a través del uso de estaciones meteorológicas inalámbricas (Rasthofer, 2014).

Este campo de la inteligencia artificial se compone de varios tipos de aprendizaje. Es importante conocerlos ya que cuentan con diferentes características distintivas, utilizándose cada uno de estos de acuerdo al ámbito del problema que se quiera solucionar. A continuación, se detalla cada uno de ellos:

a) Aprendizaje supervisado.

Según (SOTO, 2011) este tipo de aprendizaje permite realizar un análisis de datos abstractos, esto quiere decir que proporciona una respuesta de manera simple a partir de información poco clara, incompleta o imprecisa, tratando de imitar una forma de decisión que se asemeja a la humana.

El funcionamiento de este tipo de algoritmo se basa en enseñar a una computadora a pensar a través del ejemplo. De tal manera que este tipo de aprendizaje requiere como entrada un conjunto de datos históricos, así como la información de salida que requiere el usuario obtener a través del uso de los algoritmos disponibles en este tipo de aprendizaje (Calvo, 2019).

El aprendizaje supervisado es utilizado para efectuar predicciones a futuro basándose en comportamientos analizados en información histórica, haciendo uso de modelos estadísticos. Usualmente estos son empleados para estudios poblacionales climatológicos o medio ambientales (Cambronero, 2006). A demás de ayudar ampliamente en la agricultura como es el caso de la aplicación optima de nutrientes a las plantas, control de suministro de agua, o monitoreo de temperaturas para el control de fechas de cosecha, permitiendo así aprovechar todos los beneficios nutricionales y medicinales de cada fruto (JiménezTovar, 2019).

b) Aprendizaje no supervisado.

Según (SOTO, 2011) es una técnica automática de aprendizaje la cual busca en los datos de entrada que no están etiquetados (nombrados o clasificados) patrones

de comportamiento para que la computadora aprenda a tomar decisiones e identificar procesos sin la intervención de un humano.

La manera en que funciona los algoritmos de aprendizaje no supervisado es a través de la interpretación de grandes grupos de datos, para de esta manera clasificar dicha información y así poder conocer su significado. Cabe destacar que entre mayor sea la cantidad de información suministrada a dichos algoritmos mejores resultados se obtendrán (Calvo, 2019).

Este aprendizaje es de mucha utilidad en campos donde se requiera obtener un conjunto de información relevante para la evaluación de decisiones, como puede ser en marketing para obtener grupos de clientes que tengan los mismos gustos o necesidades, biología para clasificar plantas según sus características, empresas de seguros para la identificación de fraudes o estudios geológicos, para determinar lugares peligrosos según la cantidad y tipos de terremotos, entre otros (Aprendeia, 2019).

c) Aprendizaje profundo.

Aprendizaje orientado en emular la forma de aprendizaje del ser humano para conseguir algún tipo de conocimiento específico. A través del uso de varias capas de redes neuronales artificiales, siendo su utilización más reconocida en tareas como visión artificial computacional y reconocimiento de voz o el procesamiento del lenguaje natural (Banafa, 2016).

Además, este campo o área del aprendizaje máquina (machine learning) ha sido muy utilizado en los últimos años, haciendo que grandes empresas lo utilicen para mejorar sus soluciones informáticas, como es el caso de Google, optimizando procesos como el reconocimiento de voz (Google Assistant).

Otra empresa importante que hace uso de este tipo de aprendizaje es Netflix permitiendo mejorar su sistema de recomendación de contenido multimedia (Vázquez, 2019). También hay que mencionar su uso e implementación en juegos de estrategia siendo su principal exponente la empresa DeepMind, creadora del algoritmo llamado AlphaGo el cual se basa en el uso y aprendizaje de las reglas de estrategia chinas utilizadas en el juego tradicional chino denominado GO (Knight, 2017).

2.11.2. Modelos para el aprendizaje de máquina.

Existen modelos de distintos tipos que son implementados de acuerdo a su funcionalidad. Estos algoritmos son entrenados para reconocer diversas clases de patrones de datos los cuáles se requieren clasificar o predecir. El entrenamiento y la prueba de estos modelos necesita de información para su correcta validación, existiendo distintos tipos de datos que se pueden procesar entre los cuales tenemos: imágenes, audio, datos poblacionales, entre otros (Microsoft, 2019).

Los modelos para el aprendizaje automático (machine learning) se conforman de varios subcampos, los cuales se determinan según la clase de tarea o funcionalidad a cumplir. Todos estos se listan y detallan brevemente a continuación:

a) De clasificación.

Modelos en los cuales su objetivo es clasificar un conjunto de datos según el tipo de problema. Su uso está mayormente extendido en el análisis de textos donde se evalúan palabras basándose en un criterio determinado, para de esta manera categorizarlas en dos tipos: 1) positivas o 2) negativas. Un ejemplo de esto es el registro de correo no deseado (Estévez, 2016).

b) De clustering.

Modelos que cuentan con la particularidad de no conocer las categorías de clasificación de información, ya que el trabajo de este es el de encontrar automáticamente semejanzas entre cada dato y agruparlos según sus rasgos. Uno de sus usos más populares es en la medicina ya que se utilizan para catalogar grupos de pacientes según los síntomas que presentan para de esta manera identificar posibles enfermedades que estos padezcan (Agenjo, 2017).

c) De regresión (predictivos).

Modelos utilizados para la estimación de datos, en este se emplean valores continuos y una selección de características específicas para poder realizar algún tipo de predicción. Para que este tipo de algoritmo pueda funcionar se emplean fórmulas matemáticas (Rodríguez D. , 2018). Los modelos de regresión más representativos son:

1) Árboles de regresión. - Algoritmo que trabaja con variables de respuesta continuas, este realiza una comparación de datos por cada nivel implementado, logrando así una predicción específica. Un ejemplo del uso de este algoritmo es la determinación del sueldo que deberá ganar un empleado según su experiencia y número de tareas cumplidas al año (Rodrigo, 2017).

2) Series de tiempo. - Algoritmo basado en listas de fechas que se asocian a un valor, estas series proveen un modo estructurado para representar datos, visualizando así una curva que evoluciona según el pasar del tiempo. De igual manera este modelo extiende los valores históricos hacia el futuro, para establecer mediciones donde no están disponibles. Las series de tiempo son utilizadas para conocer la capacidad de producción que se tendrá a futuro, desarrollo de la población, pronóstico del clima, entre otros (Vermorel, 2012).

3) Regresión Lineal. - Algoritmo basado en árboles de regresión, es utilizado para realizar el cálculo de una relación lineal entre una variable independiente y una dependiente para de esta manera llegar a hacer una predicción. Es de gran utilidad para evaluar tendencias en fabricación, ventas o para el desarrollo de modelos de minería con datos complejos (Microsoft, 2019).

4) Regresión Logística. - Algoritmo de clasificación utilizado para predecir una salida binaria entre 0 y 1, es decir, sirve para identificar la probabilidad de un evento en función de que un suceso ocurra o no. Por ejemplo, la clasificación de que una persona sea hombre (0) o mujer (1) en función de la forma y tamaño de su mandíbula (Rodrigo, 2017).

5) Modelo Autorregresivo (AR). - Algoritmos que permiten realizar pronósticos mediante el uso de datos históricos a través del uso de la estadística, utilizándolos en distintos propósitos (agricultura, clima, negocios, etc.) Dichos al0goritmos son muy precisos puesto que analiza las semejanzas existentes de acuerdo al comportamiento de sus datos. Permitiendo optimizar su predicción lo cual hace que este algoritmo sea muy versátil y de gran estabilidad (Buto, 2018).

Además, cabe destacar que los modelos autorregresivos están conformados de algoritmos que dependen de la necesidad o propósito requerido para poder escoger uno de estos. Dichos algoritmos son:

a) **Modelo Vectorial Autorregresivo (VAR). -** Modelo basado en regresiones múltiples no relacionadas (Espasa, 2015), está más enfocado al ámbito económico, es así que se lo usa como por ejemplo para pronosticar el endeudamiento de un país, pagos futuros de mano de obra, entre otros (Deivy Yosip, 2013).
b) **Modelo Autorregresivo de Media Móvil (ARIMA). -** Modelo univariante (de una sola característica) que realiza predicciones tan precisas que ningún otro modelo de su clase puede ofrecer (Maté, 2012) además esté se enfoca en una determinada área geográfica. Se utiliza para conocer el avance de campañas publicitarias, variables meteorológicas, cambios de precios relativos a bienes sustitutivos, entre otros (Espasa, 2015).

a. Modelo Autorregresivo Integrado de Media Móvil (ARIMA)

Modelo estadístico que se basa en el uso de regresiones lineales que permiten pronosticar fenómenos climatológicos futuros a traves del uso de datos historicos que contienen patrones de comportamiento ocultos. Eliminando tendencias disparejas de dichos patrones (similitud) obteniendo así modelos estacionales los cuales son estables a lo largo del tiempo por lo tanto pueden ser predecibles de mejor manera (De Arce, 2003).

La carácterística de estacionalidad de este modelo lo hace óptimo para la predicción de datos climatológicos utilizando como entradas informacion previa (temperatura) años en el pasado, comparando y encontrando patrones de comportamiento que agudizarán la presición del pronóstico deseado (Zhang, 2003).

Cabe destacar que un aspecto de suma importania de este modelo es la ayuda que brinda a la agroindustria en su proceso de producción y cultivo de plantas, ya que gracias a este algoritmo se puede calcular una fecha ideal de cosecha para de esta manera obtener un producto de gran calidad con mayores beneficios (Luis Francisco Laurente Blanco, 2019).

En la Ecuación 2, se puede visualizar la fórmula general que utiliza este modelo para realizar predicciones en base al tiempo, siendo p el valor que controla el número de componentes autorregresivos del modelo, d es el valor que controla la cantidad de

integraciones y q el valor que controla el número de componentes de media móvil (Contreras, 2003).

Ecuación 2

Fórmula general de ARIMA

$$Y_T = \phi_1 Y_{T-1} + \phi_2 Y_{T-2} + \ldots + \phi_{Ps+p+Ds+d} Y_{T-Ps-p-sD-d} +$$

$$+\delta + U_T + \theta_1 U_{T-1} + \ldots + \theta_{Qs+q} U_{T-sQ-q}$$

Nota. En esta ecuación se muestra la formula basica para relizar una regresion con el modelo ARIMA. Tomado de (Contreras, 2003).

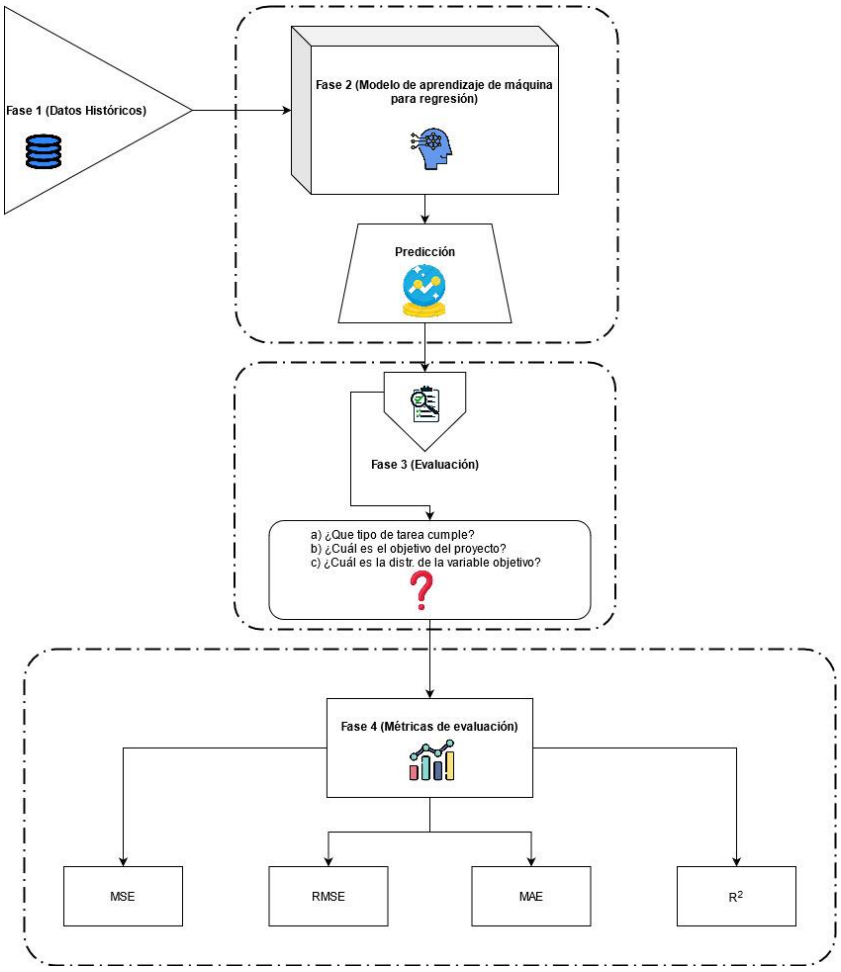
2.12. Métricas de Evaluación de algoritmos de regresión

Las métricas de evaluación permiten verificar el rendimiento que posee un modelo de aprendizaje automático (machine learning) a través del porcentaje de predicción obtenido por cada uno de los modelos utilizados dándose a conocer si estos brindan un resultado óptimo o deficiente.

En la Figura 17 se puede observar las cuatro fases necesarias para realizar el proceso que permita elegir un tipo de métrica de evaluación con el cual se valorará el resultado obtenido en el pronóstico o predicción que genere el algoritmo de regresión utilizado.

Figura 17

Proceso para elegir un tipo de métricas de evaluación para un modelo de regresión.



Nota. En esta figura se muestra el proceso para evaluar la efectividad de un modelo de regresión

Las cuatro fases utilizadas por este proceso se explican de una manera general y son las siguientes:

Fase1(Datos históricos). - Fase donde se ingresan los datos históricos al modelo de regresión para que pueda ser entrenado.

Fase 2 (Modelo de aprendizaje de máquina para regresión). - Fase donde el modelo realiza una predicción de acuerdo a los datos que se ingresaron en la fase 1(Datos históricos).

Fase 3(Evaluación). - Fase donde el modelo de predicción se evaluará según los datos de prueba suministrados, para lo cual se contestarán las siguientes interrogantes (Maldonado Schmeisser, 2020):

(a) ¿Qué tipo de tarea cumplirá, de regresión o de clasificación?
(b) ¿Cuál es el objetivo del proyecto?
(c) ¿Cuál es la distribución de la variable que se tiene por objetivo (salida que se desea obtener)?

Estas 3 interrogantes se deben responder de manera eficaz para así poder elegir correctamente la métrica adecuada al momento de evaluar diversos modelos de regresión que conforman el aprendizaje de máquina (machine learning).

Fase 4 (Métricas de evaluación). - Fase donde se analizan y evalúan las contestaciones realizadas a las preguntas planteadas en la fase 3 (Evaluación). Permitiendo elegir un tipo de métrica de evaluación acorde al modelo de regresión empleado de las cuatro métricas disponibles: 1) error cuadrático (MSE), 2) error cuadrático medio (RMSE), 3) error absoluto medio (MAE) y 4) R al cuadrado (R^2) (Maldonado Schmeisser, 2020).

A continuación, se describen de manera breve las cuatro métricas mencionadas en el párrafo que antecedió:

1) **Error cuadrático (MSE). -** Una de las métricas más simples y más utilizadas, permite realizar una estimación promedio de los errores al cuadrado tomando la diferencia entre el estimador y lo que se estima siendo aconsejable utilizarla en periodos con pequeñas desviaciones (González P. , 1990).

   En la Ecuación 3, se muestra la fórmula utilizada para calcular esta métrica que permite evaluar el desempeño de un algoritmo de aprendizaje automático (algoritmo de regresión), donde el Error de pronóstico^2 es el número de diferencias entre cada valor obtenidas elevadas al cuadrado y n es la cantidad de predicciones realizadas.

Ecuación 3

Fórmula error cuadrático

$$MSE = \frac{\sum Error\ de\ pronóstico^2}{n}$$

Nota. En esta ecuación se muestra la fórmula para realizar una evaluacion de efectividad con la metrica de error cuadratico. Tomado de (González P. , 1990).

2) **Error cuadrático medio (RMSE). -** Está métrica también es conocida como Raíz de la desviación cuadrática media, es la encargada de medir la diferencia y el valor promedio del error que existe entre dos conjuntos de datos, comparando así un valor observado con uno predefinido (GIL, 1988).

La Ecuación 4 se puede visualizarla fórmula establecida para obtener la estimación del error de predicción obtenido por el algoritmo utilizado, donde $(p_i - O_i)^2$ es la sumatoria de los errores de pronóstico obtenidos elevados al cuadrado divididos para n siendo esta la cantidad de predicciones realizadas.

Ecuación 4

Fórmula error cuadrático medio

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \left(P_i - O_i \right)^2}{n}}$$

Nota. En esta ecuación se muestra la fórmula para realizar una evaluacion de efectividad con la metrica de error cuadratico medio. Tomado de (GIL, 1988).

3) **Error absoluto medio (MAE). -** Esta métrica es empleada para obtener un promedio de todos los errores producidos por cada predicción, es decir que todas las diferencias individuales realizadas por MAE (Error absoluto medio) se ponderan por igual al promediarlas. Por ejemplo, la diferencia obtenida entre 10 y 0 corresponderá al doble de la diferencia entre 5 y 0 (De Cozar Macias, 2010).

En la Ecuación 5 se puede observar la fórmula matemática con la cual se puede realizar la estimación del error absoluto medio que se obtiene por medio del algoritmo de regresión empleado, donde el promedio de la sumatoria entre la diferencia del valor objetivo y el valor predicho $|y_i - y^2|$se la divide para N siendo este el número de predicciones efectuadas.

Ecuación 5

Fórmula error absoluto medio

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Nota. En esta ecuación se muestra la fórmula para realizar una evaluacion de efectividad con la metrica de error absoluto medio. Tomado de (De Cozar Macias, 2010).

4) **R al cuadrado (R^2). -** Métrica estrechamente relacionada con el error cuadrático (MSE), esta indica la idoneidad o integridad del modelo elegido para realizar predicciones. Se la utiliza mayormente con fines descriptivos, es por ello que tiene una escala de entre 0 y 1. Con 0 da a conocer que la predicción no es óptima y 1 indica un pronóstico perfecto (Riadi, 2004).

En la Ecuación 6 se puede observar la fórmula empleada por esta métrica para obtener el error de R al cuadrado (R^2) donde MSE (model) es la fórmula del error cuadrático descrita anteriormente en la métrica 1 dividido para MSE (baseline) como se puede visualizar en la Ecuación 7 que corresponde a la sumatoria de la diferencia al cuadrado del valor objetivo y la media de los valores observados $(y_i - y)^2$ dividiéndose este valor para N que es el número de predicciones realizadas.

Ecuación 6

Fórmula R^2

$$R^2 = 1 - \frac{\text{MSE(model)}}{\text{MSE(baseline)}}$$

Nota. En esta ecuación se muestra la fórmula para realizar una evaluacion de efectividad con la metrica de R^2. Tomado de (Riadi, 2004).

Ecuación 7

Fórmula baseline

$$\text{MSE(baseline)} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2$$

Nota. En esta ecuación se muestra la fórmula para calcular el baseline que formara parte de R^2. Tomado de (Alava, 2015).

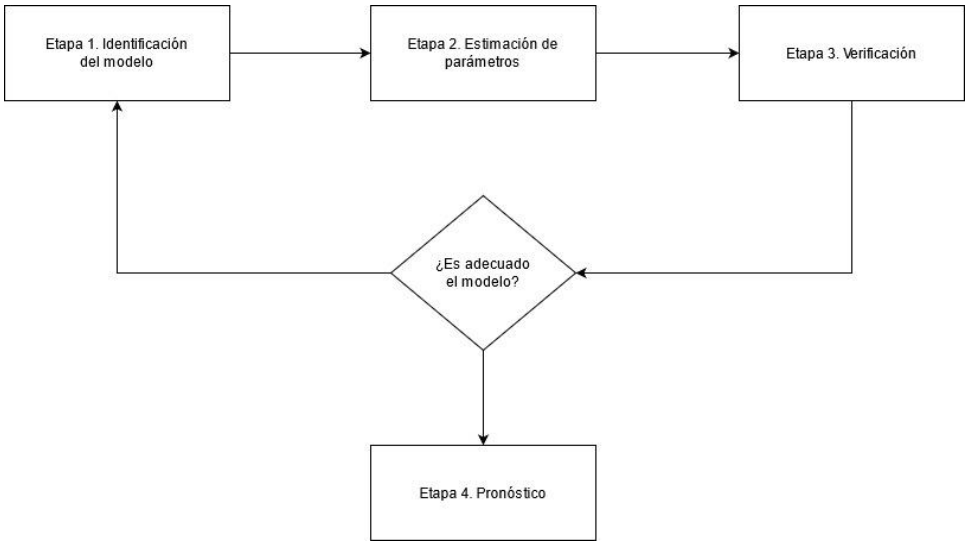
2.13. Metodología Box-Jenkins para la implementación de modelos predictivos utilizando ARIMA

Metodología que permite guiar la implementación del algoritmo de regresión ARIMA. En la Figura 18 se puede observar que dicho marco de trabajo está conformado por 4 etapas, las cuales se detallan de manera general a continuación: Etapa 1) Identificación del modelo, fase donde se evalúa si la predicción a desarrollar cumple con los parámetros suficientes para implementar ARIMA, Etapa 2) Estimación de parámetros, fase dónde se asigna un valor a las diferentes variables para alimentar al algoritmo, para que de esta manera pueda ejecutarse, Etapa 3) Verificación, fase donde se comprueba si el modelo funciona de una manera adecuada por medio de la utilización de métricas de evaluación (MSE, RMSE, MAE, R^2) en el caso de no ser óptimo se desecha el algoritmo junto con su metodología y se adopta uno más acorde a las necesidades del proyecto, y Etapa 4) Pronóstico, fase donde se realiza la predicción

lo que permite evaluar los comportamientos futuros basándose en datos reales (Danilo A. López, 2015).

Figura 18

Metodología de Box-Jenkins



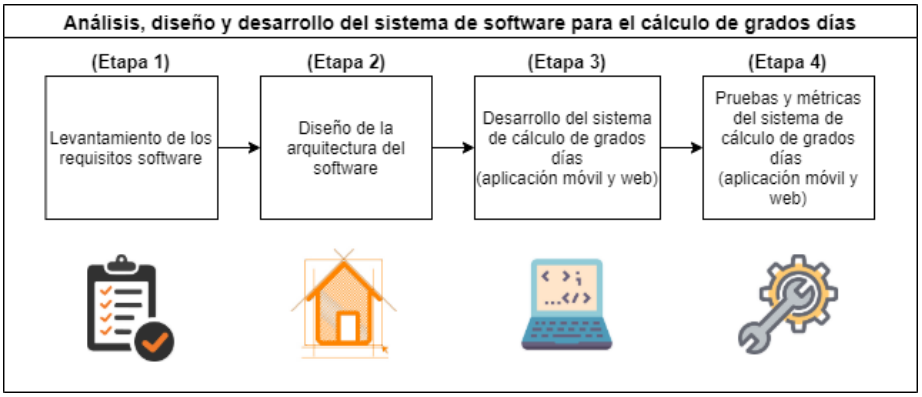
Nota. En esta figura se muestra la metodología para implementar el modelo ARIMA. Tomado de (Hernandez, 2006).

3. Análisis, diseño y desarrollo del sistema de software para el cálculo de grados días

En el presente capítulo se realiza el análisis, diseño, desarrollo e implementación de la aplicación propuesta en esta investigación basándose en la metodología SCRUM. A continuación, se observa en la Figura 19. Etapas para la creación del sistema de software propuesto en esta investigación. las fases a desarrollarse: 1) Levantamiento de Requisitos, 2) Diseño de la Arquitectura, 3) Desarrollo de la Metodología SCRUM, y 4) Desarrollo del modelo de predicción.

Figura 19

Etapas para la creación del sistema de software propuesto en esta investigación.



Nota. En esta figura se muestra las fases que conforman el presente capitulo y su secuencia.

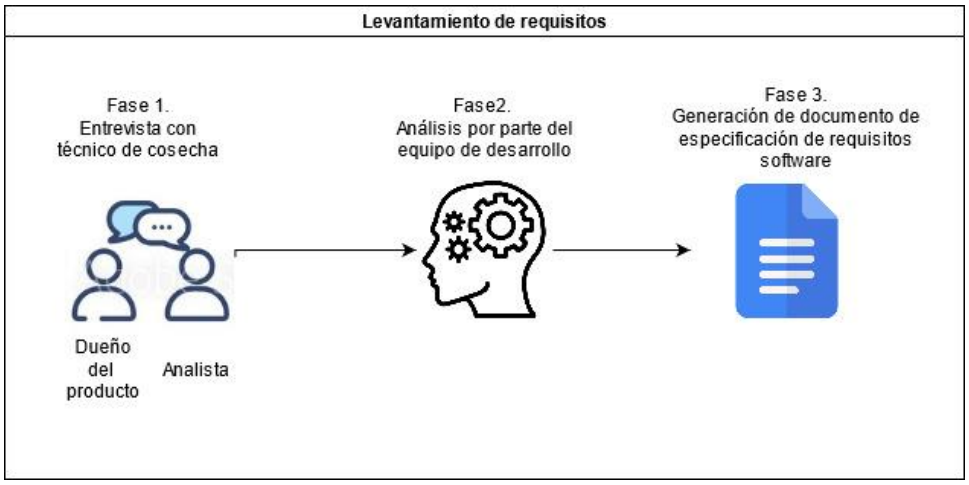
3.1. Etapa 1. Levantamiento de los requisitos de software.

En la Figura 20 se puede observar el proceso realizado en esta etapa el cual permite obtener el documento de especificación de requisitos software (ERS) mediante el levantamiento de las necesidades técnicas, las cuales son necesarias para realizar una cosecha optima y de calidad. Dichas condiciones fueron consensuadas con los ingenieros agrónomos técnicos de cosecha Sidney Galarza y Efraín López, consultores de la empresa Ecofroz para lo cual se realizaron las siguientes fases: 1) Entrevista con

el técnico de cosecha 2) Análisis por parte del equipo de desarrollo y 3) Generación del documento de especificación de requisitos software (ERS).

Figura 20

Levantamiento de requisitos



Nota. En esta figura se muestra las fases que se deben desarrollar para generar un documento de especificación de requisitos software.

A continuación, se detallan los pasos utilizados para obtener el documento de especificación de requisitos de software (ERS):

1. Fase 1 (Entrevista con los técnicos de cosecha). - Se realiza una conversación con los ingenieros agrónomos técnicos de cosecha Sidney Galarza y Efraín López, consultores de la empresa Ecofroz. Exponiéndose los inconvenientes existentes en la plantación de brócoli y proponiéndose alternativas de solución a través del desarrollo de un sistema de cálculo de grados día propósito de esta investigación. Coadyuvando al mejoramiento de la cosecha de esta hortaliza, lo cual se verá reflejado en la calidad de sus exportaciones y con esto al mejoramiento continuo del sector agroindustrial.
2. Fase 2 (Análisis por parte del equipo de desarrollo - tesistas). - Esta fase está fundamentada en la información recolectada en la Fase 1, dicha información permite a los analistas de sistemas el proponer soluciones a los inconvenientes expuestos por los ingenieros agrónomos técnicos de cosecha Sidney Galarza y Efraín López, consultores de la empresa Ecofroz. De esta manera el equipo de desarrollo a través del diálogo realizado en la Fase 1 ayudo a comprender de

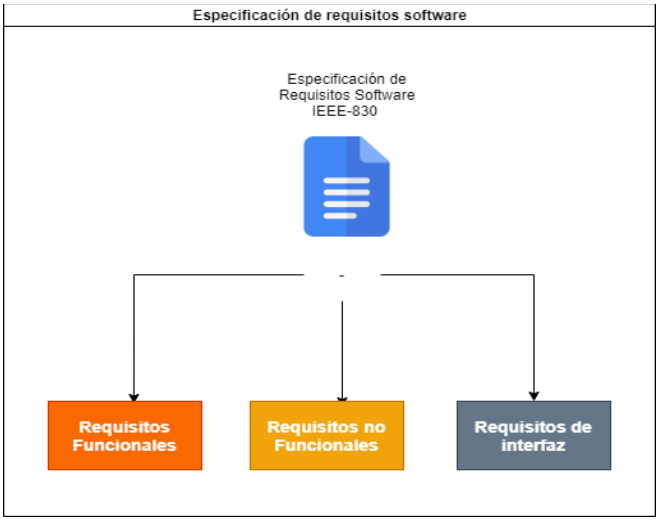
mejor manera el problema y establecer las posibles alternativas de solución. Gracias al análisis de las recomendaciones de los expertos en cosecha de brócoli, se creó un documento de especificación de requisitos de software. (ERS).

3. Fase 3 (Generación del documento de especificación de los requisitos de software). - El documento de especificación de requisitos de software se genera utilizando el estándar IEEE-830, que tiene como objetivo establecer un acuerdo entre los técnicos de cosecha (dueños del producto) y el equipo de desarrollo (tesistas) que realizan la presente investigación. Estableciéndose los compromisos de responsabilidad correspondientes de cada una de las partes. Sirviendo como apoyo al desarrollo del sistema software para el cálculo de grados días de la plantación de brócoli monitoreada por la empresa Asis-Agro y ubicada en la empresa Ecofroz. Este documento se describe en detalle en el Anexo 1 (Documento de especificación de los requisitos de software).

   Como se puede ver en la Figura 21, los documentos generados en la Especificación de Requisitos de Software (ERS) a través del estándar IEEE-830 se pueden clasificar en los siguientes requerimientos: 1) función, 2) no función y 3) interfaz.

Figura 21

Especificación de requisitos software



Nota. En esta figura se muestra los tipos de requisitos que forman parte del documento de requerimiento software

A continuación, se muestra los requerimientos obtenidos en cada una de las categorías antes mencionadas.

3.1.1. Requisitos funcionales.

Para poder realizar el desarrollo del sistema de cálculo de grados día propuesto en esta investigación es importante tener en cuenta los requerimientos funcionales ya que estos indican el comportamiento que debe tener el sistema, para generar dichos requerimientos se utilizó la Fase 1, Fase 2 y Fase 3 anteriormente explicadas. Lo cual posibilitó generar el documento de especificación de los requisitos de software (ERS).

A continuación, se muestran de manera general los requisitos funcionales que se obtuvieron para el desarrollo del sistema:

1. Gestionar usuario: Se logrará administrar usuarios en el sistema.
2. Gestionar hacienda: Se podrá gestionar haciendas existentes.
3. Gestionar hectáreas: Se podrá gestionar hectáreas existentes.
4. Gestionar cultivos: Se podrá gestionar todos los cultivos existentes.
5. Gestionar control del cultivo: Se gestiona el control del cultivo.

Detallándose de manera más específica los requerimientos funcionales obtenidos para la presente investigación en el Anexo 1 (ERS) numeral 3.2.

3.1.2. Requisitos no funcionales.

Uno de los aspectos importantes a tomar en consideración además de los requerimientos funcionales son los requerimientos no funcionales los cuales también son esenciales para el desarrollo de un sistema de calidad. Dichos requerimientos se enfocan en las restricciones o condiciones que impone el cliente al producto software a desarrollarse [poner la cita].

A continuación, se muestran los requisitos no funcionales obtenidos al realizar la fase 1 (Levantamiento de los requisitos de software) de la fase 3 (Generación del documento de especificación de requisitos software):

1. Seguridad. - La información del sistema es confidencial y nadie puede acceder a ella sin una identificación.
2. Fiabilidad. - El sistema debe funcionar de forma adecuada tomando en cuenta los requisitos sin presentar fallas.
3. Disponibilidad. - Debe ser visible en cualquier momento del día.
4. Portabilidad. - El sistema es multiplataforma es decir funciona en dispositivos móviles y web.

En el Anexo 1 (ERS) literal 3.3 se explica más detalladamente los requerimientos no funcionales.

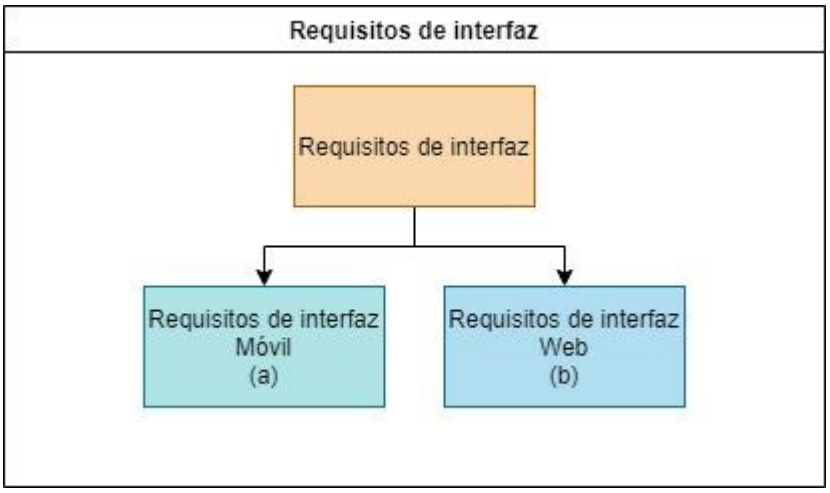
3.1.3. Requisitos de interfaz.

Las características de la calidad del software no solamente se miden a través de los requerimientos no funcionales, además de estos existen los requisitos de interfaz que determinan la facilidad de uso de una aplicación en general sin importar si es móvil o web. Para lo cual en el presente proyecto de investigación se toma en consideración la usabilidad como un atributo de suma importancia.

En la Figura 22 se puede observar que los requisitos de interfaz que fueron levantados para la presente investigación y que se encuentran en el documento de especificación de requisitos software generado en la Etapa 1 (Levantamiento de requisitos software) en su Fase 3. Requisitos que serán divididos en los siguientes: a) Requisitos de interfaz móvil y b) requisitos de interfaz web. Los cuales de explicaran en los párrafos siguientes.

Figura 22.

Requisitos de interfaz



Nota. En esta figura se muestra los dos tipos de interfase que forman parte del presente proyecto

a) Requisitos de interfaz móvil. - Son aquellos elementos visuales que se necesitan presentar de manera gráfica y ordenada de acuerdo a una secuencia lógica en una aplicación móvil para que el usuario comprenda fácilmente su utilización denominando a este proceso maquetación móvil.

En la Figura 23 se visualiza la maquetación realizada de la interfaz de la aplicación móvil, para lo cual se utilizó el programa Balsamiq WireFrames creando una versión inicial de su diseño. Diseño que fue dividido de acuerdo a los requerimientos funcionales que se encuentran en el documento de especificación de requisitos software (ERS) pudiendo ser analizados con mayor detalle en el Anexo 1 (ERS) literal 3.1.

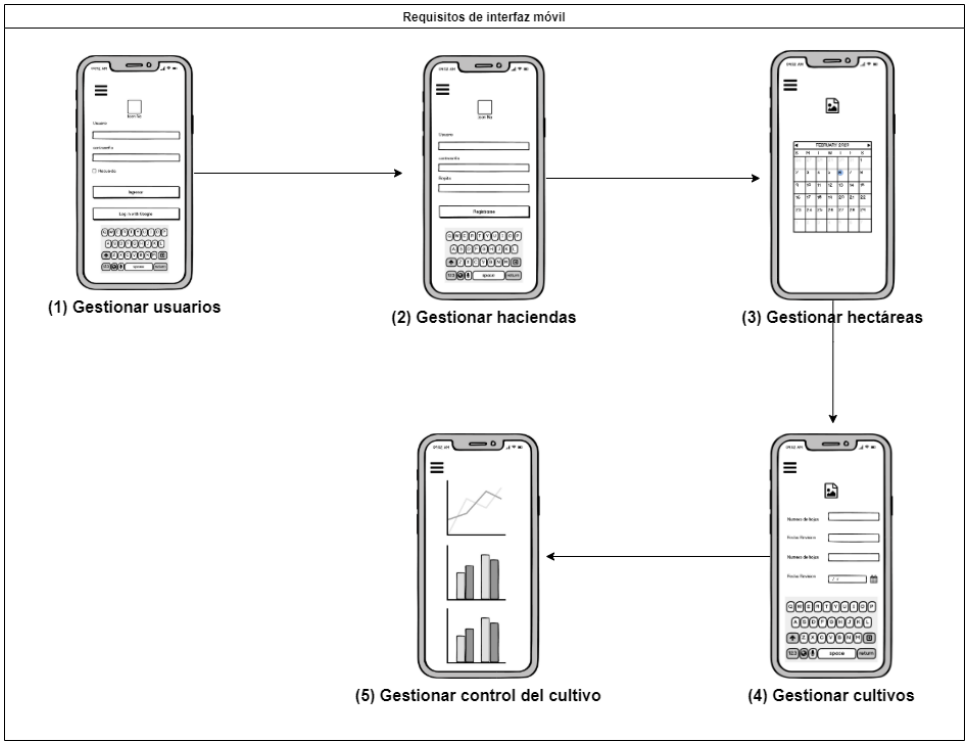
A continuación, se describe la relación existente entre los requerimientos funcionales y su maquetación.

1) Gestionar usuarios: Interfaz gráfica móvil que permite observar la lista de usuarios ingresados, dichos usuarios pueden ser editados y eliminados.
2) Gestionar haciendas: Interfaz gráfica móvil donde se observa la lista de todas las haciendas, pudiendo eliminar y editar.
3) Gestionar hectáreas: Interfaz gráfica móvil que permite observar la lista de todas las hectáreas, pudiendo editar y eliminar.

4) Gestionar cultivos: Interfaz gráfica móvil donde se observa la lista de todos los cultivos, estos cultivos se los puede editar y eliminar.
5) Gestionar control del cultivo: Interfaz gráfica móvil que permite observar las listas del control de un cultivo dando reportes del mismo.

Figura 23

Maquetación Móvil



Nota. En esta figura se muestra las vistas que forman parte del módulo móvil del sistema de software desarrollado

b) Requisitos de interfaz web. - Son aquellos elementos gráficos que son visualizados en un navegador web, dichos componentes facilitan la interacción con el usuario final (usabilidad) a través del uso de un boceto inicial denominado maquetación web.

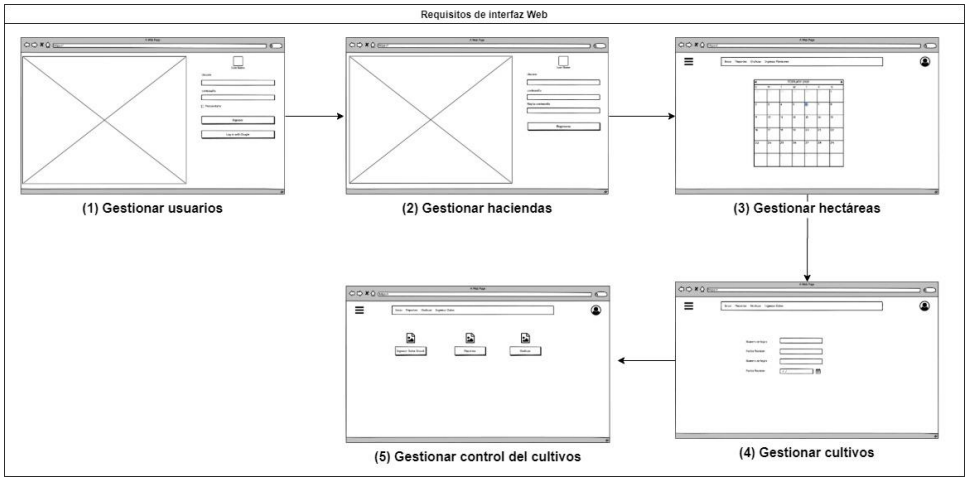
En la Figura 24 se observa un diseño inicial del sistema web, siendo de gran ayuda un programa de edición de imágenes para realizar bocetos de aplicaciones software denominado Balsamiq WireFrames. Este está divido según los requerimientos funcionales que se encuentran en el documento de especificación de requisitos software (ERS) explicado más profundamente en el Anexo 1 (ERS) literal 3.1.

La maquetación realizada se basa en los requerimientos funcionales que se detallan a continuación.

6) Gestionar usuarios: Interfaz gráfica web que permite observar la lista de usuarios ingresados, dichos usuarios pueden ser editados y eliminados.
7) Gestionar haciendas: Interfaz gráfica web donde se observa la lista de todas las haciendas, pudiendo eliminar y editar.
8) Gestionar hectáreas: Interfaz gráfica web que permite observar la lista de todas las hectáreas, pudiendo editar y eliminar.
9) Gestionar cultivos: Interfaz gráfica web donde se observa la lista de todos los cultivos, estos cultivos se los puede editar y eliminar.
10) Gestionar control del cultivo: Interfaz gráfica web que permite observar las listas del control de un cultivo dando reportes del mismo.

Figura 24

Maquetación Web



Nota. En esta figura se muestra las vistas que forman parte del módulo Web del sistema de software desarrollado

Todos los requisitos de interfaz se encuentran detallados más a detalle en el Anexo 1 (ERS) literal 3.1.

3.2. Etapa 2. Diseño de la arquitectura del software.

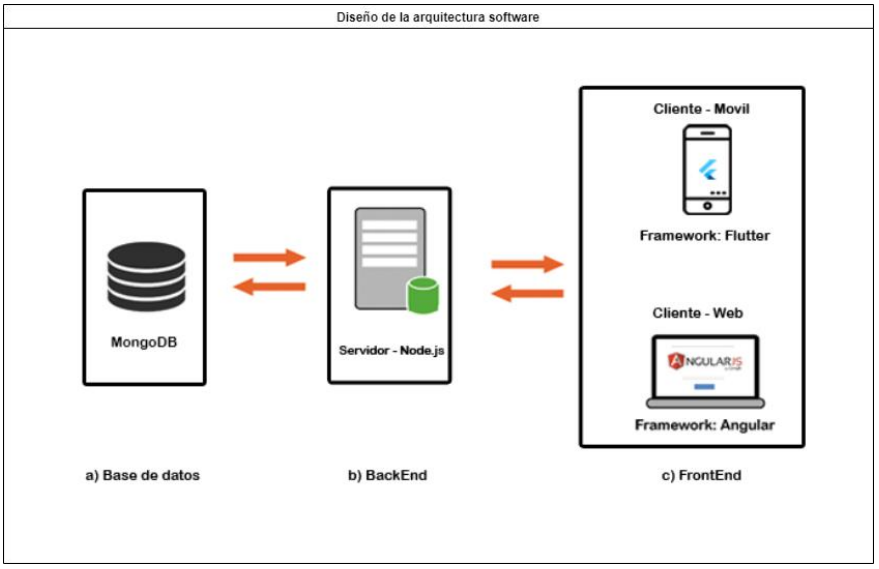
El objetivo principal de esta etapa es definir la estructura mediante la cual se desarrollará el sistema software para el cálculo de grados día en una plantación de brócoli, es decir se establecerán las piezas que se tienen que construir, el modo en el que se deben de juntar y trabajar entre ellas. Proporcionando una fuente de referencia para los analistas y diseñadores de aplicaciones, es decir, los estudiantes de tesis que desarrollaron este tema de investigación. Esta tabla se puede actualizar según los cambios de arquitectura técnica que aparezcan.

Además, el diseñar la arquitectura del software propuesto en esta investigación, orientó en la construcción de la aplicación y guio en la elaboración de los componentes que fueron necesarios para satisfacer los requisitos establecidos por el técnico consultor de la empresa Ecofroz Sr. Ing. Agrónomo Sídney Galarza los cuales se pueden observar en el Anexo 1 (ERS) numeral 3.2 requisitos funcionales.

En la Figura 25 se puede visualizar el diseño de la arquitectura propuesta para el desarrollo de la presente investigación, la cual posee los siguientes componentes: a) Base de datos (MongoDB) b) Back-end (Servidor - Node.js) y c) Front-end (cliente móvil - flutter y cliente web - angular).

Figura 25

Diseño de la arquitectura software



Nota. En esta figura se muestra el análisis para realizar la arquitectura del sistema de software

Para describir la interacción y funcionalidad entre los componentes mencionados anteriormente en la Figura 25 fue necesario utilizar el modelo 4 +1. Utilizándose aquellas vistas que dicho modelo contempla o requiere para diseñar la arquitectura software del presente proyecto de investigación detallándose dichos diagramas a continuación:

3.2.1. Vista de escenarios.

Representaciones gráficas que se utilizan para describir lo más importante de los requisitos funcionales representándolos a través de diagramas de casos de uso, técnica que permite especificar el comportamiento del software facilitando su compresión tanto a técnicos (tesistas) así como a los usuarios finales.

En la Figura 26 se visualiza el caso de uso general que diseñó para el desarrollo del sistema de cálculo de grados día resultado de un previo análisis que se realizó de la especificación de requisitos software como se puede ver en el Anexo 1. Detallándose a continuación los siguientes componentes:

1) Actores principales. - Para realizar esta vista, se identificó 3 actores que interactuaran con las funcionalidades del sistema en base a la especificación de requisitos software (ERS) como se pueden evidenciar en el Anexo 1 numeral 2.3, estos son:

Administrador. - Actor que tendrá control total de la aplicación, esto quiere decir que podrá gestionar haciendas, hectáreas, cultivos o usuarios, permitiéndole eliminar, modificar o guardar cualquier registro deseado. Así como también podrá realizar una consulta del clima de la región o de igual manera realizar predicciones para el tiempo de cosecha.

Usuario. - Actor que solo podrá entrar a la aplicación (login), registrar sus datos. Este tipo de usuario no tendrá ningún control sobre el sistema hasta que el administrador le asigne un rol (administrador o revisor).

Revisor. - Actor que tendrá control únicamente sobre el desarrollo y crecimiento del cultivo, permitiéndole así registrar por semanas la edad del brócoli, color, número de hojas, diámetro de la pella (tallo) o su altura.

2) Casos de uso. - Se representaron las funcionalidades del análisis de requisitos de software (ERS) obteniendo las siguientes: a) Gestionar hacienda, b) Gestionar hectárea, c) Gestionar cultivo, d) Gestionar control y e) Autentificar. A continuación, se describe cada uno de ellos.

Gestionar haciendas. - permite modificar, eliminar o guardar los datos (nombre, altitud, número de hectáreas) de la hacienda a gestionar. El control de este caso de uso únicamente lo podrá realizar el actor con rol de administrador.

Gestionar hectáreas. - permite modificar, eliminar o guardar los datos de la ubicación (coordenadas) que puede ser ingresada de manera manual (aplicación web) o por Sistema de Posicionamiento Global (Aplicación móvil) obteniendo latitud y longitud. Además, para obtener la temperatura de la hectárea es necesario utilizar la API Open Weather Map a través del registro de las temperaturas máximas y mínimas. El control de este caso de uso únicamente lo podrá realizar el actor con rol de administrador.

Gestionar cultivo. - permite realizar una predicción del tiempo de cosecha por medio de la consulta del clima que se encuentra registrado en la hectárea (Gestionar Hectárea), estas predicciones pueden ser diarias y mensuales. Las predicciones diarias realizan una predicción de fin del cultivo actualizándose cada 3 días, mientras que las predicciones mensuales realizan una actualización cada mes. Cabe destacar que la predicción que se realiza diariamente es más eficaz que la que se realiza mensualmente debido a que los factores climáticos son muy cambiantes. El control de este caso de uso únicamente lo podrá realizar el actor con rol de administrador.

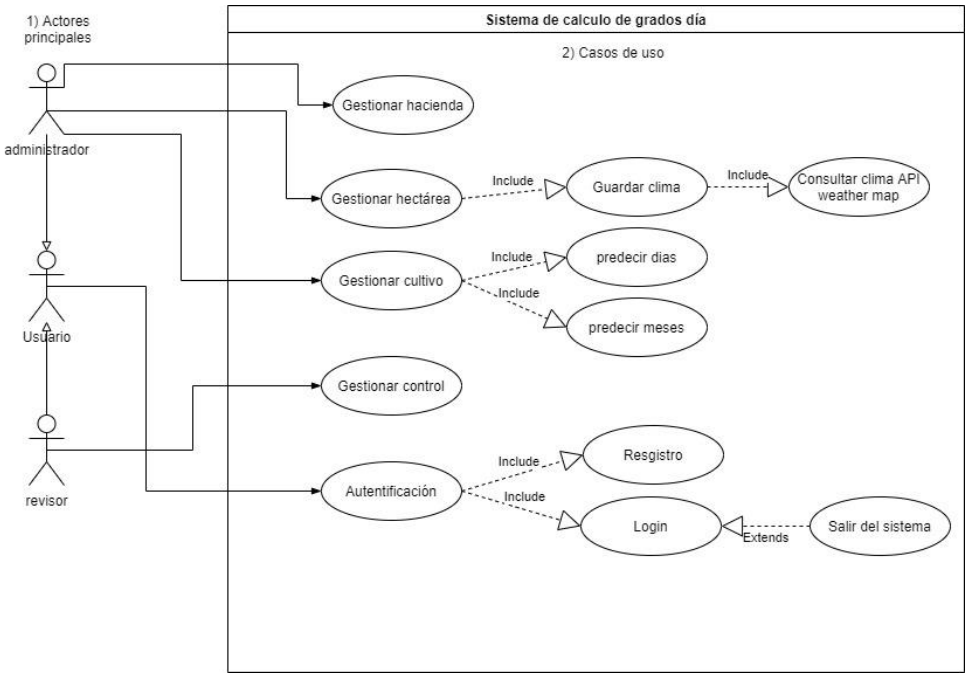
Gestionar control. - permite registrar cada semana la edad del cultivo, numero de hojas promedio de las plantas, diámetro de la pella (tallo), altura promedio de las plantas y altura, dichos datos permiten poseer un historial de la evolución y crecimiento del brócoli a lo largo del tiempo. Información que será de utilidad en futuras investigaciones. El control de este caso de uso podrá realizarlo el actor con el rol de revisor y el administrador.

Autentificación. - permite el registro de nuevos usuarios, así como la autentificación de estos. El actor con el rol de administrador podrá realizar una

modificación de datos, eliminar registros o guardarlos, también podrá asignar un rol (administrador o revisor) a algún usuario nuevo.

Figura 26

Casos de uso



Nota. En esta figura se muestra el diagrama de casos de uso con el cual se asignará roles a los diferentes usuarios del sistema

Todos los datos mencionados en la vista de escenarios fueron sustentados en el documento de especificación de requerimientos software (Anexo 1).

3.2.2. Vista lógica.

Vista en la cual se proporciona la estructura del sistema software propuesto en el presente proyecto de investigación, permitiendo determinar que atributos debe tener la base de datos y la relación que existe entre sus clases, a fin de dar una visión adecuada para el funcionamiento óptimo del sistema de software en desarrollo.

Cabe recalcar que dicha vista además permite poseer una compresión más detallada de la funcionalidad que tendrá la aplicación a desarrollarse, basándose en la descripción obtenida en el levantamiento de requisitos funcionales y no funcionales que se encuentran detallados en el anexo 1 (especificación de requisitos software).

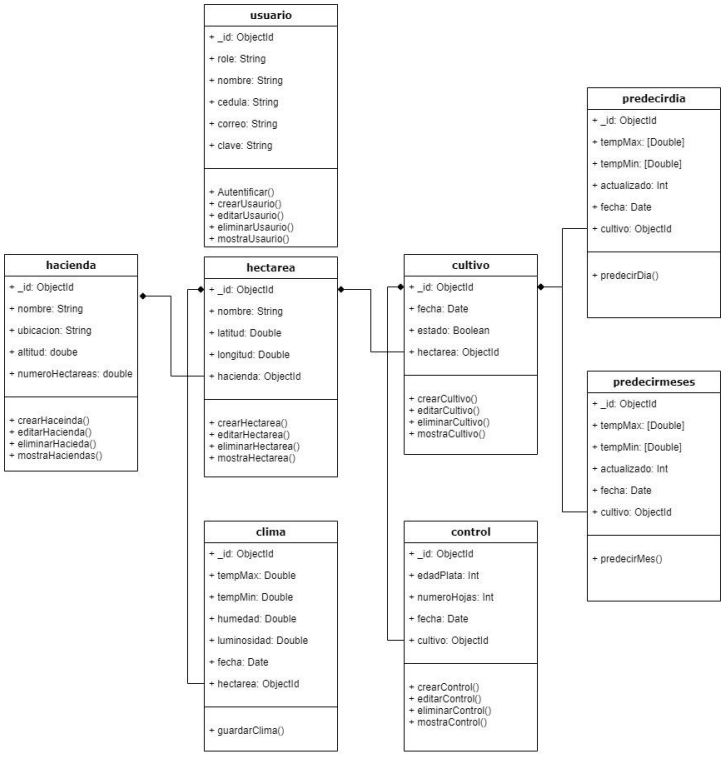
A continuación, se detallan los dos diagramas que se elaboraron y utilizaron para el desarrollo del sistema propuesto:

a) Diagrama de clases. - Representación gráfica que sirvió como referencia para la construcción de la base de datos y la forma de interactuar entre ventanas, tanto en la aplicación móvil y web.

En la Figura 27, se puede observar los atributos, funciones y relaciones que existen entre los actores y procesos que conforman el sistema propuesto en la presente investigación, permitiendo dar una solución a la problemática planteada obteniéndose el diagrama de clases. Cabe destacar que el análisis se lo realizo basándose en el documento de especificación de requisitos (anexo 1, numeral 2.2). Dicho diagrama de clases obtenido propone las siguientes clases: hacienda, hectárea, cultivo, control, clima, predecir día, predecir mes y usuario.

Figura 27

Diagrama de clases



Nota. En esta figura se muestra las colecciones y su interacción entre sí, con esto se implementará una base de datos

En la Figura 28 se puede visualizar la actividad realizada posteriormente, la cual permite que el diagrama de clases obtenido que se observó en la Figura 27 se materialicen en una base de datos no relacional concretamente Mongo DB por ser mucho más rápida que un base de datos SQL ya que se debe guardar una gran cantidad de información referente a la temperatura diaria de los cultivos de brócoli, permitiendo el cálculo de los grados días, datos sumamente importantes para pronóstico de su cosecha.

Figura 28

Colecciones de MongoDB

Nota. En esta figura se muestra las colecciones que forman parte de la base de datos Mongo DB

b) Diagramas de secuencia. - Representación visual que permite entender la interacción entre los actores y funcionalidades del sistema planteado tomando en consideración el módulo al cual pueden acceder.

En la Figura 29, se puede visualizar el diagrama de secuencia obtenido, el cual está conformado por los siguientes actores: 1) Administrador, 2) Revisor, 3) JWToken, 4) Front-end, 5) Cultivo y 6) Calculo de grados día. Dicho diagrama ayuda a entender la interacción adecuada que tendrá cada actor al momento de implementar el presente proyecto de investigación. Facilitando así la abstracción y comprensión por parte de los desarrolladores (tesistas) y los usuarios finales (Ing. Sidney Galarza e Ing. Efraín López).

A continuación, se detalla de manera general la interacción entre los actores que conforman el diagrama de secuencia planteado en esta investigación:

Administrador – revisor – JWToken

El actor administrador y revisor deben pasar por medio del actor JWToken para poder acceder al sistema, el cuál validará el tipo de privilegio que tendrá cada usuario ingresado para de esta manera poder acceder a las funcionalidades del sistema de cálculo de grados días.

Revisor - FrontEnd

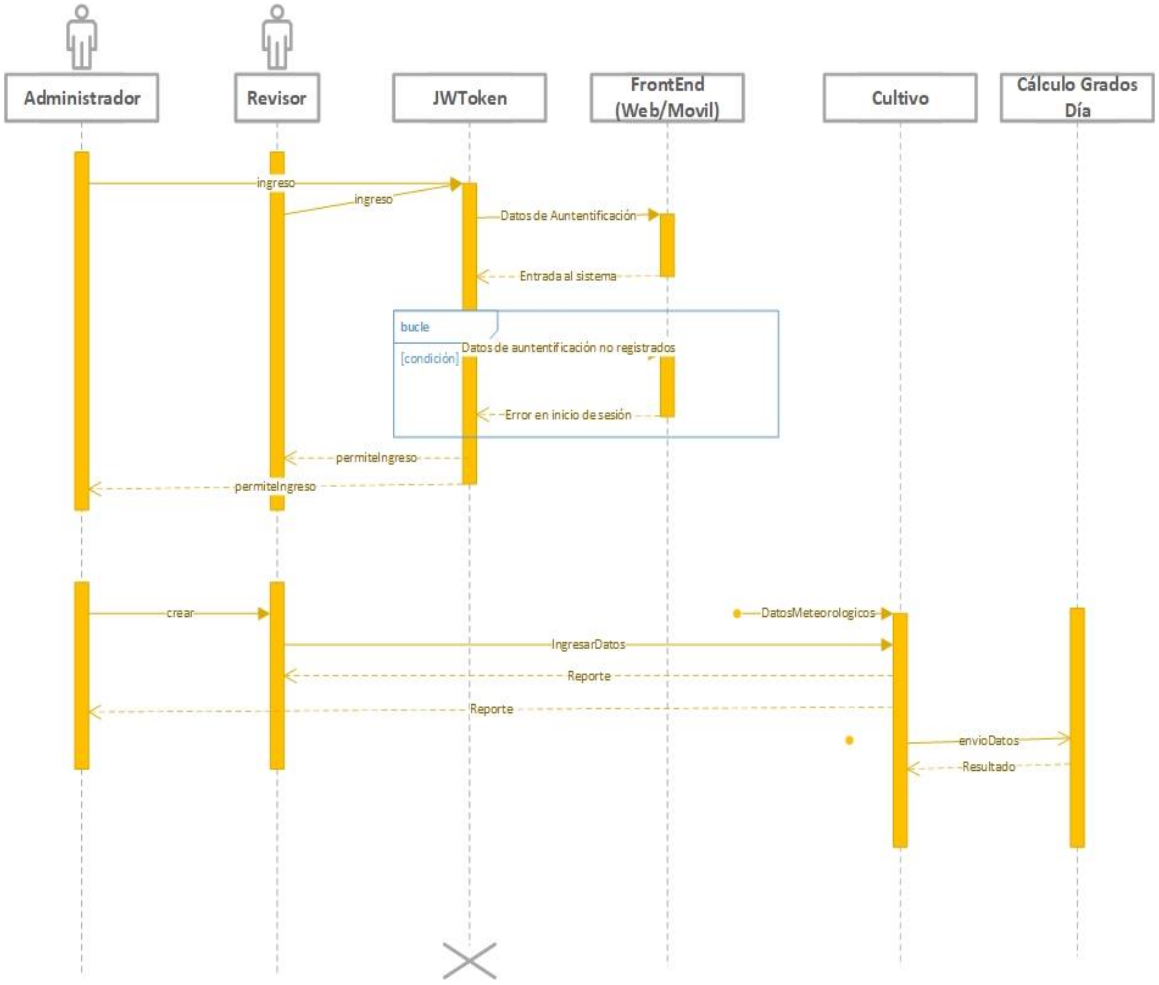
Una vez que el actor revisor haya ingresado al sistema (Autentificado) puede interactuar con el actor cultivo para de esta manera ingresar datos como: la edad del cultivo, número de hojas promedio de las plantas, diámetro de la pella (tallo) y altura. Permitiendo descubrir el estado de la plantación en cada semana de su desarrollo.

Cultivo – Calculo Grados Día

Una vez que se hayan ingresado los datos del actor cultivo es posible interactuar con el actor cálculo de grados día, para de esta manera obtener desde la base de datos las temperaturas máximas y mínimas con las cuales se realizará el cálculo de la fecha estimada de cosecha, generando así reportes del avance del cultivo junto con la predicción necesaria.

Figura 29

Diagrama de secuencia



Nota. En esta figura se muestra la secuencia con la que funciona la interacción de cada rol de usuario con el sistema de software

3.2.3. Vista física.

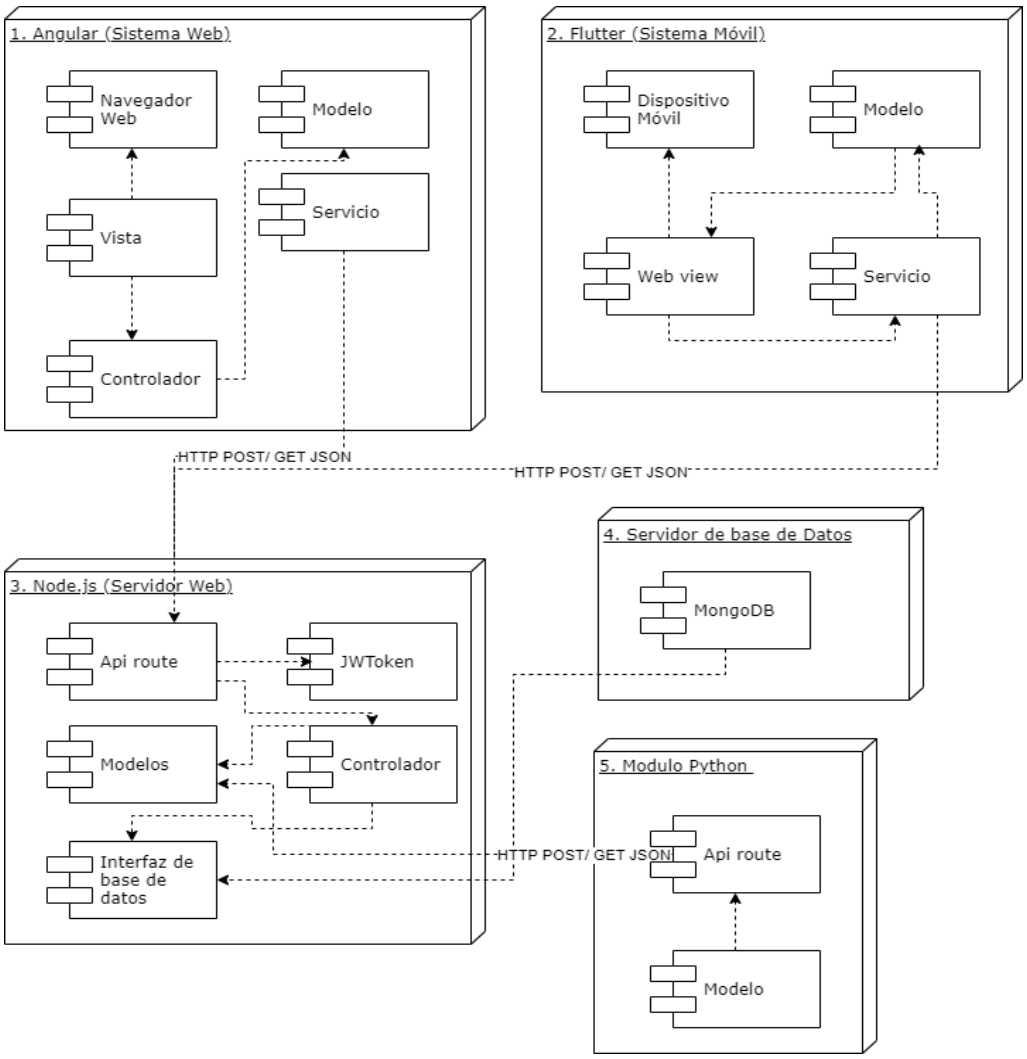
Vista mediante la cual se permite mostrar los componentes que conforman la aplicación a desarrollar tomando en consideración las conexiones existentes entre ellos, para lo cual se hace uso de su representación a través del diagrama de despliegue.

En la Figura 30, se puede observar el diagrama de despliegue obtenido donde se muestra cómo se relacionan los cinco módulos que serán usados al momento de desarrollar el sistema de cálculo de grados estos componentes son los siguientes: 1)

Angular – Sistema web, 2) Flutter – Sistema Móvil, 3) Node.js – Servidor Web, 4) MongoDB – Base de datos y 5) Modulo Python.

Figura 30.

Vista física



Nota. En esta figura se muestra las tecnologías que forman parte del proyecto y su interacción entre sí

Cada uno de estos módulos y sus conexiones existentes se detallan a continuación:

1) Angular – Sistema web. - En la Figura 30 numeral (1) se pudo visualizar la estructura utilizada al momento de usar el framework angular el mismo que está

conformado de un modelo, un controlador, un servicio, una vista y un navegador web elementos que en su conjunto permiten que el sistema en desarrollo pueda ser manejado desde un explorador web. Requiriéndose para tal propósito la existencia de una relación con el con el componente Node.js – Servidor Web conexión realizada por medio del protocolo HTTP lográndose obtener los datos enviados del servidor para su posterior procesamiento a través del controlador correspondiente y así proveer la funcionalidad necesaria para presentarlos en la vista del navegador web.

2) Flutter – Sistema Móvil. - En la Figura 30 numeral (2) se puedo observar la estructura usada al momento de emplear el framework flutter el cual está conformado por un modelo, un web view, un servicio y un dispositivo móvil todos estos elementos permiten que la aplicación a desarrollarse pueda ser accedida desde un smartphone. Para lo cual es necesario vincularse con el componente Node.js – Servidor Web. Conexión realiza por medio del protocolo HTTP de manera similar a la comunicación existente entre angular y el servidor web. Proveyéndose de información en formato JSON (texto plano), creando así un modelo que será procesado por un controlador que permite mostrar estos datos a través del navegador del celular el cual está integrado dentro de la aplicación.

3) Node.js – Servidor Web. - En la Figura 30 numeral (3) se puedo mirar la organización utilizada al momento de usar el framework Node.js el cual posee cinco elementos: 1) modelo, 2) api route, 3) controlador, 4) Interfaz de base de datos y 5) JWToken. Dichos elementos facilitan la implementación de la lógica del negocio (funciones que realizará el sistema). Mediante él envió de la información requerida por el sistema móvil o sistema web a través del uso del protocolo HTTP similar a la comunicación que se estableció anteriormente entre angular - servidor web y flutter – servidor web. Cabe destacar que un aspecto importante de este módulo es la seguridad que se brinda a la aplicación a través del uso de JWToken (middleware) para conceder acceso solo a usuarios autorizados es decir conceder permisos de autentificación al sistema para evitar infiltraciones no deseadas.

4) MongoDB – Base de datos. - En la Figura 30 numeral (4) se puedo visualizar el módulo en el cual se configura la conectividad requerida por el servidor web y la base de datos Mongo DB a fin de administrar la información requerida por la aplicación es decir gestionar todas las peticiones de ingreso, actualización o eliminación de datos requeridos por el usuario o cliente final.

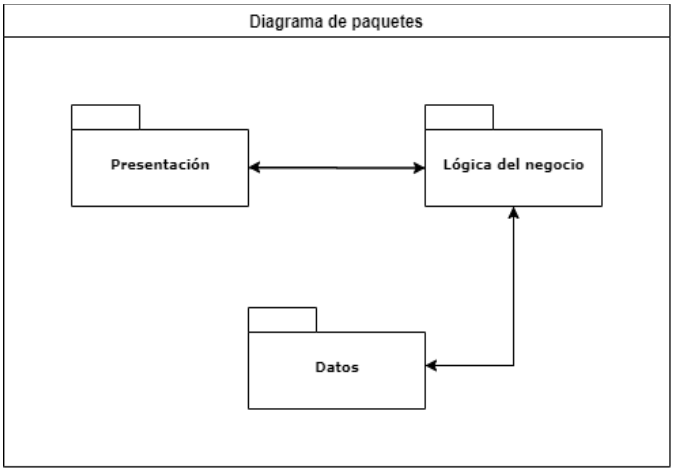
5) Modulo Python. - En la Figura 30 numeral (5) se puedo observar la organización que contiene el módulo de Python el cual está conformado de dos elementos: 1) modelo de datos y 2) api route. Estos elementos permiten enviar datos al servidor web mediante el api route con el protocolo HTTP (url's de conexión) para que posteriormente el módulo 3 (Node.js - servidor web) a través del componente denominado modelo de servidor envié a su controlador de servidor la petición y realicé la conexión con la interfaz de base de datos y de esta manera sean almacenados en la base de datos (modulo 4 - MongoDB) logrando de esta manera obtener información de predicción de temperatura máxima y minina.

3.2.4. Vista de desarrollo.

Vista mediante la cual se definirá la estructura de los paquetes del sistema en desarrollo (agrupación de funcionalidades). En la Figura 31 se puede observar el diagrama obtenido el cual está conformado de tres capas: 1) Presentación, 2) Lógica del negocio y 3) Datos.

Figura 31

Diagrama de paquetes



Nota. En esta figura se muestra los diferentes paquetes que forman parte del sistema desarrollado en este proyecto

A continuación, se detalla de manera general cada uno de las capas que conforman el diagrama de paquetes que se observó en la Figura 31:

1) Capa de presentación. - Capa encargada de mostrar de forma visual al usuario la funcionalidad del presente sistema es decir la representación gráfica del proyecto en desarrollo(vistas). Cabe destacar que las vistas a desarrollarse para el sistema web utilizaran el framework Angular y para el sistema móvil el framework Flutter.

2) Capa de la lógica del negocio. – Capa que presenta los requerimientos funcionales que posee el presente proyecto en desarrollo a fin de cumplir las necesidades de la aplicación (lógica del negocio). Estas necesidades se han detallado en la Fase 1 (Levantamiento de los requisitos de software).

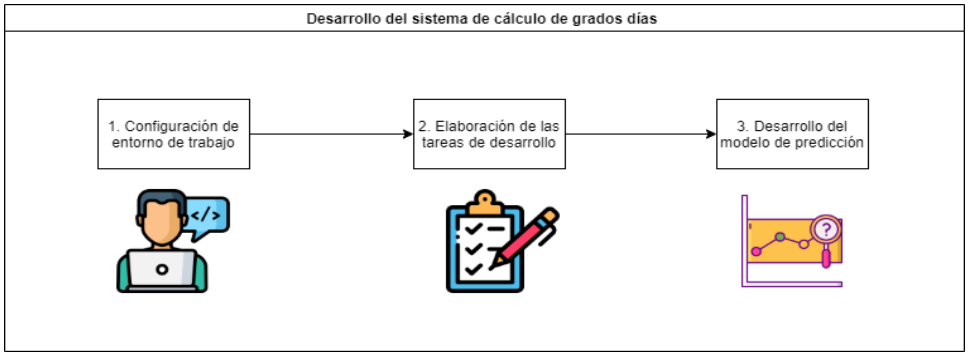
3) Capa de datos. – Capa que permite poseer una base de datos o repositorio donde serán almacenados toda la información o datos pertinentes que permitirán el desarrollo de software del presente proyecto. Además, esta capa deberá ser utilizada por la capa de presentación y la capa de la lógica del negocio.

3.3. Etapa 3. Desarrollo del sistema de cálculo de grados días (aplicación móvil y web).

Etapa en la cual se describe el desarrollo del sistema de software implementado en esta investigación. En la Figura 32 se puede visualizar los elementos que conforman esta etapa: 1) Configuración de entorno de trabajo ,2) Elaboración de las tareas de desarrollo y 3) Desarrollo del modelo de predicción.

Figura 32

Desarrollo del sistema de cálculo de grados día



Nota. En esta figura se muestra las fases que conforman la etapa 3 para el desarrollo del presente proyecto

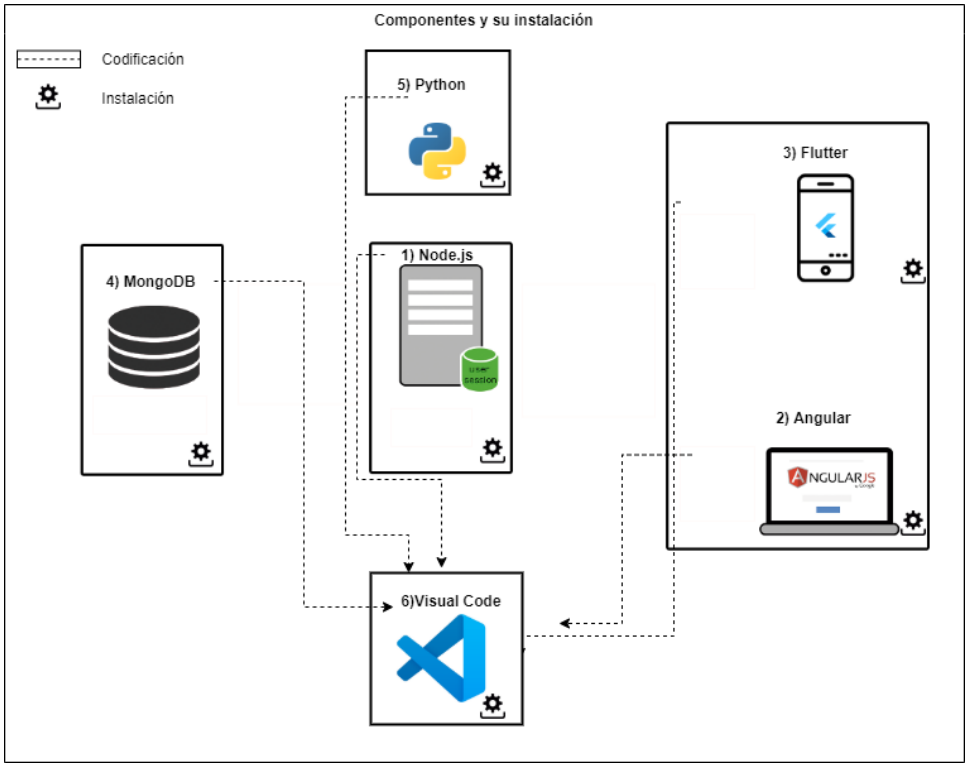
A continuación, se detallan cada una de las fases mencionadas en el párrafo que antecedió a este.

3.3.1. Configuración de entorno de trabajo.

Fase en la cual se explica la configuración del entorno de trabajo realizado en el presente proyecto de investigación. En la Figura 33 se pueden observar los componentes requeridos (Frameworks), los mismos que fueron instalados y configurados (conexiones) para la construcción del aplicativo propuesto. Cabe destacar que en esta etapa fue fundamentada en lo establecido en la Etapa 2 denominada Diseño de la arquitectura de software específicamente en la Vista Física.

Figura 33

Configuración de entorno de trabajo



Nota. En esta figura se muestra la configuración que se deberá realizar para codificar el sistema

A continuación, se describe en detalle y de manera general cada componente utilizado (tecnología requerida) y su instalación. Como se pudo visualizar en la Figura 33:

1. Node.js. - Framework que se utiliza para realizar la lógica del negocio (funciones que realizará el sistema) planteada en la especificación de requisitos software establecido en la Etapa 2 (Diseño de la arquitectura de software) en la vista física. Su instalación se ejecuta localmente con requerimientos mínimos de corei3 de cuarta generación dichas instrucciones se las puede observar en el Anexo 3 numeral 2 (Instalación Node.js).

2. Angular. - Framework que se requiere al momento de desarrollar las interfaces o vistas que el usuario final observará en cualquier computadora dentro de un navegador web establecido en la Etapa 2 (Diseño de la arquitectura de software) en la vista física. Para utilizar de una manera correcta es necesario instalar como se detalla en el Anexo 3 numeral 3 (Instalación Angular).

3. Flutter. - Framework que se necesita al momento de desarrollar las interfaces o vistas que el usuario final observara en un dispositivo móvil, en este caso a través del sistema operativo Android establecido en la Etapa 2 (Diseño de la arquitectura de software) en la vista física. Teniendo en cuenta que para su instalación se necesitó de mínimo Windows 7 o superior como se detalla en el Anexo 3 numeral 4 (Instalación Flutter).

4. Mongo DB. - Base de datos no relacional que se emplea para obtener el almacenamiento de información en una base de datos no relacional lo cual facilitara almacenar grandes cantidades de datos lo que se establecido en la Etapa 2 (Diseño de la arquitectura de software) en la vista lógica. De esta manera su instalación se la logra en un computador que haga de servidor de base de datos como se muestra en el Anexo 3 numeral 5 (Instalación Mongo DB).

5. Python. - Framework que se usa para el desarrollar inteligencia artificial a través del algoritmo de aprendizaje no supervisado (ARIMA) permitiendo predecir las temperaturas máximas y mínimas lo cual permitirá realizar la cosecha del brócoli en el tiempo adecuado siendo esto establecido en la Etapa 2 (Diseño de la arquitectura de software) en la vista física. Para su instalación es necesario requerimientos mínimos

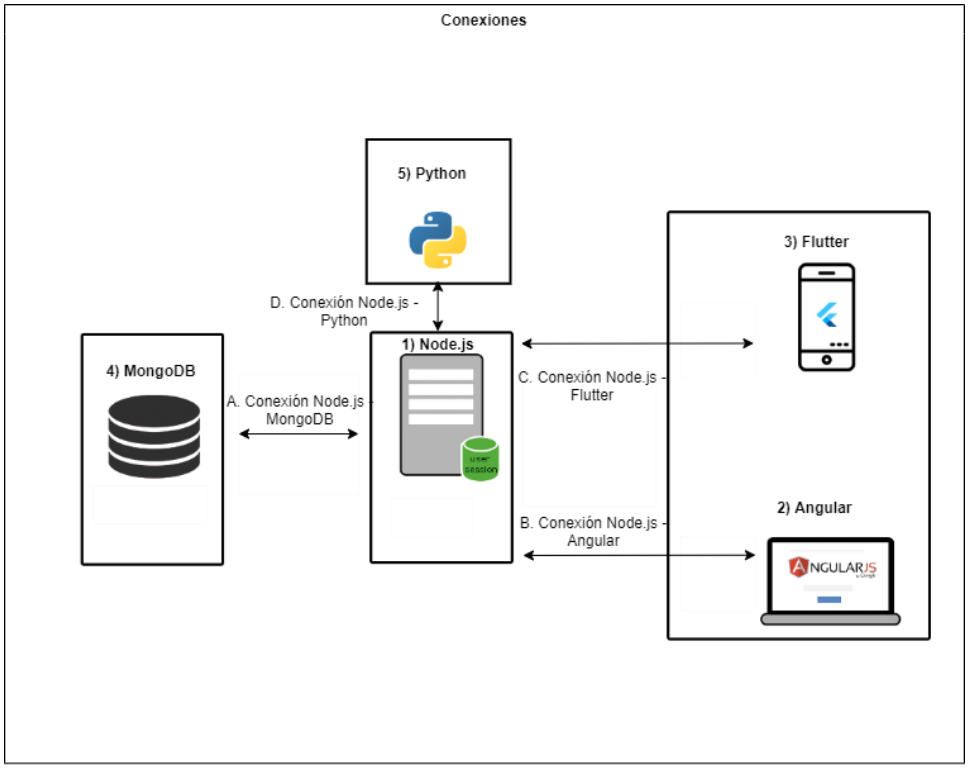
como Windows 7 o superior y un procesador corei3 de tercera generación esto se puede revisar en el Anexo 3 numeral 6 (Instalación Python).

6. Visual Code. - Editor de código fuente que se utiliza en el presente proyecto de investigación para realizar el desarrollo de software a través del uso de los diferentes frameworks utilizados. Permitiendo la generación de los scripts correspondientes los cuales facilitan obtener la funcionalidad de la aplicación móvil y web. Su instalación puede ser revisada el Anexo 3 numeral 7 (Instalación Visual Code).

Cabe destacar que para que exista comunicación entre todas las tecnologías mencionadas anteriormente se debe realizar la configuración de los entornos de trabajo utilizados (Node.js, Angular, Flutter, MongoDB, Python) permitiendo su interconexión y correcto funcionamiento.

Figura 34

Conexiones entre componentes



Nota. En esta figura se muestra la implementación de las conexiones entre los diferentes frameworks y tecnologías que forman parte del proyecto

Cada conexión existente se describe en detalle a continuación de manera general. Como se muestra en la Figura 34:

A. Conexión Node.js - MongoDB.

Interconexión que se realiza entre el entorno de trabajo Node.js (servidor web) y MongoDB (base de datos) la cual posee la información requerida por el sistema (Clima, hectáreas, cultivos, haciendas, control y usuarios) y a su vez proporciona el acceso desde la aplicación móvil o el sistema web. Cabe mencionar que dicha conexión se realiza luego de haberse realizado la instalación previa de Node.js y MongoDB como se pudo observar anteriormente en la Figura 34. Requiriéndose adicionalmente la configuración del script Node.js que viene por defecto en este framework. Como se puede observar en el Anexo 3 numeral 8 (Conexión MongoDB con NodeJS).

B. Conexión Node.js - Angular

Conectividad que se lleva a cabo entre el entorno de trabajo Node.js (Servidor web) y Angular (Sistema Web) lo que permite al cliente final visualizar las interfaces del sistema desarrollado a través del uso de un navegador web facilitando percibir su usabilidad a través de su interacción con el sistema (experiencia de usuario) es decir determinar si su percepción es positiva o negativa enfocándose en la funcionalidad descrita en el Anexo 1 (especificación de requisitos de software). Siendo necesario la codificación del script de Node.js que posee el presente framework. Todo esto se puede evidenciar en el Anexo 3 numeral 9 (Conexión Angular a NodeJS).

C. Conexión Node.js - Flutter

Interconexión que tienen los entornos de trabajo Node.js (Servidor web) y Flutter (Sistema móvil) de esta manera se consigue que el usuario logre interactuar desde un Smartphone facilitando su portabilidad enfocada en la facilidad de uso, además permite un control del cultivo de brócoli en cualquier lugar, dicha funcionalidad del sistema está definida en el Anexo 1 (especificación de requisitos de software). De esta manera es necesario configurar el script de Node.js que viene por defecto en el framework. Detallado en el Anexo 3 numeral 10 (Conexión Flutter a NodeJS).

D. Conexión Node.js - Python

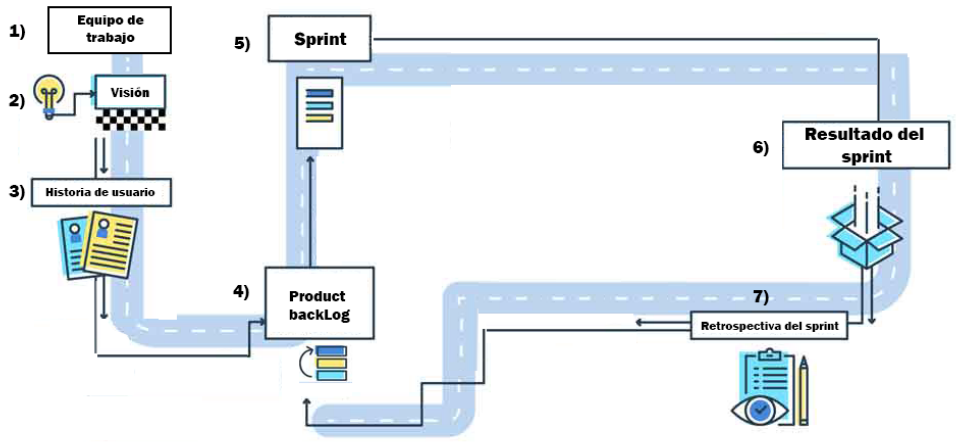
Conectividad que se realiza entre el entorno de trabajo Node.js (servidor web) y Python (modulo web) la cual envía los datos de predicción de temperatura máxima y mínima para un periodo de tiempo de 100 días consiguiendo de esta manera alimentar la base de datos del sistema de cálculo de grados día siendo procesados de manera más sencilla por las aplicaciones web y móvil. Para mayor información sobre esta conexión se puede visualizar el Anexo 3 numeral 11 (Conexión NodeJS a Python).

3.3.2. Elaboración de las tareas de desarrollo

En esta fase se hizo uso de la metodología SCRUM la cual permite desarrollar el presente proyecto de investigación facilitando la división del trabajo (esfuerzo) en etapas (tareas). En la Figura 35 se observa cómo se aplica la metodología SCRUM listándose sus etapas a continuación: 1) Equipo de trabajo, 2) Visión, 3) Historias de usuario, 4) Lista de historias de usuario (Product backlog), 5) Desarrollo de historia de usuario (Sprint), 6) Resultados de historias de usuario (Sprint result) y 7) Retrospectiva del sprint (Retrospective sprint).

Figura 35

Elaboración de las tareas de desarrollo



Nota. En esta figura se muestra las etapas por las que pasa el desarrollo del sistema para ser elaborado según SCRUM

A continuación, se realiza una breve explicación del marco de trabajo que se propuso para el desarrollo de la metodología SCRUM el cual consta de las siguientes etapas:

1. Equipo de desarrollo (SCRUM team). - En la Figura 35 numeral 1 se puede visualizar la primera etapa del marco de trabajo seleccionado en la cual están detallados los integrantes involucrados en el desarrollo del sistema de cálculo de grados días los roles existentes son los siguientes: SCRUM master a cargo del Mgs. Milton Escobar, product owner (Ing. Sídney Galarza y Ing. Efraín López) y SCRUM developer (Cristian Tapia y David Tamayo) todos los participantes son detallados más a fondo en el Anexo 3 numeral 12 (Equipo de desarrollo del proyecto).

2. Visión (vision). - En la Figura 35 numeral 2 se observa la segunda etapa de la metodología utilizada, la cual es necesaria para entender de mejor manera la problemática propuesta por los ingenieros agrónomos técnicos de cosecha Sidney Galarza y Efraín López (product owner's), quienes plantearon la necesidad de desarrollar un sistema para el cálculo de grados día el cual minimizará las pérdidas en la plantación de brócoli, mejorando el ciclo productivo de esta hortaliza. Estas necesidades fueron establecidas en la Etapa1 denominada levantamiento de requisitos de software correspondiente a la Fase 1 (Entrevista con el técnico de cosecha). Dicho dialogo previo permitió realizar la Fase 2 (Análisis por parte del equipo de desarrollo) lo que ayudo a los analistas y desarrolladores de la presente investigación (tesistas – SCRUM developer's) establecer el entendimiento del problema. Obteniendo así seis requerimientos funcionales: 1) Gestionar usuario, 2) Gestionar hacienda 3) Gestionar hectárea 4) Gestionar cultivo y 5) Gestionar control. Elaborado a partir del Anexo 1, denominado especificación de requisitos de software (ERS).

3. Historias de usuario (User Stories). - En la Figura 35 numeral 3 se puede visualizar la tercera parte de la metodología propuesta la cual utiliza lo analizado en etapa 2 (Visión) como entrada, es decir se ordenó formalmente los requerimientos funcionales llenando la plantilla correspondiente, para de esta manera guiar a los desarrolladores (tesistas - SCRUM developer's) en la definición de lo que se debe programar tomando en consideración el identificador, nombre de la historia de usuario, puntos de estimación, programador responsable y descripción. Lo cual permite administrar de manera óptima las actividades existentes dentro del proyecto. Cabe

destacar que se generaron cinco historias de usuario detallándose cada una de estas en el Anexo 2 (Historias de usuario).

4. Lista de historias de usuario (Product backlog). - En la Figura 35 numeral 4 se puede observar la cuarta etapa del presente marco de trabajo que utiliza como entrada lo analizado en la etapa 3 (Historias de usuario) de la cual se extrae el identificador y nombre que son colocados en la plantilla detallada en el Anexo 3 numeral 13 (Lista de historias de usuario). Con todos estos datos los desarrolladores (tesistas – SCRUM developer's) analizan la fecha en la que se iniciará las tareas, su posible fecha de finalización, estimación del número de horas que tomará desarrollarla, el estado y el número de sprint.

5. Desarrollo de la historia de usuario (Sprint). - En la Figura 35 numeral 5 se puede observar el sprint, el cual está conformado de varias historias de usuario que se las escoge según lo listado en la etapa 4 (Product backlog). Hay que tener en cuenta que cada sprint de este proyecto se establece con un tiempo máximo para realizar las tareas de 15 días. De esta manera el equipo de desarrollo conformado por David Tamayo y Cristian Tapia crean tareas de cada historia de usuario para así proceder a su programación.

6. Resultados de las historias de usuario (Sprint result). - En la Figura 35 numeral 6 se visualiza el Resultado de lo analizado en la etapa 6 (sprint) detallada anteriormente. Este es un requerimiento funcional ya programado por el SCRUM developer (David Tamayo y Cristian Tapia) y puesto en marcha dando solución a la especificación de requisitos software (ERS) Anexo 1 (Requerimientos Software).

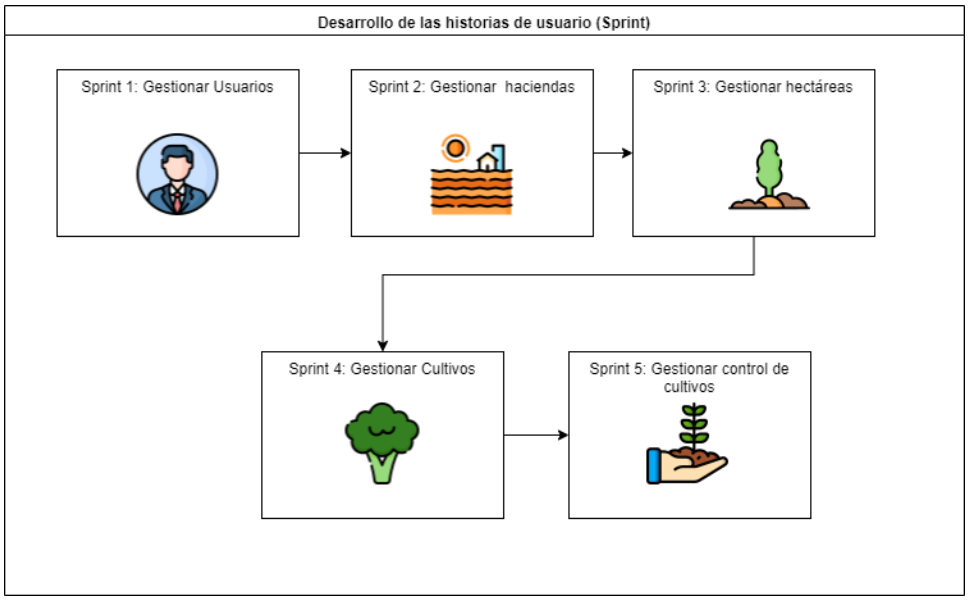
7. Retrospectiva del sprint (Sprint Retrospective). - En la Figura 35 numeral 7 se muestra la última etapa de la metodología SCRUM siendo este el indicador de cómo se desarrolló el proyecto de cálculo de grados días en relación al tiempo y si se llega a cumplir la meta en el tiempo establecido es por ello que se aplicó un gráfico denominado burndown chart para representar el avance del proyecto en función del tiempo.

Como se visualiza en la Figura 36 es importante detallar los Sprint (Desarrollo de las historias de usuario) existentes en el sistema de cálculo de grados día para una

plantación de brócoli dando a conocer todas las tareas que corresponde a cada sprint conjuntamente con avance en el transcurso del tiempo. A continuación, se detalla los cinco sprints que posee el presente sistema software:

Figura 36

Desarrollo de las historias de usuario



Nota. En esta figura se muestra los diferentes sprint que se desarrollaron para completar la elaboración del sistema.

1. **Sprint 1: Gestión de usuarios**

El sprint 1 está enfocado en la gestión de usuarios, la cual permite registrarse en el sistema para de esta manera poder acceder a la aplicación mediante el uso de credenciales asegurando así la información. El usuario ingresado podrá cerrar su sesión actual en cualquier momento, también se puede mencionar que los atributos de la tabla usuario contienen datos como: nombre, correo, cedula, rol y clave. Dicha información es obtenida de la historia de usuario H001 que se encuentra Anexo 2 (Historias de usuario), con estas es posible crear la lista de tareas con su respectivo avance la cual esta detallado en el Anexo 4 (Pila de sprint) numeral 2.

Resultado de sprint

En el presente sprint se desarrolló el login (inicio de sesión) o ingreso a la aplicación móvil (Figura 37) y web (Figura 38). El inicio de sesión es un requisito establecido en el documento de especificación de requisitos de software (Anexo 1,

índice 2.2). En las figuras mencionadas anteriormente se muestra el resultado de este sprint.

Figura 37

Resultado sprint 1 web

Nota. En esta figura se muestra la vista Web, resultante del desarrollo del primer sprint

Figura 38

Resultado sprint 1 móvil

Nota. En esta figura se muestra la vista Móvil, resultante del desarrollo del primer sprint

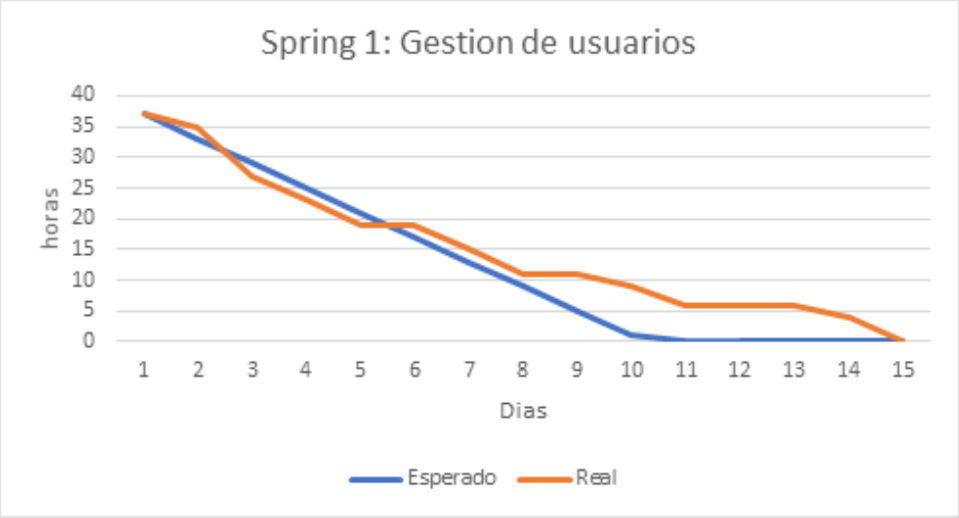
BurnDown chart

Se presenta en la Figura 39, el avance del sistema software con un gráfico BurnDown Chart el mismo que refleja el tiempo de desarrollo ideal vs el real, por lo cual

se puede observar que se terminó el desarrollo de está funcionalidad en el tiempo establecido.

Figura 39

BurnDown chart gestión usuarios



Nota. En esta figura se muestra la gráfica del tiempo de desarrollo ideal versus el tiempo de desarrollo real del primer sprint

2. Sprint 2: Gestionar hacienda

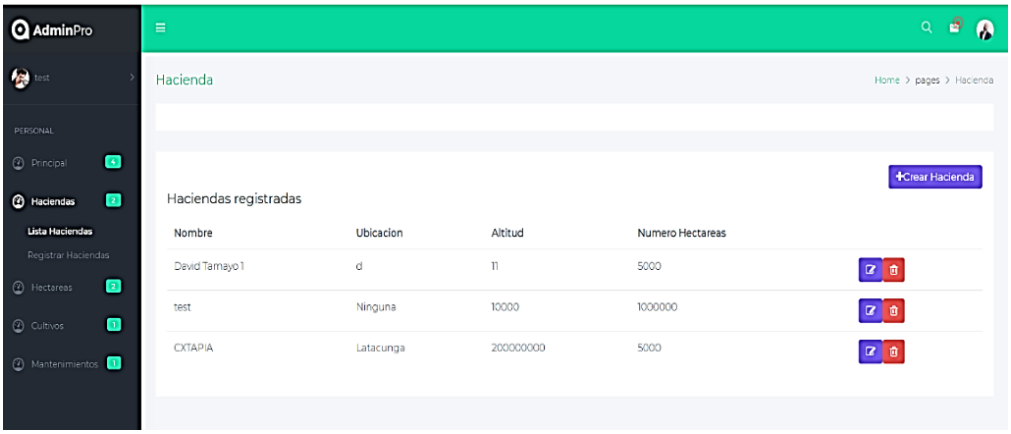
El sprint 2 está enfocado en la gestión de haciendas ingresadas, es decir se podrá crear, editar y eliminar por el administrador. También se podrá listar todas las haciendas existentes por todos los usuarios que se encuentren identificados y tengan asignado un rol. La hacienda contiene los siguientes datos: nombre, ubicación, altitud y número de hectáreas. Se detalla todo esto más a profundidad en la historia de usuario H002 que se encuentra Anexo 2, con estas historias de usuario es posible crear la lista de tareas con su respectivo avance siendo especificado en el Anexo 4 (Pila de sprint) numeral 3.

Resultado de sprint

En la Figura 40 y Figura 41 se puede observar las opciones de gestión de haciendas, las cuales pueden ser creadas, listadas, editadas o eliminadas, tanto en la aplicación web como en la móvil, así dando respuesta al requerimiento funcional del presente sprint.

Figura 40

Resultado sprint 2 web



Nota. En esta figura se muestra la vista Web, resultante del desarrollo del segundo sprint

Figura 41

Resultado sprint 2 móvil

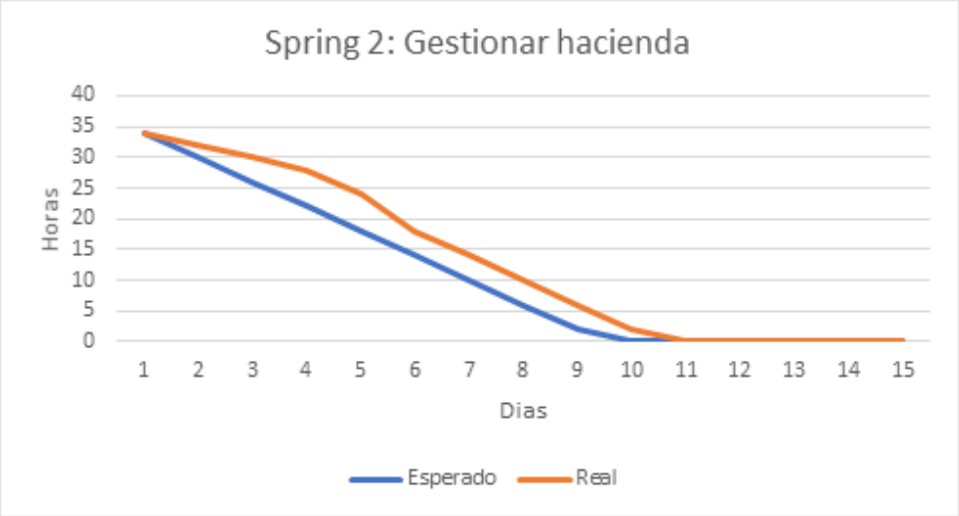
Nota. En esta figura se muestra la vista móvil, resultante del desarrollo del segundo sprint

BurnDown chart

En la Figura 42 se ve representado el avance del sistema software con un gráfico BurnDown Chart, visualizando la culminación del sprint antes de lo planeado.

Figura 42

BurnDown chart Gestionar haciendas



Nota. En esta figura se muestra la gráfica del tiempo de desarrollo ideal versus el tiempo de desarrollo real del segundo sprint

3. Sprint 3: Gestionar hectáreas

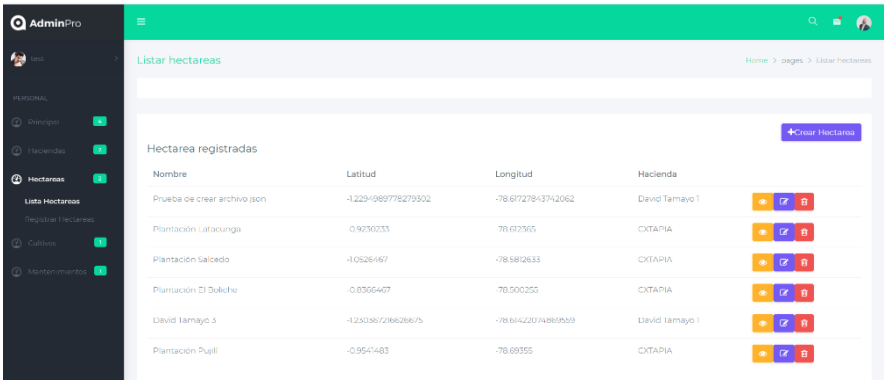
El presente sprint tiene por objetivo la gestión de hectáreas ingresadas, pudiendo así crear, editar o eliminar los parámetros por medio del usuario administrador, también se podrá listar todas las hectáreas existentes por todo tipo de usuario que se encuentre identificado, el parámetro hectárea contiene atributos como: nombre, hacienda, ubicación (latitud, longitud) dicha información se encuentra detallada en el Anexo 2 historia de usuario H003, con estas es posible crear la lista de tareas con su respectivo avance la cual esta detallado en el Anexo 4 (Pila de sprint) numeral 4.

Resultado sprint

En la Figura 43 y Figura 44 se representa los resultados de este sprint; mostrando interfaces que permiten crear, listar, editar o eliminar las hectáreas desde cualquier entorno.

Figura 43

Resultado de sprint 3 web



Nota. En esta figura se muestra la vista Web, resultante del desarrollo del tercer sprint

Figura 44

Resultado de sprint 3 móvil

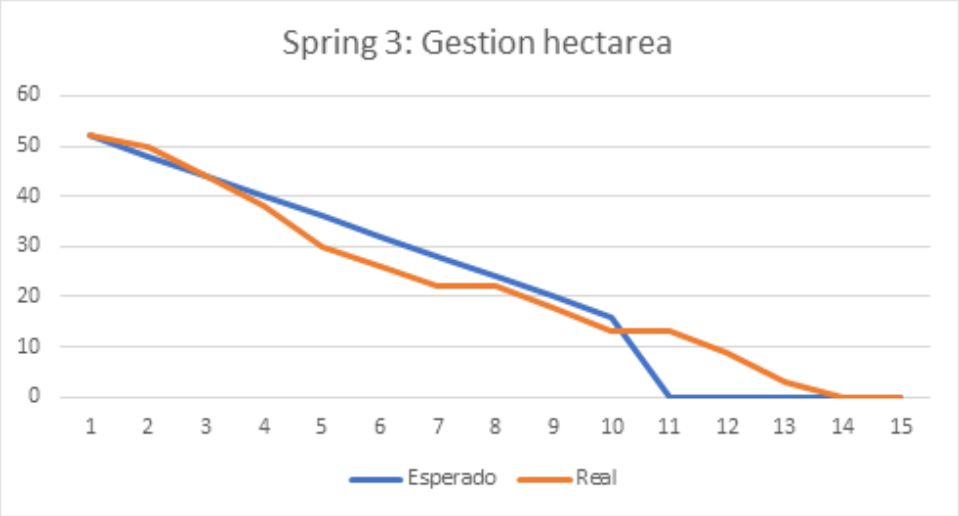
Nota. En esta figura se muestra la vista móvil, resultante del desarrollo del tercer sprint

BurnDown chart

Observando la Figura 45, la cual contiene la gráfica BurnDown Chart de avance del sistema software, refleja un tiempo estimado de desarrollo muy cercano a lo establecido en la planificación del sprint.

Figura 45

BurnDown chart Gestión hectárea



Nota. En esta figura se muestra la gráfica del tiempo de desarrollo ideal versus el tiempo de desarrollo real del tercer sprint

4. Sprint 4: Gestionar de Cultivos

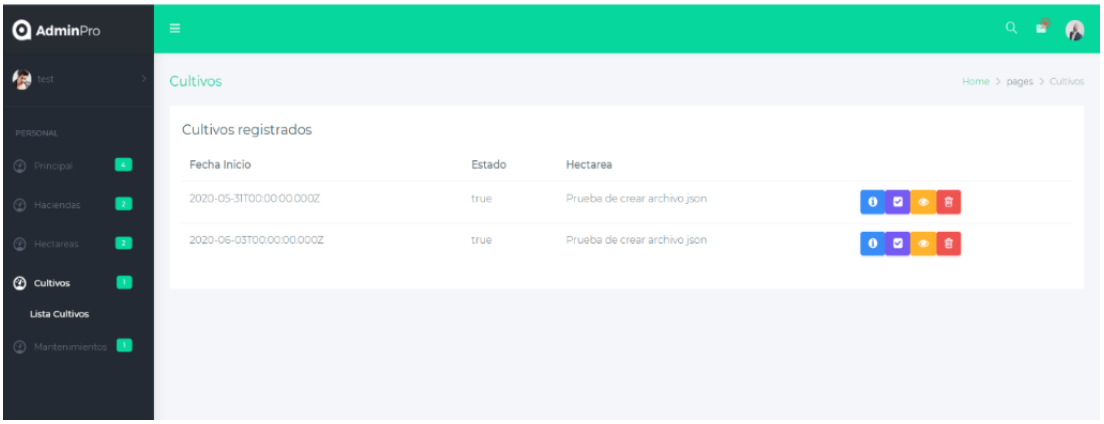
El sprint 4 tiene por objetivo la gestión de los cultivos ingresados, con la posibilidad de crear, editar o eliminar los mismos, también se podrá listar todos los cultivos existentes por cualquier tipo de usuario que se encuentre identificado, el cultivo contiene datos de fecha y estado, se podrá también visualizar la lista de cultivos existentes en cada hectárea. Las predicciones pueden ser accedidas por cualquier miembro del sistema, teniendo dos tipos de predicción: la predicción de 3 meses y la predicción que se actualiza cada 5 días estos requerimientos se pueden observar en la historia de usuario H004 encontrada en el Anexo 2 (Historias de usuario), con estas historias de usuario es posible crear la lista de tareas con su respectivo avance siendo especificado en el Anexo 4 (Pila de sprint) numeral 5.

Resultado sprint

Los resultados del sprint 4, son visibles en la Figura 46 y Figura 47. El cuál hace posible el crear, listar, editar, eliminar y realizar el control de un cultivo, estableciendo la predicción de cosecha del mismo.

Figura 46

Resultado sprint 4 web



Nota. En esta figura se muestra la vista Web, resultante del desarrollo del cuarto sprint

Figura 47

Resultado sprint 4 móvil

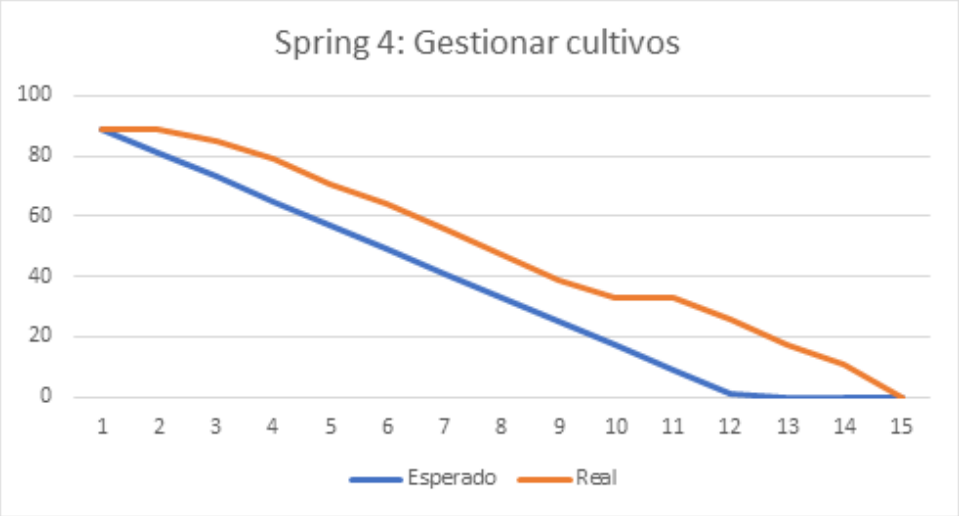
Nota. En esta figura se muestra la vista móvil, resultante del desarrollo del cuarto sprint

BurnDown chart

A continuación, en la Figura 48 se observa el avance del sprint, gracias a la representación de la gráfica Burndown Chart, este indica la existencia de un retraso en el desarrollo, sin embargo, se cumplió con el tiempo establecido.

Figura 48

BurnDown chart gestión cultivos



Nota. En esta figura se muestra la gráfica del tiempo de desarrollo ideal versus el tiempo de desarrollo real del cuarto sprint

5. Sprint 5: Gestionar control de cultivos

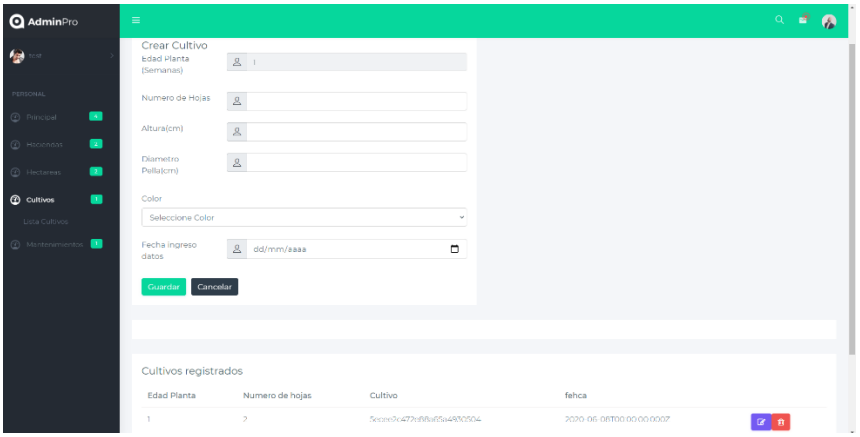
El sprint 5 se basa en la gestión del control de cultivos, permitiendo crear, editar o eliminar cada uno de estos, también se podrá listar todos los controles existentes por todo usuario que se encuentre identificados, el control del cultivo contiene los siguientes datos: Edad de la planta, número de hojas, altura, color, diámetro de la pella y fecha, las gráficas que se mostrarán en la interfaz, permitirán saber el estado de la planta con relación a datos ideales, mostrando así el monitoreo del número de hojas, tamaño de la planta y temperaturas máximas y mínimas, la historia de usuario correspondiente al presente sprint se encuentra en el Anexo 2 tabla H005, con estas es posible crear la lista de tareas con su respectivo avance la cual esta detallado en el Anexo 4 (Pila de sprint) numeral 6.

Resultado sprint

Tanto la Figura 49 como la Figura 50 indica el resultado del sprint 5, el cual se encarga de la gestión del control de plantas, mostrando gráficas y reportes del mismo.

Figura 49

Resultado sprint 5 gestión control web



Nota. En esta figura se muestra la vista Web, resultante del desarrollo del quinto sprint

Figura 50

Resultado sprint 5 gestión control móvil

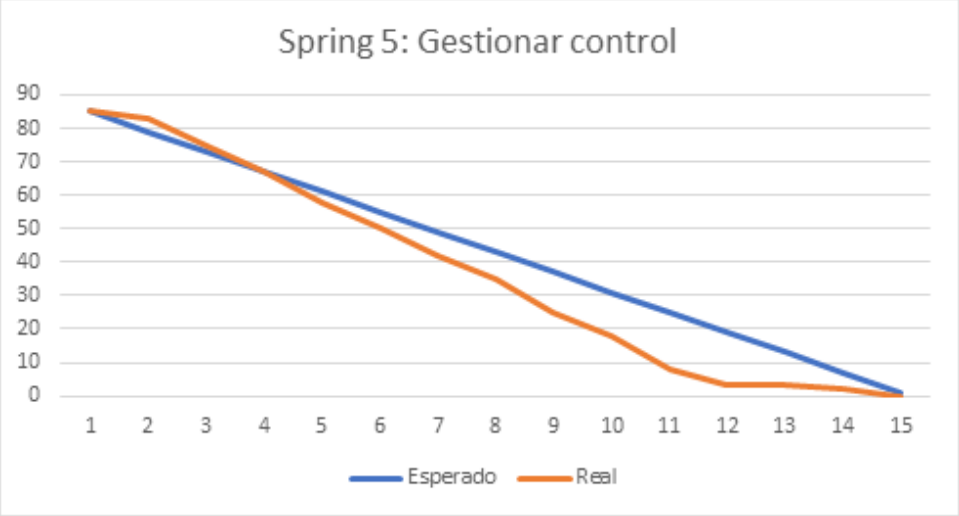
Nota. En esta figura se muestra la vista móvil, resultante del desarrollo del quinto sprint

Burndown chart

En el presente sprint se muestra que el tiempo estimado tuvo buenos resultados (Figura 51); ya que es muy parecido al tiempo ideal de desarrollo del sistema software.

Figura 51

Burndown chart gestión control



Nota. En esta figura se muestra la gráfica del tiempo de desarrollo ideal versus el tiempo de desarrollo real del quinto sprint

3.3.3. Desarrollo del modelo de predicción.

Para realizar el modelo de predicción se usó la metodología Box-Jenkins la cual tiene diversos pasos a seguir que ayudan con el desarrollo e implementación en tiempos óptimos y con errores mínimos. Es por ello que se describen a detalle a continuación los pasos a seguir para poner en funcionamiento este algoritmo:

1. **Identificación del modelo. -** Como primer paso se localizó el modelo que más serviría para el propósito de este proyecto, en este caso es el predecir temperaturas en la zona de cultivo, por lo cual después de analizar todos los modelos existentes se concluyó por un modelo autorregresivo (AR). Al observar que AR tiene muchos algoritmos en su haber para varios propósitos, se identificó uno que era óptimo para este proyecto, este se denomina modelo autorregresivo de media móvil (ARIMA) el cual tiene antecedentes muy beneficiosos en el cultivo de papas y otro tipo de frutos, ya que es del tipo estacional (analiza

patrones de comportamiento) por lo cual este sería el modelo que encajaría perfectamente en el desarrollo de este sistema.

Como se observa en la Figura 52 para implementar el algoritmo ARIMA en Python es necesario agregar una librería del paquete statsmodels la cual contiene el modelo autorregresivo utilizado en el desarrollo de este proyecto.

Figura 52

Librerías utilizadas en Python

```
from pandas import read_csv
from statsmodels.tsa.arima_model import ARIMA
```

Nota. En esta figura se muestra las librerías utilizadas en Python

2. **Estimación de parámetros.** - Los parámetros que se ingresan a este algoritmo son las temperaturas máximas y mínimas históricas. Estos datos son necesarios para poder dar una predicción a futuro, dicha información es separada para ser procesada, conformando así dos grupos que son: 1) temperatura mínima y 2) temperatura máxima.

   Como se muestra en la Figura 53, se creó la función temp_min para extraer datos de temperatura mínima de nuestro dataset (conjunto de datos de temperatura máxima y mínima) el cual contiene 40000 datos suministrados por la API Open Weather Map, como siguiente paso se estandarizó la información para que no exista mucho ruido al momento de pasarlos al modelo de predicción. Una vez estandarizada dicha información se procedió a entrenar el modelo ARIMA con una configuración simple para de esta manera obtener 100 datos de temperatura mínima y máxima futura.

   Ya entrenado los datos se debe invertir la diferencia que fue creada con la estandarización de información para obtener valores reales, esto se logra recorriendo el array que contiene los 100 datos predichos y con esto poder realizar el cálculo de inversión, obteniendo así datos que concuerdan con la realidad.

Todo el procedimiento descrito en la estimación de parámetros es el mismo en cuanto a la extracción de datos futuros de temperatura máxima variando únicamente en la obtención de parámetros del dataset (en el cual se escoge 40000 datos de temperatura máxima).

Figura 53

Implementación de ARIMA

```
def temp_min():
    #Primero extraigamos solo la temperatura del conjunto de datos.
    uni_data = df['T (degC)']
    uni_data.head()
    TRAIN_SPLIT = 40000

    #Estandarizacion de los datos
    uni_train_mean = uni_data[:TRAIN_SPLIT].mean() #calculo de la media
    uni_train_std = uni_data[:TRAIN_SPLIT].std() #calculo de la descviacion estandar
    #aqui estandarizamos
    uni_data = (uni_data-uni_train_mean)/uni_train_std
    # diferencia estacional
    X = uni_data.values[:40000]
    days_in_year = 365
    differenced = difference(X, days_in_year)
    # entrenar el  modelo
    model = ARIMA(differenced, order=(7,0,1))
    model_fit = model.fit(disp=0)
    #STEP numero de dias a predecir
    forecast = model_fit.forecast(steps=100)[0]

    # invertir el pronóstico diferenciado en algo utilizable
    history = [x for x in X]
    day = 1
    respuesta = []
    for yhat in forecast:
        inverted = inverse_difference(history, yhat, days_in_year)
        print('Day %d: %f' % (day, inverted*uni_train_std + uni_train_mean))
        respuesta.append(inverted*uni_train_std + uni_train_mean)
        history.append(inverted)
        day += 1
    return respuesta
```

Nota. En esta figura se muestra el código necesario para implementar Python en el proyecto

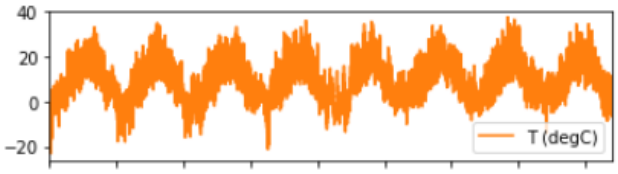
3. **Verificación. -** Una vez que se implementó este modelo en el proyecto, se procedió a realizar una prueba de eficacia del mismo, por lo que se hizo uso de unas métricas de evaluación que van acorde con el funcionamiento de ARIMA, estas se denominan error cuadrático medio (RMSE) y error absoluto medio (MAE). Con esta evaluación por medio de RMSE y MAE, se obtuvo un 93.15% de eficacia, indicando que el modelo ARIMA funciona correctamente. Se puede

observar las pruebas realizadas en la etapa 4 numeral 2 (Métricas del algoritmo de regresión).

4. **Pronóstico. -** Finalmente, se realizó un pronóstico real de la temperatura del sector donde se implementó el proyecto. En la Figura 54 se puede observar los resultados obtenidos de cada día.

Figura 54

Datos de temperatura



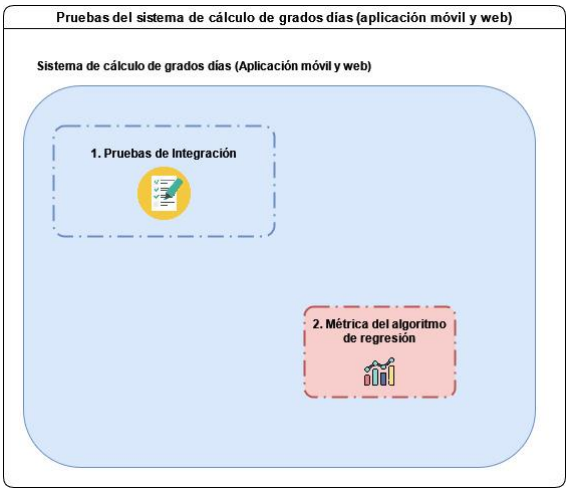
Nota. En esta figura se muestra los datos de temperatura generados por el modelo ARIMA

3.4. Etapa 4. Pruebas y métricas del sistema de cálculo de grados días (aplicación móvil y web).

Luego de haberse ejecutado la etapa 3 (desarrollo del sistema de cálculo de grados días aplicación móvil y web) que fue explicada en el capítulo 3 apartado 3.3. Es necesario que se efectuen las siguientes pruebas : 1) pruebas de integración y 2) métricas del algoritmo de regresión (ARIMA) como se observa en la Figura 55. Describiendose cada una de estas en los siguientes párrafos.

Figura 55

Pruebas y métricas efectuadas al sistema de cálculo de grados días



Nota. En esta figura se muestra las pruebas y métricas que se elaboraron en esta etapa

Según (Escobar, 2015) las pruebas requieren de un modelo para realizar las pruebas funcionales del software. En consecuencia, se requieren ejecutar cuatro etapas las cuales son las siguientes: 1) Especificación, 2) Planificación, 3) Ejecución y 4) Evaluación de resultados, razón por la cual se tomó en consideración el modelo mencionado. Con la finalidad de validar las especificaciones funcionales, lo que permite evaluar los incidentes a través de ciclos de prueba y sus correspondientes casos de prueba. Obteniéndose software de calidad minimizando lo fallos mínimos o inexistentes.

En la Figura 56 se muestra el proceso descrito por (Escobar, 2015) para efectuar pruebas funcionales detallándose cada una de sus etapas a continuación:

a) **Especificación. -** Etapa donde se analizan los problemas que tienen las funcionalidades que conforman el sistema para de esta manera realizar casos de prueba que ayudaran a determinar los fallos encontrados. Si los usuarios finales (clientes) están de acuerdo con este análisis se continua con el siguiente paso (Planificación), de lo contrario se procede a considerar los inconvenientes presentados por dichos usuarios.

b) **Planificación.** – Etapa don se planifica el tiempo y orden en el que se realizará las correcciones del sistema de software, estableciendo ciclos

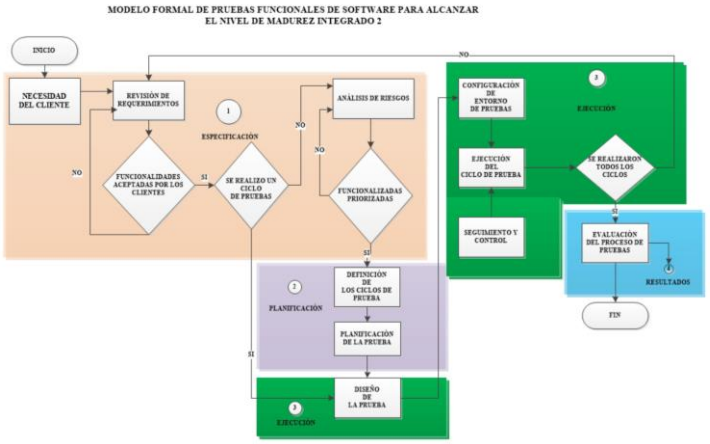
(en este caso prueba 1 y 2). De igual forma se establece el análisis de errores según su funcionalidad y riesgo.

c) **Ejecución.** – Etapa donde se realiza la corrección de los errores según lo planificado en la fase b (Planificación).

d) **Evaluación y resultados.** – Etapa donde se indica la cantidad de fallos corregidos al igual que la optimización que se obtuvo al ejecutar estas pruebas. Verificando así si el sistema desarrollado cumple con todos los parámetros indicados por los usuarios finales.

Figura 56

Modelo propuesto para Pruebas Funcionales de Software.



Nota. En esta figura se muestra el modelo para efectuar pruebas finuncionales para un sistema de softaware. Tomado de (Escobar, 2015)

A continuación se describe lo realizado en las pruebas y métricas que se efectuan al sistema de cálculo de grados días:

1) Pruebas de integración. - Pruebas que permiten evaluar y verificar el cumplimiento de las 29 funcionalidades que el sistema desarrollado debe poseer de acuerdo a lo establecido en la etapa 1 (Levantamiento de requisitos) como se visualiza en el anexo 1 (Especificación de requisitos software). Dichas pruebas se materializan en 22 casos de prueba los cuales fueron divididos en 13 casos para la aplicación móvil y 9 casos para aplicación web los cuales se pueden revisar en el Anexo 5 (Casos de prueba del sistema de cálculos de grados día).

Cabe destacar que la evaluación de la validez de las características del sistema de cálculos de grados día se realizó por los ingenieros agrónomos técnicos de cosecha Sídney Galarza y Efraín López (product owners). Permitiendo así determinar el grado de satisfacción al utilizar el sistema desarrollado. Para lo cual se requirió establecer una prioridad de corrección de errores de acuerdo a las necesidades de los usuarios finales. Dicho proceso de mantenimiento fue planificado semanalmente concluyéndose dicho procedimiento en 12 semanas como se puede visualizar en la Tabla 7.

Tabla 7

Proceso de corrección de los casos de prueba

CASOS DE PRUEBA - SEMANAS	1	2	3	4	5	6	7	8	9	10	11	12
Registro de usuario – Web												
Editar perfil de usuario – Móvil												
Cerrar sesión – Web												
Crear haciendas – Móvil												
Listar haciendas – Movil												
Editar haciendas – Movil												
Eliminar haciendas – Movil												
Crear hectáreas – Movil												
Listar hectáreas – Movil												
Editar hectáreas – Movil												
Eliminar hectáreas – Movil												
Crear cultivos – Movil												
Listar cultivos – Movil												
Editar cultivos – Movil												
Eliminar cultivos – Movil												
Visualizar producción cosecha 3 meses - Movil												
Visualizar producción cosecha 5 días - Movil												
Crear control – Web												

CASOS DE PRUEBA - SEMANAS	1	2	3	4	5	6	7	8	9	10	11	12
Listar control – Web												
Editar control – Web												
Eliminar control – Web												
Gráfica de control de cultivo - Web												

Nota. En esta tabla se muestra la planificación de corrección de fallos presentados

En la Figura 57, se puede visualizar la evaluación y verificación de las funcionalidades totales del sistema de cálculo de grados días para optimizar el ciclo productivo del brócoli, determinándose aquellos incidentes surgidos al utilizarlo y los casos de prueba generados que permitan las correcciones necesarias de los errores presentados. Comparándose los resultados obtenidos en la primera y segunda prueba permitiéndose determinar la integración del sistema según el uso y satisfacción de las necesidades de los usuarios finales. Obteniéndose 31 incidentes en la prueba1 generándose 22 casos de prueba para su respectiva corrección lo que permitió reducir el número de errores. Eliminándose un 87.1% de fallos (27 errores) manteniéndose un 12.9% restante (4 errores). Posteriormente se realiza la prueba 2, evaluando por segunda ocasión el sistema desarrollado en esta investigación. Reflejándose la reducción de 22 a 0 casos de prueba. Cabe destacar que se conservan 4 errores no prioritarios (ajenos a los desarrolladores) debido al versionamiento existente de las siguientes tecnologías: 1) Android 7.0, 2) Node.js 12, 3) Python 3.1 y 3) Flutter 1.22.0 cómo se puede observar en la Tabla 8.

Tabla 8

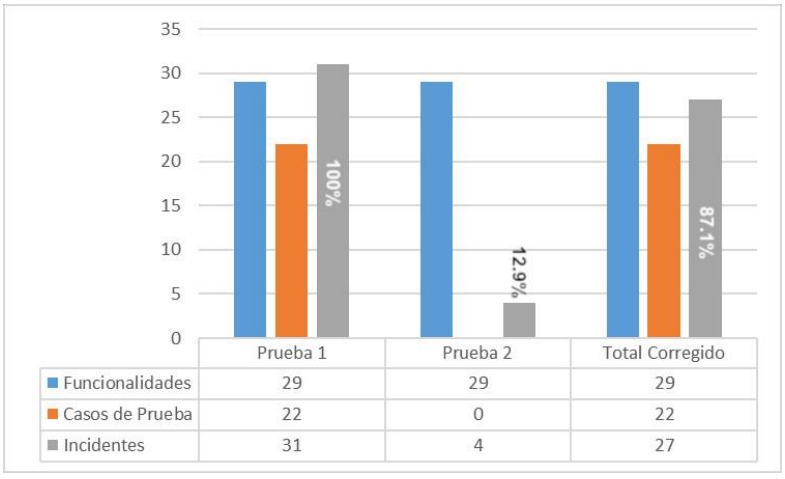
Incidentes presentados en varias tecnologías

Tecnología	Versión	Incidente no prioritario
Android	7.0	Al leer datos del GPS del Smartphone este se tarda en calibrarlo demorando así la consulta de dicha información.
Node.js	12	Cierre automatizado de cualquier sesión de usuario al cumplir un tiempo de terminado
Python	3.1	Los 5 a 10 minutos de tiempo de espera para generar una predicción.
Flutter	1.22.0	El gráfico seguimiento del desarrollo del cultivo no es actualizable en tiempo real cuando se tiene una sesión abierta.

Nota. En esta tabla se muestra los errores no prioritarios que conserva el sistema desarrollado en esta investigación

Figura 57

Funcionalidades, casos de prueba e incidentes del sistema de cálculo de grados día (aplicación móvil y web)

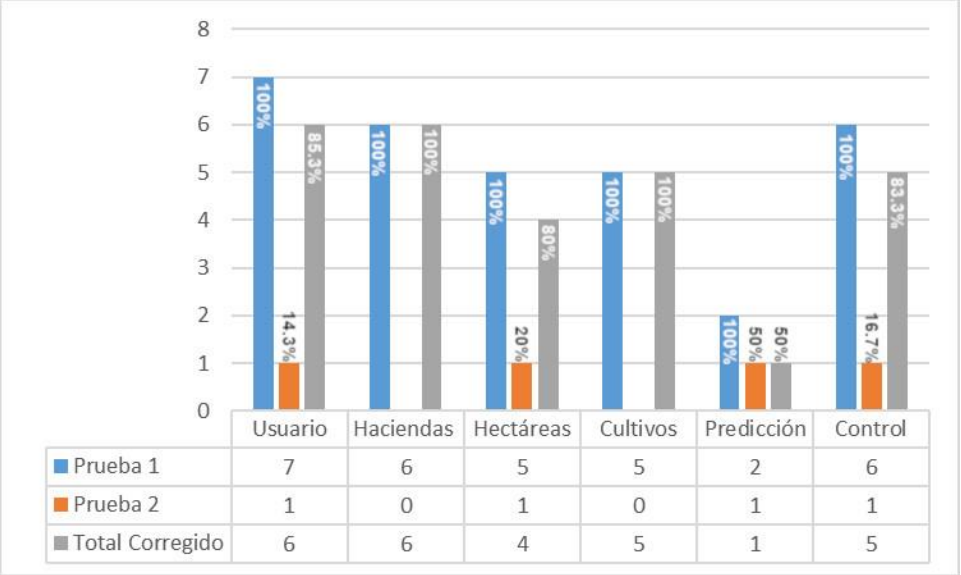


Nota. En esta figura se muestra la comparación de los datos resultantes de la primera y segunda prueba realizada al sistema de software

En la Figura 58, se reflejan los incidentes presentados por el software implementado, los cuales están clasificados según su funcionalidad. En la primera prueba realizada al sistema se puede evidenciar que existen 7 incidentes para la funcionalidad de usuarios, 6 para haciendas, 5 para hectáreas, 5 para cultivos, 2 para predicción y 6 para control. Esto disminuye drásticamente en la segunda prueba ejecutada al software ya que únicamente existe 1 acontecimiento para usuarios (14.3%), 1 para hectáreas (20%), 1 para predicción (50%) y 1 para control (16.7%). Concluyéndose que se logró suprimir varios errores para cada funcionalidad que conforma el sistema, siendo el porcentaje de corrección para cada una de estas el de: 85.3% para usuarios, 100% para haciendas, 80% para hectáreas, 100% para cultivos, 50% para predicción y 83.3% para control. Los fallos que se mantuvieron en este análisis son los tratados anteriormente en la Tabla 8. (Incidentes presentados en varias tecnologías).

Figura 58

Incidentes según su funcionalidad del sistema de cálculo de grados día (aplicación móvil y web)



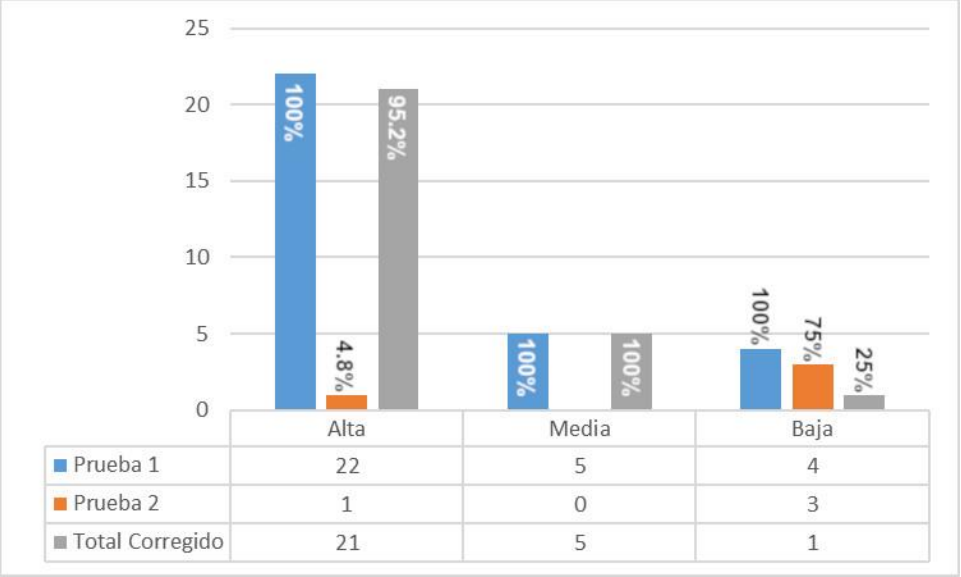
Nota. En esta figura se muestra los datos resultantes de la evaluación de incidentes según su funcionalidad

En la Figura 59 se puede observar el nivel de criticidad que tiene cada incidente presentado en las dos evaluaciones de funcionamiento ejecutadas al sistema para el

cálculo de grados días en una plantación de brócoli. La primera prueba se conforma de 22 fallos de alta prioridad, 5 de prioridad media y 4 de baja prioridad, mientras que en la segunda prueba efectuada existe 1 error de alta prioridad (4.8%) y 3 de baja prioridad (75%). Por lo cual se concluye que se redujo las incidencias en un 95.2% para altas, 100% para medias y 25% para bajas. Conservando únicamente los 4 errores tratados en este apartado y mencionados en la Tabla 8 (Incidentes presentados en varias tecnologías).

Figura 59

Nivel de criticidad de los incidentes del sistema de cálculo de grados día (aplicación móvil y web)



Nota. En esta figura se muestra los incidentes presentados según su nivel de criticidad

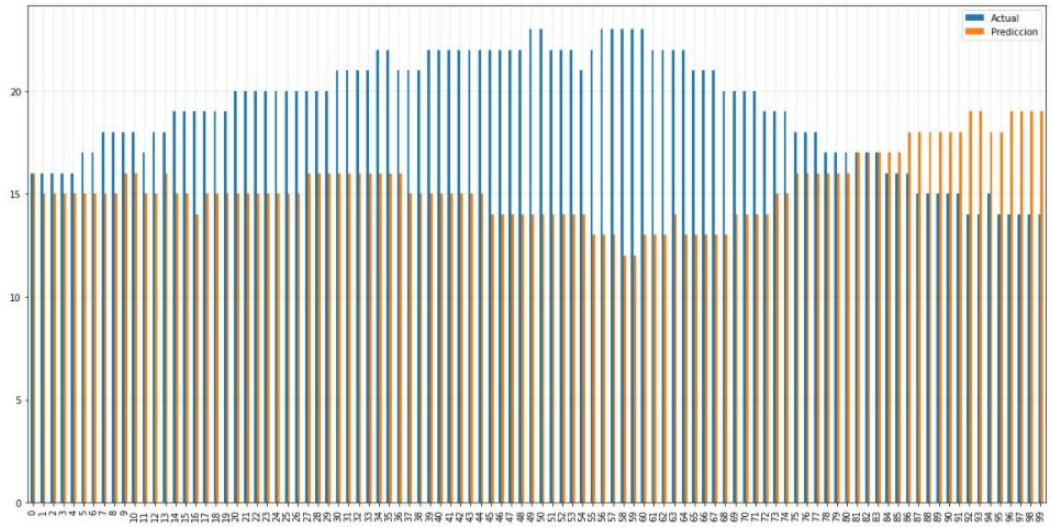
2) Métrica del algoritmo de regresión. - Evaluación que permite realizar una verificación de eficacia de la predicción suministrada por el algoritmo de regresión empleado en este proyecto, detallándose su aplicación en el capítulo 3, etapa 3 apartado 3.3.3. Para esto se requirió hacer uso de las métricas de evaluación denominadas como error absoluto medio y error cuadrático medio.

Para verificar la eficacia del método de predicción aplicado a este proyecto, fue necesario basarse en el trabajo de (Erik Giovany Montes Páez, 2016), donde se indica que se debe realizar como primer paso una comparación del pronóstico suministrado por ARIMA (datos de color naranja) versus datos climatológicos reales (datos de color

azul), el resultado se lo puede visualizar en la Figura 60. La diferencia entre estos dos grupos de información es de vital importancia para estimar un porcentaje de error por medio de métricas de evaluación, este procedimiento es detallado posteriormente.

Figura 60

Datos de predicción de prueba vs datos reales



Nota. En esta figura se muestra el diagrama de la comparación de los datos reales y los datos de prueba

Una vez comparados estos dos grupos de información, se procede a evaluar la eficacia del algoritmo ARIMA por medio de las métricas de error absoluto medio y error cuadrático medio, para ello se debe calcular la diferencia entre los datos de predicción de prueba y datos reales. Lo cual da como resultado un diagrama de dispersión para cada método de evaluación de la eficacia del modelo implementado (ARIMA).

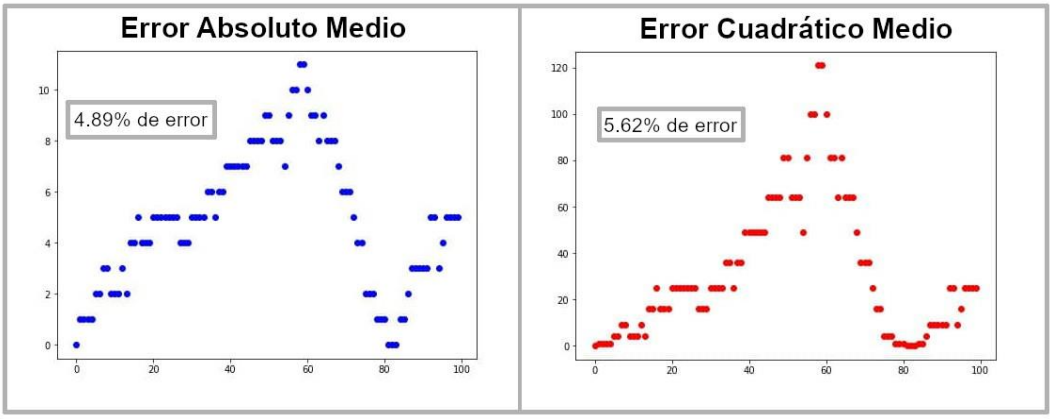
En la Figura 61 se puede visualizar la gráfica resultante de cada métrica utilizada, tomando el eje cartesiano x para mostrar el avance en días (100 en total) de estos diagramas de dispersión junto con el eje y que muestra el valor del rango de la temperatura (en grados centígrados). Como se observa a simple vista la distribución de cada punto es casi similar en los dos esquemas, variándose únicamente en posiciones iniciales y finales.

Numéricamente, la métrica de error absoluto medio muestra un porcentaje de error del 4.89% frente al 5.62% indicada por la métrica de error cuadrático medio, lo

cual refleja un fallo ínfimo de efectividad por parte del modelo ARIMA. Con el argumento descrito en el párrafo anterior (similitud de los diagramas de dispersión) junto con este, se concluye que el algoritmo de regresión implementado en esta investigación proporciona datos de confianza que servirán para optimizar el ciclo productivo del brócoli por medio del cálculo de grados días.

Figura 61

Diagramas de dispersión y porcentajes de error resultantes de las métricas de evaluación (Error Absoluto Medio y Error Cuadrático Medio)



Nota. En esta figura se muestra el diagrama de dispersión generado por las métricas de error absoluto medio y erros cuadrático medio junto con su porcentaje

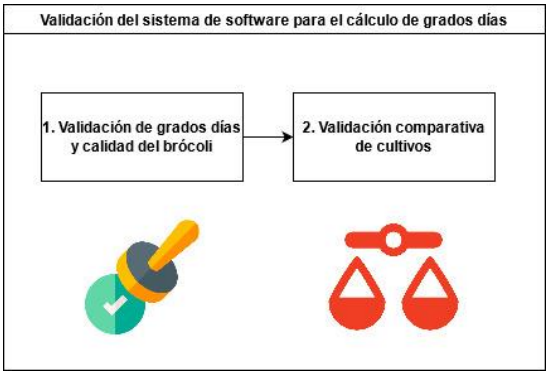
4. Validación del sistema de software para el cálculo de grados días

En este capítulo del presente proyecto de investigación se determina la validez y la efectividad del sistema desarrollado para el cálculo de grados día en una plantación de brócoli el cual busca optimizar el ciclo productivo de esta hortaliza por medio de la predicción o estimación de una fecha de cosecha idónea.

Como se puede visualizar en Figura 62, este capítulo cuenta con 2 etapas. Estas son: 1) Validación de grados días y calidad del brócoli, 2) Validación comparativa de cultivos.

Figura 62

Etapas de validación del sistema de software



Nota. En esta figura se muestra las fases que conforman el proceso de validación del sistema de software

4.1. Validación de grados días y calidad del brócoli

El proceso de cultivo de brócoli tiene un tiempo aproximado de 3 meses, en consecuencia, este es un factor importante para saber cuándo la cosecha estará preparada. Los actores principales del presente proyecto (agrónomos) tienen basto conocimiento sobre el tema.

Para el análisis de la calidad del brócoli se trabajó con dos grupos de esta hortaliza. Para el primer grupo se efectuó un seguimiento según lo previsto por el sistema de cálculo de grados día mientras que en el segundo se realizó una

monitorización manual de las plantas. En la Figura 63 se puede observar el cultivo donde se aplicó el seguimiento del software desarrollado.

Figura 63

Plantación de brócoli en Lasso

Nota. En esta figura se muestra el lugar donde se realizó las pruebas y la implementación del sistema desarrollado

a) Grados Días

Teniendo en cuenta los datos históricos suministrados por parte de los ingenieros agrónomos técnicos de cosecha Efraín López y Sidney Galarza, se observó 5 grupos de brócoli que ya fueron cosechados de la forma tradicional (cgdm) comparándolos con cinco grupos de cosecha que fueron sometidos al sistema de cálculos de grados día (cgda). Los datos de comparación de estos grupos se los puede visualizar en la Tabla 9. Toda esta información se la examinará más a profundidad posteriormente.

Tabla 9

Comparación del método manual con el sistema de grados días

Manual			Sistema grados días			diferencia días	diferencia grados día	% días	% cgd
grupo	días	cgdm	Cosecha	días	cgda				
1	130	1020	1	91	910	39	110	30%	10,78%
2	130	1040	2	93	911	37	129	28%	12,40%
3	130	1015	3	90	909	40	106	31%	10,44%
4	130	1030	4	95	910	35	120	27%	11,65%
5	130	1025	5	93	914	37	111	28%	10,83%

Nota. En esta tabla se muestra la comparación de resultados entre el sistema manual y el software para el cálculo de grados días

En la Figura 64 se analiza la comparación del tiempo de crecimiento (en días) de las agrupaciones de brócoli que cuentan con el uso del sistema de software y sin el uso del mismo. Como se puede observar los grupos con monitoreo manual (grupo 1, 2 3, 4 y 5) tienen 130 días fijos de crecimiento, mientras que la agrupación que cuenta con seguimiento por parte del sistema de cálculo de gradados días tiene un desarrollo de: 91 días (70%) para el grupo 1, 93 días (72%) para el grupo 2, 90 días (69%) para el grupo 3, 95 días (73%) para el grupo 4 y 93 días (72%) para el grupo 5. Concluyéndose que el grupo 1 se optimizó en un 30% (39 días), el grupo 2 en un 28% (37 días), el grupo 3 en un 31% (40 días), el grupo 4 en un 27% (35 días) y el grupo 5 en un 28% (37 días). Mejorándose así el ciclo productivo del brócoli, puesto que mientras más tiempo tome cosechar los cultivos de esta hortaliza, más recursos son gastados e incluso no se llega a planificar un tiempo de entrega preciso para los compradores de este fruto que residen fuera del país, afectando a su exportación.

Figura 64

Días de desarrollo y cosecha de los grupos de brócoli

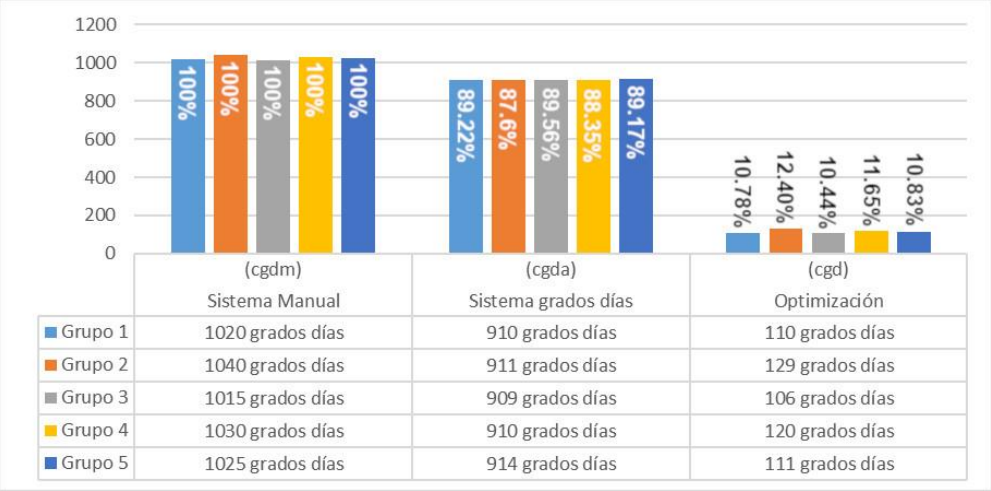
	Sistema Manual	Sistema grados días	Optimización
Grupo 1	130 días	91 dias	39 días
Grupo 2	130 días	93 dias	37 días
Grupo 3	130 días	90 dias	40 días
Grupo 4	130 días	95 dias	35 días
Grupo 5	130 días	93 dias	37 días

Nota. En esta figura se muestra la comparación de los días de desarrollo del cultivo por parte del sistema manual y el de cálculo grados días

En la Figura 65, se representa la diferencia del cálculo de grados días efectuado por el método manual (cdgm) y por el sistema de software (cdga). Observándose que los grupos 1, 2, 3, 4 y 5 del monitoreo manual (cdgm) cuentan con cifras en gados días de 1020, 1040, 1015, 1030, 1025 respectivamente, mientras que los grupos 1, 2 3, 4 y 5 sometidos al seguimiento del sistema implementado en este proyecto (cdga), cuentan con cantidades (grados días) de 910 (89.22%), 911 (87.6%), 909 (89.56%), 910 (88.35%) y 914 (89.17%) correspondientemente. Obteniéndose una mejora del: 10.78% (110 grados días) para el grupo 1, 12.40% (129 grados días) para el grupo 2, 10.44% (106 grados días) para el grupo 3, 11.65% (120 grados días) para el grupo 4 y 10.83% (11 grados días) para el grupo 5. Resolviéndose que los conjuntos de brócoli sometidos al control del aplicativo desarrollado en esta tesis tienen una mayor calidad, puesto que mientras esta hortaliza más se acerque a los 909 grados días, mejores características y propiedades nutricionales tendrá.

Figura 65

Grados días de los grupos de brócoli



	(cgdm) Sistema Manual	(cgda) Sistema grados días	(cgd) Optimización
Grupo 1	1020 grados días	910 grados días	110 grados días
Grupo 2	1040 grados días	911 grados días	129 grados días
Grupo 3	1015 grados días	909 grados días	106 grados días
Grupo 4	1030 grados días	910 grados días	120 grados días
Grupo 5	1025 grados días	914 grados días	111 grados días

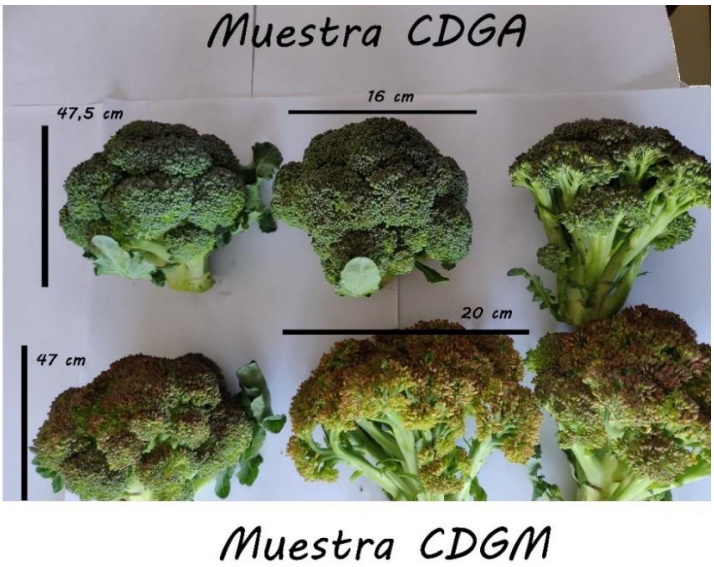
Nota. En esta figura se muestra la comparación de los grados días de desarrollo del cultivo por parte del sistema manual y del sistema de software

b) Calidad del brócoli

Con el propósito de analizar la efectividad del sistema de cálculo de grados día se analizaron 6 muestras de brócoli, esto se representa en Figura 66. En esta imagen se separan estos dos grupos siendo el de la parte de arriba la muestra sometida al sistema desarrollado y, en la parte de abajo la muestra manual.

Figura 66

Comparativa de muestras



Nota. En esta figura se muestra una comparativa física de muestras que se sometieron al sistema de software y al sistema manual

Estas muestras evidencian la mejora que tiene la implementación de este tipo de sistemas en los cultivos, ya que como se puede observar el grupo que se sometió a este software es más llamativo que el que se lo cosecho de manera tradicional. Todo el análisis de los parámetros mostrados anteriormente se los detalla en la Tabla 10.

Tabla 10

Análisis de muestras

Indicadores	Cálculo de grados días automático	Cálculo de grados días manual
Altura	El promedio de altura de estas muestras es de 47.5 centímetros	El promedio de la altura de estas muestras es de 47 centímetros
Diámetro	Se observa que presentan un diámetro promedio de 16 cm, de igual manera se pude notar a simple vista que las inflorescencias se encuentran agrupadas de una manera que indica una mejor calidad	Se indica un diámetro promedio de 20 cm, hay que mencionar que las inflorescencias presenciadas en estas muestras se encuentran un poco más separas por lo que indica que el brócoli no tiene tanta calidad

Indicadores	Cálculo de grados días automático	Cálculo de grados días manual
Color y sensación	Se muestra un color más llamativo lo que indica que tiene más propiedades nutricionales y un mejor sabor	Estas muestras indican un color más amarillento, esto quiere decir que contienen menores propiedades nutricionales y un peor sabor

Nota. En esta tabla se muestra la comparación de características entre el método tradicional y el sistema para el cálculo de grados días

4.2. Validación comparativa de cultivos

Para esta etapa de validación se comparó los resultados que se obtuvieron del cultivo donde se implementó el sistema de cálculo de grados días de este proyecto con los del sistema tradicional y con la información recabada por (Ayala, 2010). Este análisis se lo realizó gracias a la experiencia de los ingenieros agrónomos técnicos de cosecha Sidney Galarza y Efraín López, los cuáles certificaron la validez de estos resultados, esto se puede evidenciar en el Anexo 6, índice 2 (Certificación). Todo este análisis se lo puede observar en la Tabla 11.

Tabla 11

Comparativo entre diferentes técnicas de cosecha

N°	Elemento de análisis	Sistema tradicional	Según AYALA	Sistema grados días
1	Tiempo de desarrollo	130 días	85 a 90 días	+/-90 días
2	Numero de hojas del brócoli	14-20	17	16-18
3	Altura del brócoli	47cm	48cm	48cm
4	Diámetro del brócoli	12-20 cm	11.4–18.8 cm	11-17 cm
5	Color de la pella	Verde oscuro	Verde oscuro	Verde oscuro
6	Peso del brócoli	160 – 200 gr	151 - 270 gr	160 – 200 gr
7	Temperatura umbral del brócoli	4 °C	4 °C	4 °C
8	Formación de pella del brócoli	612 grados día	632.9 grados día	609 grados día

Nota. En esta tabla se muestra los resultados en comparación de la investigación de Ayala, el método tradicional y el sistema de grados días.

El análisis comparativo muestra que sin afectar todos los parámetros mencionados la implementación del sistema es más eficiente en cuanto al tiempo de desarrollo del brócoli, haciendo hincapié en que no afecta al cultivo; pero al cosecharlo al tiempo que es debido se llega a un producto de mayor calidad con mayores propiedades nutricionales.

Los documentos de entrega y recepción del sistema para el cálculo de grados días en una plantación de brócoli se ven reflejados en el Anexo 6, índice 3 (Acta entrega - recepción). Esto certifica que el proyecto desarrollado cumple con todos los estándares de calidad impuestos por los ingenieros agrónomos técnicos de cosecha Sidney Galarza y Efraín López.

La carta de aceptación evidenciada en el Anexo 6, índice 4 (Acta de validación y aceptación) indica la validez de los resultados obtenidos por el sistema de software (sistema para el cálculo de grados días en una plantación de brócoli), los cuales fueron tratados y analizados en este capítulo.

5. Conclusiones.

- Se desarrollo e implemento satisfactoriamente el sistema software para el cálculo de grados día en una plantación de brócoli en la Empresa Ecofroz S.A. en el transcurso del cuarto trimestre del año 2019". Optimizando el ciclo productivo reflejando su calidad en el incremento de exportaciones debido al mejoramiento que se obtuvo en su tiempo de cosecha.

- El marco teórico permitió comprender las bases conceptuales necesarias sobre tecnologías de front-end, back-end, bases de datos no relacionales, modelo de aprendizaje supervisado y Apis meteorológicas que posibilitaron profundizar las temáticas necesarias para la consecución del tema propuesto en la presente investigación.

- La descripción de características y las funcionalidades por medio del estándar IEEE-830 posibilitó la construcción del documento de especificación de requisitos software de la presente investigación. Actividad de suma importancia por parte de los ingenieros de software. Permitiendo establecer un acuerdo entre el desarrollador y el usuario final.

- La descripción de características y las funcionalidades por medio del estándar IEEE-830 permite establecer una base clara de las necesidades de la empresa y su relación con las características esenciales para el desarrollo del sistema software propuesto mejorando el tiempo de cosecha de los cultivos de brócoli.

- El modelo 4+1 permite construir de manera fácil y adecuada una arquitectura de software acorde a las exigencias agrícolas propuestas por expertos agrónomos de la presente investigación. Así como también da cabida a implementar módulos acordes al internet de las cosas por su compatibilidad con todo tipo de software facilitando la implementación de módulos diversos que utilicen API's de estación meteorológica y Smartphones.
- El modelo 4+1 facilita el desarrollo de complementos de la presente investigación tomando en consideración aspectos de seguridad, transaccionalidad, interoperabilidad, escalabilidad y licenciamiento. Debido a la simplicidad que prestan

los diagramas UML para integrar funcionalidades, peticiones de información conexiones ente módulos internos y externos del sistema.

- El uso de la metodología SCRUM es de gran utilidad para el desarrollo de la presente investigación debido a su optimización del tiempo, agilidad en el desarrollo de tareas y capacidad de llevar una documentación clara. Logrando así obtener un sistema de calidad.

- El modelo de madurez de pruebas integrado ayudó a ejecutar ciclos de evaluación que permitió el análisis de casos de prueba con los que se logró corregir fallos clave en el sistema desarrollado en este proyecto entregándose un software confiable al cliente.

- El procesamiento de información de los cultivos de brócoli que utilizaron al sistema desarrollado en el proyecto desarrollado, en comparación con los cultivos tradicionales que efectúan un monitoreo manual permitió mejorar el tiempo de desarrollo en un 29% reduciéndose sus días de avance de 130 a 93 en promedio, gracias a esto también se logró un incremento de calidad en un 9.35% debido a su reducción de grados días de 1025 al 910.

6. Recomendaciones

- Se recomienda aplicar el cálculo de grados día a las plantaciones no solamente brócoli sino también a otros tipos de cultivos. Lo que permitirá establecer un tiempo óptimo de cosecha, mejorando su calidad y con esto sus beneficios nutritivos.

- Se recomienda utilizar el framework Node.js para el desarrollo de la lógica de sistemas software por disponer de varias herramientas que facilitan la comunicación entre distintas tecnologías y ayudan en el desarrollo de las funcionalidades que están descritas en los documentos de especificación de requerimientos software.

- Se recomienda usar el framework Angular para el diseño de las interfaces que son utilizadas por aplicaciones web debido a su compatibilidad con todos los navegadores web y su facilidad de uso mejorando la experiencia de usuario.

- Se recomienda usar el framework Flutter para el desarrollo de aplicaciones móviles debido a su facilidad de desarrollo y baja curva de aprendizaje.

- Se recomienda diseñar la arquitectura 4+1 por que posee una guía de referencia que permite establecer una ruta para la construcción de un sistema software. Facilitando a los programadores cumplir con el desarrollo de las funcionalidades establecidas en el documento de especificación de requisitos software es decir entender de mejor manera como se construirá el proyecto.

- Se recomienda utilizar la metodología Scrum para el desarrollo de sistemas de software debido a que facilita la creación de tareas y asignación de actividades de desarrollo a los programadores controlando sus tiempos, resultados y detección temprana de errores.

7. Bibliografía

Agenjo, O. A. (2017). *Universidad Politécnica de Madrid.* Recuperado de ttp://oa.upm.es/48075/1/TFG_OSCAR_ARTURO_GARRIDO_AGENJO.pdf

Aharon Cuevas, R. R. (2015). *Arquitectura de Software*. Recuperado el 13 de marzo de 2020, de https://sites.google.com/site/softwarearchitecturedocument/5-otras-vistas/5-1-vista-fisica

Alava, N. (2015). *Ingenieria en Software*. Recuperado de https://ingenieriaensoftwarenathalyalava.wordpress.com/2015/04/25/modelos-de-procesos-prescriptivos/

Alcaraz, M. (2014). Internet de las cosas. *Universidad Católica Nuestra Señora de la Asunción*, 2-3.

Alegsa, L. (2020). *Definición de aplicación web.* Recuperado el 16 de febrero de 2020, de https://www.alegsa.com.ar/Dic/aplicacion_web.php

Almeida Granja, J. P. (2012). *Elaboracion de un plan estrategico para la reactivacion y reposicionamiento de la empresa Fresh Food CIA. LTDA. en el sector agroindustrial del Ecuador.*

Almenara, C. (1992). Diseño de software informático. . *Universidad de Sevilla.*

Aluja, T. (2001). La minería de datos, entre la estadística y la inteligencia artificial. *Qüestiió: quaderns d'estadística i investigació operativa*, 479-498.

Aprendeia. (2019). *Aprendeia*. Recuperado de https://aprendeia.com/todo-sobre-aprendizaje-no-supervisado-en-machine-learning/

APROFEL. (2016). *Estudio de impacto del sector brocolero en el marco de las negociaciones del tratado de libre comercio.* Quito.

Arnold, G. W. (1959). Radiation effects in silica at low temperatures. *Physical Review*.

Ayala, P. V. (2010). Efecto de la aplicacion de diferentes herbicidas sobre el rendimiento de brocoli. *Univerisdad de Chile falcultad de ciencias agronomicas*.

Bahit, E. (2012). Scrum y eXtreme Programming para programadores.

Banafa, A. (2016). *OpenMind BBVA*. Recuperado de https://www.bbvaopenmind.com/tecnologia/mundo-digital/que-es-el-aprendijaze-profundo/

Barros López, J. G. (2010). Atlas climatológico del Ecuador (Bachelor's thesis, QUITO/EPN/2010).

Benítez, L. F. (2017). LA IMPLEMENTACIÓN DEL MOBILE MARKETING COMO HERRAMIENTA MULTIDISCIPLINAR EN EL SECTOR TURÍSTICO Y AEROPORTUARIO. Universidad de Malaga.

Bernd , B., & Allen, H. (2014). *Object-Oriented Software Engineering.* Ciudad de Mexico: Editorial Prentice Hall.

Bianco, V. (2011). *Orticoltura.* Bologna, Italia: Pàtron Editore.

Bonaparte, U. J. (2012). Proyectos UML Diagramas de clases y aplicaciones Java en NetBeans 6.9. 1. Proyecto. *Universidad Tecnológica Nacional–UTN, Facultad Regional Tucumán.*

Borja, J. (2015). Introducción a la agronomía.

Bree, R. T. (2016). Using Microsoft Excel to code and thematically analyse qualitative data: a simple, cost-effective approach. *All Ireland Journal of Higher Education*, 8(2).

Buto, P. O. (2018). Modelos Autorregresivos espaciales para la simulación y pronósticos de enfermedades desde condiciones climáticas. *Revista Cubana de meteorología*, 12(1).

Cabello, M. V. (2010). Introducción a las bases de datos relacionales. *Vision Libros.*

Calero, C. &. (2010). *Calidad del producto y proceso software.* Editorial Ra-Ma.

Callejas-Cuervo, M. A.-A.-C. (2017). Modelos de calidad del software, un estado del arte.

Calvo, D. (2019). *Diego Calvo*. Recuperado de https://www.diegocalvo.es/aprendizaje-supervisado/

Cambronero, C. G. (2006). Algoritmos de aprendizaje: knn & kmeans. Intelgencia en Redes de Comunicación. *Universidad Carlos III de Madrid*, 23.

Campderrich , F. (2014). *Ingeniería de software.* Barcelona: Editorial UOC.

Canós, J. H. (2012). Metodologías ágiles en el desarrollo de software.

Careuno. (2018). Qué es y cómo empezar con AngularJS.

Careuno, A. (18 de diciembre de 2018). *Qué es y cómo empezar con AngularJS.* Recuperado el 18 de junio de 2020, de https://www.phonegapspain.com/que-es-y-como-empezar-con-angularjs/

Carrasco, L. (12 de noviembre de 2017). *¿Qué es un algoritmo?* Recuperado el 18 de junio de 2020, de https://concepto.de/algoritmo-en-informatica/

Casseres, E. (1984). *Producción de hortalizas.* San José.

Castillo Ponce, R. y. (2010). Econometría práctica. *Mexicali, Mexico: Universidad Autónoma del Estado de Baja California*.

Castro-Silva, J. A. (2006). Sistema de riego autónomo basado en la Internet de las Cosas.

Catalunya, U. D. (2019). *Universidad De Catalunya*. Recuperado de http://informatica.blogs.uoc.edu/2019/09/05/movil-hibrido-flutter/

CESA. (2015). *Plan Rector de proyecto : apoyo a la transformación y comercialización de productos agrícolas y rurales.* Quito.

Chakray. (12 de agosto de 2017). *Qué son los microservicios.* Recuperado el 19 de junio de 2020, de https://www.chakray.com/es/que-son-los-microservicios-definicion-caracteristicas-y-ventajas-y-desventajas/

Chávez, F. (2012). *El cultivo del brócoli. Curso internacional de producción de hortalizas para la exportación.* Quito: Ec, Proexant.

CISET. (23 de marzo de 2019). *Software - Concepto y tipos de software.* Recuperado el 23 de junio de 2020, de https://www.ciset.es/glosario/480-software

Clasificación Del Software. (2017). Recuperado el 05 de agosto de 2020, de Mitecnologico.com: http://mitecnologico.com/sistemas/Main/Clasificaci%c3%b3nDelSoftware

Colla, P. (2012). Marco para evaluar el valor en metodología SCRUM. La Plata.

Contreras, J. E. (2003). ARIMA models to predict next-day electricity prices. *IEEE transactions on power systems*, 1014-1020.

Coronel, C. M. (2011). Base de datos: diseño, implementación y administración. *Cengage Learning Editores*.

Danilo A. López, N. Y. (2015). Desarrollo de un Modelo Predictivo para la Estimación del Comportamiento de Variables en una Infraestructura de Red. Recuperado de https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-07642015000500018

Daubenmire, R. (2000). Plant and Environment. A text-book of autoecology. *John Wiley and Sons*, p. 424.

De Arce, R. &. (2003). Modelos Arima. . *Programa CITUS: Técnicas de Variables Financieras.*

De Cozar Macias, O. D. (2010). Ajuste orbusto de primitivas elipticas basado en el error absoluto medio. *Universidad de Málaga*.

Deivy Yosip, D. R. (2013). *Universidad Nacional de Trujillo.* Recuperado de http://dspace.unitru.edu.pe/bitstream/handle/UNITRU/8786/DIONICIO%20ROSADO%2C%20Deivy%20Yosip.pdf?sequence=1&isAllowed=y

Dimas, J. (12 de mayo de 2018). *lenguaje Dart.* Recuperado el 17 de junio de 2020, de https://codigofacilito.com/cursos/dart

Dimes, T. (2015). *Conceptos Básicos de Scrum: Desarrollo de software Agile y manejo de proyectos Agile. .* Babelcube Inc.

Dutoit, B. B. (2014). *Object-Oriented Software Engineering Bruegge Dutoit.*

Eich, B. (2020). *JSON.* Recuperado el 10 de septiembre de 2019, de https://www.json.org/json-

es.html#:~:text=JSON%20(JavaScript%20Object%20Notation%20%2D%20Notaci%C3%B3n,es%20simple%20interpretarlo%20y%20generarlo.

Engel, E. &. (2001). Prediciendo el precio del cobre:¿ Más allá del camino aleatorio?

Enguídanos, A. M. (1995). Utilidad de los modelos de predicción de la crisis empresarial. *Revista española de financiación y contabilidad*, 281-300.

Erik Giovany Montes Páez, F. E. (2016). Aplicación de series de tiempo en la realización de pronósticos de producción. 79-88. Recuperado de https://dialnet.unirioja.es/servlet/articulo?codigo=6371161

ESCOBAR, M. E. (2014). DISEÑO DE UN MODELO PARA EL DESARROLLO DE PRUEBA DE SOFTWARE A FIN DE ALCANZAR EL NIVEL 2 DE TEST MATURITY MODEL EN LA EMPRESA SIREDCOM EN LA CIUDAD DE QUITO. Recuperado de http://repositorio.espe.edu.ec/bitstream/21000/8291/1/T-ESPEL-MAS-0009.pdf

Escobar, M. E. (2015). Modelo formal de pruebas funcionales de software para alcanzar el Nivel de Madurez Integrado 2. *Facultad de ingenieria de la UPTC*, 31-42.

Espasa, A. (2015). *IESTA.* Recuperado de http://www.iesta.edu.uy/wp-content/uploads/2015/11/Tema-4-Modelos-VAR-recursivos.pdf

Estévez, M. (2016). *Inteligencia Analítica*. Recuperado de https://inteligencia-analitica.com/acercamiento-modelos-clasificacion/

FAO. (2016). *La importancia de la agricultura en la actualidad.* [En línea]. DIsponible desde: http://www.fao.org/3/a0015s/a0015s04.htm. Recuperado el 21 de febrero de 2020

FAO. (2017). *Perspectivas para el medio ambiente. Agricultura y medio ambiente.* [En lpine]. Disponible desde: http://www.fao.org/3/y3557s/y3557s11.htm. Recuperado el 12 de abril de 2020

Fernández, E. (2 de junio de 2016). *Introducción a TypeScript.* Recuperado el 17 de junio de 2020, de https://desarrolloweb.com/articulos/introduccion-a-typescript.html

Figueroa, R. G. (2008). Metodologías tradicionales vs. metodologías ágiles. *Universidad Técnica Particular de Loja, Escuela de Ciencias de la Computación*, 9.

Francescangeli, N., Stoppani , I., & Martí , H. (2014). Aptitud de modelos de temperaturas y de tiempo térmico en brócoli (Brassica oleracea var. italica) . *Revista Agriscientia*, pp. 51-57.

Gálvez Neculman, H. A. (2019). Desarrollo de Back-End basado en una Arquitectura de Microservicios para la gestión y reventa de dominios. *CL de la empresa Haulmer Inc*.

García, F. R. (2002). *Gestión Integrada de modelado y de la medición del proceso software.* Escuela Superior de Informática. Universidad de Castilla–La Mancha.: Grupo Alarcos.

García, G. A. (2018). PROPUESTA DE UN MODELO DE REFERENCIA BASADO EN INTERNET DE LAS COSAS PARA DISEÑAR SOLUCIONES UTILIZANDO TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES.

GIL, M. A. (1988). Acotación del decrecimiento absoluto del error cuadrático medio del estimador insesgado de la diversidad: un criterio de elección del tamaño muestral. *Revista de la Real Academia de Ciencias Exactas Físicas y Naturales*, 129-139.

Gino, I. B. (2020). U.S. Patent No. 10,824,483. Washington, DC: U.S. *Patent and Trademark Office*.

GITBook. (2018). Recuperado de https://mcazorla.gitbooks.io/programacion-en-el-servidor/content/introduccion_a_codeigniter/estructura_del_framework_codeigniter.html

GONZÁLEZ, L. L. (2004). El diseño de interfaz gráfica de usuario para publicaciones digitales.

González, M. (2019). *Análisis de series temporales: Modelos ARIMA.* Madrid: Universidad del Pa´ıs Vasco (UPV-EHU). Recuperado el 20 de julio de 2020, de https://addi.ehu.es/bitstream/handle/10810/12492/04-09gon.pdf

González, P. (1990). Error cuadrático medio de predicción para modelos estructurales de series temporales. Universidad del País Vasco. Departamento de Análisis Económico.

Goonight, J. (12 de enero de 2018). *La inteligencia artificial (IA).* Recuperado el 19 de junio de 2020, de https://www.sas.com/es_mx/insights/analytics/what-is-artificial-intelligence.html

Grau, J. L. (2016). *Proagilist*. Recuperado de https://proagilist.es/blog/agilidad-y-gestion-agil/agile-scrum/la-metodologia-xp/

Guamán, J. (2017). Estudio del proceso de floreteo y su incidencia en la calidad del brócoli en la empresa ECOFROZ S.A. ubicada en el Cantón Mejía. *Trabajo de Titulación bajo la modalidad Estudio Técnico*. Ambato.[En línea]. Disponible desde:http://repositorio.uti.edu.ec/bitstream/123456789/55/1/8.%20Proyecto%20Final%20Julio%20A.%20Guam%C3%A1n%20C.%20Empastado.pdf: Universidad Tecnológica Indoamérica. Recuperado el 14 de junio de 2020

Gutierrez, D. (11 de junio de 2011). *Metodologías y procesos de análisis de software.* Recuperado el 17 de junio de 2020, de https://sites.google.com/site/proyectoadpmodelosdedesarrollo/home/modelo-en-cascada

Gutiérrez, J. (23 de enero de 2017). *¿Qué es un framework web?* Recuperado el 18 de junio de 2020, de http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf

Haughee, E. (2013). Instant Sublime Text Starter. *Packt Publishing*.

Hernandez, P. A. (2006). The effect of sample size and species characteristics on performance of different species distribution modeling methods. *Ecography*, 773-785.

IBM Corp. (2006). Recuperado el 05 de febrero de 2020, de https://cgrw01.cgr.go.cr/rup/RUP.es/LargeProjects/core.base_rup/guidances/concepts/logical_view_C135365E.html

ISARQ. (2018). *Ingenieria de software y Arquitectura*. Recuperado de https://isarq.com/index.php/2018/06/29/ques-es-api-para-que-sirve/

Izaurralde, M. P. (2013). Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario. . En T. d. especialidad.

Jadhav, M. A. (2015). Single page application using angularjs. *International Journal of Computer Science and Information Technologies*, 2876-2879.

JiménezTovar, D. J. (2019). *Unibague.* Recuperado de .https://repositorio.unibague.edu.co/bitstream/20.500.12313/1308/1/Trabajo%20de%20grado.pdf

Joskowicz, J. (2008). Reglas y prácticas en eXtreme Programming. Universidad de Vigo.

Js, N. &. (2016). Node. js. *Tradução de: SILVA, AG Disponível em*.

Knight, W. (2017). *MIT Technology Review*. Recuperado de https://www.technologyreview.es/s/9679/alphago-zero-ha-derrotado-su-hermano-mayor-en-100-0-sin-ayuda-humana

Korth, H. &. (1993). *Fundamentos de bases de datos.* Madrid.

Kruchten, P. (1995). Planos Arquitectónicos: El Modelo de 4+ 1 Vistas de la Arquitectura del Software. *IEEE Software*, 42-50.

Lafuente, A. (12 de julio de 2018). *Bases de datos relacionales vs. no relacionales: ¿qué es mejor?* Recuperado el 18 de junio de 2020, de https://aukera.es/blog/bases-de-datos-relacionales-vs-no-relacionales/

Lanas, S. (16 de noviembre de 2018). *Qué es la arquitectura de microservicios.* Recuperado el 18 de junio de 2020, de https://www.evaluandosoftware.com/que-es-la-arquitectura-de-microservicios/

Laplana Martín, P. (2019). Definición de una aplicación multiplataforma en dispositivos móviles para eventos accesibles.

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

López, J. (2018). *Algoritmos y programación.* Bogotá: Fundación Gabriel Piedrahita Uribe. Recuperado el 15 de diciembre de 2019, de https://libros.metabiblioteca.org/bitstream/001/169/8/AlgoritmosProgramacion.pdf

Lucas, J. (4 de septiembre de 2019). *Qué es NodeJS y para qué sirve.* Recuperado el 18 de junio de 2020, de https://openwebinars.net/blog/que-es-nodejs/

Luis Francisco Laurente Blanco, F. L. (2019). *RIIARn*. Recuperado de http://riiarn.agro.umsa.bo/index.php/RIIARn/article/view/21

Luna, F. M. (2017). *PROGRAMACION WEB Full Stack 2-HTML5: Desarrollo frontend y backend-Curso visual y práctico* (Vol. 2). RedUsers.

MAcgillvray, J. (2015). *Vegetable production.* New York, NY, U.S.A.: McGraw-Hill Book Co.

Maida, E. P. (2015). Metodologías de desarrollo de software. Tesis de Licenciatura en Sistemas y Computación. Facultad de Química e Ingeniería “Fray Rogelio Bacon”. *Universidad Católica Argentina*.

Maldeadora, N. (12 de diciembre de 2018). *Qué es Frontend y Backend.* Recuperado el 18 de junio de 2020, de https://platzi.com/blog/que-es-frontend-y-backend/

Maldonado Schmeisser, E. L. (2020). Evaluación del dimorfismo sexual a partir de variables métricas de cráneo y postcráneo mediante un análisis de regresión logística binaria y análisis de funciones discriminantes en una población subactual de Santiago, Chile.

Mamani, M. V. (2017). Sistema web de bajo costo para monitorear y controlar un invernadero agrícola. Ingeniare. *Revista chilena de ingeniería*, 599-618.

Mansutti Rosón, C. A. (2018). API unificada de acceso a datos meteorológicos para la agricultura.

Manuel, C. (2012). *MÉTODO ÁGIL SCRUM, APLICADO A LA IMPLANTACIÓN DE UN SISTEMA INFORMÁTICO PARA EL PROCESO DE RECOLECCIÓN MASIVA DE INFORMACIÓN CON TECNOLOGÍA MÓVIL.*

Maqueira López, L. A. (2016). Influencia de la temperatura ambiental y la fecha de siembra sobre la duración de las fases fenológicas en cuatro cultivares de arroz (Oryza sativa L.). . *Cultivos Tropicales*, 65-70.

Marquès, P. (1995). Metodología para la elaboración de software educativo. Barcelona (España).

Martelo, E. M. (2014). Prototipo De Una Aplicación Móvil Con Realidad Aumentada Para Mostrar Puntos De Información De Ubicación De La Universidad Simón Bolívar En Barranquilla Colombia Mediante El Uso Del Navegador Móvil Junaio. *Investigación e Innovación en Ingenierías*.

Martínez, A. (2003). Postcosecha y mercadeo de hortalizas de clima frío bajo prácticas de producción sostenible. *U. Jorge Tadeo Lozano*.

Maté, C. (2012). *Slide Share.* Recuperado de https://es.slideshare.net/juan_churqui/modelo-arima-14236175

McKinney, W. &. (2015). pandas: powerful Python data analysis toolkit. Pandas. *Pandas—Powerful Python Data Analysis Toolkit*, 1625.

Mendes Calo, K. E. (2010). Evaluación de metodologías ágiles para desarrollo de software. *In XII Workshop de Investigadores en Ciencias de la Computación.*

Méndez Camacho, M. M. (2016). *Aplicación de Scrum para crear Sistema de Estandarización de Planes de Trabajo en sistema web 2.0 para el CNE a través de la reutilización de Software, utilizando herramientas de Software libre (Bachelor's thesis, Quito, 2016.).*

Microsoft. (2019). *Docs Microsoft*. Recuperado de https://docs.microsoft.com/es-es/windows/ai/windows-ml/what-is-a-machine-learning-model

Mongodb. (2 de febreo de 2020). *La base de datos líder para aplicaciones modernas.* Recuperado el 18 de junio de 2020, de https://www.mongodb.com/es

Moreno Hermenejildo, I. Y. (2017). *Diseño de aplicación móvil para buscar ubicación de farmacias de turno utilizando la metodología SCRUM (Bachelor's thesis, Espol).*

Moreno, A. I. (2016). Diseño y evaluación de VISP, una aplicación móvil para la práctica de la competencia oral. *Revista Iberoamericana de Educación a Distancia*, 63-81.

Múñoz, R. (24 de octubre de 2014). *Node.js, javascript en servidor.* Recuperado el 17 de junio de 2020, de http://www.cantabriatic.com/node-js/

Murcia Pérez, E. S. (2013). Módulo Web Front-end para el desarrollo de simulación a partir de Weibull, Ji Cuadrado y Beta.

Norvig, S. J. (2004). Inteligencia Artificial: un enfoque moderno.

numpy. (2020). *numpy.org*. Recuperado el 06 de abril de 2020, de https://numpy.org/

Olarte, L. (23 de abril de 2018). *Lenguaje de Programación.* Recuperado el 17 de junio de 2020, de http://conogasi.org/articulos/lenguaje-de-programacion/

Ordóñez, L. T. (2010). Las exportaciones ecuatorianas tras diez años de dolarización y el papel de CORPEI. ECUADOR COMERCIO EXTERIOR. 125.

Organización de las Naciones Unidas para Alimentación y Agricultura. (14 de julio de 2018). *El futuro de la alimentación y la agricultura.* Recuperado el 10 de noviembre de 2019, de http://www.fao.org/publications/fofa/es/

Payne, A. &. (2012). Desarrollo. Madrid: Alianza Editorial.

Pérez, D. (27 de octubre de 2017). *¿Qué son las bases de datos?* Recuperado el 18 de junio de 2020, de http://www.maestrosdelweb.com/que-son-las-bases-de-datos/

Pérez, J. (2013). Caracterización y análisis de los sistemas de terrazas agrícolas en el Valle de Toluca, México. *Rev. agric. soc. desarro. SCIELO. [En línea]. Disponible desde: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1870-54722013000400002, vol. 10*(num.). Recuperado el 18 de enero de 2020

Pourrut, P. (1983). *Los climas del Ecuador: fundamentos explicativos. ORSTOM y Programa Nacional de Regionalización Agraria del Ministerio de Agricultura y Ganadería.* Quito.

Presma, R. (2012). *Ingeniería de Software: un enfoque práctico.* Madrid: Editorial McGraw.Hill/Interamericana.

Pressman, R. S. (1988). Ingeniería del software.

ProEcuador. (2016). *Perfil del Brócoli.* Quito: Ministerio de Comercio Exterior.

Raffino, M. (14 de febrero de 2020). *Base de datos.* Recuperado el 18 de junio de 2020, de https://concepto.de/base-de-datos/

Rahmat, R. F. (February de 2018). Implementation of Bayesian model averaging on the weather data forecasting applications utilizing open weather map. (I. Publishing., Ed.) *In IOP Conference Series: Materials Science and Engineering, 309*.

Rasthofer, S. A. (2014). A machine-learning approach for classifying and categorizing android sources and sinks. *In NDSS, 14*, 1125.

Raza, S. (21 de abril de 2018). *Cómo funcionan los microservicios.* Recuperado el 18 de junio de 2020, de https://www.evaluandosoftware.com/que-es-la-arquitectura-de-microservicios/

Rendón, Y. (19 de mayo de 2019). *Bases de datos relacionales vs. no relacionales.* Recuperado el 18 de junio de 2020, de https://www.pragma.com.co/academia/lecciones/bases-de-datos-relacionales-vs.-no-relacionales

Reyes, M. (20 de agosto de 2015). *Desarrollador web: Front-end, back-end y full stack. ¿Quién es quién?* Recuperado el 18 de junio de 2020, de https://www.campusmvp.es/recursos/post/Desarrollador-web-Front-end-back-end-y-full-stack-Quien-es-quien.aspx

Reynoso, C. (2004). Introducción a la Arquitectura de Software. *Universidad de Buenos Aires*, 33.

Riadi, F. M. (2004). Métricas para la valuación y medición del desempeño financiero y una prueba para EVA (r) en el mercado de telecomunicaciones chileno. *CAPIC REVIEW*, 4.

Riera Tubón, E. C. (2004). Desarrollo de una aplicación web con base de datos geográfica y turística para la provincia de Cotopaxi.

Rodrigo, J. A. (2017). *Ciencia de Datos*. Recuperado de https://www.cienciadedatos.net/documentos/33_arboles_de_prediccion_bagging_random_forest_boosting.

Rodríguez, D. (2018). *Analytics Lane.* Recuperado de https://www.analyticslane.com/2018/11/26/diferencias-entre-regresion-y-clasificacion-en-aprendizaje-automatico/

Rodríguez, M. (2014). Aprendemos el concepto, uso y cálculo de los Grados día. *Revista INESEM. [En linea]. Disponible desde: https://revistadigital.inesem.es/gestion-integrada/uso-concepto-grados-dia-degree-days/*. Recuperado el 05 de marzo de 2020

Roger., P. (2010). *Ingeniería del Software. Un enfoque práctico.* Madrid: McGraw-Hill Interamericana.

Rojas Jaén, M. C. (2019). Seguridad en los datos e implantación de la NTP-ISO/IEC 27001: 2014 en la Sub Gerencia de Gestión de Base de Datos del RENIEC.

Romero, P. Á. (2006). Arquitectura de software, esquemas y servicios. *Ingeniería Industrial*.

S.L., I. (2015). Recuperado de https://www.imaginanet.com/pdfinet/SCRUM%20es%20una%20metodolog%C3%ADa%20para%20la%20programaci%C3%B3n%20de%20aplicaciones%20m%C3%B3viles%20y%20Web.pdf

Sánchez, J. (2004). Principios sobre bases de datos relacionales. *Informe, Creative Commons*, 11-20.

Santos, B. O.-O.-D. (2010). *Producción de hortalizas en ambientes protegidos: estructuras para la agricultura protegida.* Gainesville, FL.

Schacherbauer, W. S. (2001). A flexible multiband frontend for software radios using high IF and active interference cancellation.

Scheidereiter, G. D. (2008). Aplicación de los Modelos Autorregresivos Integrados de Media Móvil (ARIMA) a las Series de Precipitaciones de Lluvia. *Universida Tecnológica Nacional.*

Schejtman, A. (1998). Agroindustria y pequeña agricultura: experiencias y opciones de transformación. En: Agroindustria y pequeña agricultura: vínculos, potencialidades y oportunidades comerciales-LC/G.

Schwaber, K. &. (2013). *La guía de scrum: La guía definitiva de scrum, las reglas del juego.* Recuperado el 09 de febrero de 2020, de scrumguides: http://www. scrumguides. org/docs/scrumguide/v1/Scrum-Guide-ES.pdf.

Scott, J. T. (2019). Gold Tree Solar Farm-Machine Learning to Predict Solar Power Generation.

Sicilia, M. A. (2008). *OpenStax CNX*. Recuperado de https://cnx.org/contents/OfSwDu71@1/Disparadores-en-bases-de-datos-relacionales

Silberschatz, A. K. (2002). *Fundamentos de bases de datos.* Madrid.

Sistel. (12 de marzo de 2017). *Mongo DB. Rendimiento, agilidad y escalabilidad con la base de datos NoSQL.* Recuperado el 17 de junio de 2020, de https://www.sistel.es/business-platform/mongodb

Solis, W. (21 de enero de 2016). *¿Qué es JavaScript?* Recuperado el 17 de junio de 2020, de https://uniwebsidad.com/libros/javascript/capitulo-1

Sommerville, I. (2005). Ingeniería del software. *Pearson educación.*

Soria, E. G. (2016). La producción de brócoli bajo riego en Guanajuato.

SOTO, C. &. (2011). Aprendizaje supervisado para la discriminación y clasificación difusa. *Dyna*, 26-33.

Stevens, P. P. (2002). *Utilización de UML en Ingeniería del Software con Objetos y Componentes* (Vol. 14). Addison Wesley.

Suaza, K. V. (2015). Mejora de historias de usuario y casos de prueba de metodologías ágiles con base en TDD.

Toledo, J. (2003). *Cultivo del brócoli.*

Toro, A. D. (2001). Metodología para el análisis de requisitos de sistemas Software. *Universidad de Sevilla-2001.*

Torres, J. (2018). DEEP LEARNING Introducci—n pr‡ ctica con Keras.

Torres, V. R. (2008). Modelo estadístico para la medición del impacto de la innovación o transferencia tecnológica en la rama agropecuaria. *Revista Cubana de Ciencia Agrícola*, 133-139.

Trigás Gallego, M. (2013). Metodologia scrum.

Ullman, J. D. (1999). *Introducción a los Sistemas de Bases de Datos.* Prentice Hall.

Valbuena, S. J. (2014). Sistemas para almacenar grandes volúmenes de datos. *Revista GTI*, 17-28.

Valencia Altamirano, D. G. (2018). Análisis de frameworks de desarrollo de api rest y su impacto en el rendimiento de aplicaciones web con arquitectura Spa (Master's thesis).

Van Rossum, G. &. (1991). Guía de aprendizaje de Python. Release.

Vázquez, J. C. (June de 2019). RNA-AP: Redes Neurales Artificiales con Aprendizaje Profundo. *In XXI Workshop de Investigadores en Ciencias de la Computación*.

Vermorel, J. (2012). *Lokad*. Recuperado de https://www.lokad.com/es/que-es-el-pronostico-de-series-de-tiempo

Viejo, D. (2020). *Arquitectura de desarrollo web con django y apps con flutter.* Madrid: Universitat Oberta de Catalunya.

Vigliola, M. (2015). *Manual de horticultura.* Buenos Aires: Editorial Hemisferio sur, S.A.

Vivanco, J. (2019). *Medium*. Recuperado de https://medium.com/@javiervivanco/el-modelo-c4-de-documentaci%C3%B3n-para-la-arquitectura-de-software-424704528390

Volosky, M. (2012). *Hortalizas. Cultivo y producción en Chile.* Santiago: Editorial Universitaria.

Winston, W. L. (2005). *Investigación de operaciones: aplicaciones y algoritmos (Vol. 4).* Thomson.

Zapata, C. M. (2013). GENERACIÓN DEL DIAGRAMA DE SECUENCIAS DE UML 2.1. 1 DESDE ESQUEMAS PRECONCEPTUALES (GENERATION OF UML 2.1. 1 SEQUENCE DIAGRAM FROM PRE-CONCEPTUAL SCHEMES). *Revista EIA*, 89-103.

Zavala , A. (2015). Minimización de los costos de producción en la Sección de Corte mediante la optimización del uso de los recursos, ECOFROZ, S.A. *Trabajo de graduación presentado como requisito parcial para optar al título de Ingeniera en Gestión de Agronegocios*. Tegucigalpa, Honduras. [En línea]. Disponible desde: https://bdigital.zamorano.edu/bitstream/11036/1957/1/T1794.pdf: Carrera de Gestión de Agronegocios. Recuperado el 16 de noviembre de 2019

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. . *Neurocomputing*, 159-175.

Anexos