UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS COMPUTACIONALES

TEMA:

ESTUDIO DEL FRAMEWORK ANGULAR PARA DESARROLLAR APLICACIONES SINGLE-PAGE. DESARROLLO DEL SISTEMA DE INFORMACIÓN, SEGUIMIENTO Y CONTROL PARA LA HACIENDA GANADERA LA VEGA

AUTOR:

EDISON PATRICIO FARINANGO CAIZA

DIRECTOR:

MSC. PEDRO DAVID GRANDA GUDIÑO

IBARRA – ECUADOR

MAYO 2020

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1004356646		
APELLIDOS Y NOMBRES:	FARINANGO CAIZA EDISON PATRICIO		
DIRECCIÓN:	SAN PABLO - OTAVALO		
EMAIL:	epfarianango@utn.edu.ec		
TELÉFONO FIJO:		**TELÉFONO MÓVIL:**	0996416130

DATOS DE LA OBRA	
TÍTULO:	ESTUDIO DEL FRAMEWORK ANGULAR PARA DESARROLLAR APLICACIONES SINGLE-PAGE. DESARROLLO DEL SISTEMA DE INFORMACIÓN, SEGUIMIENTO Y CONTROL PARA LA HACIENDA GANADERA LA VEGA
AUTOR (ES):	FARINANGO CAIZA EDISON PATRICIO
FECHA: DD/MM/AAAA	25/05/2020
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	■ **PREGRADO** ☐ **POSGRADO**
TITULO POR EL QUE OPTA:	**INGENIERO EN SISTEMAS COMPUTACIONALES**
ASESOR /DIRECTOR:	**MSC. PEDRO GRANDA**

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 25 días del mes de mayo de 2020

EL AUTOR:

(Firma).....
Nombre: EDISON PATRICIO FARINANGO CAIZA

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN DEL DIRECTOR

Por medio del presente yo MSc Pedro Granda, certifico que el presente trabajo de titulación "**ESTUDIO DEL FRAMEWORK ANGULAR PARA DESARROLLAR APLICACIONES SINGLE-PAGE. DESARROLLO DEL SISTEMA DE INFORMACIÓN, SEGUIMIENTO Y CONTROL PARA LA HACIENDA GANADERA LA VEGA**" fue realizado en su totalidad por el Sr. Edison Patricio Farinango Caiza, portador de la cédula de identidad: 1004356646, bajo mi supervisión.

Es todo cuanto puedo certificar en honor a la verdad.

Atentamente,

MSc. Pedro Granda.

DIRECTOR DE TESIS

ADMINISTRADORA DE LA HACIENDA GANADERA LA VEGA

CERTIFICA

QUE: El señor EDISON PATRICIO FARINANGO CAIZA con cédula de identidad 1004356646 estudiante de la Facultad de Ingeniería en Ciencias Aplicadas – de la carrera de Ingeniería en Sistemas Computacionales, ha desarrollado con los requerimientos entregados por parte de la Hacienda Ganadera LA VEGA, el Proyecto de Tesis "**ESTUDIO DEL FRAMEWORK ANGULAR PARA DESARROLLAR APLICACIONES SINGLE-PAGE. DESARROLLO DEL SISTEMA DE INFORMACIÓN, SEGUIMIENTO Y CONTROL PARA LA HACIENDA GANADERA LA VEGA**".

QUE: El proyecto fue entregado a la Hacienda Ganadera LA VEGA el 22 de mayo del 2020.

Es todo cuanto puedo certificar, facultando al interesado hacer uso de este certificado como estime conveniente.

Atentamente,

Hda. LA VEGA

Ing. Marcela Amaya Garzón
ADMINISTRADORA HACIENDA LA VEGA.
CI.: 171940332-9

DEDICATORIA

“Tu promesa no se retrasará ni un día. Recuerda cuánto has progresado no cuánto te falta. No estás donde quieres pero tampoco estás dónde estabas”.- Daniel Habif

Dedico este proyecto principalmente a Dios y a mi hermano Diego que desde el cielo guían mi camino. A mis padres Esteban y María por todo su esfuerzo y apoyo incondicional; esencial para permitirme haber llegado hasta este momento tan importante en mi formación profesional.

A mis hermanas por la motivación constante que me ha servido para el cumplimiento de mis objetivos planteados, en especial a mi hermana Mireya que siempre ha estado junto a mí brindándome su apoyo.

A cada una de aquellas personas que de forma directa o indirecta contribuyeron con mi formación académica

Edison Patricio Farinango Caiza

AGRADECIMIENTO

A mis padres Esteban y María, por todo el esfuerzo y atención que han dedicado en mí, enseñándome a no desfallecer, a siempre perseverar con honradez y responsabilidad, ellos son mi motivación para avanzar y cumplir mis objetivos trazados.

A la Universidad Técnica del Norte por permitirme ser parte de ella y nutrirme de conocimiento, a los docentes de la carrera de Ingeniería en Sistemas Computacionales que con su profesionalismo y dedicación impartieron enseñanzas a lo largo de mi formación académica.

Mi sincero agradecimiento aquellas personas que supieron orientarme en la construcción y desarrollo del presente Tema de Trabajo de Titulación. A mi Director de Tesis Ing. Pedro Granda por el apoyo, confianza y la orientación investigativa.

Sin más me gustaría poder extender mi más sincero agradecimiento a cada una de las personas que han contribuido de una u otra forma para el cumplimiento de cada una de las metas y objetivos que se han presentado durante esta mi formación profesional.

Edison Patricio Farinango Caiza

TABLA DE CONTENIDO

pp.

ÍNDICE DE FIGURAS

Figura **pp.**

ÍNDICE DE CUADROS

Tabla **pp.**

RESUMEN

El presente documento describe el estudio del Framework Angular que ha venido surgiendo los últimos años, con el objetivo de demostrar que Angular es recomendable para el desarrollo de aplicaciones escalables, llegando a solventar requerimientos mínimos así también como requerimientos empresariales, para demostrar se desarrolla e implementa el Sistema de información, seguimiento y control de la hacienda ganadera “La Vega”, a su vez la documentación elaborada sobre el Framework ayuda a los lectores interesados a aprender el Framework y herramientas adicionales en menor tiempo.

Inicialmente se realiza una introducción general del tema de trabajo de grado en el que se detalla los antecedentes, situación actual, planteamiento del problema, los objetivos, el alcance y la justificación.

En el capítulo uno; se realiza la revisión bibliográfica del Framework, se detalla el ámbito social sobre el cual se desarrollará el sistema, la situación actual del framework, la arquitectura, metodología de desarrollo, herramientas y configuraciones adicionales y características principales del Framework.

En el capítulo dos, se explica el desarrollo del Sistema con la ayuda de la metodología SCRUM, seguridad, pipes y configuraciones para el desarrollo consumiendo API Rest, por último se da a conocer la implementación del sistema en base a la metodología de desarrollo.

En el capítulo tres, se muestra la evaluación y resultados de usabilidad del software, se detalla las herramientas actuales del Framework para la elaboración de la documentación, además, el análisis de flexibilidad, compatibilidad y adaptabilidad del Framework Angular con respecto a otro. Por último muestra las conclusiones y recomendaciones del tema.

En el capítulo tres, se muestra la evaluación y resultados de usabilidad del software, se detalla las herramientas actuales del Framework para la elaboración de la documentación, además, el análisis de flexibilidad, compatibilidad y adaptabilidad del Framework Angular con respecto a otro. Por último muestra las conclusiones y recomendaciones del tema.

Palabras Claves: Sistema Web, Desarrollo web, API Rest, Componentes, Pipes, SCRUM, usabilidad, SPA, Aplicación en una sola página, Módulos, Angular CLI.

ABSTRACT

This document describes the study of the Angular Framework that has been emerging in recent years, with the aim of demonstrating that Angular is recommended for the development of scalable applications, reaching minimum requirements as well as business requirements, to demonstrate it is developed and implemented The information, monitoring and control system of the “La Vega” livestock farm, in turn the documentation prepared on the Framework helps interested readers to learn the Framework and additional tools in less time.

Initially, a general introduction to the subject of undergraduate work is carried out, detailing the background, current situation, problem statement, objectives, scope and justification.

In chapter one; The bibliographic review of the Framework is carried out, the social scope on which the system will be developed, the current situation of the framework, the architecture, development methodology, additional tools and configurations and main features of the Framework are detailed.

In chapter two, the development of the System is explained with the help of the SCRUM methodology, security, pipes and configurations for the development consuming API Rest, finally the implementation of the system based on the development methodology is disclosed.

In chapter three, the evaluation and usability results of the software are shown, the current tools of the Framework for the preparation of the documentation are detailed, in addition, the analysis of flexibility, compatibility and adaptability of the Angular Framework with respect to another. Finally it shows the conclusions and recommendations of the topic.

In chapter three, the evaluation and usability results of the software are shown, the current tools of the Framework for the preparation of the documentation are detailed, in addition, the analysis of flexibility, compatibility and adaptability of the Angular Framework with respect to another. Finally it shows the conclusions and recommendations of the topic.

Keywords: Web System, Web Development, API Rest, Components, Pipes, SCRUM, Usability, SPA, Single Page Application, Modules, Angular CLI.

INTRODUCCIÓN

Antecedentes

En la actualidad la implementación de nuevas TIC's es común en las empresas y organizaciones que buscan un constante crecimiento para mejorar los procedimientos y ofrecer sus productos o servicios a nivel mundial. Ecuador en los últimos años ha ido avanzando en el uso de las TIC's, ubicándose en el puesto 83 de un total de 137 países, la Figura 1 hace referencia a los reportes del GCI[1] que se encuentra en el sitio web World Economic Forum en el que hace referencia al período 2016-2019 (Wolrd Economic Forum, 2016).

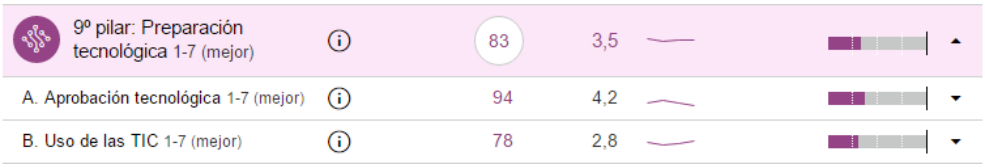


Figura 1: Uso de las TICS
Fuente: (Wolrd Economic Forum, 2016)

En Ecuador la Ganadería específicamente la producción lechera ha sido uno de los campos que ha venido creciendo significativamente, pero a la vez la mayoría de Haciendas Ganaderas no cuentan con un control adecuado en sus recursos lo que les impide ser más eficientes, por lo cual se puede implementar un sistema para que mejore la eficiencia de la administración y acceso a la información.

Las herramientas y plataformas para desarrollo de aplicaciones web han evolucionado constantemente, pasando del contenido de las páginas con texto estático a un contenido rico e interactivo (Menéndez & Barzanallana, 2016). En la actualidad los usuarios han optado por seguir usando las tradicionales, sin darse cuenta de los beneficios que ofrecen las nuevas plataformas de desarrollo web.

[1] **GCI:** Índice de Competitividad Global

Un claro ejemplo son las single-page application, o aplicación de página única que es una aplicación en la que todas las funcionalidades se desarrollan en una única página, cargando toda la lógica necesaria una vez al cargar la página o cargándola de forma dinámica en el momento en que sea necesario, con el propósito de dar una experiencia más fluida a los usuarios como si se tratara de una aplicación de escritorio (De La Fuente, 2016).

Actualmente las empresas dedicadas al desarrollo de servicios en la web y aplicaciones han notado que los frameworks son una herramienta indispensable para generar ganancias en menos tiempo. No obstante como lo menciona Acens (2018): cada framework conlleva una cantidad de tiempo para llegar a manejarlo con eficiencia ya sea por el lenguaje de programación, complejidad, tecnologías o poca información en el idioma nativo del interesado.

Situación Actual

ANGULAR es un framework de código abierto basado en Web Components, escrito en TypeScript[2], la última versión estable es 8.2.11 (actualizado el 15 de octubre del 2019), Permite desarrollar aplicaciones auto-actualizables, construido para ser más rápido, su código es moderno, cuenta con una API simplificada lo que conlleva a mejorar la productividad, además es de gran ayuda para los desarrolladores frontend (García, 2016).

Conviene recordar que Angular no es una nueva versión de Angular JS, sino que se trata de un nuevo framework; escrito completamente desde cero. Es decir, si tiene conocimientos en Angular JS, pasar a Angular no es directo. Valerio Barreto (2016) afirma: Angular busca dar a conocer la técnica de desarrollo de tipo Single Page Applications (SPA) que funciona en dispositivos móviles y de escritorio gracias a que es un framework Cross Platform[3], se maneja un desarrollo basado en el modelo vista controlador (MVC) y la ejecución del lado del cliente.

Actualmente el manejo de la información de la ganadería no se encontraba tecnificada, todos los registros se realizan manualmente, teniendo como debilidad y

[2] **TypeScript:** Lenguaje de programación libre y de código abierto, súper conjunto de JavaScript, basados en clases.
[3] **Cross Platform:** Programa o dispositivo que puede utilizarse sin inconvenientes en distintas plataformas de hardware y sistemas operativos.

riesgo la manipulación de la información, por lo cual toma más tiempo al momento de presentar información requerida y también en la toma de decisiones, esta situación se viene dando por la falta de un sistema web que permita administrar y organizar la información de manera correcta.

Prospectiva

Con el presente estudio a realizarse se elaborará la documentación necesaria para que estudiantes y profesionales interesados en el estudio de Angular puedan ampliar sus conocimientos sobre esta herramienta.

El estado en el que se encuentra la Hacienda ganadera La Vega impide que se lleve un control ordenado de la información y recursos, por lo tanto, a futuro se pretende solucionar este problema con el sistema web a desarrollarse.

Se agregarán módulos como: control de acceso basado en roles, registro y producción diaria de tanto de la leche como del ganado. Estos módulos mejorarán las características y prestaciones en cuanto a la seguridad, roles de usuario, manejo y un mejor control de la información con la que trabaja la hacienda ganadera “La Vega”. Se prevé que la hacienda ganadera La Vega estará conectado en un servicio Cloud que estará integrado el sistema web.

Planteamiento del Problema

La forma en la que se realiza el registro y el despliegue de información de la hacienda La Vega es en ocasiones inadecuados, ya que actualmente el manejo y control de los procesos - como se muestra en la Tabla 1 - relacionados a la ganadería no se encuentra tecnificada. Lo cual no permite que se realice el registro y despliegue de información satisfactoria acorde a la necesidad; por la inexistencia de datos almacenados digitalmente. Por lo cual se pretende realizar el estudio del framework Angular para el desarrollo de una aplicación capaz de gestionar, automatizar y mejorar la administración.

TABLA 1: Procesos relacionados a la administración de una hacienda ganadera.

Procesos	
Inicio	Ver – crear hacienda.
	Ver – crear animal.
Novedades	Compra -venta de productos.
	Potreros - traslado de animales.
	Producción histórica de forrajes.
	Alimentación.
	Mastitis.
Reproducción	Celos - palpaciones.
	Celos - tacto.
	Partos.
	Abortos.
Producción	Pesaje de leche individual – total.
	Pesaje de animales.
	Selección – descarte de animales.
	Secado – Destete.

Fuente: Elaboración propia.

Formulación del Problema

¿Cómo mejorar la versatilidad en las aplicaciones web mediante la implementación del framework Angular?, Figura 2.

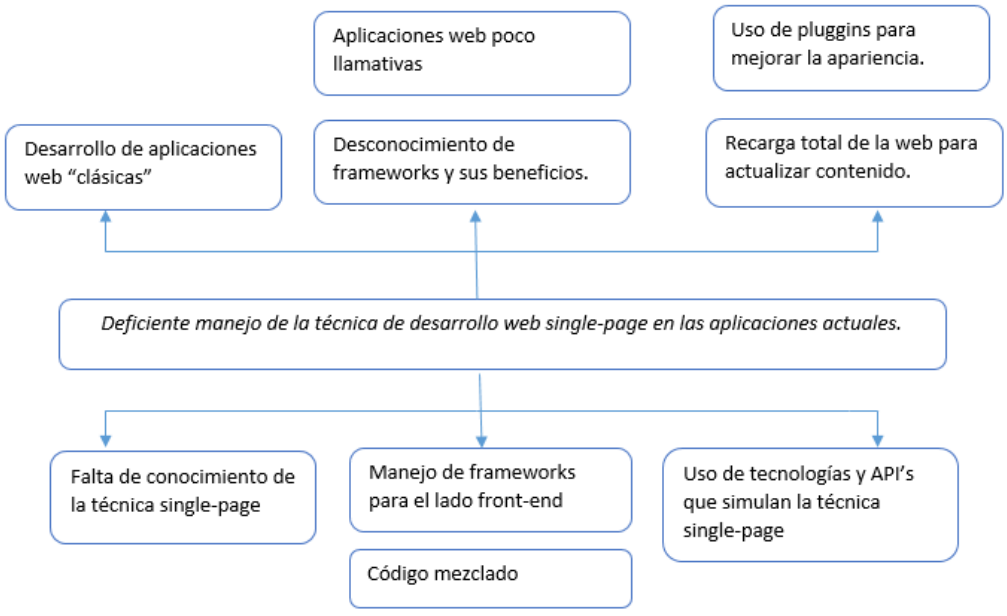


Figura 2: Árbol de Problemas
Fuente: Elaboración propia.

Delimitación del Problema

Delimitación de Estudio

El presente trabajo de grado está orientado en la línea de investigación de desarrollo y aplicación de Software.

Delimitación en el Espacio

Es sistema web se desarrolló en la Hacienda La Vega, perteneciente a la parroquia San Pablo del Lago del cantón Otavalo, provincia de Imbabura.

Objetivos

Objetivo General

Establecer un estudio del Framework Angular para desarrollar aplicaciones single-page en la implementación del Sistema de Información, Seguimiento y Control para la Hacienda Ganadera La Vega.

Objetivos Específicos

- Analizar la situación actual del framework Angular.
- Elaborar documentación técnica que servirá como guía para las personas interesadas en el framework.
- Implementar un sistema web capaz de gestionar, automatizar y mejorar la administración desarrollado en Angular para el sistema de Información, Seguimiento y Control para la Hacienda Ganadera “La Vega” usando metodología SCRUM.
- Realizar un análisis de impactos del uso de la herramienta con respecto a otro framework.

Alcance

El propósito del presente estudio del framework Angular es desarrollar aplicaciones single-page, además de facilitar a las personas interesadas una guía adecuada para el aprendizaje y desarrollo de aplicaciones web más versátiles.

Con los conocimientos adquiridos se desarrolló un sistema que permite llevar el control y administración de una hacienda ganadera mediante la metodología SCRUM, en diferentes secciones como: ganado en sus diferentes etapas, registro individual y ficha médica, movimiento de animales, registro de productos e insumos agrícolas y pecuarios, alertas de medidas sanitarias e información reproductiva, como se muestra en la Figura 3 (Agrocalidad, 2012).

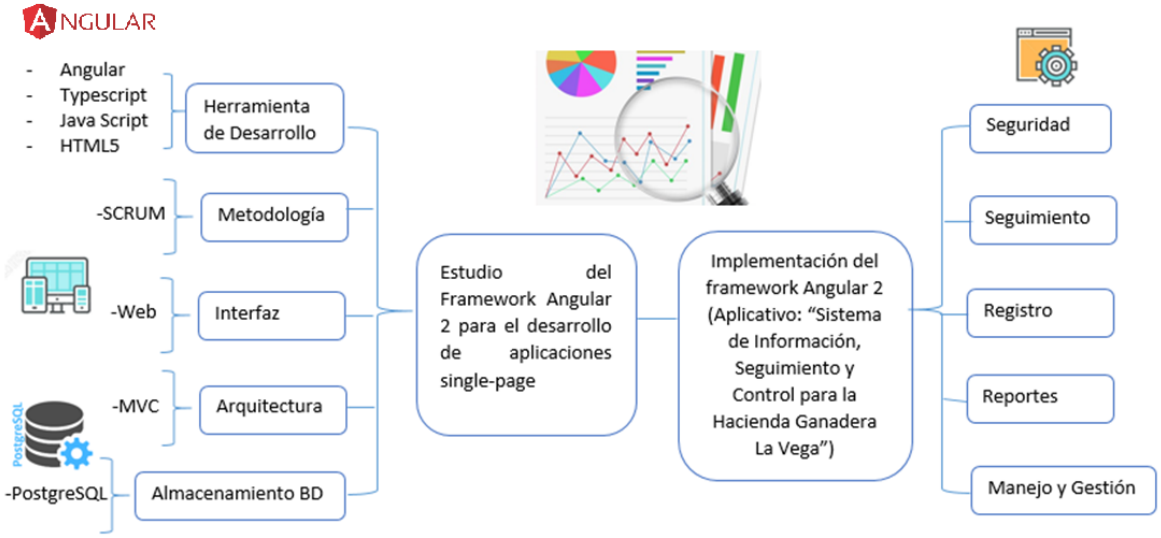


Figura 3: Mapa mental de la idea
Fuente: Elaboración Propia

Limitación

Para la investigación del presente estudio es necesario de ebooks, los cuales se consultaron a través de internet; debido a que actualmente no se encuentra bibliografía en libros físicos en las bibliotecas del país.

Como aporte al desarrollo de las interfaces se necesita tener conocimiento en ciertos lenguajes que permitan construir un sistema visualmente agradable y funcional como es el caso de las herramientas JQuery, JavaScript, CSS3 y HTML5.

Administración del Sistema

- **Seguridades**

Controla el tipo de autenticación de los usuarios al sistema por medio de un formulario de login, cuenta con filtros para accesos no autorizados dependiendo el con el que cuente el usuario.

- **Usuarios y Roles**

Dicho módulo es administrado por parte del usuario administrador, permite agregar, modificar, eliminar y consultar la información de un determinado objeto o usuarios al sistema web los cuales tendrán permisos de acceso a módulos específicos. El acceso a la información también se controla a través de una función o rol del usuario que requiere dicho acceso.

Módulos del Sistema

- **Módulo de Información de la Hacienda**

En este módulo del sistema se administra la información general de la hacienda ganadera, tales como propietarios, administradores, ubicación, logo, etc.

- **Módulo de Ganado**

En el presente módulo del sistema se administra los registros del ganado en diferentes aspectos como: etapas de crecimiento, información individual y ficha médica, información de crías, catálogo de toros con sus respectivos historiales y registro de mortalidad.

- **Módulo de Reproducción**

En el presente módulo se maneja el control reproductivo como lo es el registro de hembras reproductoras, etapas de gestación y medicamentos aplicados con fines reproductivos.

- **Módulo de Sanidad**

Se maneja los registros e históricos de los días de vacunación, controles sanitarios, desparasitación o algún otro tratamiento, lo cual permite realizar un completo seguimiento a cada grupo de ganado.

- **Módulo de Potreros**

En este módulo se podrán identificar los potreros de la hacienda ganadera según el pastoreo, cantidad de alimento disponible y el tiempo que toma en la recuperación.

- **Módulo de Alimentación**

En este módulo se administra la suministración de alimento en cantidad y calidad, de acuerdo al propósito y características del ganado.

- **Módulo de Reportes Reproductivos**

Permite la administración de la información genética, ayudando a llevar un historial de descendencia.

- **Módulo de Reporte de Eventos**

Permite administrar y visualizar sucesos importantes como chequeos, registros de compras y stock de la alimentación en la hacienda ganadera.

Para el desarrollo del sistema web se aplicará la metodología SCRUM que está basada en el modelo de las metodologías de desarrollo ágiles, incrementales, basadas en iteraciones y revisiones continuas (Navarro, Fernández, & Morales, 2013). Conjuntamente con el modelo de arquitectura de software MVC, tal como se describe en la Figura 4.

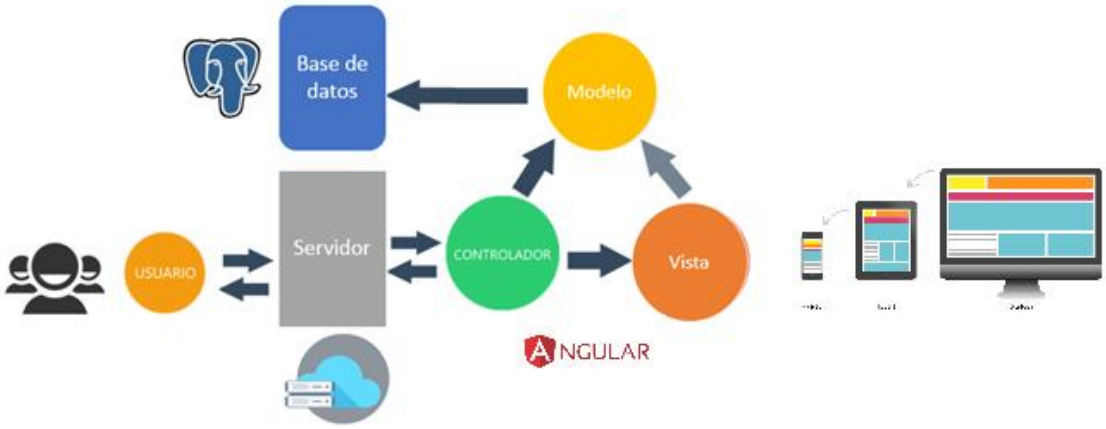


Figura 4: Arquitectura MVC
Fuente: Adaptado de (Navarro et al., 2013)

Justificación

El avance tecnológico y la implementación de TIC's hoy en día ha hecho que las empresas o instituciones busquen la manera de automatizar sus procesos para facilitar el desempeño. Mediante el análisis de tendencias en búsqueda a través de la herramienta de comparación Google Trends realizada con términos referentes a frameworks, da a conocer un incremento constante de la tendencia de popularidad de Angular. Es por aquello que se prevé desarrollar un sistema en base al estudio del framework Angular el cual permitió agilizar los procesos de control y administración de la Hacienda Ganadera La Vega.

El estudio es un aporte para los estudiantes y personas de habla hispana que estén interesadas en realizar las debidas investigaciones de nuevas metodologías y herramientas tales como los frameworks para realizar de mejor manera el desarrollo de software, ya que actualmente la documentación en español a nivel mundial es mínima.

CAPÍTULO I

Revisión Bibliográfica

El presente capítulo contiene temas relacionados al ámbito actual en el que se encuentra la hacienda ganadera La Vega, así como los componentes y situación actual del framework del tema de estudio, además de la arquitectura y metodología a implementarse para el desarrollo del sistema.

1.1. Estado de las haciendas ganaderas en el Ecuador.

El consumo mundial de lácteos que son producidos principalmente de las haciendas ganaderas, se estima que crecerá el 36% entre 2014 y 2024, con un consumo de alrededor de 713 millones de toneladas de leche líquida. Además, se dice que en los próximos años la globalización de la industria se acelerará con un impacto significativo en el comercio nacional e internacional (Jonsson, 2014).

El presidente y CEO de Tetra Pak Group Jonsson (2014) manifestó:

> Se prevé que el consumo mundial de leche blanca aumente el 1,8% de 2014 a 2017, es decir, de alrededor de 212 mil millones de litros a cerca de 223 mil millones de litros, superando el crecimiento de 1,2% del período 2010–2013.

En Ecuador las haciendas ganaderas y la ganadería en general se ve reflejada en los beneficios de producción con la que cuenta cada región; es decir en las regiones Costa y Amazonía producen principalmente ganado de carne, a diferencia de la región Sierra que gracias a sus valles fértiles permite la producción de leche, en cada región el Ministerio de Agricultura, Ganadería y Pesca (MAGAP) implementa planes para el beneficio y producción en dichas regiones.

La ganadería dedicada a la producción de leche, es la actividad económica de mayor importancia del sector agropecuario en el Ecuador; como se describe en la Tabla 2,

a tal punto que los ganaderos exhiben como insignia el hecho de que el país ahorra 500 millones USD$ anuales al no tener que importar el producto. Según datos recogidos en el Censo Agropecuario del año 2014 se registraron 4.486.020 cabezas de ganado vacuno (Resabala & Recalde, 2017).

TABLA 2: Actividad económica en base a la producción de leche.

Regiones	%
Sierra	51%
Costa	36%
Oriente	13%
Total	**100%**

Fuente: (Resabala & Recalde, 2017)

El gerente general de la Asociación de Ganaderos de la Sierra y Oriente (AGSO), prevé que en el país la producción de leche seguirá creciendo, actualmente se producen 5.500.000 litros de leche a diario, según estudios la mayoría de las haciendas que se dedican a la producción de leche no cuentan con tecnología orientada a la sistematización de resultados de producción (Grijalva, 2016)

Para tal efecto la Tabla 3 hace énfasis al Número de vacas ordeñadas, producción y destino de la leche, según región y provincia.

TABLA 3: Encuesta de Superficie y Producción Agropecuaria Continua 2016.

REGIÓN Y PROVINCIA	NÚMERO TOTAL DE VACAS ORDEÑADAS	PRODUCCIÓN TOTAL DE LECHE (Litros)	DESTINO PRINCIPAL DE LA LECHE (Litros)				
			Vendida en líquido	Consumo en los terrenos	Alimentación al balde	Procesada en los terrenos	Destinada a otros fines
REGIÓN SIERRA	570.270	4.106.855	3.369.942	330.721	98.486	295.328	12.377
IMBABURA	24.808	246.454	226.239	11.885	4.805	3.341	184

Fuente: (Grijalva, 2016)

1.1.1. Impacto de las Tics en el sector ganadero.

La utilización y aplicación de las Tics en la Ganadería se encuentran presente hace algunos años, en países como España, Estados Unidos, Uruguay, México, Argentina, entre otros referentes, principalmente para la venta de cárnicos, subastas electrónicas de cabezas de ganado, obtención de equipos o herramientas y registros ganaderos para diferentes certificaciones mediante tiendas en línea (Granda, 2014).

La implementación de dichos sistemas ha permitido mejorar la administración ganadera ya que poseen herramientas y características importantes, las mismas que podrían ser mejoradas debido a que la mayoría se encuentran desarrollados para trabajos específicos.

En los últimos años Ecuador ha crecido en la implementación de Tics en los diferentes campos tales como industriales o empresariales, actualmente no solo es necesario en dichos campos, sino que también debe servir en el mejoramiento de la agricultura y ganadería.

En ese contexto, como menciona Rodrigues (2012): las tecnologías de la información y de las comunicaciones (TIC) se convierten en una poderosa herramienta para el acceso y la organización del conocimiento disponible para los agricultores y ganaderos. Las TIC cumplen el rol de facilitar la implementación de tecnologías –nuevas y tradicionales– y de transformar las maneras de aprendizaje e interacción entre productores.

En el país las principales limitaciones para la integración de las TIC en la agricultura parecen estar en los niveles educativos y en las prácticas culturales y tradicionales de las comunidades rurales. El Ministerio de Agricultura, Ganadería y Pesca (MAGAP) entre uno de sus trabajos se encuentra el proyecto El Ordeño, el cual es una plataforma productiva de la leche que funciona como una plataforma virtual para los productores, eso ha reducido significativamente el tiempo necesario para disponer de esos recursos. Sin embargo, actualmente el piloto del plan tiene problemas debido a la cobertura de Internet en diferentes sectores (Rodrigues, 2012).

Dicho proyecto ha sido de gran importancia en el impacto de las Tics a nivel de empresas: automatización y control en base a un software, uso de enlaces privados para suplir la deficiencia de cobertura de Internet en haciendas ubicadas en zonas

rurales, uso de telefonía IP y controles biométricos, además, sitios web de asociaciones han mejorado en la administración y uso de la información del sector ganadero.

El desarrollo de las TIC en la agricultura y ganadería debe estar enfocado en el pequeño productor. Es preciso solucionar la poca cobertura de Internet en las zonas rurales, al igual que la cobertura de las empresas operadoras de telefonía celular en el sector rural. Las TIC con mayor potencial para el sector es el Internet, pues de aquí se derivan múltiples productos, entre ellas las aplicaciones web (Rodrigues, 2012).

1.1.2. Importancia de implementar un software para la administración de haciendas ganaderas.

Actualmente los productores de ganado se ven encaminados a convertirse en empresarios eficientes, se conoce que la actividad ganadera se refleja en la obtención de uno o varios productos que generen ganancias, sin embargo para poder obtener dichos resultado es conveniente la implementación de un sistema capaz de generar registros que permitan obtener resultados, y que estos a su vez puedan ser medidos para posteriormente comparar con las metas planteadas.

Los sistemas de registros tales como Software Ganadero SG y ProGAN incluyen la mayoría de funcionalidades que integran el proceso de producción. No obstante a pesar de su importancia existe desconocimiento de parte de los productores de cómo llevarlos y las ventajas que estos significan a la hora de la toma de decisiones en el ámbito de producción o metas propuestas (Domic, 2015).

El objetivo del sistema de administración ganadera desarrollado en el tema de estudio, busca brindar que los registros de producción tengan herramientas que permita manejar la información, seguimiento y control de los animales, de tal manera que ayude al productor o administrador a tomar mejores decisiones.

No obstante para implementar un sistema de Información, Seguimiento y Control de haciendas ganaderas se requiere de herramientas que permitan la identificación de

todos los animales que conforman el hato[4], el mismo que debe ser único, permanente, visible y fácil de aplicar (Domic, 2015).

1.2. Situación actual del framework Angular.

Angular es un framework mantenido por Google para el desarrollo de aplicaciones web del lado del cliente, debido a la buena expectativa y mejoramiento que dejó el conocido framework AngularJS la empresa optó por escribir desde cero este nuevo framework, el mismo que mejoró notablemente su integración con módulos, extensiones y componentes.

AngularJS y Angular no tienen nada que ver el uno con el otro, ya que es otro framework y no simplemente una nueva versión, migrar una aplicación de AngularJS v1.x a Angular todavía no está al 100%. Lo que pretende Google es aportar un framework nuevo que incluya todas las nuevas normas de desarrollo y ha intentado corregir todos los problemas que presentaba AngularJS v1.x (Jdonsan, 2016).

Angular comenzó en el 2010 bajo el nombre de AngularJS, se ha popularizado y cada vez trata de estandarizar sus componentes para crear aplicaciones SPA modernas, es decir aplicaciones de una sola carga en la cual todas sus secciones y carga de datos se realizan de manera asíncrona sin necesidad de refrescar la página.

Retos y necesidades de la nueva versión de Angular

Desde su creación Angular ha sido el framework preferido por la mayoría de los desarrolladores JavaScript. De ser una plataforma para la creación de Web Apps, ha evolucionado como motor de una enorme cantidad de proyectos del ámbito empresarial y de ahí para aplicaciones en la Web Mobile Híbrida, llevando la tecnología al límite de sus posibilidades (Alvarez & Basalo, 2018).

Todos aquellos retos son el motivo por el que comenzaron a detectarse problemas en Angular 1, o necesidades donde no se alcanzaba una solución a la altura de lo deseable. Entre las necesidades más notables se menciona a las siguientes:

[4] **Hato:** Conjunto de animales de ganado mayor o menor.

- **JavaScript**

Encontraron problemas en la creación de aplicaciones debido al propio JavaScript. Ya que es un lenguaje con carácter dinámico, asíncrono y de complicada depuración. Al ser tan particular resultó difícil adaptarse a él, sobre todo para personas que están acostumbradas a manejar lenguajes más tradicionales como Java o C#, porque muchas cosas que serían básicas en esos lenguajes no funcionan igualmente en JavaScript.

Encontraron problemas en la creación de aplicaciones debido al propio JavaScript. Ya que es un lenguaje con carácter dinámico, asíncrono y de complicada depuración. Al ser tan particular resultó difícil adaptarse a él, sobre todo para personas que están acostumbradas a manejar lenguajes más tradicionales como Java o C#, porque muchas cosas que serían básicas en esos lenguajes no funcionan igualmente en JavaScript.

- **Desarrollo del lado del cliente**

Angular lleva al navegador mucha programación que antes estaba del lado del servidor, comenzando por el renderizado de las vistas. Uno de los retos y necesidades es la sobrecarga en el navegador, haciendo que algunas aplicaciones sean lentas usando Angular 1 como motor (Alvarez & Basalo, 2018).

1.2.1. Últimas novedades de Angular.

En julio del 2016 fue lanzada la primera versión oficial estable de Angular que aprovecha las últimas mejoras de los estándares Frontend, estas mejoras son los web components[5] que recomienda la W3C desde julio 2014 y permiten extender el HTML; la nueva especificación de JavaScript que desde junio 2015 cuenta con la especificación ECMAScript[6] 6 de la que se nutre TypeScript para añadirle tipos de datos, interfaces y mejoras en la programación orientada a objetos (Aguilera, 2016)

Angular trata de mantener ciertos objetivos como lo es la consistencia; es decir simplicidad para aplicaciones pequeñas y escalabilidad para aplicaciones grandes. Desde la actualización 2 de Angular hasta la 8 (v8.2.11 estable octubre, 2019) añade

[5] **Web components:** Conjunto de APIs Web que permiten crear etiquetas HTML personalizables.
[6] **ECMAScript:** Estándar que define cómo ser interpretado el lenguaje en cada una de las tecnologías.

una gran cantidad de funcionalidades para navegadores modernos y mejora de rutas. Sin embargo las que más resaltan son las siguientes:

- **HTML Imports**

Permite importar un código HTML en otro HTML, se utiliza para la distribución de las librerías de web components.

- **Shadow DOM**

Es la tecnología que permite encapsular el web component para que no se vea afectado por el DOM general de la página a no ser lo requiera explícitamente. En otras palabras, es todo el contenido que encapsula la etiqueta del web component. En algunos navegadores se puede ver inspeccionando la página (Aguilera, 2016).

- **Carga Diferencial de JavaScript**

La carga diferencial o Differential Loading consiste en generar un proyecto, CLI producirá paquetes o bundles de JavaScript antiguo (ES2015) y moderno (ES2015+), esto ayudará a los navegadores modernos a cargar más deprisa las páginas al tener ya el compilador para ES2015+.

- **CSS**

Angular ha implementado el respaldo de SASS para Bazel, las reglas se agregan al área de trabajo para un proyecto que requiere la extensión SASS a CSS. Con SASS, los desarrolladores pueden escribir estilos visuales para un sitio web en un lenguaje más avanzado que se compila en CSS, sin embargo, el desarrollado puede elegir el estilo visual con el que mejor se adapte.

- **Ivy**

Con la versión 8 se pretende implementar el motor Ivy o el View Engine clásico para generar los proyectos, aunque Ivy no será viable para todo los usos, ya que su lanzamiento definitivo será en versiones posteriores, no obstante las ventajas más notable de dicho motor son las siguientes (ImaginaGroup, 2019):

- Código más fácil de depurar, incluso a medida que las aplicaciones crecen.
- Mejora de la comprobación de tipos de plantillas.
- Mejora de la retrocompatibilidad.

1.2.2. Principales características de Angular.

Problemas dificultosos de solucionar con la tecnología AngularJS, han sido los que han incitado a sus creadores a desarrollar desde cero una nueva versión del framework. La nueva herramienta se orienta a dar soluciones a los usos dados por los desarrolladores, además, llevar a JavaScript a un nuevo nivel que se asemeje a lenguajes más tradicionales, a continuación las principales características:

a) Simplicidad

Aguilera (2016) manifiesta que una de las característica que ha buscado el equipo de Angular es la de simplificar el framework, es decir quedarse solo con lo bueno y mejorando o eliminando lo malo de la primera versión. Entre lo malo destaca la utilización directa del **$scope** sobre el diseño de los servicios y la carga de módulos, que ahora se resuelve de forma nativa con ES6 y la construcción de web components en lugar de directivas para extender el HTML como se aprecia en la Figura 5.

TypeScript provee una sintaxis mucha más entendible y que trata de asemejarse a lenguajes de programación orientados a objetos (POO) como es el caso de los tradicionales Java o C#, los templates dan a conocer las características de la vista de un componente y sus relaciones con otros componentes.

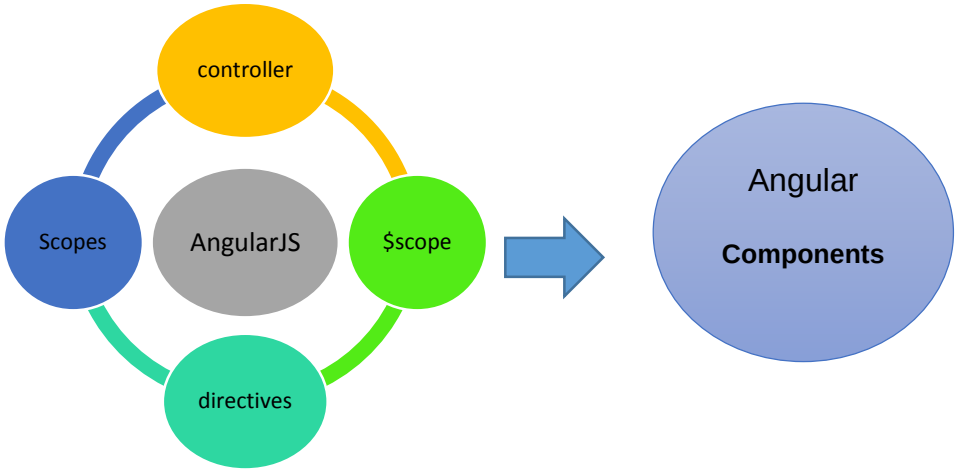


Figura 5: Eliminación del objeto $scope en Angular
Fuente: Adaptado de (Aguilera, 2016)

b) Rendimiento

Se dice que las buenas prácticas adoptadas en el diseño han mejoran el rendimiento general, una de ellas es la detección de cambios en la vista que antes se

hacía con un ciclo **$digest** que consumía muchos ciclos de CPU y ahora se realiza con un sistema reactivo que supone una significativa mejora de rendimiento (Aguilera, 2016).

Referente a esta mejora, ahora el Data Binding[7] es unidireccional, para hacer un Data Binding bidireccional como en AngularJS, se tiene que hacer de manera explícita. Está optimizado para móviles donde los ciclos de CPU y memoria son parámetros críticos para el correcto funcionamiento (Aguilera, 2016).

Inyección de dependencias

Dicha característica ha sido heredada de la versión AngularJS, en la Figura 67 destaca el mecanismo para proporcionar nuevas instancias de una clase con todas aquellas dependencias que requiere, la mayoría son servicios, reduce el acoplamiento entre clases y favorece el testing. Gracias a TypeScript Angular sabe de qué servicios depende un componente con tan solo revisar su constructor (Oriol, 2016).

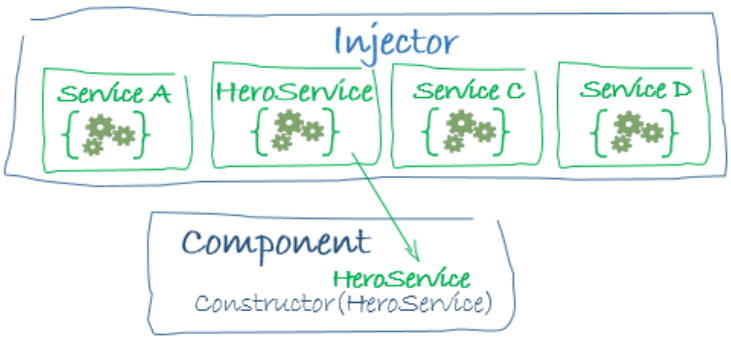


Figura 6: Inyector de dependencias
Fuente: (Oriol, 2016)

c) Testabilidad

Esta característica está muy ligada a la anterior. Con Angular no hay excusas para que el código no esté completamente testeado, ya que cualquier elemento de la aplicación es fácilmente testeable gracias a frameworks como Karma y Jasmine (Aguilera, 2016).

[7] **Data Binding:** Coordina partes de una plantilla con partes de un Component.

d) Mejoras en el Router

El router en las aplicaciones single page es el elemento que permite simular la navegación entre páginas, permitiendo cambiar la vista dinámica principal al clicar sobre un enlace (Aguilera, 2016). En la versión actual cuenta con ciertas mejoras:

Creación de rutas más sencillas y que vaya de acuerdo con el estándar de la industria, en versiones anteriores de Angular la configuración de una ruta era de la siguiente manera: **{path: '/admin', loadChildren: './admin/admin.module#AdminModule'}**

En esta nueva versión de Angular la configuración de una ruta será de la siguiente manera: **{path: `/admin`, loadChildren: () => import(`./admin/admin.module`).then(m => m.AdminModule)}**

Esto permite a los editores de código Visual Studio y WebStorm entender fácilmente y validar estas importaciones en el proyecto.

e) Cambios en el Lazy loading

Cargar módulos a través de lazy loading, también llamada carga diferida, permite reducir el tamaño inicial de la aplicación y aumentar la velocidad de carga inicial. La forma de hacerlo a partir de ahora cambia un poco con el fin de que sea un poco más estándar y se adecúe a lo que se hace en la industria como un estándar informal (Valiñas, 2019).

Antes:

path: '/dashboard,
loadChildren: './dashboard/dashboard.module#DashboardModule'

A partir de la versión 8 se escribe de la siguiente forma, con una Promise:

path: `/dashboard`,
loadChildren: () => import(`./dashboard/dashboard.module`).then(m => m.DashboardModule)

Como se puede observar el cambio no supone un gran impacto, pero sí se deberá actualizar la aplicación si se usa esta técnica de carga en diferido.

1.2.3. Actualización de Angular a una versión superior.

El presente estudio aborda información y documentación referente al framework Angular, en su versión 8.1.1 (actualmente estable). Es necesario recalcar que poco

después de salir la versión de Angular 2 (en adelante), el framework cambió de nombre a “Angular”.

Las personas que han seguido a la comunidad de Angular desde hace un tiempo comprenden el cambio de Angular 1 a Angular, el mismo que fue un cambio total, con nuevas API's, nuevos patrones; debido a que Angular se ha reprogramado desde cero. Cambiar a una versión superior a Angular no será como migrar de Angular 1. No será una re-programación completa, se considera un simple cambio en algunas de las bibliotecas del núcleo que exigen un cambio principal de versión SEMVER[8] (CampusMVP, 2019).

Como expresa Minar (2017) uno de los core team de Angular; SEMVER permite a los desarrolladores no sólo razonar sobre cualquier actualización que se haga. Como lo demuestra la Figura 78 una versión semántica consta de tres números:

Figura 7: *Versión semántica*
Fuente: (Minar, 2017)

Cuando se corrige un error y lo liberen, aumenta el último número (patch). Si se agregan una nueva función, aumentan el segundo número (minor) y cada vez que liberan un cambio de fuerte aumentan el primer número (major).

A partir de la liberación de Angular el team de Angular ha decidido sacar versiones consecutivas utilizando el SEMVER, por lo tanto se dice que cada 6 meses habrá una versión de Angular nueva.

Desde aquel entonces se han ido publicando diversas entregas con números basados en el versionado semántico, pero siempre bajo la misma base tecnológica. Por tanto, el mencionado estudio es válido tanto si se desea conocer Angular 8.1.1 (versión

[8] **SEMVER:** Semantic Versioning se relaciona al contexto de añadir significado a los números de las versiones.

estable actualmente) o Angular (versión superior), ya que el framework en sí continúa manteniendo sus mismas bases.

1.2.4. Ventajas de desarrollar en Angular

El autor Munguía (2016) en la comparación de Angular y React JS propone las siguientes ventajas:

- **f)** Angular es un framework web robusto mantenido por Google.
- **g)** Escrito en TypeScript que es un superset de ES6 y ES5.
- **h)** Angular está orientado a componentes, por lo que cada sección de una página puede ser un componente y ser reutilizado en futuros desarrollos.
- **i)** Incluye lo necesario para construir una aplicación como manejo de formularios, Routing[9], cliente HTTP, herramientas para escribir pruebas unitarias, entre otros.
- **j)** Angular sigue un patrón MVC en el que los modelos y las vistas tienen comunicación bidireccional.
- **k)** Combinado con Ionic, se puede usar para construir aplicaciones móviles.
- **l)** Es orientado a componentes.
- **m)** Los principales editores e IDEs ofrecen ya extensiones para poder trabajar con este framework de la manera más cómoda posible.

1.2.5. Principales ventajas frente a otros framework.

En la actualidad existen diferentes frameworks que cuentan con una característica específica en el desarrollo Frontend, uno que sin duda ha sido de gran reconocimiento es el framework React JS que en sí vendría siendo una librería. Para separar dudas en la Tabla 4 se hace una comparativa básica conjuntamente con otro framework como lo es Angular JS.

[9] **Routing:** es el encargado de reconocer cuál es la ruta que el usuario desea mostrar, presentando la pantalla correcta en cada momento.

TABLA 4: Aspectos básicos frente a React y Angular JS

Característica	Angular JS	Angular	React
Versión	1.5.7	8.2.11	15.2.1
Autor	Google	Google	Facebook
Lenguaje	JS/HTML	JS/TypeScript	JS/JSX
Tamaño	143k	764k	151k
Estrellas en Github	50.5k	13.7k	45.1k
Contribuidores en Github	1501	301	741

Fuente: (Munguía, 2016)

Vinculando a la comparativa anterior, en la Tabla 5 se hace una comparativa más detallada, algo que recalcar en la comparativa de Angular y React JS; es que Angular cuenta con una buena manipulación de DOM, la misma que puede ser con bibliotecas jQuery, mientras que comparando Angular y Angular JS; es que según datos oficiales Angular puede llegar a ser hasta 5 veces más rápido.

TABLA 5: Comparativa frente a React y Angular JS

Característica	Angular JS	Angular	React
Diseño de código	JS en HTML	JS en HTML	Centrado en JS
Herramientas	Pocas	Muchas	Muchas
Curva de aprendizaje	Alta	Media	Baja
Debug	Bien HTML / Mal JS	Bien HTML / Bien JS	Mal HTML / Bien JS
¿Falla cuándo?	Ejecución	Ejecución	Compilación
Binding	2 vías	2 vías	Una dirección
Templating	HTML	TypeScript	JSX
Móvil	No tiene soporte	Ionic Framework	React Native
MVC	Sí	Sí	Solo vista
Renderizado	Cliente	Servidor	Servidor

Fuente: Elaboración Propia

Al comparar dichas evidencias se dice que Angular es framework más robusto con una curva de aprendizaje en constante crecimiento, a diferencia de React que es una librería que sin duda es ágil; sin embargo, la curva de aprendizaje no es tan alta.

1.2.6. Análisis.

Haciendo mención al artículo otorgado por Acosta (2019), Ingeniero en Informática. Especialidad en Inteligencia Artificial. Cuenta con amplia experiencia en el mundo educativo, así como en el desarrollo de proyectos web y trazabilidad multiplataforma, quién hace un análisis de lo que se debe saber acerca del framework Angular.

Sobre Angular recalca que se trata de un framework utilizado para el desarrollo de aplicaciones web de una sola página, mismo que está desarrollado en TypeScript. El propósito de Angular, es el de generar una mayor cantidad de aplicaciones basadas en un navegador, permitiendo el uso del modelo vista controlador (MVC).

En dicho artículo hace mención de los siguientes puntos, para lo cual se representa en la Tabla 6 para comprender mejor:

TABLA 6: Análisis del framework Angular

Característica	
Requerimientos e instalación del framework Angular	- Es necesario disponer de Node.js, el cual ayuda a construir software escalable. - Angular depende de las características y funcionalidades proporcionadas por bibliotecas que están disponibles al desarrollador mediante paquetes npm (Node.js Package Manager). - Angular CLI (Command Line Interface), para crear proyectos, generar códigos de aplicaciones y bibliotecas.
Utilización de Angular	- Desarrollar un Frontend con Angular es la opción elegida por una gran cantidad de desarrolladores. - El framework se caracteriza por eliminar códigos innecesarios y garantizar aplicaciones más ligeras y rápidas. Incluye plantillas, enlace con datos, modularización, gestión de API RESTful, inyección de dependencias y manejo de AJAX.
Ventajas de utilizar Angular	- Varias aplicaciones de Google también utilizan Angular, por ende su estabilidad parece garantizada. - Un frontend con Angular se crea utilizando el lenguaje TypeScript; es un superscript para JavaScript, que garantiza una mayor seguridad.

	- Las pruebas son simples. Con la separación de módulos se puede cargar los servicios necesarios, mientras se realizan las pruebas automáticas.
Angular nos proporciona estructura modular y consistencia de código	- Angular organiza código en buckets. Que pueden ser componentes, directivas, pipes (tuberías) o servicios. - Estos buckets se pueden definir como módulos que facilitan la organización de la funcionalidad de la aplicación, segregándola en características y fragmentos reutilizables. - El framework angular se basa en componentes, que comienzan de la misma forma.

Fuente: (Acosta, 2019)

Las tendencias de búsqueda que se observan en la Figura 8 trata sobre los frameworks Frontend, misma que fue obtenida de la herramienta Google Trends[10], se muestra el crecimiento del interés en este framework con respecto a otros que han tenido interés por la comunidad de desarrolladores.

El gráfico sin duda hace notar la considerable acogida que ha tenido el framework Angular con los desarrolladores, Angular sin duda se adapta a equipos grandes y pequeños, además, se aprende más fácil; así como del soporte y resolución de errores que brinda la comunidad de Angular. A diferencia de React en el que han presentado dificultad para dominarlo. Vale la pena considerar Vue JS que de a poco ha crecido sin embargo no existe gran cantidad de documentación de apoyo a los desarrolladores (Elrom, 2020).

[10] **Google Trends:** Herramienta que representa con cuánta frecuencia se realiza una búsqueda de un término particular en varias regiones del mundo.

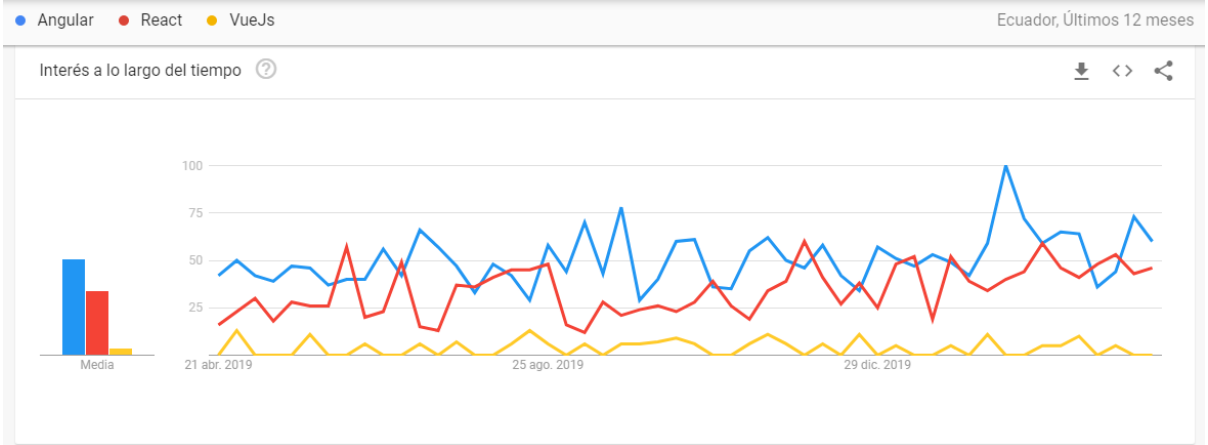


Figura 8: Búsqueda de frameworks Frontend, revisado el 20-04-2020
Fuente: Obtenido de Google Trends

- **Conclusión de la situación actual del framework Angular.**

Reusabilidad: La estructura de Angular basada en componentes hace que los componentes sean altamente reutilizables en toda la aplicación.

Pruebas unitarias simplificadas: Al ser independientes entre sí, los componentes hacen que las pruebas unitarias sean mucho más fáciles.

Sin embargo, no cumple todas las normas establecidas. Angular requiere que los desarrolladores dividan una aplicación en diferentes componentes de MVC y construyan un código que pueda unirlos. Solamente se pide dividir la aplicación y Angular se encarga de todo lo demás

Mejor legibilidad: La coherencia en la codificación hace que la lectura del código sea una tarea fácil para los nuevos desarrolladores. Lo que aumenta su productividad.

Facilidad de mantenimiento: Los componentes pueden ser reemplazados con mejores implementaciones. En pocas palabras, permite el mantenimiento y actualización eficiente del código.

En el Anexo B se detalla la documentación técnica del Framework Angular, adicional, se muestran los pasos necesarios para poder empezar a desarrollar. Los literales o pasos a seguir son interpretados por el autor.

1.3. Patrón de diseño MVC.

El patrón de diseño MVC separa el software en tres partes, es decir, modelo, vista y controlador. Se define como una arquitectura de software que separa los datos (Modelo) de la vista (Vista) y de cómo procesarlo (controlador). Con la separación en los diferentes aspectos, puede simplificar el desarrollo de sistemas complejos porque permite enfocarse más en ciertos aspectos, como se demuestra en la Figura 9 (Budiman, Puspitasari, Wati, Haviluddin, & Rahim, 2019).

Es una arquitectura importante puesto que se utiliza tanto en componentes gráficos básicos hasta sistemas empresariales; la mayoría de los frameworks modernos utilizan MVC (o alguna adaptación del MVC) para la arquitectura, entre ellos se menciona a Ruby on Rails, Django, Angular y muchos otros más (Hernandez, 2015).

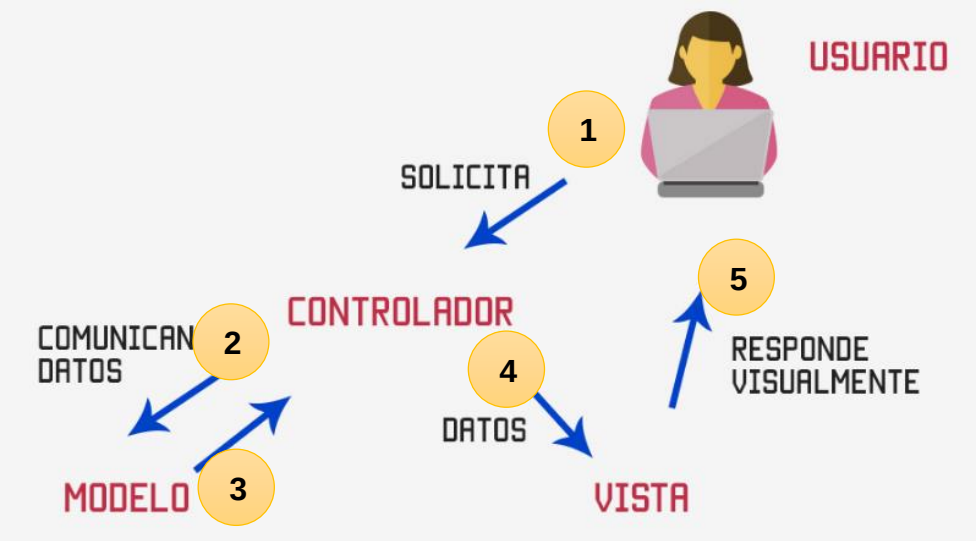


Figura 9: Arquitectura MVC
Fuente: Adaptado de (Hernandez, 2015)

1.3.1. Modelo

El modelo o capa de datos, es donde se manipulan los datos directamente con el gestor de base de datos que para el presente estudio se requiere de PostgreSQL.

1.3.2. Vista

La vista contiene las representaciones de la interfaz de usuario es decir las páginas web; mismas que pueden incluir HTML, CSS y archivos JavaScript, también se conoce como la capa de aplicación.

1.3.3. Controlador

El controlador o capa lógica, define el comportamiento de la aplicación, es el encargado de coordinar el modelo y la vista para responder las peticiones que el usuario presente.

En el sitio web de la empresa IBM[11] Knowledge Center (2019) recalca que la mayoría de las aplicaciones hoy en día siguen este patrón, muchas con ligeras variaciones. Por ejemplo, algunas aplicaciones combinan la vista y el controlador en una clase porque ya están estrechamente unidos. Todas las variaciones recomiendan enérgicamente la separación de los datos de su presentación. Esto no sólo simplifica la estructura de una aplicación sino que también permite reutilizar el código.

1.4. MVC en Angular.

Angular no tiene un MVC clásico, ya que el modelo tiene mucha relación con la vista. Por lo cual se maneja el concepto de two-way data binding, es decir, en la vista podría modificar el modelo y en el modelo podría modificar la vista.

Sin embargo la arquitectura MVC en Angular, no en un nivel teórico pero en base a la experiencia de desarrolladores backend manifiestan lo siguiente: los ***Servicios*** serían los Modelos o backend, se dispone de ***Componentes*** para la Vista y los componentes de páginas o ***Enrutadores*** serían los Controladores. Cada una de estas capas consta de varios módulos, de los cuales algunos son esenciales y otros opcionales.

En la Figura 10 se presenta la arquitectura general de Angular y como se relacionan los mismos elementos para el desarrollo de aplicaciones SPA, obtenida de la página oficial de angular (Angular, 2018).

[11] **IBM:** Empresa líder en la investigación, desarrollo y fabricación de las tecnologías de la información más avanzadas del sector.

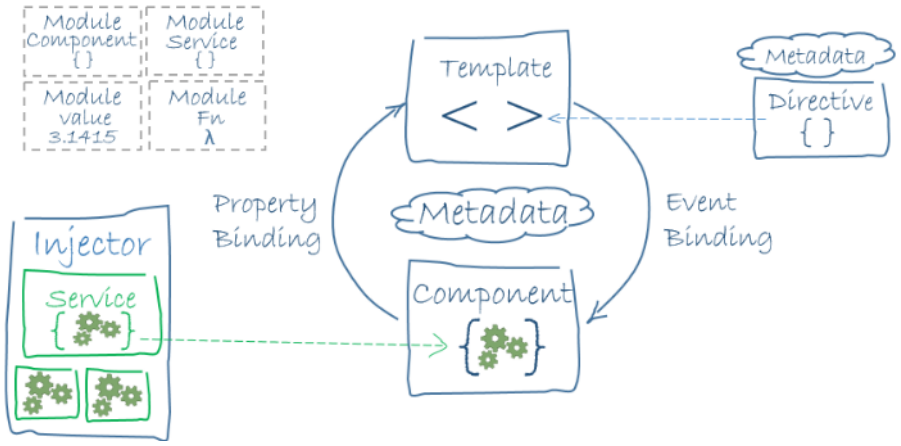


Figura 10: Arquitectura Angular
Fuente: (Angular, 2018)

En base a la Figura 10 se observa que primeramente empieza a escribir clases de componentes para administrar plantillas HTML, agregando lógica de aplicaciones en servicios y componentes, servicios en módulos. Luego, inicia la aplicación arrancando el módulo raíz. Angular asume el control, presentando el contenido de su aplicación en un navegador y respondiendo a las interacciones del usuario de acuerdo con las instrucciones que ha proporcionado (Angular, 2018).

Angular es un framework completo para construir aplicaciones de lado del cliente con HTML y TypeScript, es decir, con el objetivo de que el peso de la lógica y el renderizado lo lleve el propio navegador, en lugar del servidor (Oriol, 2019).

1.5. Herramientas para desarrollar una aplicación en Angular.

Para desarrollar en forma efectiva una aplicación en Angular y en base al tema de estudio se debe instalar al menos las siguientes herramientas básicas:

1.5.1. NodeJS 6.1.1.4 LTS

Node.js® es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Node.js usa un modelo de operaciones ECMAScript sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. El ecosistema de paquetes de Node.js, npm, es el ecosistema más grande de librerías de código abierto en el mundo (Node.js-Foundation, 2019).

NodeJS permite el desarrollo backend usando únicamente JavaScript, asume la facilidad de desarrollar servicios y API REST[12]; se acopla con Angular para el desarrollo de aplicaciones SPA, permitiendo la navegación asíncrona.

1.5.2. Google Chrome

Angular es compatible con la mayoría de navegadores como Google Chrome, Mozilla Firefox en sus últimas versiones, IE desde la versión 9, Edge desde la versión 13, Safari desde la versión 7, IOS desde la versión 7, Android desde la versión JellyBean, de los cuales el navegador Google Chrome es el recomendado al ejecutar aplicaciones hechas en dicho framework (Farina, 2015).

Entre las principales características de Angular es el soporte a navegadores y es Chrome quien posee la mayor compatibilidad para las herramientas y dependencias del framework.

1.5.3. TypeScript

TypeScript es un lenguaje de programación de código abierto desarrollado y presentado por Microsoft hace unos tres años. Es un súper conjunto de JavaScript que esencialmente añade capacidades de POO como es el tipado estático y objetos basados en clases (Quijano, 2015).

La Figura 11 hace especificación a que JavaScript que desde junio del 2015 cuenta con una nueva relación con ECMAScript 6 de la que se nutre TypeScript para añadir tipos de datos, interfaces y mejoras en la programación orientada a objetos.

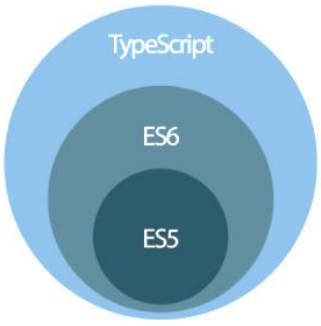


Figura 11: Concepto TypeScript
Fuente: (Quijano, 2015)

[12] **API REST:** Es un servicio que puede usarse desde cualquier dispositivo que entienda el protocolo HTTP, además interactúa directamente con la base de datos.

1.5.4. Angular CLI

Alvarez & Basalo (2018) expresan que es una de las herramientas esenciales para desarrollar con el nuevo framework Angular. Es un paso esencial y necesario antes de comenzar a ver código, puesto que se necesita de esta herramienta para poder iniciar una aplicación Angular. Angular CLI (Command Line Interface) ahorra escribir mucho código y permite partir de un esquema de aplicación avanzado y capaz de facilitar los flujos de desarrollo, depuración, testing o deploy.

Angular CLI es una herramienta oficial del propio equipo de Angular. En definitiva, facilita mucho el proceso de inicio de cualquier aplicación con Angular, ya que ofrece el esqueleto de archivos y carpetas que se necesita, junto con una cantidad de herramientas ya configuradas. Durante la etapa de producción o testing permite preparar los archivos que deben ser subidos al servidor, transpilar las fuentes, etc.

1.5.5. Visual Studio Code

Visual Studio Code es un editor de texto moderno, accesible, gratuito y open source; una herramienta personalizable, pero también productiva sin tocar el archivo de configuración. Su sistema IntelliSense[13] es muy potente y detecta las librerías importadas, muestra la definición de los métodos al ponernos encima con el cursor, además de ofrecer autocompletado al escribir.

1.5.6. Postman

Actualmente Postman dispone de una aplicación nativa, está compuesto por diferentes herramientas y utilidades gratuitas que permiten realizar tareas diferentes dentro del mundo API REST, permite hacer peticiones a APIs y generar colecciones de peticiones que permiten probarlas de una manera rápida y sencilla (Cuervo, 2019). Como se muestra en la Figura 12 un test con Postman tiene tres partes:

Descripción del Test: un texto que describe la finalidad del test.

[13] **IntelliSense:** Permiten obtener información acerca del código que utiliza, realiza un seguimiento de los parámetros que escribe y a agrega llamadas a propiedades y a métodos.

Código asociado al Test: código que utiliza el api de Postman pm para poder codificar los tests unitarios.

Resultado de la ejecución del Test: información sobre los tests que se han ejecutado correctamente, incorrectamente o que se han obviado.

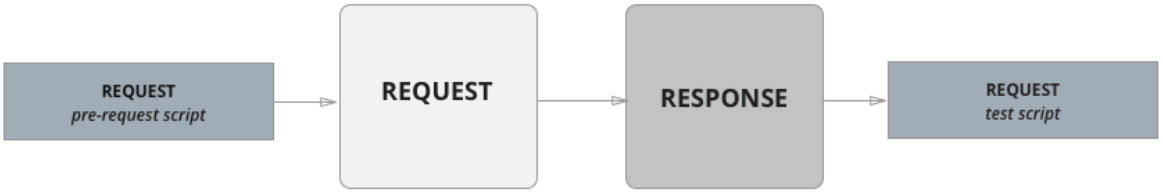


Figura 12: Concepto TypeScript
Fuente: (Quijano, 2015)

Postman es una herramienta que se convierte en indispensable como ayuda al desarrollo, permite al equipo mantener una colección actualizada de todas las llamadas de servicio o una colección que permita probar rápidamente la integración con APIs de terceros. Además, la posibilidad de exportar toda la colección en un fichero .json permite al equipo versionar la colección en el sistema de control de versiones (Redondo, 2017).

1.5.7. PostgreSQL

PostgreSQL es un sistema de gestión de base de datos objeto-relacional bajo la licencia BSD28 que permite ver y modificar el código libremente. De la misma forma este gestor de base de datos no tiene que anhelar ninguna funcionalidad a otros motores de datos pagados (Medina, 2017).

PostgreSQL maneja el modelo cliente/servidor y aplica multiprocesos para consolidar un buen rendimiento en el sistema. También PostgreSQL soporta toda clase de consultas SQL29, abarcando subconsultas, transacciones y funciones definidas por el usuario. Otra ventaja de PostgreSQL es el control de concurrencias multiversión; que permite que se realice transacciones sin bloqueos de lectura.

En el Anexo B, sección 2 se encuentra los pasos de instalación de las herramientas mencionadas.

1.6. Características Web Component con Angular CLI.

1.6.1. Componentes.

Como se detalla anteriormente sobre los Componentes se dice que las aplicaciones en Angular se desarrollan como árboles de componentes, estructurado de la siguiente manera:

a) **Nivel Raíz:** Toda aplicación parte de un componente raíz. Suele recibir el nombre de la aplicación desarrollada y el sufijo App o simplemente App.
b) **Nivel Troncal:** Generalmente dos o tres componentes troncales para la estructura de las páginas. Es común el patrón Navegador-Contenedor, con algún elemento auxiliar para ayudas, mensajes, menús complejos.
c) **Nivel de Ramas:** En este símil, las ramas equivalen a rutas o vistas de la aplicación. En un SPA cada ruta tiene una vista asociada que se carga dentro del componente troncal contenedor.
d) **Nivel de Hojas:** Cada una de las vistas está a su vez formada por múltiples componentes de negocio a modo de hojas.

1.6.2. Módulos.

Alvarez & Basalo (2018) argumenta que los árboles de componentes pueden ocultar fácilmente el bosque de una aplicación. Los módulos son agrupaciones de componentes. Ayudan a mantener un orden y a encapsular funcionalidad para crear aplicaciones desacopladas con bloques re-utilizables.

Los módulos se estructuran de la siguiente manera.

a) ***Importa*** componentes que otros módulos exportan.
b) ***Declara*** los componentes que el mismo fabrica.
c) ***Exporta*** algunos de estos componentes, para que los consuman otros módulos.

En el Anexo B, sección 4 se describe la estructura del código y archivos generados para el desarrollo de módulos y componentes del sistema a desarrollarse con Angular CLI.

1.7. Características de aplicaciones SPA.

De acuerdo con Alvarez (2016) Es un tipo de aplicación web donde todas las pantallas las muestra en la misma página, sin recargar el navegador. Técnicamente, una aplicación SPA es un sitio donde existe un único punto de entrada, generalmente el archivo index.html En la aplicación no hay ningún otro archivo HTML al que se pueda acceder de manera separada y que muestre un contenido o parte de la aplicación, toda la acción se produce dentro del mismo index.html.

1.7.1. Varias vistas

Aunque técnicamente tenga una sola página, lo que sí cuenta la aplicación es con varias vistas, se irán intercambiando vistas distintas, produciendo el efecto de que tiene varias páginas, cuando realmente todo es la misma, únicamente intercambiando vistas.

El efecto de las SPA es que cargan muy rápido sus pantallas. De hecho aunque parezcan páginas distintas, realmente es la misma página, por eso la respuesta es muchas veces instantánea para pasar de una página a otra. Otra característica es que suele comunicar con un servidor y éste le envía los datos (Alvarez, 2016).

1.7.2. Diferencia con las aplicaciones web clásicas.

Habitualmente la lógica de negocio de aplicaciones web se realiza íntegramente en el lado del servidor, y se confía la propia naturaleza del sistema de URLs el mostrar una vista de aplicación u otra. Para el navegador, cada URL diferente es completamente independiente del resto: Pese a tener los mismos estilos y/o plantillas, para la gran mayoría de páginas web dinámicas, implica que al cambiar entre vistas se sufrirá el problema de la latencia en la web (Santos & Serrano, 2017).

Para mejor comprensión se observa en la Figura 13 la diferencia entre una aplicación single-page y una aplicación tradicional.

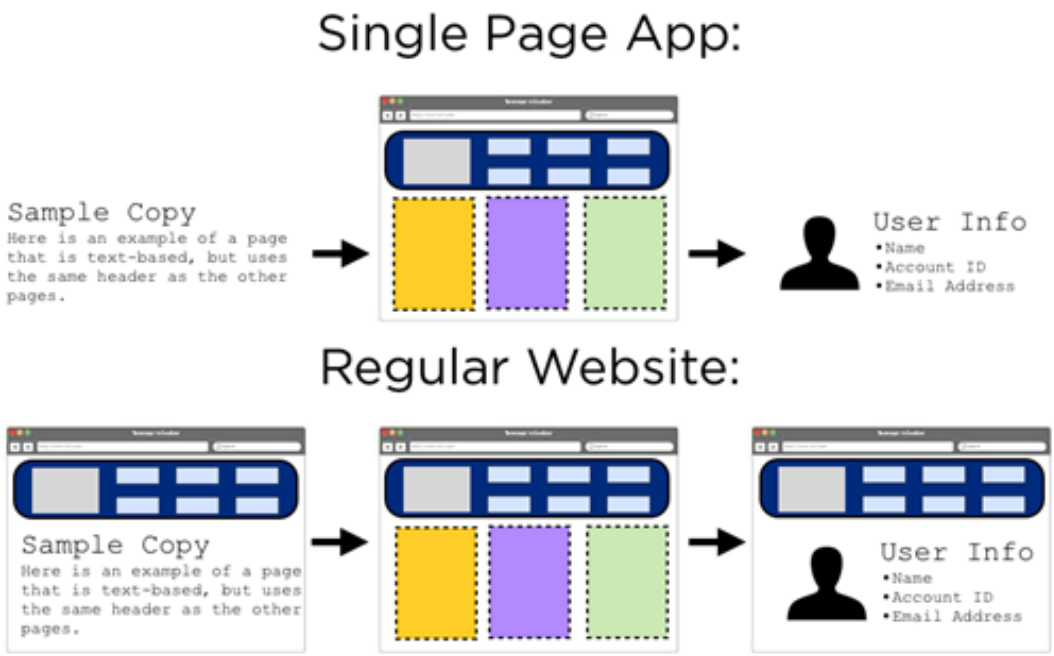


Figura 13: Lógica de aplicaciones SPA
Fuente: (Carvajal, 2017)

1.7.3. Ciclo de vida de las aplicaciones SPA

Citando a Burró (2014) menciona que una aplicación SPA podría realizar cualquier función que desarrolle una aplicación tradicional de escritorio ya que el tiempo de respuesta es mucho más rápido que el de una aplicación web tradicional.

Como se muestra en la Figura 14 la aplicación se envía al navegador y la página no se recarga durante el uso de la aplicación, es decir después de la primera solicitud o llamada, ya no vuelve a interactuar con el servidor, debido a que después de la primera solicitud el resto se producen mediante llamadas AJAX, misma que se encargan de devolver los datos necesarios.

Las SPA's gestionan toda la interactividad con el usuario desde el lado cliente, cargando recursos del servidor (vistas, datos, etc.)

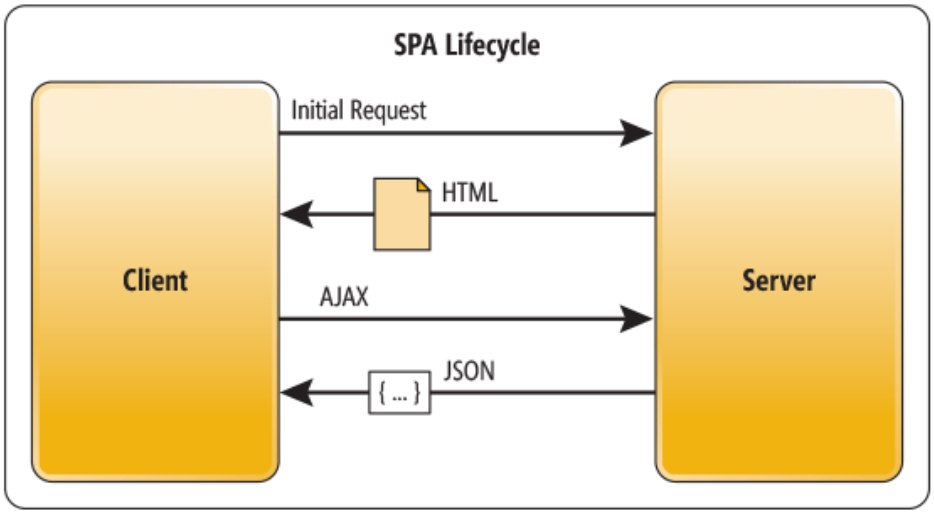


Figura 14: Ciclo de vida de aplicaciones SPA
Fuente: (Burró, 2014)

1.7.4. Ventajas de las Single Page Applications.

La ventaja de estas aplicaciones es que la mayoría de datos se encuentra disponibles vía API, permitiendo comunicación con otras aplicaciones desde el backend, que a su vez cuenta con las siguientes ventajas:

a) Reduce considerablemente las fases de trabajo en arquitectura de información por utilizar una única plantilla.
b) La plantilla única hace que también se reduzca la cantidad de código utilizado.
c) Al contar con un único diseño general se le puede dedicar todo el tiempo a optimizar los pequeños detalles y a probar todo tipo de diseños.
d) UI más rápidas.
e) Son más interactivas.
f) Son más interactivas.

1.8. Metodologías de desarrollo.

Como lo mencionan en el sitio web OK Hosting (2016) una metodología de software es una manera de interpretar la realidad o la disciplina en cuestión. De hecho, la metodología destinada al desarrollo de software se considera como una estructura utilizada para planificar y controlar el procedimiento de creación de un sistema de información especializada.

1.8.1. Metodologías de desarrollo tradicionales vs ágiles

Dichas metodologías son utilizadas para la documentación durante el ciclo de desarrollo del proyecto, el objetivo de las tradicionales es cumplir el plan de proyecto, a diferencia de las ágiles que hacen importancia a la capacidad de respuesta en los cambios, la confianza con el equipo de desarrollo, además, mantener una buena relación con el cliente.

Metodologías tradicionales conocidas como RUP y MSF cuentan con el hándicap[14] las cuales implican altos costes al implementar dichos cambios en el ciclo de desarrollo, a diferencia de las metodologías ágiles, como indica su nombre; se basan en una planificación adaptativa, permitiendo potenciar el desarrollo de software a gran escala. Entre las más destacadas son XP, Scrum, o Iconix (Gonzáles, 2017).

Las metodologías tradicionales buscan siempre una fuerte planificación y documentación durante el desarrollo, en la Figura 15 muestra la arquitectura de la metodología en cascada la misma que ha servido de base para el resto de metodologías tradicionales.

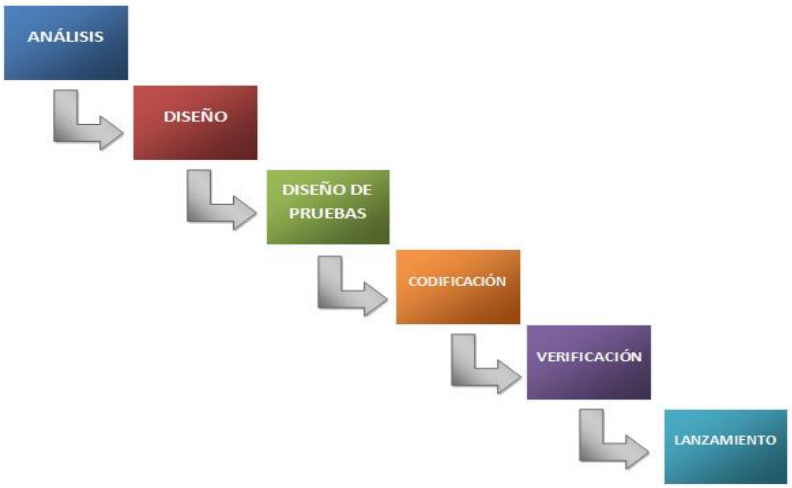


Figura 15: Metodología de desarrollo en Cascada
Fuente: (Camacho, 2016)

En la Tabla 7 se pone a consideración un cuadro comparativo, dando a conocer las principales características de las metodologías tradicionales y ágiles, en base a la experiencia de desarrolladores de software.

[14] **Hándicap:** condición en la cual se advierte una desventaja de una cosa en relación con otra.

TABLA 7: Cuadro comparativo: Metodologías tradicionales y ágiles

Metodologías Tradicionales	Metodologías Ágiles
Rigidez ante los cambios, de manera lentos o moderada	- Flexibilidad ante los cambios del proyecto de forma moderada a rápida
- Los clientes interactúan con el equipo de desarrollo mediante reuniones	- Los clientes hacen parte del equipo de desarrollo
- Grupos de gran tamaño y varias veces distribuidos en diferentes sitios	- Grupos pequeños (promedio 10 participantes in situ) en el mismo lugar.
- Dependencia de la arquitectura de software mediante modelos	- Menor dependencia de la arquitectura de software
- Poco Feedback lo que extiende el tiempo de entrega	- Continuo Feedback acortando el tiempo de entrega
- Mínimos roles	- Diversidad de roles
- Basadas en normas de estándares de desarrollo	- Basadas en heurísticas a partir de prácticas de producción de código
- Procesos muy controlados por políticas y normas	- Procesos menos controlados, pocas políticas y normas

Fuente: Elaboración Propia

1.8.2. Metodología de Desarrollo ágil SCRUM

¿Qué es SCRUM?

Scrum es una metodología ágil que sigue los principios del manifiesto ágil en cuanto a comunicación, colaboración, software funcional, auto organización, y flexibilidad ante cambios del negocio, con respecto a la metodología cuenta con 3 pilares fundamentales:

- Transparencia: toda la información que se maneja dentro del equipo es visible para todos.
- Inspección: desde un punto de vista técnico permitir una inspección por terceros.
- Adaptabilidad: adaptable a cambios o problemas en el momento de desarrollo.

Scrum está construido para ser adaptativo e iterativo en ciclos de trabajo que se llaman Sprint, al final de cada sprint los stakeholders (todas las personas interesadas

en el proyecto) y el equipo se reúnen para evaluar los incrementos de la funcionalidad del producto. En resumen, Scrum es un conjunto de roles, responsabilidades y reuniones. (Medina, 2017). La asignación de roles y responsabilidades al equipo de trabajo beneficia al desarrollo de software debido a que cada una de las partes se compromete a colaborar y realizar los trabajos específicos que abarca la metodología SCRUM.

a) **Product Owner:** Es el responsable o único dueño del producto, es decir si llegase el caso de hacer una modificación, el Product Owner es quien podrá hacer dicha modificación. Además del mantenimiento actualizado de la pila del producto en todo momento durante la ejecución del proyecto.
b) **Scrum master:** Actúa como facilitador para el Product Owner y el equipo. Remueve cualquier obstáculo que tenga el equipo para lograr las metas del sprint. Esto ayuda al equipo a permanecer creativo y productivo.
c) **Team:** El equipo es responsable de la realización del trabajo. Un equipo normalmente tiene entre 3 a 9 miembros: desarrolladores, diseñadores, testers, etc.
d) **Team:** El equipo es responsable de la realización del trabajo. Un equipo normalmente tiene entre 3 a 9 miembros: desarrolladores, diseñadores, testers, etc.

   Stakeholders: Son personas que pueden estar presentes en las reuniones del Team Scrum, pero no pueden emitir algún comentario.

Como lo menciona Nonaka & Takeuch (2019) SCRUM comprende las siguientes fases:

a) **Pre-juego**.

Planificación: Definición de una nueva versión basada en la pila actual, junto a una estimación de coste y prolongación de tiempo del proceso, En un sistema nuevo, esta fase abarca la visión del análisis, y en un caso de mejoras del sistema comprende un análisis de alcance limitado. Arquitectura: Diseño de la implementación de las funcionalidades de las tareas, incluyendo la modificación de la arquitectura y diseño general.

b) Juego.

Desarrollo de sprint: Desarrollo de las actividades de la nueva versión con respecto a las variables de tiempo, requisitos, costo y competencia. La iteración de variables define el final de esta fase, en donde el sistema va evolucionando mediante las múltiples iteraciones de desarrollo de sprint.

c) Post-juego.

Test: Pruebas antes del lanzamiento de la versión del sistema. Preparación para el lanzamiento de la versión final, incluyendo la documentación.

El modelo Scrum sugiere que el proyecto va a progresar por medio de una serie de sprints, esto mantiene la metodología ágil, los sprints son bloques de tiempo de no más de un mes, generalmente duran entre 2 a 4 semanas.

- Al principio del sprint se hace una reunión de planeación en donde los miembros del equipo, determinan la cantidad de items a desarrollar y crean el sprint backlog, priorizando las tareas.
- Una historia de usuario terminada implica que esta haya sido desarrollada, probada e integrada al producto.
- Cada día del sprint los miembros asisten a una reunión que dura 15 minutos, en esta cada miembro comparte su prioridad en el día, en qué va a trabajar e identifica obstáculos que le impidan progresar.
- Al finalizar el sprint, se tiene una reunión de revisión en la que el equipo demuestra la nueva característica a los stakeholder quienes dan feedback.
- Antes de iniciar un nuevo sprint se hace una reunión de retrospectiva, donde todo el equipo se reúne e identifican errores y oportunidades del sprint que terminó.

Complementando con lo mencionado en el apartado anterior, a continuación en la Figura 16 se representa el flujo de un sprint:

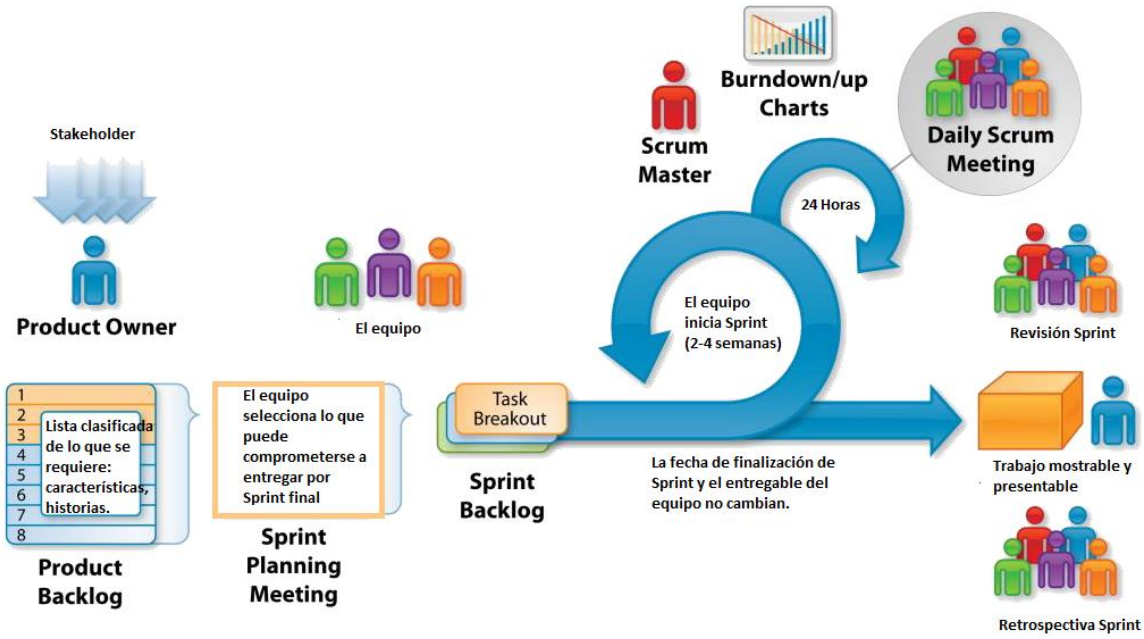


Figura 16: Flujo de Sptrint en SCRUM
Fuente: Elaboración propia

CAPÍTULO II

Desarrollo

En el presente capítulo se desarrolla e implementa el Software “Sistema de Información, Seguimiento y Control para la Hacienda Ganadera La Vega”, aplicando las fases de la metodología SCRUM, además se especifica los roles de usuario, sus módulos principales incluyendo diagramas de modelamiento para comprender la lógica de negocio.

2.1. Situación actual de la Hacienda Ganadera “La Vega”.

Actualmente la Hacienda “La Vega” está ubicada en la parroquia de San Pablo de Lago, cantón Otavalo. Se dedica a la actividad ganadera, específicamente al ganado lechero, tiene una producción diaria de 900 litros con un total de 120 cabezas de ganado entre vacas de producción, vacas secas, vaconas y terneras; ubicadas en 20 hectáreas de pasto natural.

Los registros que lleva la administración de la hacienda son: registros de ganado, reproductivos, potreros, inseminación, sanitarios e historiales de producción; dicha información se registra de forma manual y se almacena de manera insegura, por lo cual la administración de la hacienda ha visto la necesidad de adquirir un software que permita automatizar el proceso de administración y sistematizar su información.

2.2. Pre-juego

Para la planificación de los Sprint, la metodología de Scrum nos brinda los artefactos que ayudaran a la realización de las tareas programadas.

2.2.1. Roles de la Aplicación

Roles y responsabilidades del equipo de trabajo en el desarrollo de software en base a la metodología SCRUM. La Tabla 8 describe a las personas que están relacionadas con el proyecto:

TABLA 8: Asignación de Roles del Sistema

ROL	NOMBRE	DESCRIPCIÓN
Propietario del Proyecto (Product Owner)	Ing. Pedro Granda	Director de Tesis
Jefe de Proyecto (Scrum Master)	Patricio Farinango	Tesista
Equipo de desarrolladores (Team)	Patricio Farinango	Tesista

Fuente: Elaboración Propia

2.2.2. Product Backlog

Es el equivalente a los requisitos del sistema que ayuda a establecer las funcionalidades de la misma tal como se describe en la Tabla 9:

TABLA 9: Product Backlog del Sistema

N°	Tarea	Estimación	Responsable
P1	Análisis y estructuración de la base de datos y requerimientos del sistema		Patricio Farinango
P2	Análisis y estructuración de las herramientas para Backend y Frontend		Patricio Farinango
P3	Instalación y configuración de las herramientas del lado Backend		Patricio Farinango
P4	Análisis de API's, desarrollo y validaciones de Registro y Login del sistema del lado Backend		Patricio Farinango
P5	Análisis y desarrollo de seguridades del Usuario en el sistema del lado Backend		Patricio Farinango
P6	Instalación y configuración de las herramientas del lado Frontend		Patricio Farinango
P7	Módulo de seguridades del Registro y Login al sistema del lado Frontend		Patricio Farinango
P8	Desarrollo del Perfil Usuario y Panel de administración del sistema del lado Frontend		Patricio Farinango

P9	Módulo usuario y roles		Patricio Farinango
P10	Módulo de ganadería		Patricio Farinango
P11	Módulo de reproducción de ganadería		Patricio Farinango
P12	Módulo de sanidad de ganadería		Patricio Farinango
P13	Módulo de potreros		Patricio Farinango

Fuente: Elaboración Propia

2.2.3. Arquitectura de la interfaz.

La Figura 17 hace énfasis de manera general a la interfaz utilizada en el acceso al sistema, y a la vez los diferentes roles necesarios.

Figura 17: Arquitectura de la Interfaz de acceso al sistema
Fuente: Elaboración propia

- **Rol Administrador del Sistema:** Este rol se encarga de las altas y bajas de usuarios registrados, además, de los componentes del sistema.
- **Rol Administrador de la Hacienda:** Este rol se encarga de la administración de la hacienda en el sistema, así, por ejemplo:
  a) Ingreso de nuevas cabezas de ganado.
  b) Ingreso de nuevos usuario colaboradores o empleados para usar el sistema.
  c) Edición y creación de reportes de la hacienda acorde a la necesidad.
  d) Edición de ciertos datos de la hacienda.
  e) Aprobación de compras y ventas de recursos para la hacienda.
  f) Visualización de reportes.

- **Rol Usuario Colaborador del Sistema:**
  a) Editar datos del perfil.
  b) Ingreso de datos para creación de reportes.
  c) Ingreso de datos para cálculos tales como producción diaria de leche, edades, genética del ganado.
  d) Ingreso de datos para registros del sistema como inseminación, sanidad, alimentación, entre otros.
  e) Visualización de reportes.

2.2.4. Sprint Backlog

Recursos y costos asignados para los Sprints

Debido a que el presente sistema a desarrollarse es un proyecto de titulación; los recursos asignados para el sistema son el estudiante Tesista y el computador que se usa para realizarlo.

Sprint 1: Análisis y estructuración de la base de datos y requerimientos del sistema

Como se especifica en la Tabla 10, el Sprint 1 realiza las tareas de requerimientos del sistema, además de la inicialización del sistema como la instalación y configuración del motor de base de datos, dicho sprint definirá las bases de desarrollo del sistema.

- **Priorización**

Las tareas para el presente Sprint son priorizadas acorde al proceso del sistema en base a la línea de desarrollo, por lo cual las tareas S1, S3 y S4 tienen prioridad alta ya que son bases que permiten el análisis del funcionamiento del sistema, seguidamente la tarea S2; que tiene prioridad media ya que es necesaria para asegurar la integridad de los datos al evitar repeticiones innecesarias.

TABLA 10: Especificación del Sprint 1

[P1] Análisis y estructuración de la base de datos y requerimientos del sistema					
N°	**Sprint**	**Tarea**	**Responsable**	**Estimación**	**Observación**
S1	1	Instalación y configuración de las herramientas para diseño y administración de la Base de Datos	Patricio Farinango		Realizado
S2	1	Creación y Diseño Físico de la Base de Datos del sistema.	Patricio Farinango		Realizado
S3	1	Creación de tablas de acuerdo a los requerimientos de la aplicación, relación con claves primarias y foráneas	Patricio Farinango		Realizado
S4	1	Validación de la Base de Datos del sistema	Patricio Farinango		Realizado
Fecha inicio: **Fecha de culminación:**					

Fuente: Elaboración Propia

Sprint 2: Análisis y estructuración de las herramientas para Backend y Frontend

En el Sprint 2 realiza un análisis previo de las herramientas a utilizar tanto del lado backend como del Frontend, la Tabla 11 describe como del lado del backend se utilizará las herramientas NodeJS y Express. El tiempo de ejecución está diseñado para ser utilizado fuera del contexto de un navegador, mientras que Express es el framework de NodeJS más popular para desarrollar aplicaciones.

Del lado del Frontend se utilizará en framework del tema de estudio, que es Angular que permite crear y mantener páginas web en una sola página, conjuntamente con la línea de comandos Angular CLI para facilitarnos al momento de empezar a desarrollar.

- **Priorización**

Las tareas para el presente Sprint mantienen son de gran importancia para el desarrollo del sistema, la tarea S6 tiene prioridad alta ya que es el tema de estudio y es necesario dar a conocer todos sus beneficios y las herramientas que brinda el framework Angular, seguidamente la tarea S5 tiene prioridad media ya que trata la implementación de herramientas que se acoplan mejor al framework pero no quiere

decir que sean la únicas herramientas con las que se puede desarrollar del lado del backend.

TABLA 11: Especificación del Sprint 2

[P2] Análisis y estructuración de las herramientas para el desarrollo del sistema Backend y Frontend					
N°	**Sprint**	**Tarea**	**Responsable**	**Estimación**	**Observación**
S5	2	Investigación y análisis las herramientas Nodejs y Express del lado Backend	Patricio Farinango		Realizado
S6	2	Investigación y análisis las herramientas Angular y Angular CLI del lado Frontend	Patricio Farinango		Realizado
Fecha inicio: **Fecha de culminación:**					

Fuente: Elaboración Propia

Sprint 3: Instalación y configuración de las herramientas del lado Backend

La Tabla 12 describe al Sprint 3, en el cual se realiza las tareas de instalación y configuración de dependencias necesarias para el entorno de desarrollo, además, se genera la estructura de la aplicación express conjuntamente con el servidor NodeJS y conexión con la base de datos PostgreSQL. Una vez realizada la conexión se procede a la creación de los controladores de los objetos REST y seguidamente crear las rutas.

- **Priorización**

Las tareas para el presente Sprint son priorizadas acorde al proceso de desarrollo del sistema, por lo cual las tareas S7, S9 y S11 tienen prioridad alta debido a que son bases esenciales para iniciar con el desarrollo del backend, las tareas S10 y S12 tienen prioridad media ya que es el resultado de la ejecución correctas de las tareas con prioridad alta, y por último la tarea S8 con prioridad mínima la cual da a conocer la estructura del proyecto backend.

TABLA 12: Especificación del Sprint 3

[P3] Instalación y configuración de las herramientas para el desarrollo del sistema del lado Backend					
N°	**Sprint**	**Tarea**	**Responsable**	**Estimación**	**Observación**
S7	3	Instalación y configuración de dependencias	Patricio Farinango		Realizado
S8	3	Análisis del entorno de desarrollo	Patricio Farinango		Realizado
S9	3	Conexión con la base de datos	Patricio Farinango		Realizado
S10	3	Iniciando servidor web con NodeJS y Express	Patricio Farinango		Realizado
S11	3	Análisis y creación de controladores de objetos RESTful (POST, GET, PUT y DELETE)	Patricio Farinango		Realizado
S12	3	Crear Rutas	Patricio Farinango		Realizado
Fecha inicio: **Fecha de culminación:**					

Fuente: Elaboración Propia

Sprint 4: Análisis de API's, desarrollo y validaciones de Registro y Login del sistema del lado Backend

El Sprint 4 realiza las tareas de validación en el registro y login, como se describe en la Tabla 13 dicho sprint analizará los objetos JSON que se consumen desde el Frontend del sistema.

- **Priorización**

Las tareas para el Sprint 4 son priorizadas acorde al proceso del sistema en base a la línea de desarrollo, por lo cual la tarea S15 se considera prioridad alta ya que es parte del funcionamiento inicial del sistema, las tareas S13, S14 y S15 tienen prioridad media debido que permiten el análisis del funcionamiento del Product Backlog anterior.

TABLA 13: Especificación del Sprint 4

[P4] Análisis de API's, desarrollo y validaciones de Registro y Login del sistema del lado Backend					
N°	**Sprint**	**Tarea**	**Responsable**	**Estimación**	**Observación**
S13	4	Testeo de API´s REST mediante el software Postman	Patricio Farinango		Realizado
S14	4	Análisis de los Objetos JSON	Patricio Farinango		Realizado
S15	4	Validaciones Registro y Login	Patricio Farinango		Realizado
S16	4	Otras validaciones requeridas	Patricio Farinango		Realizado
Fecha inicio: **Fecha de culminación:**					

Fuente: Elaboración Propia

Sprint 5: Análisis y desarrollo de seguridades del Usuario en el sistema del lado Backend

El Sprint 5 realiza las tareas de autenticación utilizando JWT y creación de middleware, la Tabla 14 describe los Sprint quienes definirán la seguridad en la autenticación del usuario en el sistema.

- **Priorización**

Las tareas para el presente Sprint son priorizadas acorde al proceso del sistema en base a la línea de desarrollo y necesarias para un buen funcionamiento, por lo cual las tareas S17, S18 y S19 son prioridad alta ya que conjuntamente brindar una óptima seguridad de acceso y administración al sistema. La tarea S20 tiene prioridad media ya que es necesaria para asegurar las cabeceras HTTP y CORS.

TABLA 14: Especificación del Sprint 5

[P5] Análisis y desarrollo de seguridades del Usuario en el sistema del lado Backend					
N°	**Sprint**	**Tarea**	**Responsable**	**Estimación**	**Observación**
S17	5	Creación Tokens de Autenticación con JWT	Patricio Farinango		Realizado
S18	5	Creación Middleware para Autenticación	Patricio Farinango		Realizado

S19	5	Creación Middleware para Roles	Patricio Farinango		Realizado
S20	5	Configuración de cabeceras HTTP y acceso CORS	Patricio Farinango		Realizado
Fecha inicio: **Fecha de culminación:**					

Fuente: Elaboración Propia

Sprint 6: Instalación y configuración de las herramientas del lado Frontend

La Tabla 15 describe al Sprint 6 el cual se encarga del análisis, instalación y configuración de las herramientas para el lado del Frontend como lo es el framework Angular y el complemento Angular CLI que permite generar utilidades de manera automática desde la línea de comandos.

- **Priorización**

Las tareas para el presente Sprint son priorizadas acorde al proceso del sistema en base a la línea de desarrollo, por lo cual las tareas S21, S22, S23, S24 y S25 tienen prioridad alta ya que son necesarios; desde la instalación de dependencias como el análisis del archivo package.json además del análisis de la estructura del entorno de desarrollo, la de creación de módulos y componentes básicos. Dichas tareas son bases que permiten el funcionamiento del correcto del sistema en base a los lineamientos del tema de estudio, seguidamente la tarea S26 y S27 tiene prioridad media ya que permiten el direccionamiento y diseño del sistema.

TABLA 15: Especificación del Sprint 6

[P6] Instalación y configuración de las herramientas para el desarrollo del sistema del lado Frontend					
N°	**Sprint**	**Tarea**	**Responsable**	**Estimación**	**Observación**
S21	6	Instalación de dependencias	Patricio Farinango		Realizado
S22	6	Análisis y configuración del archivo package.json	Patricio Farinango		Realizado
S23	6	Análisis e Instalación de módulos	Patricio Farinango		Realizado
S24	6	Análisis del entorno de desarrollo	Patricio Farinango		Realizado

S25	6	Análisis y creación de Components	Patricio Farinango		Realizado
S26	6	Routing y Páginas	Patricio Farinango		Realizado
S27	6	Bootstrap y Maquetación Web	Patricio Farinango		Realizado
Fecha inicio: **Fecha de culminación:**					

Fuente: Elaboración Propia

Sprint 7: Módulo de seguridades del Registro y Login al sistema del lado Frontend

El Sprint 7 realiza las tareas de inicialización del sistema como el control de acceso o login en base a las credenciales del usuario, dicho sprint define el inicio del sistema, como detalla la Tabla 16.

- **Priorización**

Las tareas para el presente Sprint son priorizadas acorde al proceso del sistema en base a la línea de ejecución del usuario, las actividades del sprint tienen similitud en el grado de importancia ya que todas trabajan conjuntamente con la finalidad de brindar seguridad en el registro y acceso al sistema.

TABLA 16: Especificación del Sprint 7

[P7] Módulo de seguridades del Registro y Login al sistema del lado Frontend					
N°	**Sprint**	**Tarea**	**Responsable**	**Estimación**	**Observación**
S28	7	Desarrollo de la página Principal	Patricio Farinango		Realizado
S29	7	Desarrollo de la página de Registro y Login de Usuarios	Patricio Farinango		Realizado
S30	7	Autenticación y Login de Usuarios	Patricio Farinango		Realizado
S31	7	Logout de la aplicación	Patricio Farinango		Realizado
Fecha inicio: **Fecha de culminación:**					

Fuente: Elaboración Propia

Sprint 8: Desarrollo del Perfil Usuario y Panel de administración del sistema del lado Frontend

La Tabla 17 describe al Sprint 8, mismo que realiza las tareas de carga de imágenes, además, de realizar un panel de administración con el fin de modificar el perfil del usuario.

- **Priorización**

Las tareas para el presente Sprint se priorizan acorde al proceso de desarrollo del sistema, por lo cual las tareas S35 y S36 tienen prioridad alta ya que son necesarios para la carga y preview de imágenes. Dichas tareas son bases que permiten el funcionamiento del correcto del sistema en base a los lineamientos del tema de estudio, seguidamente la tarea S33 y S34 tiene prioridad media ya que forman parte de la funcionalidad adicional del sistema.

TABLA 17: Especificación del Sprint 8

[P8] Desarrollo del Perfil Usuario y Panel de administración del sistema del lado Frontend					
N°	**Sprint**	**Tarea**	**Responsable**	**Estimación**	**Observación**
S33	8	Sección Perfil de Usuario	Patricio Farinango		Realizado
S34	8	Desarrollo del Panel de Administración	Patricio Farinango		Realizado
S35	8	Configuración de Pipe para imagen de usuarios	Patricio Farinango		Realizado
S36	8	Subir imagen desde el frontend hacia el backend	Patricio Farinango		Realizado
Fecha inicio: **Fecha de culminación:**					

Fuente: Elaboración Propia

Sprint 9: Módulo usuario y roles

El Sprint 9 realiza las tareas de crear, editar y eliminar registros de usuarios, tanto del backend como del Frontend, también la carga de imágenes, dicho desarrollo se detalla en la Tabla 18.

- **Priorización**

Las tareas elegidas para el presente Sprint fueron tomadas en cuenta acorde al proceso de desarrollo del sistema, el nivel de priorización es alta ya que como en los anteriores Sprints son continuaciones del login y perfil de usuarios, con dicho sprint termina el módulo de login y administración de usuarios.

TABLA 18: Especificación del Sprint 9

[P9] Módulo usuario y roles					
N°	**Sprint**	**Tarea**	**Responsable**	**Estimación**	**Observación**
S37	9	CRUD's de usuarios desde el backend	Patricio Farinango		Realizado
S38	9	Testeo de API´s REST mediante el software Postman	Patricio Farinango		Realizado
S39	9	CRUD's de usuarios desde el frontend y carga de imágenes	Patricio Farinango		Realizado
S40	9	Búsqueda de usuarios	Patricio Farinango		Realizado
S41	9	Carga de imagen de usuario	Patricio Farinango		Realizado
Fecha inicio: **Fecha de culminación:**					

Fuente: Elaboración Propia

Sprint 10: Módulo de ganadería

Tal como se aprecia en la Tabla 19 el Sprint 10 realiza las tareas de crear, editar y eliminar registros de ganadería, tanto del backend como del Frontend, así también la carga de imágenes, información individual, crías y desplegar catálogo de padres disponibles. Adicionalmente se prevé desplegar información relacionada a la producción de leche individual y total.

- **Priorización**

Las tareas elegidas para el presente Sprint fueron consideradas acorde al proceso de desarrollo del sistema, el nivel de priorización es alta ya que son primordiales para el proceso de negocio y requerimientos del sistema.

TABLA 19: Especificación del Sprint 10

[P10] Módulo de ganadería					
N°	**Sprint**	**Tarea**	**Responsable**	**Estimación**	**Observación**
S42	10	CRUD's del ganado desde el backend	Patricio Farinango		Realizado
S43	10	Testeo de API´s REST mediante el software Postman	Patricio Farinango		Realizado
S44	10	CRUD's del ganado desde el frontend y carga de imágenes	Patricio Farinango		Realizado
S45	10	Información individual, crías y catálogo de padres	Patricio Farinango		Realizado
S46	10	Registro de etapas	Patricio Farinango		Realizado
47	10	Registros de producción individuales y totales	Patricio Farinango		Realizado
Fecha inicio: **Fecha de culminación:**					

Fuente: Elaboración Propia

Sprint 11: Módulo de reproducción de ganadería

El Sprint 11 realiza las tareas de crear, editar y eliminar registros de reproducción de ganadería, tanto del backend como del Frontend. Adicionalmente se prevé desplegar información individual de hembras reproductoras y etapas de gestación; como muestra la Tabla 20.

- **Priorización**

Las tareas elegidas para el presente Sprint fueron consideradas acorde al proceso de desarrollo del sistema, el nivel de priorización es me ya que siguen en secuencia al módulo anterior y es necesario dicha información. S49 es considerado nivel medio ya que es un requerimiento que se lo puede realizar en base a otros métodos.

TABLA 20: Especificación del Sprint 11

[P11] Módulo de reproducción de ganadería					
N°	**Sprint**	**Tarea**	**Responsable**	**Estimación**	**Observación**
S48	11	CRUD's de reproducción desde el backend	Patricio Farinango		Realizado

S49	11	Testeo de API´s REST mediante el software Postman	Patricio Farinango		Realizado
S50	11	CRUD's de reproducciones del ganado desde el frontend	Patricio Farinango		Realizado
S51	11	Información individual de hembras reproductoras y etapas de gestación	Patricio Farinango		Realizado
Fecha inicio: **Fecha de culminación:**					

Fuente: Elaboración Propia

Sprint 12: Módulo de sanidad de ganadería

La Tabla 21 describe al Sprint 12 el cual realiza el CRUD de los registros de sanidad o aplicación de medicamentos, tanto del backend como del Frontend, desplegar información de los medicamentos aplicados, así también los registros e históricos de vacunas y tratamientos.

- **Priorización**

Las tareas elegidas para este Sprint son consideradas de mayor prioridad para el funcionamiento del sistema, son tareas que despliegan información a tomar en consideración para el usuario.

TABLA 21: Especificación del Sprint 12

[P12] Módulo de sanidad de ganadería					
N°	**Sprint**	**Tarea**	**Responsable**	**Estimación**	**Observación**
S52	12	CRUD's de registros de medicamentos desde el backend	Patricio Farinango		Realizado
S53	12	Testeo de API´s REST mediante el software Postman	Patricio Farinango		Realizado
S54	12	CRUD's de registros de medicamentos desde el frontend	Patricio Farinango		Realizado
S55	12	Información individual de medicamentos aplicados	Patricio Farinango		Realizado

S56	12	Información de registros e históricos de vacunación y tratamientos	Patricio Farinango		Realizado
Fecha inicio: **Fecha de culminación:**					

Fuente: Elaboración Propia

Sprint 13: Módulo de potreros

El Sprint 13 realiza el CRUD de los registros de potreros, que es necesario para el correcto funcionamiento del Sprint 10 [P10], la Tabla 22 muestra información relacionada al potrero, ubicación, tiempo de recuperación y carga de imágenes.

- **Priorización**

Las tareas elegidas para este Sprint son consideradas de mayor prioridad para el funcionamiento del sistema, son tareas que despliegan información a tomar en consideración para el módulo de ganadería y el registro de información de otros módulos.

TABLA 22: Especificación del Sprint 13

[P13] Módulo de potreros					
N°	**Sprint**	**Tarea**	**Responsable**	**Estimación**	**Observación**
S57	13	CRUD's de registros de potreros desde el backend	Patricio Farinango		Realizado
S58	13	Testeo de API´s REST mediante el software Postman	Patricio Farinango		Realizado
S59	13	CRUD's de registros de potreros desde el Frontend y carga de imágenes	Patricio Farinango		Realizado
S60	13	Información individual de potreros y etapas recuperación	Patricio Farinango		Realizado
Fecha inicio: **Fecha de culminación:**					

Fuente: Elaboración Propia

2.2.5. Resumen de la planificación de Sprint

La Tabla 23 describe de manera general la planificación del proyecto, tomando en cuenta las fechas de inicio y fin; así también los recursos y requerimientos necesarios para el desarrollo de cada Sprint.

TABLA 23: Planificación general del proyecto

ID	Fecha Inicio	Fecha Fin	Nombre Sprint	Recursos	Requerimientos
P1	05-Febrero-2018	23-Febrero-2018	Sprint 1	Tesista, recursos oficina	S1, S2, S3, S4
P2	26-Febrero-2018	23-Marzo-2018	Sprint 2	Tesista, computadora	S5, S6
P3	26-Marzo-2018	06-Abril-2018	Sprint 3	Tesista, computadora	S7, S8, S9, S10, S11, S12
P4	09-Abril-2018	27-Abril-2018	Sprint 4	Tesista, computadora	S13, S14, S15, S16
P5	30-Abril-2018	11-Mayo-2018	Sprint 5	Tesista, computadora	S17, S18, S19, S20
P6	14-Mayo-2018	08-Junio-2018	Sprint 6	Tesista, computadora	S21, S22, S23, S24, S25, S26, S27
P7	11-Junio-2018	29-Junio-2018	Sprint 7	Tesista, computadora	S28, S29, S30, S31, S32
P8	02-Julio-2018	20-Julio-2018	Sprint 8	Tesista, computadora	S33, S35, S36
P9	23-Julio-2018	10-Agosto-2018	Sprint 9	Tesista, computadora	S37, S38, S39, S40, S41
P10	13-Agosto-2018	31-Agosto-2018	Sprint 10	Tesista, computadora	S42, S43, S44, S45, S46, S47
P11	03-Septiembre-2018	21-Septiembre-2018	Sprint 11	Tesista, computadora	S48, S49, S50, S51
P12	24-Septiembre-2018	12-Octubre-2018	Sprint 12	Tesista, computadora	S52, S53, S54, S55, S56
P13	15-Octubre-2018	31-Octubre-2018	Sprint 13	Tesista, computadora	S57, S58, S59, S60

Fuente: Elaboración Propia

2.3. Juego (Desarrollo de Sprints)

La implementación de los Sprint da a conocer la programación, interfaces y el acceso a datos que utiliza cada tarea para poder generar las pruebas entregables en base a cada Sprint.

2.3.1. Esquema de la solución del sistema.

Es necesario conocer la estructura de la implementación del sistema para así poder comprender, interpretar y mantener cada sprint en secuencia a la funcionalidad de dicho sistema, la solución del mismo se basa en la arquitectura SOA como se observa en la Figura 18 donde el cliente o usuario consume servicios publicados por el servidor, el cual maneja toda la lógica el negocio y el acceso a datos.

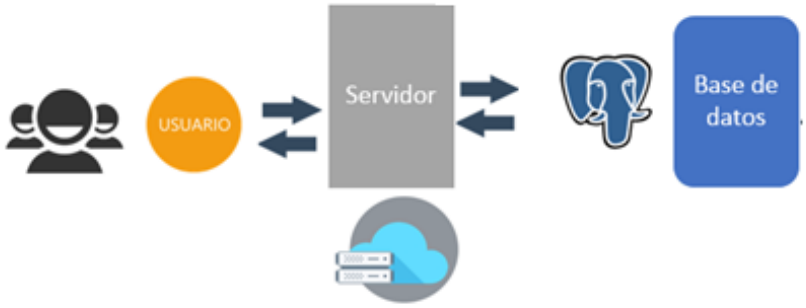


Figura 18: Esquema de la solución del sistema
Fuente: Elaboración propia

2.3.2. Diseño de la base de datos.

En el Anexo C se define el modelo físico de la base de datos para el sistema de Información, seguimiento y control de la Hacienda ganadera La Vega el mismo que consta de 27 tablas relacionadas para crea la estructura de datos para el sistema.

2.3.3. Iteración 1: Análisis y estructuración de la base de datos del sistema.

Instalación y configuración de las herramientas para diseño y administración de la Base de Datos.

Descargar el motor de base de datos PosgreSQL en el enlace: https://www.postgresql.org/download/, seguidamente ejecute el instalador

manteniendo las configuraciones predeterminadas; como lo es el puerto (5432), una vez finalizada la instalación inicie PostgreSQL desde pgAdmin para administrar el entorno PostgreSQL.

Creación y Diseño Físico de la Base de Datos del sistema.

Para el análisis, diseño y construcción de la base de datos se utilizó PowerDesigner, una herramienta orientada a modelos de datos a nivel físico y conceptual. Para la instalación es necesario seguir las configuraciones predeterminadas que brinda el asistente de instalación.

Creación de tablas de acuerdo a los requerimientos de la aplicación, con sus correspondientes claves principales y foráneas.

La creación de la base de datos está diseñada a los requerimientos del cliente, en la cual se describe cada una de las tablas, sus atributos y tipos de datos.

En el Anexo C se describe el diseño y descripción de las tablas generadas para el desarrollo del sistema.

2.3.4. Iteración 2: Análisis y estructuración de las herramientas para el desarrollo del sistema Backend y Frontend.

Previo al análisis de las herramientas a utilizarse tanto para Backend como Frontend, se formó una arquitectura basada en la conocida MEAN Stack, las herramientas a utilizarse como PostgreSQL, Express.js, Angular.js y Node.js. Como describe la Figura 19 los cuatro son productos fuertemente ligados al mundo Javascript, además de ser herramientas fuertemente consolidadas y escalables para el desarrollo de aplicaciones web: Frontend, Backend y Base de Datos.

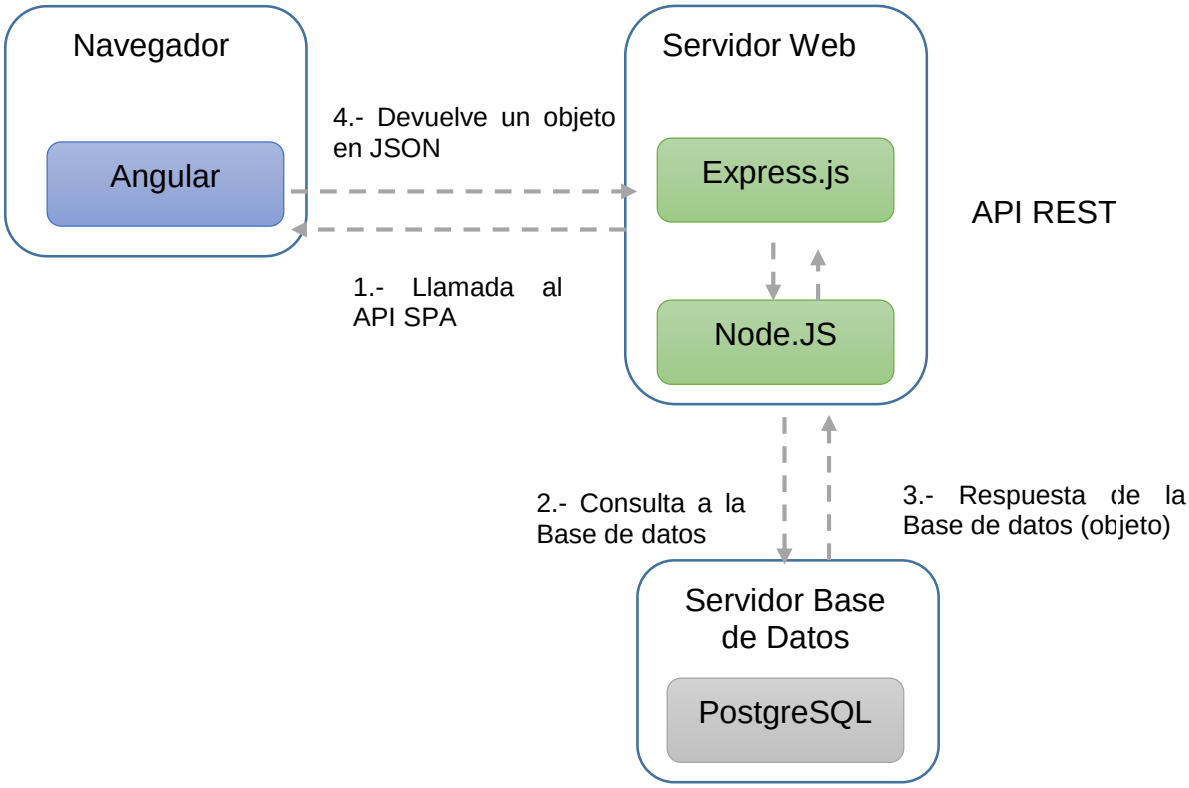


Figura 19: Flujo de las aplicaciones a utilizarse.
Fuente: Elaboración propia

2.3.5. Iteración 3: Instalación y configuración de las herramientas para el desarrollo del sistema del lado Backend.

Instalación y configuración de dependencias.

Abrir una línea de comandos cmd y ubicarse en la ruta en la cual se configurará el sistema del lado Backend, para inicializar NodeJS como se observa en la Figura 20 digitar el comando **npm init**, seguidamente confirmar la versión de instalación y la ejecución del archivo EntryPoint.

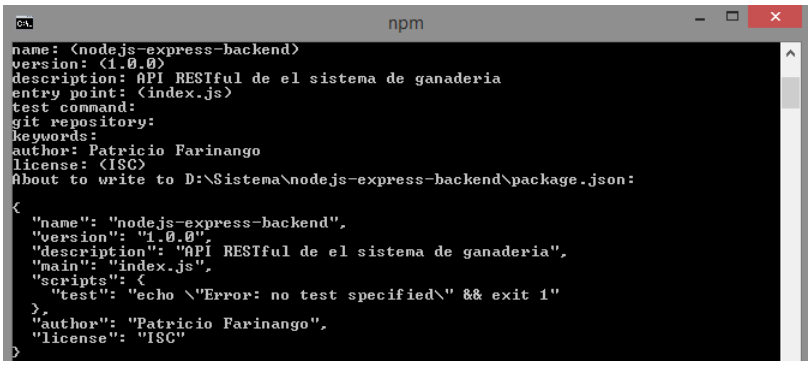


Figura 20: Configuración del archivo EntryPoint.
Fuente: Elaboración propia

Dichos pasos permiten generar el archivo package.json como muestra en la Figura 21, el cual el permite importar paquetes necesarios para el desarrollo de la aplicación. En la misma ruta de instalación de NodeJS ejecute la siguiente línea de comando: **npm install –save express** lo cual permite iniciar la descargas de las dependencias.

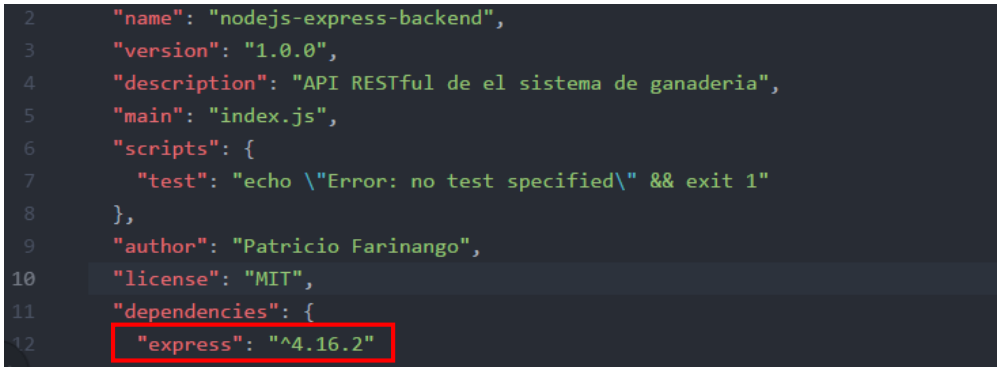


Figura 21: Instalación de Express.js
Fuente: Elaboración propia

Análisis del entorno de desarrollo.

En el directorio de la carpeta del proyecto se puede apreciar los archivos generados y los módulos disponibles para poder trabajar.

A continuación en la Tabla 24 se detalla los módulos utilizados en el backend para el desarrollo de la aplicación, los mismos que se instalan anteponiendo el comando **npm** perteneciente a NodeJS y seguido de **install**.

TABLA 24: Módulos necesarios para el desarrollo Backend

Paquete	Comando	Detalle
bcrypt (1.0.3)	npm install –save bcrypt	Permite cifrar las contraseñas
body-parser (1.18.2)	npm install –save body-parser	Convierte una petición desde cualquier formato a un .json de javascript utilizable.
connect-multiparty (2.1.0)	npm install –save connect-multiparty	Permite subir ficheros.
jwt-simple (0.5.1)	npm install –save jwt-simple	Genera tokens de jwt, para la autenticación y seguridad de la API
moment (2.19.2)	npm install –save moment	Paquete para procesar fechas y tiempo

bluebird (3.5.1)	npm install –save bluebird	ORM que brinda una serie de funcionalidades para poder trabajar con PostgreSQL dentro de NodeJS
pg-promise (5.9.7)	npm install –save pg-promise	Admite una variedad de formas en que se pueden suministrar los nombres de SQL
nodemon (1.12.1)	npm install –save-dev nodemon	Dependencia para el desarrollo, el cual reinicia el servidor automáticamente mientras se realiza cambios.(agregar la expresión –dev para instalar únicamente como dependencia de desarrollo)

Fuente: Elaboración Propia

Conexión con la base de datos.

Para establecer la conexión con la base de datos PostgreSQL desde NodeJS se necesita el ORM **pg-promise** y **blubird.**

```
var connectionString = 'postgres://postgres:postgres@localhost:5432/{nombre de la bdd};
// nombre de la base de datos
```

Iniciando servidor web con NodeJS y Express.

Se configura dentro de un nuevo archivo JavaScript en el cual se cargan los módulos registradores de solicitudes http. Seguidamente se configura las variables de entorno en la cual se asigna el entorno de producción, host y el puerto de entrada.

```
app.set("env", process.env.NODE_ENV || "development");
app.set("host", process.env.HOST || "0.0.0.0");
app.set("port", process.env.PORT || 3000);
```

Análisis y creación de controladores de objetos RESTful (POST, GET, PUT y DELETE).

La Figura 22 contiene las configuraciones para las cabeceras y CORS, esto permite que las aplicaciones cliente de otros dominios usen el API Server.

```
app.use(function(req, res, next) {
    res.header("Access-Control-Allow-Origin", '*');
    res.header("Access-Control-Allow-Headers", 'Authorization, X-API-KEY, Origin, X-Requested-With
    res.header('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OPTIONS');
    res.header('Allow', 'GET, POST, PUT, DELETE, OPTIONS');
    next();
});
```

Figura 22: Configuraciones del para crear los controladores
Fuente: Elaboración propia

Crear Rutas.

Crear un archivo (index.js) .js en el cual se cargaran las rutas y declarar una variable con la ruta del archivo en cual se alojarán las rutas, tal como se muestra en la Figura 23, seguidamente implementar el método.

```
var api = require('./api/index');
router.get('/', function(req, res, next) {
      res.status(200)
      .json({
        status: 'éxito!',
        message: 'El servidor local con NodeJS y Express esta ejecutandose!'
      });
});
```

Figura 23: Configuraciones del para crear las rutas
Fuente: Elaboración propia

Una vez configurada la conexión de bdd y la del servidor express, ejecute el comando **npm start**, si las configuración son validadas entonces se iniciará el servidor Express, y por ende se puede hacer uso de las APIs.

En el Anexo A, sección Conexión se describe todo el código y configuraciones necesarias.

2.3.6. Iteración 4: Análisis de APIs, desarrollo y validaciones de registro y login del sistema del lado Backend.

De acuerdo con el desarrollo del proyecto se realizar un test a las rutas creadas, mediante la herramienta Postman, como se aprecia en la Figura 24 dicha herramienta devuelve un objeto en formato JSON, el cual se interpreta del lado Frontend.

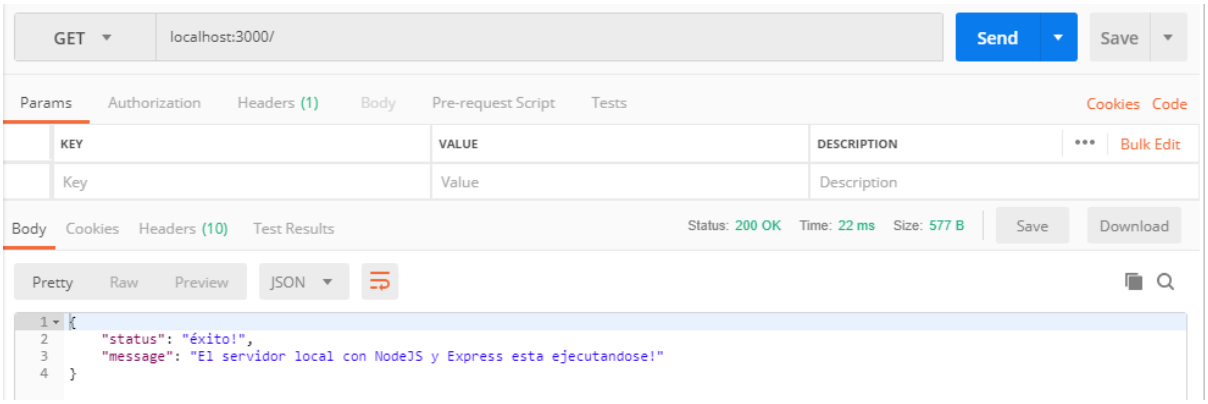


Figura 24: Test del puerto del servidor Backend
Fuente: Elaboración propia

En el Anexo A, sección módulo Registro y Login del lado Backend se describe todo el código y validaciones necesarias.

2.3.7. Iteración 5: Análisis y desarrollo de seguridades del usuario en el sistema del lado Backend.

Parte de la seguridad del sistema se da en el módulo de Login, como muestra la Figura 25 genera un Token de autenticación mediante la función creada.

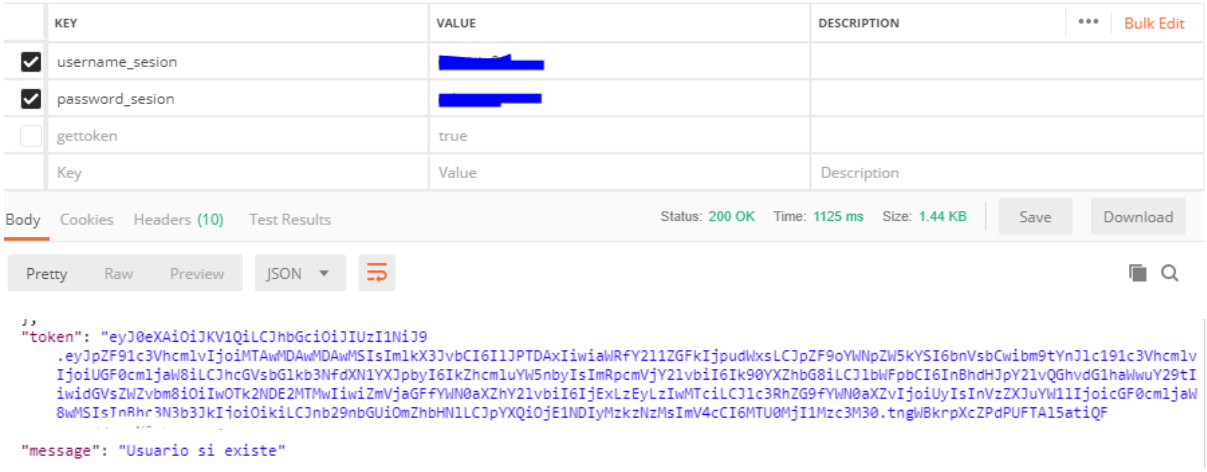


Figura 25: Token generado en base a la función Backend
Fuente: Elaboración propia

En el Anexo A, sección módulo Seguridades del usuario del lado Backend se describe todo el código, creación de middleware y validaciones necesarias.

2.3.8. Iteración 6: Instalación y configuración de las herramientas para el desarrollo del sistema del lado Frontend.

Mediante la línea de comandos cmd ubicarse en el director del proyecto, seguidamente digitar el comando ng new [nombre del proyecto] para inicializar el proyecto de Angular, como resultado obtenemos el directorio con la estructura básica como se observa en la Figura 26.

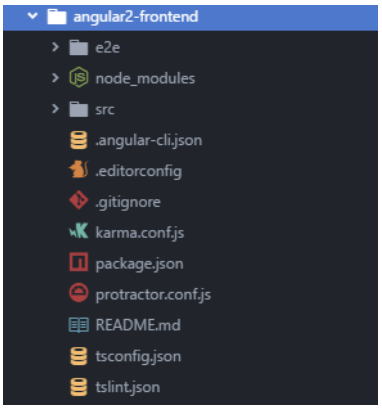


Figura 26: Estructura básica de un proyecto Angular
Fuente: Elaboración propia

Una vez revisado el archivo de configuración package.json, se procede a instalar dichos módulos, en la línea de comandos cmd digitar el comando **npm install**, por ultimo iniciar el servidor Frontend con el comando **ng serve**.

En el Anexo A, sección módulo Instalación y configuración de las herramientas Frontend se describe todo el código e instalaciones individuales de módulos.

2.3.9. Iteración 7: Módulo de seguridades del registro y login al sistema del lado Frontend.

En la presente iteración se desarrolló la interfaz principal conjuntamente con las validaciones necesarias y como muestra la Figura 27.

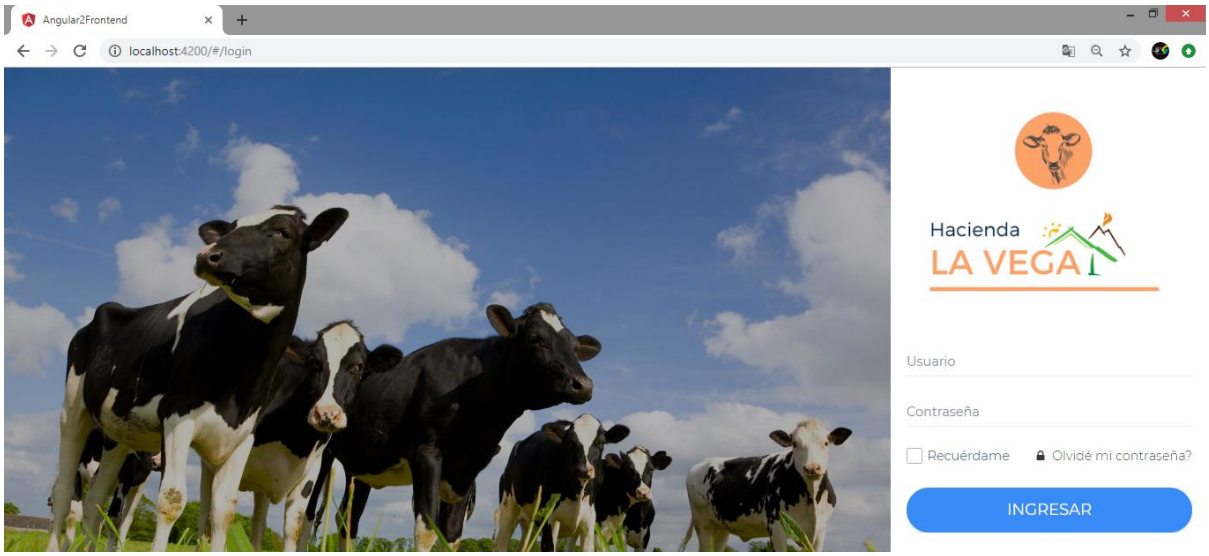


Figura 27: Interfaz principal del sistema
Fuente: Elaboración propia

En el Anexo A, sección módulo Login y Registro se describe todo el código y configuraciones de autenticación Backend y Frontend.

2.3.10. Iteración 8: Desarrollo del perfil usuario y panel de administración del sistema del lado Frontend.

En la Figura 28 se desarrolló la interfaz principal conjuntamente con las validaciones necesarias del módulo encargado del perfil de usuario y del panel de administración con funciones acorde al perfil.

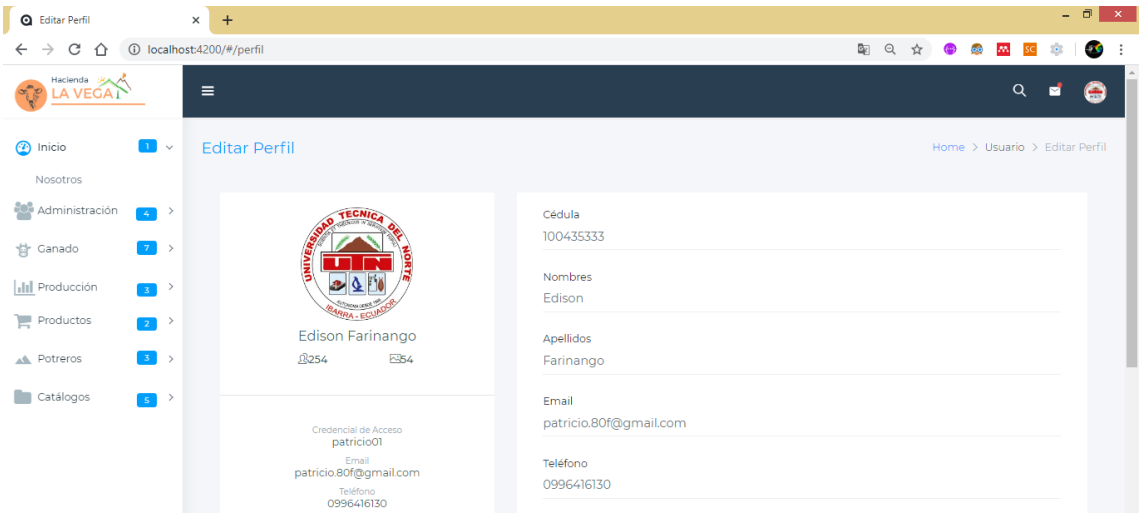


Figura 28: Interfaz principal del módulo perfil de usuario y actualizar imagen de perfil
Fuente: Elaboración propia

En el Anexo A, sección módulo Perfil de Usuario y Panel de Administración se describe todo el código y configuraciones de autenticación Backend y Frontend.

2.3.11. Iteración 9: Módulo usuario y roles.

La presente iteración tiene como resultado el módulo que permite administrar a los usuarios del sistema en base al rol, En la Figura 29 de observa la interfaz principal de la lista de usuarios.

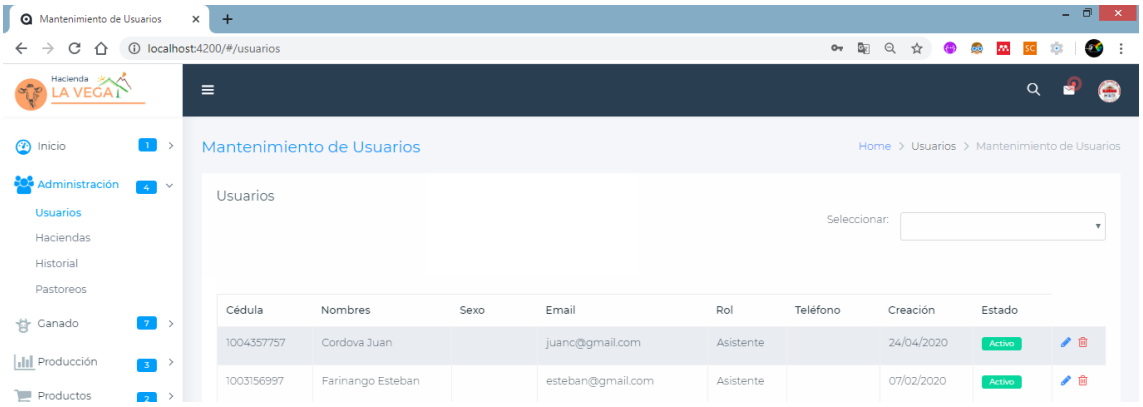


Figura 29: Interfaz principal del módulo usuario y roles
Fuente: Elaboración propia

En el Anexo A, sección módulo Usuarios y Roles se describe todo el código desarrollado para Backend y Frontend, así también se detalla los test de las APIs generadas.

2.3.12. Iteración 10: Módulo de ganadería.

En la Figura 30 de observa como resultado la interfaz actualizar del módulo, el cual permite llevar un registro diario del ganado.

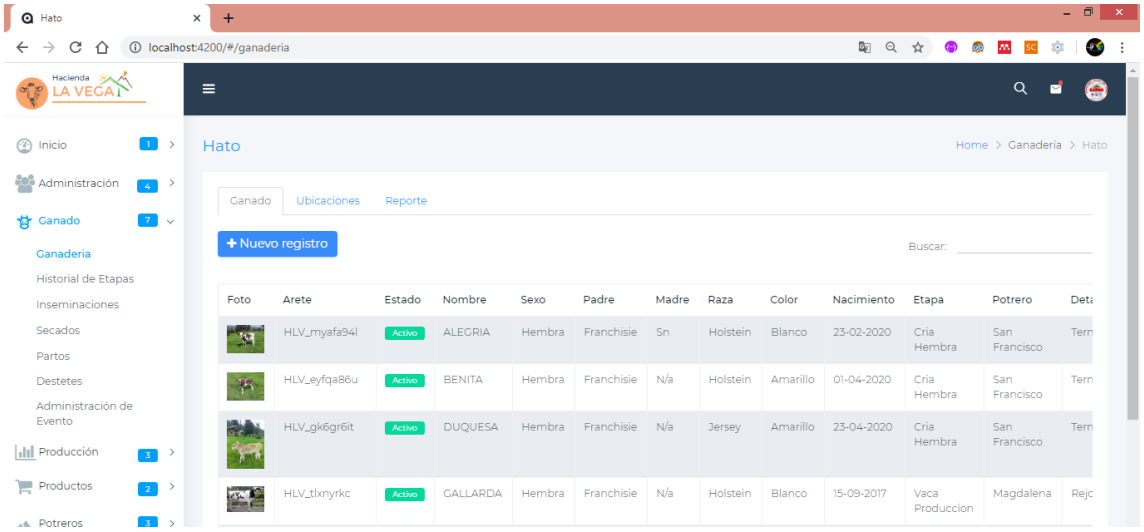


Figura 30: Interfaz de la función actualizar del módulo ganadería
Fuente: Elaboración propia

En el Anexo A, sección módulo Ganadería se describe todo el código desarrollado para Backend y Frontend, así también se detalla otras funcionalidades del módulo.

2.3.13. Iteración 11: Módulo de reproducción de ganadería.

En la Figura 31 de observa como resultado la interfaz principal del módulo, el cual permite llevar un registro diario de los partos e inseminaciones del ganado.

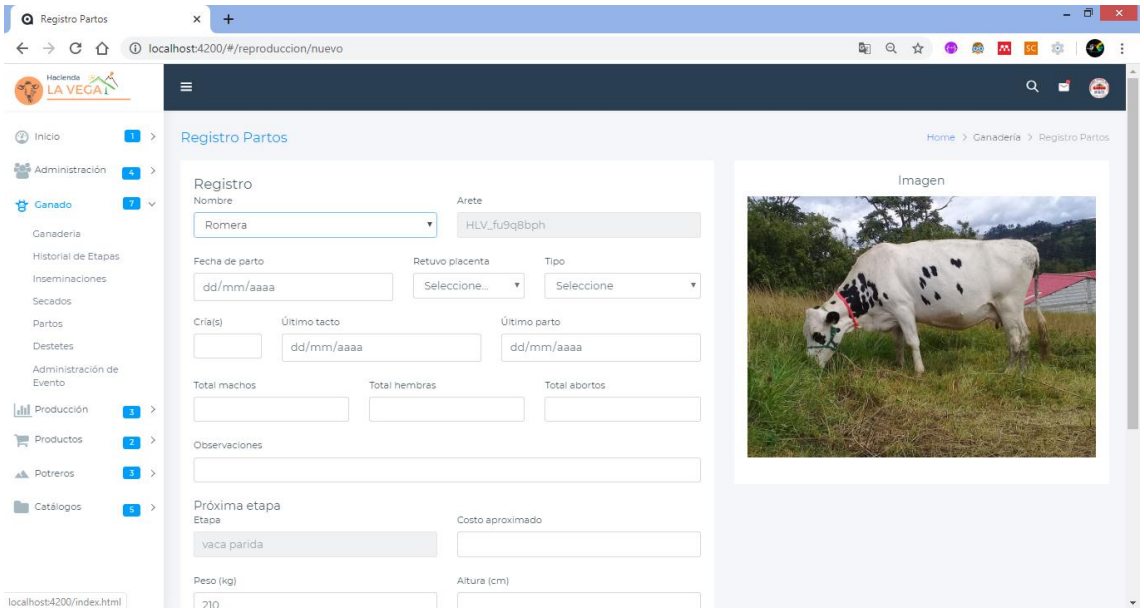


Figura 31: Interfaz principal módulo reproducción ganadería
Fuente: Elaboración propia

En el Anexo A, sección módulo reproducción Ganadería se describe todo el código desarrollado para Backend y Frontend, así también se detalla otras funcionalidades del módulo.

2.3.14. Iteración 12: Módulo de sanidad de ganadería.

En la presente iteración se desarrolló el módulo y sus respectivas funciones tal como se muestra en la Figura 32 la interfaz de la función nuevo registro de aplicación del medicamento.

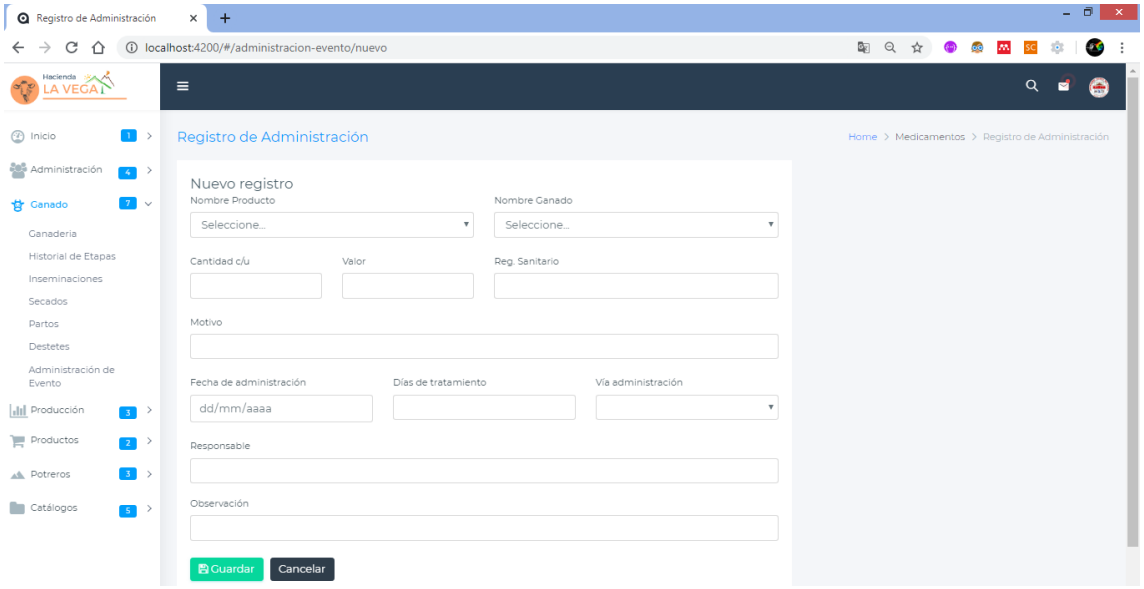


Figura 32: Interfaz de la funcionalidad nuevo registro del módulo sanidad
Fuente: Elaboración propia

En el Anexo A, sección módulo Sanidad de Ganadería se describe todo el código desarrollado para Backend y Frontend, así también se detalla otras funcionalidades del módulo.

2.3.15. Iteración 13: Módulo de potreros.

La Figura 33 demuestra como resultado la interfaz principal de la función asignar potrero del módulo potreros, el cual permite llevar un registro diario de la ubicación del ganado.

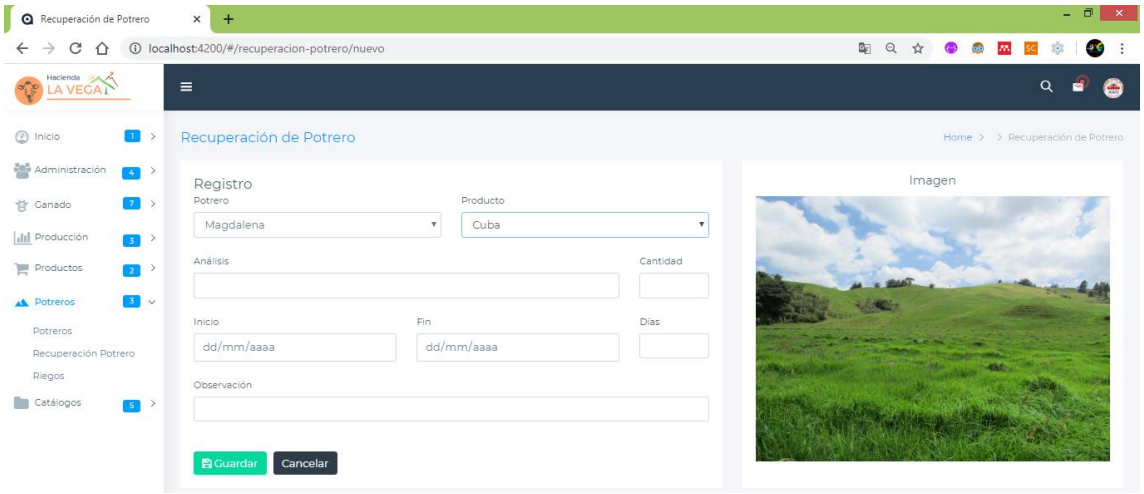


Figura 33: Interfaz de la función asignar a un potrero
Fuente: Elaboración propia

En el Anexo A, sección módulo Potreros se describe todo el código desarrollado para Backend y Frontend, así también se detalla otras funcionalidades del módulo.

2.4. Post-juego

2.4.1. Pruebas de análisis y estructuración de la base de datos del sistema.

TABLA 25: Pruebas de análisis y estructuración de la base de datos del sistema

ID	Variables	Se espera	Se obtuvo
1	Instalación y configuración de las herramientas	Las herramientas tienen que estar configuradas acorde a la necesidad del desarrollador.	Las herramientas se instalaron y configuraron correctamente.

2	Modelamiento y diseño físico de la base del sistema.	La base de datos debe tener un diseño físico al cual se pueda acceder a las relaciones de forma rápida, eficiente y segura.	La base de datos se implementó con el diseño físico con relaciones eficientes y seguras.
3	Validación y normalización de la base de datos del sistema.	La base de datos debe contar con un análisis detallado lo cual permita perfeccionar la información.	La base de datos contó con un análisis detallado de la información.

Fuente: Elaboración Propia

2.4.2. Pruebas de análisis y estructuración de las herramientas para el desarrollo del sistema Backend y Frontend.

TABLA 26: Pruebas de análisis y estructuración de las herramientas para el desarrollo del sistema Backend y Frontend

ID	Variables	Se espera	Se obtuvo
1	Investigación y análisis las herramientas Nodejs y Express del lado Backend	El desarrollador tiene que tener en claro el lenguaje de programación así también las ventajas y desventajas.	El desarrollador determinó los requerimientos de las herramientas a utilizarse.
2	Investigación y análisis las herramientas Angular y Angular CLI del lado Frontend	El desarrollador tiene que tener el conocimiento adecuado de las herramientas y los componentes necesarios.	El desarrollador contó con el conocimiento necesario en base a la documentación del presente tema de investigación.

Fuente: Elaboración Propia

2.4.3. Pruebas de instalación y configuración de las herramientas para el desarrollo del sistema del lado Backend.

TABLA 27: Pruebas de instalación y configuración de las herramientas para el desarrollo del sistema del lado Backend.

ID	Variables	Se espera	Se obtuvo
1	Instalación y configuración de dependencias.	Las herramientas a utilizarse en el Backend deben estar instaladas y configuradas	Las herramientas y dependencias iniciales a utilizarse se instalaron y configuraron correctamente.

2	Análisis del entorno de desarrollo.	Tener en claro la mayor parte del entorno de desarrollo y sus funcionalidades.	El desarrollador contó con el conocimiento adecuado e identificó las funcionalidades del entorno de desarrollo.
3	Conexión con la base de datos desde el servidor.	Crear una conexión a la base de datos desde el servidor Backend mediante una función que permita el uso de archivos JSON para mejorar el rendimiento.	Se creó la conexión al servidor Express Backend configurando un estándar de las respuestas a peticiones http.

Fuente: Elaboración Propia

2.4.4. Pruebas de análisis de APIs, desarrollo y validaciones de registro y login del sistema del lado Backend.

TABLA 28: Pruebas de análisis de APIs, desarrollo y validaciones de registro y login del sistema del lado Backend.

ID	Variables	Se espera	Se obtuvo
1	Testeo de API´s REST mediante el software Postman	Analizar los objetos JSON para posteriormente testear en el software Postman	Se identificó el formato y variables que conforman el objeto JSON, antes y después de realizar el test.
2	Registro y Login	El registro y login en el Backend debe permitir realizar las peticiones básicas, así también crear la configuración para encriptar la contraseña.	Registro y login permite realizar un análisis a la respuesta en el Backend, además de realizar la encriptación de una sola vía.

Fuente: Elaboración Propia

2.4.5. Pruebas de análisis y desarrollo de seguridades del usuario en el sistema del lado Backend.

TABLA 29: Pruebas de análisis y desarrollo de seguridades del usuario en el sistema del lado Backend.

ID	Variables	Se espera	Se obtuvo
1	Tokens de Autenticación con JWT.	Obtener un token después de un login correcto para el uso posterior en el sistema.	El sistema generó el token de acuerdo al login de usuario, mismo que se verificó y usó en las

			peticiones necesarias en el sistema.
2	Middleware para Autenticación y Roles.	Desarrollar un middleware para separar el token de autenticación y roles.	El sistema usó el middleware para bloquear o permitir el acceso a ciertas rutas específicas.
3	Cabeceras HTTP y acceso CORS	Desarrollar un middleware para manejar el acceso Cors en Express JS y HTTP del sistema.	El sistema utilizó la función del middleware para validar el acceso de CORS y cabeceras HTTP.

Fuente: Elaboración Propia

2.4.6. Pruebas de instalación y configuración de las herramientas para el desarrollo del sistema del lado Frontend.

TABLA 30: Pruebas de instalación y configuración de las herramientas para el desarrollo del sistema del lado Frontend.

ID	Variables	Se espera	Se obtuvo
1	Instalación y configuración de dependencias.	Se deberá instalar los módulos necesarios, de igual manera la instalación de la línea de comandos Angular CLI.	La estructura básica del directorio de archivos del sistema, además, la línea de comandos de Angular CLI
2	Análisis del entorno de desarrollo.	Tener en claro configuraciones del entorno de desarrollo y sus funcionalidades.	El desarrollador contó con el conocimiento adecuado e identificó las funcionalidades del entorno de desarrollo con la ayuda de la documentación del presente tema de investigación.
3	Crear módulos, componentes y servicios.	Se deberá tener un nivel considerable de conocimiento de la línea de comandos de Angular CLI.	Se utilizó la línea de comandos identificando las banderas básicas, con el comando respectivo se creó los módulos, componentes y servicios acorde a la secuencia del desarrollo del sistema.
4	Bootstrap y Maquetación Web	Mostrar un interfaz agradable al usuario.	Se implementó una plantilla conjuntamente con la instalación de Bootstrap y librerías externas como JQuery.

Fuente: Elaboración Propia

2.4.7. Pruebas de módulo de seguridades del registro y login al sistema del lado Frontend.

TABLA 31: Pruebas de módulo de seguridades del registro y login al sistema del lado Frontend.

ID	Variables	Se espera	Se obtuvo
1	Usuario incorrecto	El sistema deberá mostrar un mensaje de acceso incorrecto al sistema.	El sistema mostró el mensaje de alerta al ingresar al sistema.
2	Contraseña incorrecta	El sistema tiene que entregar un mensaje de acceso incorrecto al sistema.	El sistema mostró el mensaje de alerta al ingresar al sistema.
3	Usuario y Clave correctos	El sistema debe permitir el acceso al sistema correspondiente al rol con el que inició sesión.	El sistema permitió el ingreso al sistema, con su respectivo rol.
5	Logout de la aplicación	El sistema finaliza la sesión de trabajo del usuario, bloqueando el acceso a diferentes rutas.	El sistema bloquea las rutas, debido a la petición de Logout emitida por el usuario.

Fuente: Elaboración Propia

2.4.8. Pruebas de desarrollo del perfil usuario y panel de administración del sistema del lado Frontend.

TABLA 32: Pruebas de desarrollo del perfil usuario y panel de administración del sistema del lado Frontend.

ID	Variables	Se espera	Se obtuvo
1	Perfil de Usuario	El sistema deberá contener un módulo destinado a la información del usuario logeado.	El sistema cuenta con el modulo perfil de usuario en el cual permite mostrar la información personal.
2	Desarrollo del Panel de Administracion	El sistema mostrará un módulo en el cual se actualiza la información del usuario e imagen.	El sistema cuenta con un módulo en el cual el usuario actualiza la información personal e imagen.
3	Subir imagen desde el frontend hacia el backend	El sistema almacenará las imágenes de perfil en el servidor Backend.	El sistema almacena la imagen de usuario en el servidor Backend.

Fuente: Elaboración Propia

2.4.9. Pruebas del módulo de usuario y roles.

TABLA 33: Pruebas del módulo de usuario y roles.

ID	Variables	Se espera	Se obtuvo
1	Usuario y contraseña con rol de administrador.	El sistema deberá permitir tener mayor control de los registros del sistema.	El sistema mediante el rol de Administrador tiene mayor control a la información del sistema, principalmente en las acciones de actualizar y eliminar.
2	Usuario y contraseña con rol de colaborador.	El sistema deberá permitir el control a las acciones básicas del sistema.	El sistema permite por medio del rol Colaborador el control de las funciones básicas del sistema.
3	Usuario y contraseña con rol de usuario invitado.	El sistema permitirá desplegar información de interés hacia el usuario invitado.	El sistema muestra información útil para un usuario con rol de Invitado.

Fuente: Elaboración Propia

2.4.10. Pruebas del módulo de ganadería.

TABLA 34: Pruebas del módulo de ganadería.

ID	Variables	Se espera	Se obtuvo
1	Lista de registro	El sistema mostrará la interfaz con toda la lista de registros existentes en la base de datos, dicha interfaz tendrá los accesos necesarios con las funcionalidades como nuevo, editar, eliminar y buscar.	El sistema mostró una interfaz con los registros disponibles para el rol de dicho usuario, además de los accesos a las funcionalidades como nuevo, editar, eliminar y buscar.
2	Nuevo registro	El sistema deberá desplegar la interfaz para ingresar la información del nuevo animal.	El sistema permitió ingresar la información referente al nuevo registro.
3	Guardar registro	Al hacer clic en el botón guardar el sistema comprobará la validez de los datos, así también resaltará los campos requeridos obligatoriamente.	El sistema mostró todos los campos que son requeridos para la creación del registro.

5	Actualizar registro	Al hacer clic en el botón guardar luego de haber editar los campos necesarios, el sistema comprobará la validez de los mismo y procederá a actualizar dicho registro.	El sistema muestra información mensajes de advertencia al actualizar y comprueba la información ingresada, para posteriormente actualizarla.
5	Eliminar registro	Al hacer clic en el botón eliminar el sistema mostrará un mensaje de advertencia.	Al pretender eliminar el registro el sistema mostró un mensaje de advertencia con información relevante del registro, para posteriormente aceptar y eliminar.

Fuente: Elaboración Propia

2.4.11. Pruebas del módulo de reproducción de ganadería.

TABLA 35: Pruebas del módulo de reproducción ganadería.

ID	Variables	Se espera	Se obtuvo
1	Lista de registro	El sistema mostrará la interfaz con toda la lista de registros existentes en la base de datos, dicha interfaz tendrá los accesos necesarios con las funcionalidades como nuevo, editar, eliminar y buscar.	El sistema mostró una interfaz con los registros disponibles para el rol de dicho usuario, además de los accesos a las funcionalidades como nuevo, editar, eliminar y buscar.
2	Nuevo registro	El sistema deberá desplegar la interfaz para ingresar la información del nuevo animal.	El sistema permitió ingresar la información referente al nuevo registro.
3	Guardar registro	Al hacer clic en el botón guardar el sistema comprobará la validez de los datos, así también resaltará los campos requeridos obligatoriamente.	El sistema mostró todos los campos que son requeridos para la creación del registro.
5	Actualizar registro	Al hacer clic en el botón guardar luego de haber editar los campos necesarios, el sistema comprobará la validez de los mismo y procederá a actualizar dicho registro.	El sistema muestra información mensajes de advertencia al actualizar y comprueba la información ingresada, para posteriormente actualizarla.
5	Eliminar registro	Al hacer clic en el botón eliminar el sistema mostrará un mensaje de advertencia.	Al pretender eliminar el registro el sistema mostró un mensaje de advertencia con

			información relevante del registro, para posteriormente aceptar y eliminar.

Fuente: Elaboración Propia

2.4.12. Pruebas del módulo de sanidad de ganadería.

TABLA 36: Pruebas del módulo de sanidad de ganadería.

ID	Variables	Se espera	Se obtuvo
2	Nuevo registro	El sistema deberá desplegar la interfaz para ingresar la información del nuevo registro de sanidad.	El sistema permitió ingresar la información referente al nuevo registro de sanidad.
3	Guardar registro	Al hacer clic en el botón guardar el sistema comprobará la validez de los datos, así también resaltará los campos requeridos obligatoriamente.	El sistema mostró todos los campos que son requeridos para la creación del registro.
5	Actualizar registro	Al hacer clic en el botón guardar luego de haber editar los campos necesarios, el sistema comprobará la validez de los mismo y procederá a actualizar dicho registro.	El sistema muestra información mensajes de advertencia al actualizar y comprueba la información ingresada, para posteriormente actualizarla.
5	Eliminar registro	Al hacer clic en el botón eliminar el sistema mostrará un mensaje de advertencia.	Al pretender eliminar el registro el sistema mostró un mensaje de advertencia con información relevante del registro, para posteriormente aceptar y eliminar.

Fuente: Elaboración Propia

2.4.13. Pruebas del módulo de potreros.

TABLA 37: Pruebas del módulo de potreros de ganadería.

ID	Variables	Se espera	Se obtuvo
1	Lista de registro	El sistema mostrará la interfaz con toda la lista de registros existentes en la base de datos, dicha interfaz tendrá los accesos necesarios con las	El sistema mostró una interfaz con los registros disponibles en el módulo potreros para el rol de dicho usuario, además de los

		funcionalidades como nuevo, editar, eliminar y buscar.	accesos a las funcionalidades como nuevo, editar, eliminar y buscar.
2	Nuevo registro	El sistema deberá desplegar la interfaz para ingresar la información del nuevo registro de potreros.	El sistema permitió ingresar la información referente al nuevo registro de potreros.
3	Guardar registro	Al hacer clic en el botón guardar el sistema comprobará la validez de los datos, así también resaltará los campos requeridos obligatoriamente.	El sistema mostró todos los campos que son requeridos para la creación del registro.
5	Actualizar registro	Al hacer clic en el botón guardar luego de haber editar los campos necesarios, el sistema comprobará la validez de los mismo y procederá a actualizar dicho registro.	El sistema muestra información mensajes de advertencia al actualizar y comprueba la información ingresada, para posteriormente actualizarla.
5	Eliminar registro	Al hacer clic en el botón eliminar el sistema mostrará un mensaje de advertencia.	Al pretender eliminar el registro el sistema mostró un mensaje de advertencia con información relevante del registro, para posteriormente aceptar y eliminar.

Fuente: Elaboración Propia

2.5. Implementación del sistema.

Para poder ejecutar el Sistema de Información, seguimiento y control para la hacienda ganadera La Vega; vamos a necesitar de un servidor web, un gestor de base de datos, software base y conexión a internet.

2.5.1. Sistema operativo.

El sistema operativo sobre el que va a ejecutarse la aplicación va a ser un sistema operativo Windows; en la versión de Windows 8.1 en español, el mismo que puede ser reemplazado por un sistema operativo Windows Server 2012.

Dentro de nuestro sistema operativo que escogimos se necesita instalar IIS que es un servidor web con características de Microsoft .Net Framework 4.0.

2.5.2. Hardware.

El hardware que se requiere para que el sistema pueda funcionar de forma correcta requiere de un computador con características de procesamiento aceptables como por ejemplo:

- Procesador Core i3 o superior.
- Memoria Ram 4GB o superior
- Disco duro de 500GB o superior.

2.5.3. Herramientas adicionales.

Para proceder a la implementación del sistema es necesario instalar las herramientas de gestión de base de datos PostgreSQL y su interfaz PGAdmin III, adicionalmente del entorno de ejecución NodeJS y Angular CLI.

2.5.4. Instalación del sistema.

Una vez instalado todo el software de aplicación sobre el software base procedemos con la instalación del sistema. En el Anexo A (Manual de Usuario del sistema) se describe los comandos y archivos generados para la implementación del mismo.

CAPÍTULO III

Resultados

3.1. Evaluación e Interpretación.

3.1.1. Evaluación e Interpretación de usabilidad del software desarrollado.

La evaluación de usabilidad del Sistema Web se obtuvo en base a los resultados de la encuesta digital aplicada (Anexo E. Formato de encuesta de usabilidad del software Sistema de información, seguimiento y control de la hacienda ganadera “La Vega”) mediante la herramienta Forms de Office 365, las preguntas aplicadas fueron proporcionadas por el Sistema de Usabilidad Escalable (SUS).

En la Tabla 38 se muestran los resultados obtenidos por cada pregunta en base a la encuesta realizada a 10 personas.

TABLA 38: Resultado de las encuetas.

Preguntas	Totalmente desacuerdo	En desacuerdo	Ni de acuerdo, ni en desacuerdo	De acuerdo	Totalmente de acuerdo
1	1	0	0	5	4
2	0	3	4	2	1
3	0	1	0	5	4
4	0	3	1	4	2
5	0	0	2	1	7
6	4	3	2	1	0
7	1	0	0	4	5
8	1	5	2	1	1
9	0	0	2	3	5
10	4	2	1	2	1

Fuente: Elaboración Propia

A continuación se detalla las gráficas con los resultados obtenidos en cada pregunta.

Pregunta 1. Creo que usaría este sistema frecuentemente.

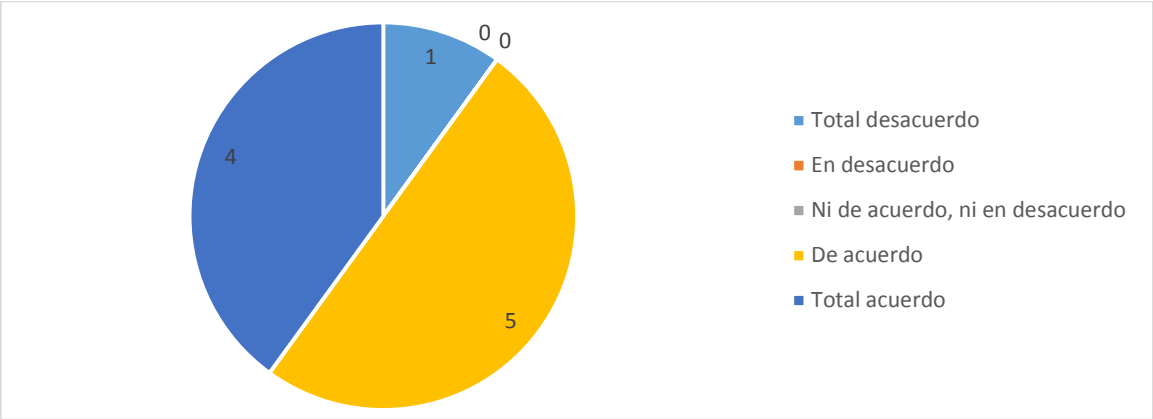


Figura 34: Representación gráfica de los resultados de la pregunta 1.
Fuente: Elaboración propia

Pregunta 2. Encuentro este sistema innecesariamente complejo.

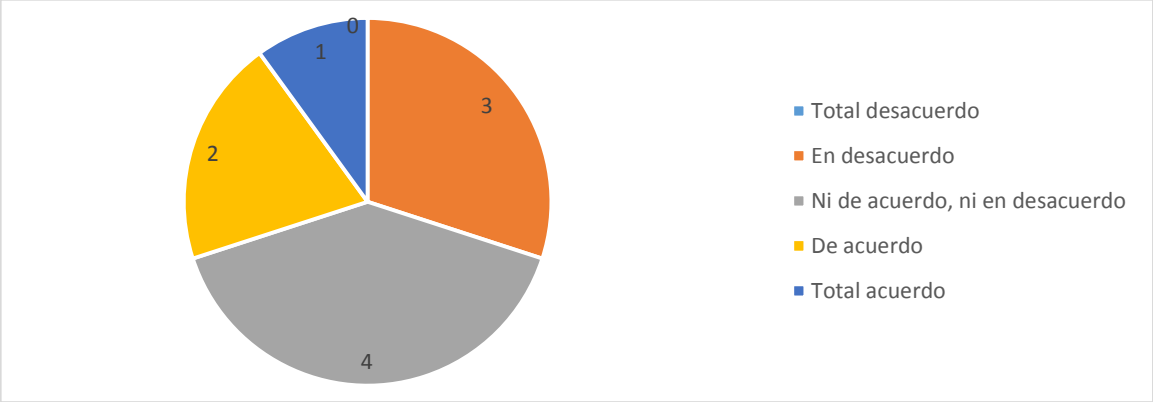


Figura 35: Representación gráfica de los resultados de la pregunta 2.
Fuente: Elaboración propia

Pregunta 3. Creo que el sistema fue fácil de usar.

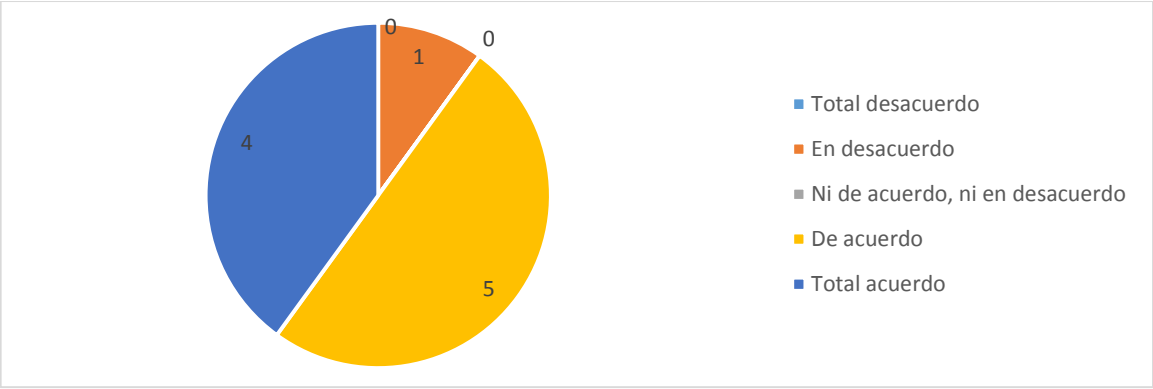


Figura 36: Representación gráfica de los resultados de la pregunta 3.
Fuente: Elaboración propia

Pregunta 4. Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar este sistema.

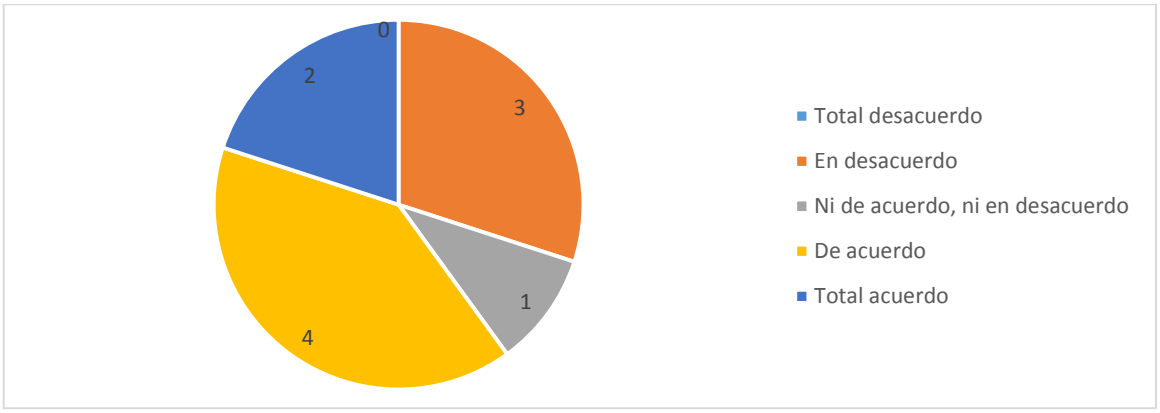


Figura 37: Representación gráfica de los resultados de la pregunta 4.
Fuente: Elaboración propia

Pregunta 5. Las funciones de este sistema están bien integradas.

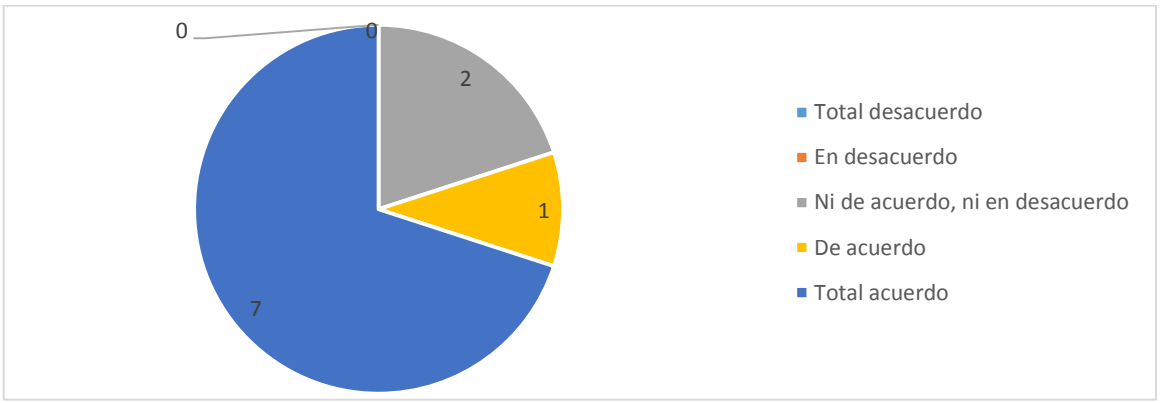


Figura 38: Representación gráfica de los resultados de la pregunta 5.
Fuente: Elaboración propia

Pregunta 6. Creo que el sistema es muy inconsistente.

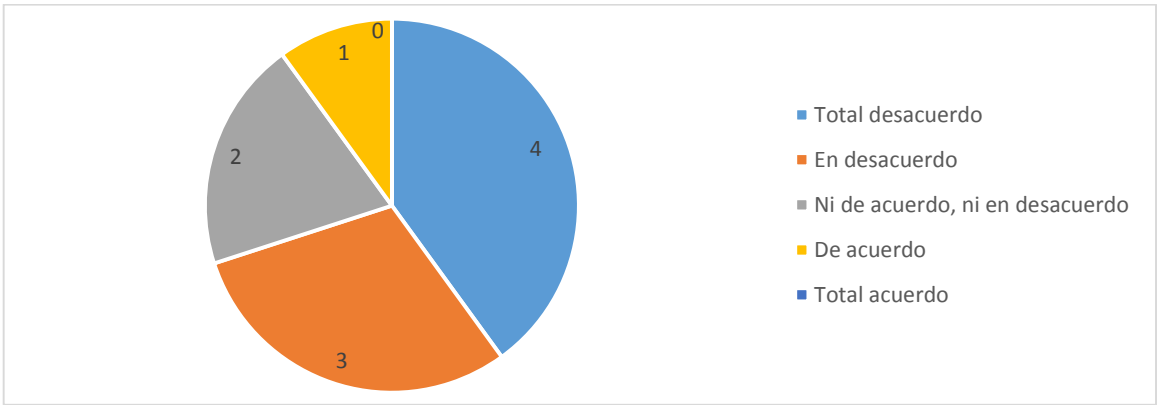


Figura 39: Representación gráfica de los resultados de la pregunta 6.
Fuente: Elaboración propia

Pregunta 7. Imagino que la mayoría de la gente aprendería a usar este sistema en forma muy rápida.

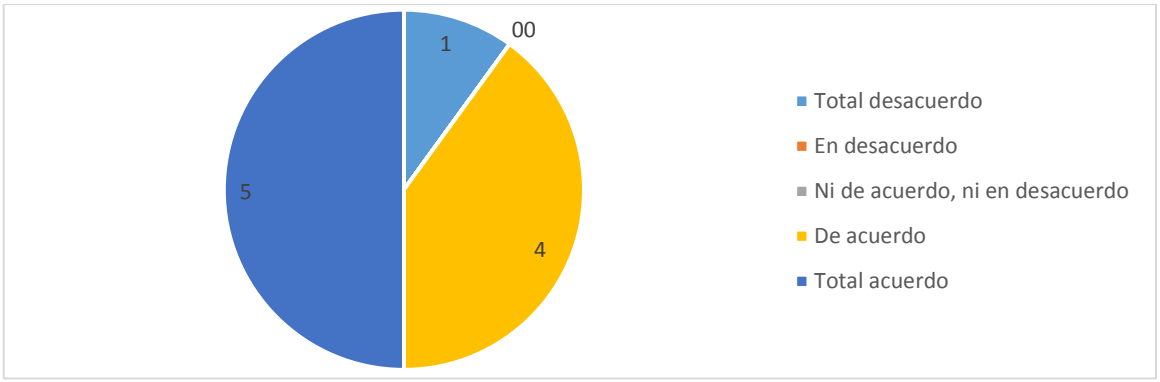


Figura 40: Representación gráfica de los resultados de la pregunta 7.
Fuente: Elaboración propia

Pregunta 8. Encuentro que el sistema es muy difícil de usar.

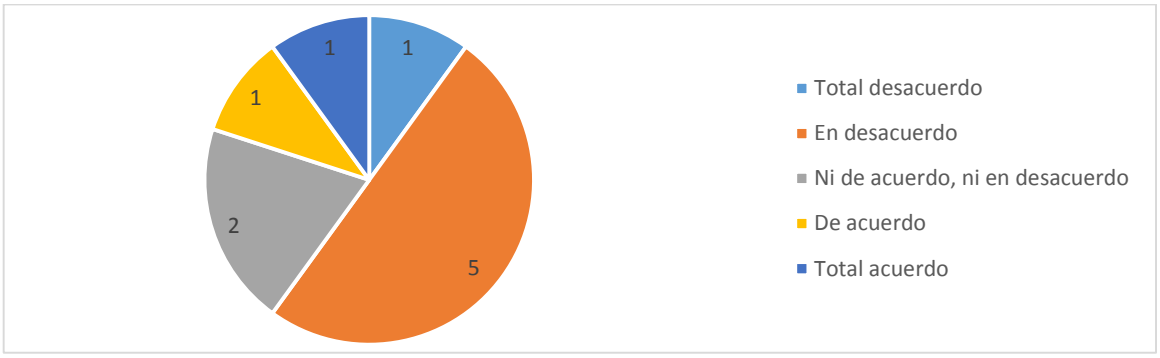


Figura 41: Representación gráfica de los resultados de la pregunta 8.
Fuente: Elaboración propia

Pregunta 9. Me siento confiado al usar este sistema.

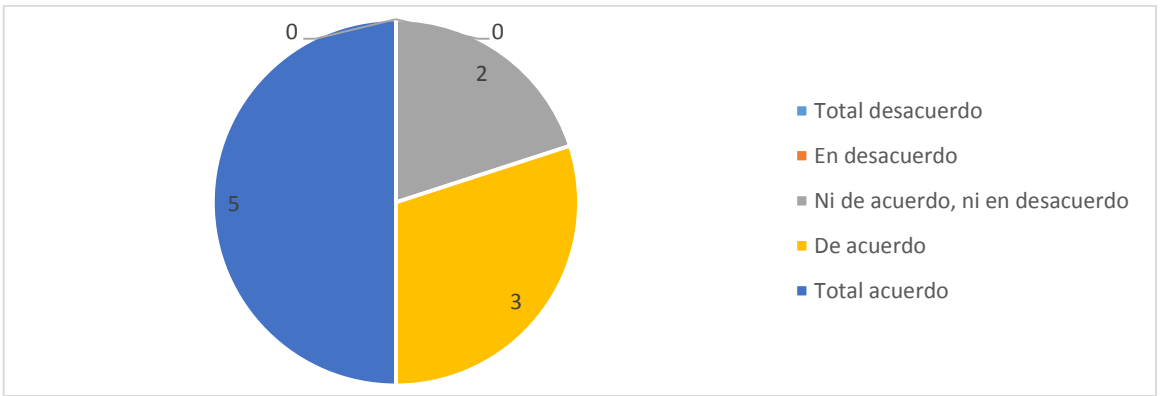


Figura 42: Representación gráfica de los resultados de la pregunta 9.
Fuente: Elaboración propia

Pregunta 10. Necesité aprender muchas cosas antes de ser capaz de usar este sistema.

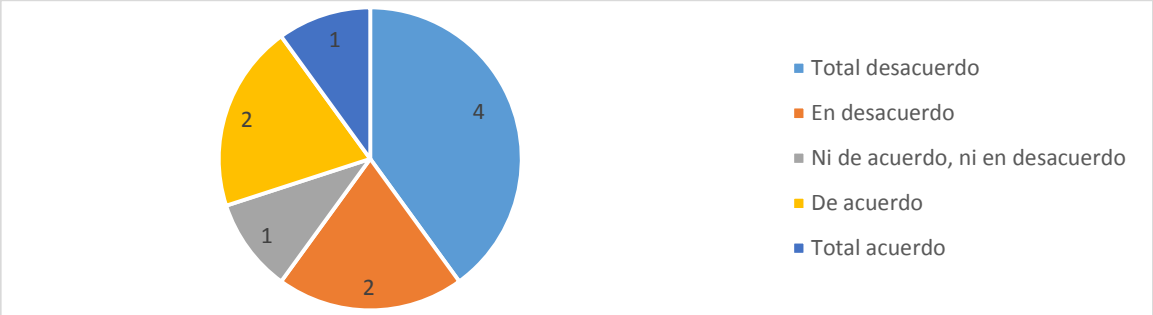


Figura 43: Representación gráfica de los resultados de la pregunta 10.
Fuente: Elaboración propia

3.1.2. Análisis e interpretación de resultados.

Se debe tomar en cuenta que los resultados obtenidos anteriormente no deben ser tratados como porcentajes, ya que pueden derivar a respuestas erróneas, además, es importante que no se altere el orden de las preguntas, mismo orden que es necesario para el proceso de interpretación de los datos.

Como primer paso es calcular los valores obtenidos asignando, los cuales puede ser puntuada de 1 a 5, donde 1 significa Total desacuerdo y 5 significa Total acuerdo. Por ejemplo en la Pregunta 1 respondieron 5 la opción “De acuerdo” por lo cual se multiplica por 4.

$$Pregunta\ 1\ (Respuesta: De\ acuerdo): 5 * 4 = 20$$

En la siguiente Tabla 39 se muestran los valores de los resultados calculado según las medidas SUS.

TABLA 39: Resultado de las encuetas.

Preguntas	Totalmente desacuerdo	En desacuerdo	Ni de acuerdo, ni en desacuerdo	De acuerdo	Totalmente de acuerdo
1	1	0	0	20	20
2	0	6	12	8	5
3	0	2	0	20	20
4	0	6	3	16	10
5	0	0	6	4	35

6	4	6	6	4	0
7	1	0	0	16	25
8	1	10	6	4	5
9	0	0	6	12	25
10	4	4	3	8	5

Fuente: Elaboración Propia

Las preguntas de SUS están diseñadas de forma que se alternan entre negativas y positivas, por tal motivo deben ser interpretadas y tratadas de manera distinta. Se debe separar en dos grupos las preguntas, las pares y las impares, para obtener dos resultados parciales. A continuación, Sumar los resultados promediados que se obtuvieron en las encuestas considerando las siguientes restricciones.

Las preguntas impares (1,3,5,7 y 9) van a tomar el valor resultante del promedio y se les restará 1.

$$Parcial\ 1 = \sum(promedio\ preguntas\ impares) - 1$$

TABLA 40: Parcial 1 de las preguntas impares.

Preguntas	Promedios
1	3,1
3	3,2
5	3,5
7	3,2
9	3,3
Parcial 1	16,3

Fuente: Elaboración Propia

Las preguntas pares (2,4,6,8 y 10), será de 5 menos el valor resultante del promedio.

$$Parcial\ 2 = 5 - \sum promedio\ preguntas\ pares$$

TABLA 41: Parcial 2 de las preguntas pares.

Preguntas	Promedios
2	1,9
4	1,5
6	3
8	2,4
10	2,6
Parcial 2	11,4

Fuente: Elaboración Propia

Finalmente se suman los dos resultados parciales y se multiplica por 2.5. En una escala percentil de 0 a 100.

$$Resultado = (Parcial\ 1 + Parcial\ 2) * 2.5$$
$$Resultado = (16.3 + 11.4) * 2.5$$
$$\boldsymbol{Resultado = 69.25}$$

De acuerdo con las investigaciones y pruebas previas hechas con el modelo SUS, se estima que un puntaje superior 68 se considera superior al promedio, mientras que un valor inferior a 68 se considera por debajo del promedio.

3.2. Documentación Técnica de Angular.

Mediante la recolección de información obtenida en la Revisión Bibliográfica acerca del Framework Angular, se describe a continuación las principales características que se usan al desarrollar un software, al final de la sección se elaborará la documentación técnica que servirá como guía para las personas interesadas en el framework.

3.2.1. Módulo

En relación a los módulos (Oriol, 2019) expresa que las aplicaciones de Angular son modulares, aunque ahora no hace falta una sintaxis específica de Angular para definir módulos, sino que Angular utiliza el sistema de módulos que define el estándar ECMAScript 2015. En la Figura 44 se describe lo anteriormente mencionado, es decir, un módulo declara los componentes, las directivas o pipes, además, puede iniciar bootstrap para que dichos componentes sean visibles.

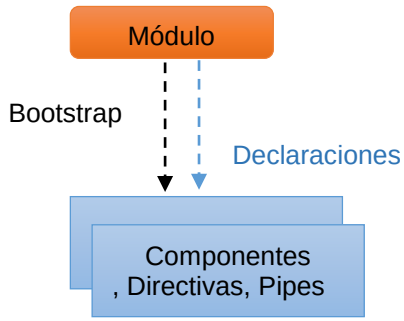


Figura 44: Arquitectura simple de un Módulo Angular
Fuente: Elaboración propia

Un módulo, típicamente es un conjunto de código dedicado a cumplir un único objetivo describir cómo encajan las partes de la aplicación facilitando la creación de contenido reusable. Cada app de Angular tiene al menos una clase de módulo convencionalmente llamado AppModule (Angular, 2018). Un módulo Angular, ya sea una raíz o característica, es una clase con un decorador **@NgModule**.

Así mismo en la Figura 45 se observa que el módulo principal exporta otros módulos, ya sean propios, de angular o de librerías de terceros, al igual que puede exportar componentes, directivas o pipes.

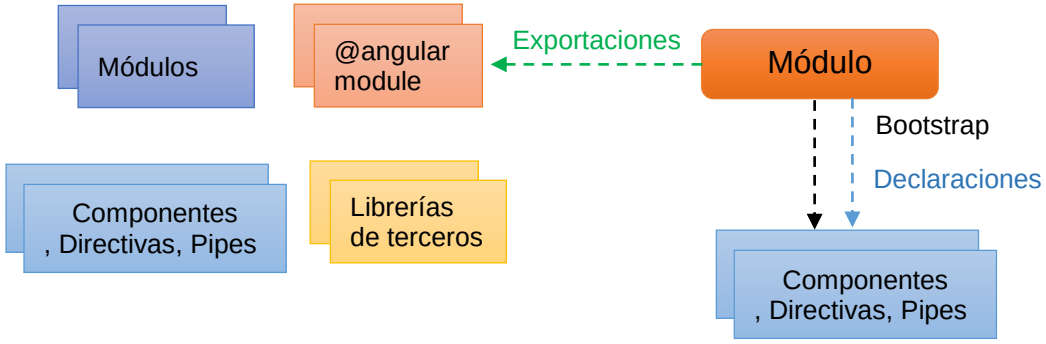


Figura 45: Arquitectura compuesta de un Módulo Angular
Fuente: Elaboración propia

Tipos de módulos

Se puede considerar 3 tipos principales de NgModule con lo cual se puede hacer:

a) Módulos de páginas.
b) Módulos de servicios.
c) Módulos de componentes reutilizables.

Considerando a los módulos de componentes son los que se utiliza siempre, de lo contrario la aplicación estaría vacía. Los módulos de páginas y servicios son opcionales pero según vaya creciendo la aplicación será necesaria su implementación.

Módulos de páginas

Los módulos de páginas son módulos con enrutamiento. Permiten separar y organizar las diferentes áreas de la aplicación. Se cargan solo una vez en el AppModule (López, 2018). Es decir; la aplicación podría tener un AppModule para la

portada y el inicio de la página, AccountModule para las páginas de registro, inicio de sesión y cierre de sesión; luego, un HaciendaModule para la lista de haciendas y páginas de detalles de la hacienda, ProductModule para lo relacionado con los productos y SharedModule para poner todo lo común/compartido.

Dichos módulos contienen 3 directorios:

/shared: servicios e interfaces que varias páginas necesitaran del mismo servicio.

/pages: componentes enrutados.

/components: componentes de presentación puros.

Módulos de servicios

Son módulos con servicios que se necesita a través de toda la aplicación, como los servicios tienen generalmente un alcance global, estos módulos se cargan solo una vez en el AppModule, y luego se puede acceder a los servicios en cualquier lugar (incluso en los módulos con carga diferida) (López, 2018).

Como dicho módulo se usará desde el exterior, debe hacer un punto de entrada, donde puede exportar el NgModule, los servicios y tal vez las interfaces y los tokens de inyección.

Para ello se debe importar la librería del decorador NgModule:

```
import { NgModule } from '@angular/core';
```

Seguidamente se importa en un arreglo del decorador los componentes necesarios en base al módulo en que se encuentra trabajando.

```
@NgModule({
  declarations: [
    //aquí los componentes
  ],
  imports: [
    //aquí los módulos
  ]
}) export class AppModule { } //nombre del modulo
```

a) Decorador

Los decoradores son una función de lenguaje JavaScript, implementada en TypeScript, para aplicar un decorador, se coloca arriba o a la izquierda del artículo

que decora, cabe mencionar que se incluye paréntesis () al aplicar un decorador (Angular, 2018).

Angular tiene su propio conjunto de decoradores para ayudarlo a trabajar con las piezas de su aplicación. La Figura 46 representa un claro ejemplo de un decorador **@Component** que identifica una clase como un componente Angular y un decorador **@Input** aplicado a la propiedad **name** de ese componente.

```
1.  @Component({...})
2.  export class AppComponent {
3.    constructor(@Inject('SpecialFoo') public foo:Foo) {}
4.    @Input() name:string;
5.  }
```

Figura 46: Lógica de un decorador
Fuente: (Angular, 2018)

b) Decorador NgModule

NgModule es una función de decorador que toma un único objeto de metadatos cuyas propiedades describen el módulo. La tabla 42 describe las propiedades más importantes.

TABLA 42: Propiedades NgModule

Propiedades	Descripción
declarations	Las clases de vista que pertenecen a este módulo. Angular tiene tres tipos de clases de vista: components, directives y pipes.
exports	Es el subconjunto de declaraciones que deberían ser visibles y utilizables en las plantillas de componentes de otros módulos.
imports	Otros módulos cuyas clases exportadas son necesarias por plantillas de componentes declaradas en este módulo.
providers	Creadores de servicios que este módulo contribuye a la colección global de servicios; Se vuelven accesibles en todas las partes de la aplicación.
bootstrap	La vista principal de la aplicación, denominada componente raíz, que aloja todas las demás vistas de la aplicación. Solo el módulo raíz debería establecer esta propiedad bootstrap.

Fuente: (Angular, 2018)

Cada nombre de librería Angular comienza con el prefijo **@angular**. Los cuales se instalan con el administrador de paquetes npm, se importan partes de ellos con la palabra reservada **import**. La Figura 47 da a conocer como también se permite importar módulos, los módulos de Angular proveen de servicios, lo cual significa que hacen disponibles los servicios para todos los componentes, directivas y pipes que estén declaradas en ese módulo.

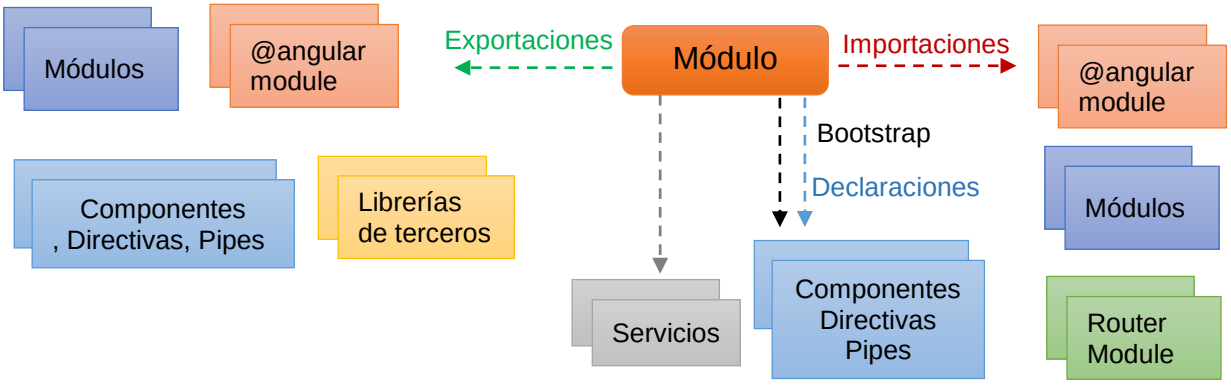


Figura 47: Arquitectura de exportación e importación de Módulos Angular
Fuente: Elaboración propia

En el Anexo B sección 6 Módulos se describe la estructura del código y archivos generados para el desarrollo de módulos en base al sistema a desarrollarse.

3.2.2. Componente

Un Component (componente) controla una zona de espacio HTML que se denomina vista. Dicho componente es una clase que cuenta con propiedades (información de la vista) y métodos (el comportamiento) los cuales permiten modificar la vista. Como se expresa en la Figura 48 se utiliza la palabra **export** ya que los componentes se exportan.

```
export class BuscarComponent{
  texto: string;
  buscar(texto: string){
    ...
  }
}
```

Figura 48: Estructura codificación de Component
Fuente: Elaboración Propia

Se dice que una aplicación de angular por lo menos necesita de un Componente el mismo que será la raíz, por lo general se tendrá más de uno. En la Figura 49 se observa que cada componente tiene una plantilla, y el componente raíz puede tener subcomponentes los cuales permitirán crear funcionalidades definidas dentro de cada componente, esto con la ventaja de poder reutilizar en varias apps o áreas en la misma app.

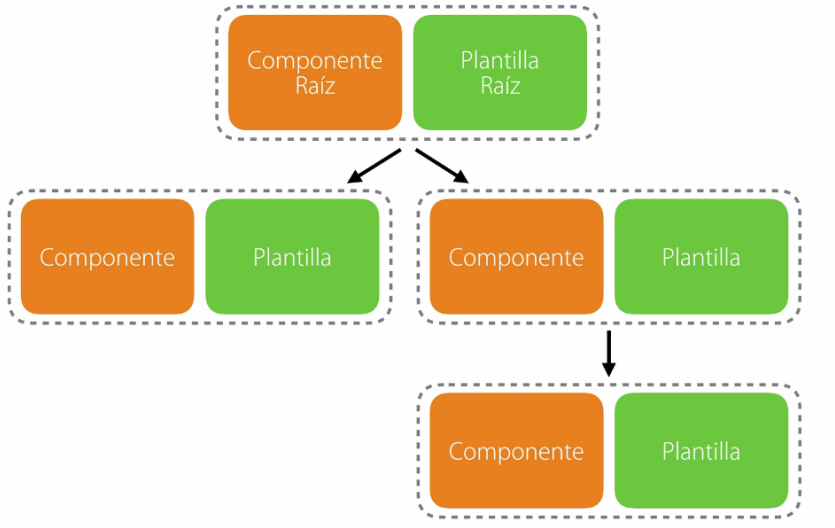


Figura 49: Estructura Component
Fuente: Elaboración Propia

Los componentes son como etiquetas HTML nuevas que pueden ir creándose acorde a la necesidad del negocio. Pueden ser cosas diversas, desde una sección de navegación a un formulario, o un campo de formulario. Para definir el contenido de esta nueva etiqueta, el componente; usa un poco de HTML con su CSS y por supuesto, un poco de JavaScript para definir su funcionalidad.

De manera didáctica la Figura 50 explica la estructura de una aplicación; la misma que a primera instancia cuenta con tres componentes como lo es: el Encabezado, Men Izq y Área principal, a su vez se puede decir que dentro del componente Encabezado cuenta con los subcomponentes Buscar y Notificaciones.

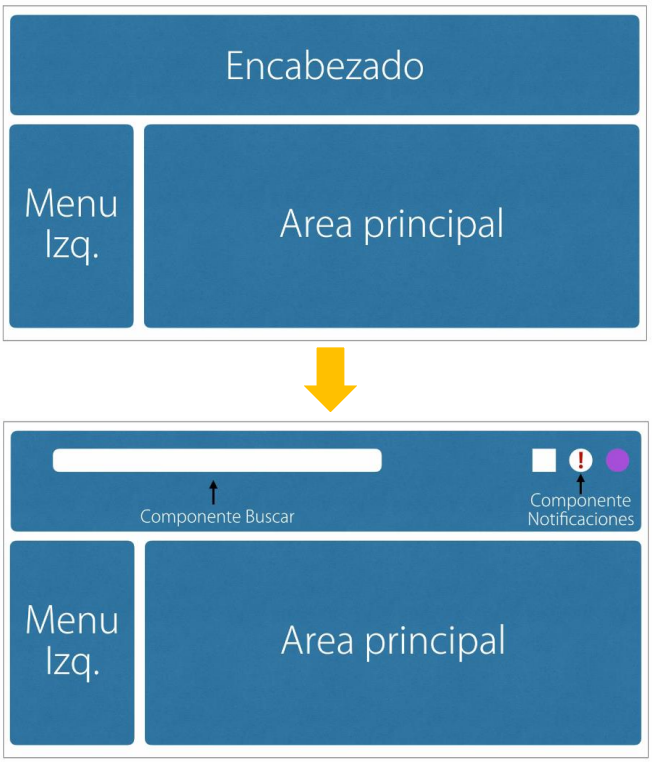


Figura 50: Estructura didáctica de Component
Fuente: Elaboración Propia

a) Identificar el componente inicial

La aplicación de Angular se desarrolla en el directorio "src". Allí se encuentra el archivo index.html raíz de la aplicación. Inicialmente dicho directorio no tiene ningún contenido en sí, pero se encuentra la estructura del uso de un componente, una etiqueta que no pertenece al HTML y que se llama de una manera similar al nombre del proyecto.

<proyecto-angular-app>Loading...**</proyecto-angular-app>**

Este es el componente (***lapp***) donde se alojará la aplicación Angular. Todos los demás componentes estarán debajo de éste, unos dentro de otros. Todo lo que ocurra en la aplicación estará dentro de dicho componente.

b) Estructura de un componente

El código del componente se genera en la carpeta "src/app" como demuestra la Figura 51. Allí se encuentra varios ficheros que forman el componente completo, separados por el tipo de código que se generará según la necesidad del desarrollador.

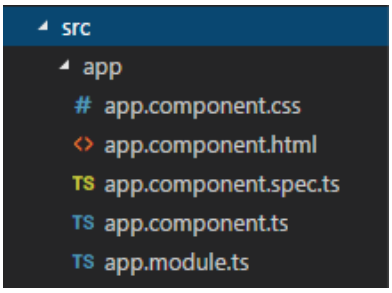


Figura 51: Archivos generados de un Component
Fuente: Elaboración Propia

- **app.component.html:** Equivale a lo que se conoce por "vista" en la arquitectura MVC.
- **app.component.css:** Permite colocar estilos al contenido, siendo que éstos están encapsulados en el componente y no salen afuera.
- **app.component.ts:** Es el archivo TypeScript, que se traduce a JavaScript antes de presentarse al navegador. Equivalente al controlador en el MVC.
- **app.component.spec.ts:** Un archivo TypeScript destinado a tareas de testing de componentes.

Import

El import de "component" dentro de @angular/core, una función decoradora que hace la acción de registrar el componente. La clase que hace las veces de controlador**.**

Una de ellas es el "selector" de este componente, o el nombre de la etiqueta que se usará cuando se desee representar. Luego está asociada la vista (propiedad "templateUrl") al archivo .html del componente y su estilo (propiedad "styleUrls") a un array de todas las hojas de estilo que se necesite (Alvarez & Basalo, 2018).

```
import { Component } from '@angular/core';
@Component({
  selector: 'proyecto-angular-app',
  templateUrl: 'proyecto-angular.component.html',
  styleUrls: ['proyecto-angular.component.css']
})
```

Export

En la clase del componente, se debe colocar con un export para que se conozca fuera de este módulo, es la parte que representa el controlador en una arquitectura

MVC. En ella se encontrará todas las propiedades y métodos que se deseen usar desde la vista.

```
export class ProyectoAngularAppComponent { }
```

Esas propiedades representan el modelo de datos y se podrán usar expresiones en las vistas para poder visualizarlas.

Angular crea, actualiza y destruye componentes a medida que el usuario se mueve a través de la aplicación. Su aplicación puede actuar en cada momento de este ciclo de vida a través de ganchos de ciclo de vida opcionales (Angular, 2018).

c) Componentes re-utilizables

Como plantea López (2018) se dice que son módulos de componentes que la mayoría se relacionan con Interfaz de Usuario que se desea reutilizar en diferentes proyectos. Como los componentes tienen un alcance local, estos módulos se importan en los módulos de cada página donde los necesita.

Se debe considerar que no se debe confiar en un servicio, ya que los servicios son a menudo específicos para una aplicación en particular. ¿Por qué? Al menos debido a la API URL. Proporcionar los datos será el papel del componente de páginas. El componente UI[15] solo recupera datos pasados por otra persona (López, 2018).

En el Anexo B sección 6 Componentes se describe la estructura del código y archivos generados para el desarrollo de componentes en base al sistema a desarrollarse.

3.2.3. Template

La Figura 52 explica la estructura del Template; el cual permite definir la vista de un Component, el template en Angular se parece a HTML, pero decorado con otros componentes y algunas directivas: expresiones de Angular que enriquecen el comportamiento del template (Oriol, 2019). En la siguiente figura muestra un ejemplo de lista de tareas:

[15] **UI:** User Interfaz o Interfaz de Usuario es el medio que permite la interacción de un usuario con un sistema computacional.

```
<h2>To Do List</h2>
<p><i>List of Tasks</i></p>
<div *ngFor="let todo of todos" (click)="selectTodo(todo)">
  {{todo.subject}}
</div>
<todo-detail *ngIf="selectedTodo" [todo]="selectedTodo"></todo-detail>
```

Figura 52: Ejemplo estructura de Template
Fuente: (Angular, 2018)

Se observa, además de elementos HTML conocidos como <h2> y , hay otros elementos desconocidos como lo son:

- *ngFor
- {{todo.subject}}
- (click)
- [todo]
- <todo-detail>

Todos ellos forman parte de la sintaxis de templates de Angular, los cuales se explicarán a detalle en el bloque de Data Binding y Directivas.

3.2.4. Metadatos

Es conveniente revisar la Figura 53, en la misma se puede observar que en realidad es solo una clase de TypeScript, con una declaración muy similar a la de ES6 y sin nada de Angular por ningún lado. Significa que realmente se trata de una clase de Javascript, hasta que indique que se trata de un componente, gracias a los metadatos de Angular (Oriol, 2019).

Los metadatos le dicen a Angular cómo procesar una clase, la forma de añadir metadatos la clase en TypeScript es mediante el patrón decorador justo antes de la declaración de la clase.

```
import { Component } from '@angular/core';

@Component({
  selector:    'todo-list',
  templateUrl: 'todo-list.component.html',
  styleUrls: ['todo-list.component.css'],
  moduleId:    module.id,
  directives:  [TodoDetailComponent],
  providers:   [TodoService]
})
export class TodoListComponent { ... }
```

Figura 53: Combinación de Component con metadata Angular
Fuente: http://blog.enriqueoriol.com/2016/06/introduccion-a-angular-2-parte-i-componente.html

El decorador **@Component**, identifica la clase inmediatamente debajo de ella como una clase de componente, toma un objeto de configuración requerido con la información que Angular necesita para crear y presentar el componente y su vista.

En la Tabla 43 se describen las opciones de configuración @Component más útiles:

TABLA 43: Opciones de configuración @Component

Propiedades	Descripción
selector	Es un selector de CSS que indica a Angular que debe crear e instanciar el componente cuando se encuentra un elemento con ese nombre en el HTML. Es decir, cuando Angular se encuentra <todo-list></todo-list> en un template HTML, entre esos tags inserta una instancia de TodoListComponent.
templateUrl	La url en la que se encuentra el template que trata de vincular al componente.
styleUrls	Un array con urls a archivos de estilos que se desea aplicar al componente. Angular implementa una característica interesante de HTML5 denominada Shadow DOM, que permite aislar los estilos de un componente con respecto al resto.
directives	Sirve para facilitar la Inyección de Dependencias. Ingresa un array de los Componentes y las Directivas que utiliza el Component.
providers	Igual que con las directivas, aquí se define los servicios de los que depende el Component.

Fuente: (Angular, 2018)

3.2.5. Data Binding

Oriol (2019) da a conocer que una de las principales características de Angular es que abstrae la lógica pull/push asociada a insertar y actualizar valores en el HTML y convertir las respuestas de usuario (inputs, clicks, etc) en acciones concretas. Escribir toda esa lógica a mano suele ser tedioso y propenso a errores, y Angular lo resuelve gracias al Data Binding.

Como muestra la Figura 54 el Data Binding es el mecanismo que sirve para sincronizar el modelo (Component) y la vista (partes de una plantilla).

```
<div>{{todo.subject}}</div>
<todo-detail [todo]="selectedTodo"></todo-detail>
<div (click)="selectTodo(todo)"></div>
```

Figura 54: Data Binding Angular
Fuente: (Angular, 2018)

Angular dispone de 4 formas de Data Binding, tres de una sola dirección y uno de doble dirección:

- **Interpolación: (Hacia el DOM)**

Al hacer {{***todo.subject***}}, Angular se encarga de insertar el valor de esa propiedad del componente entre las etiquetas donde se ha definido. Es decir, evalúa ***todo.subject*** e introduce su resultado en el DOM.

- **Property binding: (Hacia el DOM)**

Al hacer [todo]="***selectedTodo***", Angular está pasando el objeto ***selectedTodo*** del Componente padre a la propiedad todo del Componente hijo, en este caso de TodoDetailComponent.

- **Event binding: (Desde el DOM)**

Al hacer (click)="***selectTodo***(todo)", le indica a Angular que cuando se produzca un evento click sobre esa etiqueta , llame al método ***selectTodo*** del Component, pasando como atributo el objeto todo presente en ese contexto.

- **Two-way binding: (Desde/Hacia el DOM)**

Es una cuarta forma importante que combina la propiedad y el enlace de eventos en una sola notación, utilizando la directiva ngModel.

Angular procesa todos los Data Binding una vez por cada ciclo de eventos JavaScript, desde la raíz del árbol de componentes de la aplicación hasta todos los componentes secundarios.

3.2.6. Directiva

Los Template angulares son dinámicos cuando Angular los renderiza, transforma el DOM de acuerdo con las instrucciones dadas por las directivas. Una directiva es una clase con un decorador ***@Directive***. Un componente es una directiva con una plantilla; un decorador ***@Component*** es en realidad un decorador ***@Directive*** extendido con características orientadas al Template.

En Angular existen tres tipos de directivas.

Componente.

Un Componente es una Directiva con template. De hecho ***@Component*** es un decorador ***@Directive*** extendido con características propias de los templates. Hay otros dos tipos de directivas, las estructurales y atributo, y normalmente se ven en forma de etiquetas de elementos HTML como atributos (Oriol, 2019).

Directivas Estructurales.

Como se muestra en la Figura 55 las directivas estructurales comienzan por asterisco (*), alteran el diseño agregando, eliminando y reemplazando elementos en DOM.

La plantilla de ejemplo utiliza dos directivas estructurales integradas:

```
<div *ngFor="let todo of todos"></div>
<todo-detail *ngIf="selectedTodo"></todo-detail>
```

Figura 55: Elementos de Directivas Estructurales
Fuente: (Angular, 2018)

- ***ngIf**: Si la condición se cumple, su elemento se inserta en el DOM, en caso contrario, se elimina del DOM.

- ***ngFor**: repite su elemento en el DOM una vez por cada item que hay en el iterador que se le pasa, siguiendo una sintaxis de ES6.

Directivas Atributo.

Las directivas atributo sin template, alteran la apariencia o el comportamiento de un elemento existente en el DOM, en los template parecen atributos HTML normales, de ahí el nombre. A continuación se detalla las directivas de atributo más utilizadas:

- **ngModel:** Implementa un mecanismo de binding bi-direccional. El ejemplo típico es con el elemento HTML <input>, donde asigna la propiedad value a mostrar y además responde a eventos de modificación.
- **ngClass:** Esta directiva permite añadir/eliminar varias clases a un elemento de forma simultánea y dinámica.
- **ngStyle:** De forma semejante a ngClass, esta directiva permite asignar varios estilos inline a un elemento.

3.2.7. Servicio

Los servicios son fundamentales en Angular, si bien en dicho framework se definen a través de simples clases. Todo valor, función o característica que nuestra aplicación necesita, desde constantes a la lógica de negocio, se encapsula dentro de un servicio (Oriol, 2019). Un servicio es típicamente una clase con un propósito estrecho y bien definido. Debería hacer algo específico y hacerlo bien.

Otro rasgo de los Componentes es que son grandes consumidores de servicios. No recuperan datos del servidor, ni validan inputs de usuario, ni logean nada directamente en consola. Encargan todo este tipo de tareas a los Servicios.

@Injectable

Este decorador afirma de que el servicio espera utilizar otros servicios y genera los metadatos que necesita el servicio para detectar la Inyección de Dependencias (DI) en el constructor. No es necesario ponerlo si el servicio si no tiene DI de otros servicios, pero es recomendable para evitar errores si en el futuro se desea añadir alguna dependencia (Angular, 2018).

Los Guideline[16] de Angular recomiendan utilizar siempre el decorador @Injectable para definir servicios.

Dependency Injection en el Constructor

Con la ayuda de TypeScript pasa explícitamente un objeto tipo Logger, como se describe en la Figura 56 en el constructor de TodoService. De este modo, Angular es capaz de inferir la Inyección de Dependencias.

3.2.8. Dependency Injection

Tal como mencionan en la documentación oficial de Angular (2018) la inyección de dependencia (DI) es una forma de proporcionar una nueva instancia de una clase con las dependencias completamente formadas. La mayoría de las dependencias son servicios. Angular usa la inyección de dependencia para proporcionar nuevos componentes con los servicios que necesitan.

En la siguiente Figura 56 se observa que el componente TodoListComponent, necesita el servicio TodoService en el constructor:

```
//app/todo-list.component.ts
import {TodoService} from '../shared/todo.service';

export class TodoListComponent{
  //...some stuff...
  constructor(private service: TodoService) { }
}
```

Figura 56: Ejemplo de Inyección de dependencias (DI)
Fuente: (Angular, 2018)

Inyectando servicios a otros servicios.

Cuando un servicio necesita que se pase en el constructor la inyección de dependencias, se decora con **@Injectable**.

El Injector es el principal mecanismo detrás de la DI, a nivel interno, un inyector dispone de un contenedor con las instancias de servicios que crea él mismo. Si una instancia no está en el contenedor, el inyector crea una nueva y la añade al contenedor antes de devolver el servicio a Angular (Oriol, 2019)

[16] **Guideline:** Son pautas y documentación sobre un tema para desarrollar muestras.

Dicho de otro modo, la primera vez que se inyecta un servicio, el inyector lo instancia y lo guarda en un contenedor. Cuando se inyecta un servicio, antes de nada el inyector busca en su contenedor para ver si ya existe una instancia. A continuación en la Figura 57 especifica dicho proceso:

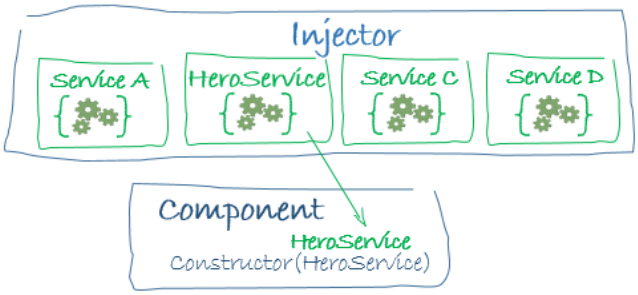


Figura 57: Estructura inyección de servicios
Fuente: (Angular, 2018)

Provider.

El provider es cualquier cosa que puede crear o devolver un servicio, es decir la propia clase define el servicio. Los providers pueden registrarse en cualquier nivel del árbol de componentes de la aplicación a través de los metadatos de componentes, o bien durante la fase de Bootstrapping[17] de la aplicación (Angular, 2018).

Librerías principales de Angular.

- **a)** @angular/core
- **b)** @angular/common
- **c)** @angular/router
- **d)** @angular/http

En el Anexo B sección 6 mediante un ejemplo se describe la implementación de las principales características de la documentación de Angular.

[17] **Bootstrapping:** Hace referencia a empezar algo sin recursos o con muy pocos recursos.

3.3. Análisis de Impactos

3.3.1. Análisis comparativo del Framework Angular vs React.

Los procesos de análisis de datos ayudan a identificar los datos útiles con el propósito de sacar conclusiones que permitan tomar mejores decisiones. Dentro de estos procesos se encuentra el análisis confirmatorio de datos que a través de indicadores confirman o no una hipótesis (Parra Olivares, 2002).

El presente estudio midió la eficiencia del DOM y velocidad de procesamiento de las peticiones y respuestas; con las herramientas de benchmarking de código abierto para frameworks Angular. Las mismas que se encuentran en el repositorio GitHub.

Benchmarking Angular y React es un proyecto que intenta medir la sobrecarga mínima de los Frameworks sin que se use plantillas HTML, manipulación de base de datos o información de depuración. El proyecto inicialmente hace la comparativa entre los tres Frameworks, los cuales actualmente se encuentran entre los más utilizados por los desarrolladores Frontend en un marco JavaScript.

El Anexo D provee de la información sobre las herramientas y configuración del mismo desde el repositorio GitHub.

Para medir el rendimiento de entornos de producción de Angular y React se realizó con la herramienta LogRocket; es un DVR para aplicaciones web, que graba literalmente todo lo que sucede en el sitio. En lugar de adivinar por qué ocurren los problemas, puede agregar e informar sobre problemas de rendimiento para comprender rápidamente la causa raíz (Borrelli, 2019).

Manipulación DOM.

Como se aprecia en la Figura 58 React tiene más ventaja sobre angular, el tamaño notablemente pequeño que se aprecia en la figura ayudó mucho a diferenciar cuando se trata de tiempo de inicio. Con respecto a Angular podemos admitir que funciona bastante bien.

Name Duration for...	angular-v8.0.1-keyed	react-v16.8.6-keyed
create rows creating 1,000 rows	123.0 ±3.8 (1.00)	137.4 ±3.7 (1.12)
replace all rows updating all 1,000 rows (5 warmup runs).	115.6 ±1.4 (1.07)	108.7 ±1.6 (1.00)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16x CPU slowdown.	114.0 ±5.2 (1.00)	136.7 ±3.6 (1.20)
select row highlighting a selected row. (5 warmup runs). 16x CPU slowdown.	26.0 ±2.5 (1.00)	30.4 ±3.5 (1.17)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	337.7 ±2.6 (4.89)	344.5 ±4.5 (4.98)
remove row removing one row. (5 warmup runs).	36.4 ±0.6 (1.00)	39.7 ±0.5 (1.09)
create many rows creating 10,000 rows	1,211.7 ± 30.2 (1.04)	1,443.8 ± 36.0 (1.24)

Figura 58: Manipulación DOM de Angular y React.
Fuente: (Srivastav, 2019)

Criterio de comparación.

La Figura 59 da a conocer el tamaño del paquete js de Angular, el cual es casi doble al de React y Vue. Los tamaños de paquete de React y Vue están muy cerca, pero con Vue teniendo el paquete más liviano tendría mayor relevancia.

Search this file...

	Angular	React	Vue
1			
2	65.5 KB	36.3 KB	30.8 KB

test.csv hosted with ♥ by GitHub view raw

Figura 59: Comparativa de ficheros js.
Fuente: (Srivastav, 2019)

Aceleración de CPU y Red.

En términos simples, es el tiempo necesario para ver realmente algo en la pantalla. En las Figuras 60 y 61 respectivamente está la comparación, está comparación se realiza utilizando la auditoría Lighthouse.

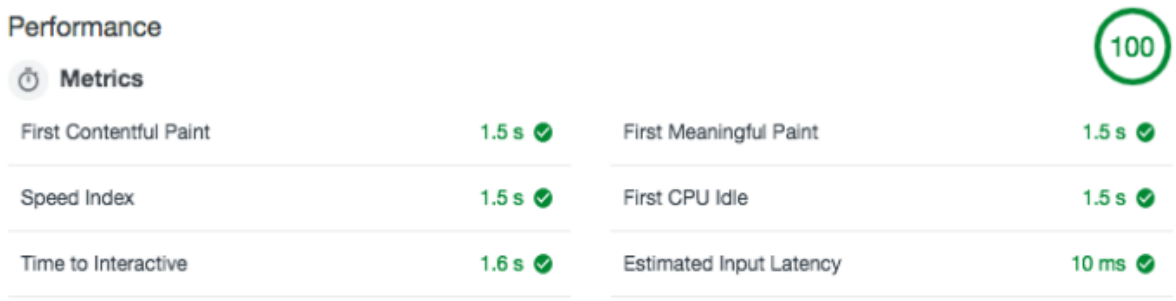


Figura 60: Aceleración de CPU en Angular
Fuente: (Srivastav, 2019)

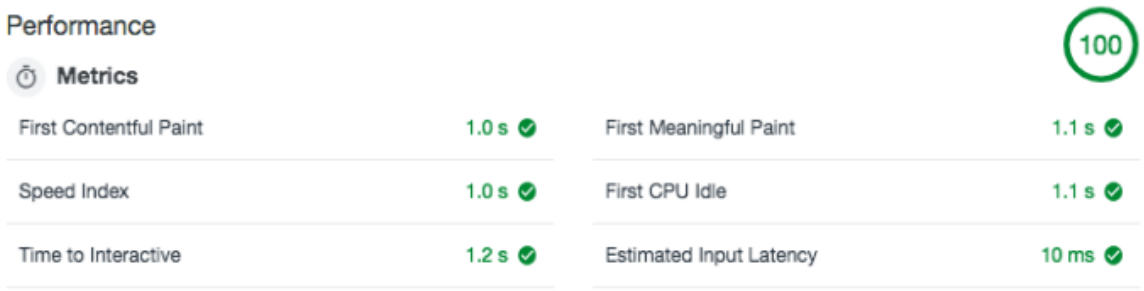


Figura 61: Aceleración de CPU en React
Fuente: (Srivastav, 2019)

En base a dicha auditoría podemos apreciar que el desempeño de React es mucho mejor con un tiempo de 1.0s, sin embargo 1.5s de Angular se considera satisfactoria.

Rendimiento del tiempo de carga.

Para la presente comparativa como se muestra en las Figuras 62 y 63 respectivamente se hace uso de las herramientas de desarrollo de Chrome, el rendimiento del tiempo de carga se considera como el tiempo necesario para el primer render, sin el tiempo de espera de la API.

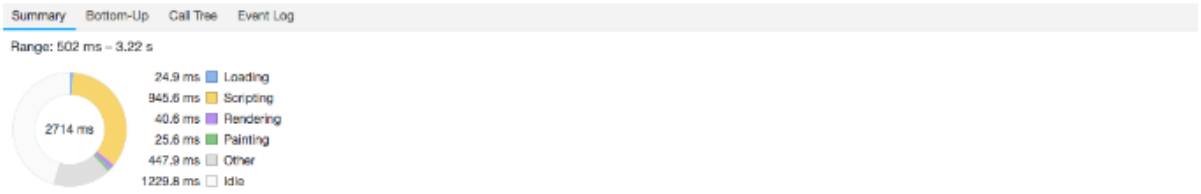


Figura 62: Rendimiento del tiempo de carga en Angular
Fuente: (Srivastav, 2019)

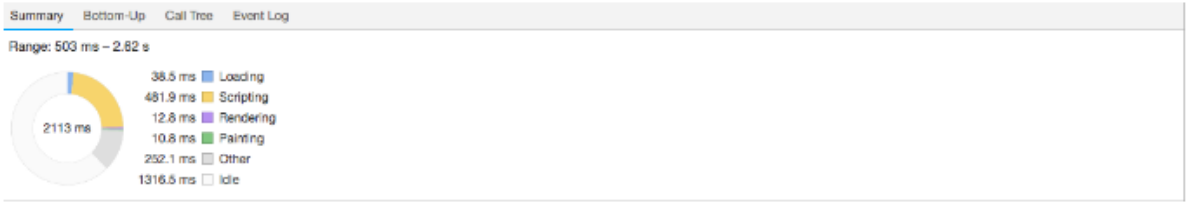


Figura 63: Rendimiento del tiempo de carga en React
Fuente: (Srivastav, 2019)

En base a la anterior auditoría y excluyendo el tiempo de inactividad, a continuación en la Figura 64 se muestra el tiempo necesario para el primer renderizado.

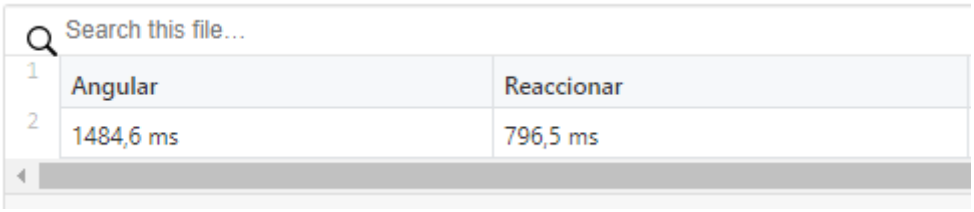


Figura 64: Tiempo necesario para el primer renderizado
Fuente: (Srivastav, 2019)

Como se puede apreciar React tardó menos, a diferencia de Angular que tardó casi el doble de tiempo en el primer renderizado.

Curva de aprendizaje.

Después del análisis anterior se puede apreciar las ventajas que tienen dichos Frameworks, los mismos que ha venido actualmente aportando notablemente en el desarrollo Frontend, sin embargo, como opinión personal Angular es más fácil de aprender, React tarda un poco en la curva de aprendizaje.

Como lo menciona Borrelli (2019) los pros generales de Angular con respecto a React son:

- Angular replica los cambios realizados en el modelo instantáneamente en las vistas de una manera fácil, eficiente e intuitiva.
- Angular tiene mucha tracción en el campo, con muchas comunidades y profesionales que lo apoyan.

3.3.2. Análisis de impactos del Sistema de Información, seguimiento y control de la Hacienda La Vega.

De acuerdo con las investigaciones y pruebas previas hechas con el modelo SUS, el mismo que da a conocer que si se estima un puntaje superior a 68 se considera superior al promedio, mientras que un valor inferior a 68 se considera por debajo del promedio. Sin embargo, en la Figura 65 da a conocer los rangos percentiles y los puntajes SUS se adaptan, lo cual nos permite saber si el puntaje SUS es de 68, el rango percentil equivalente es del 50%.

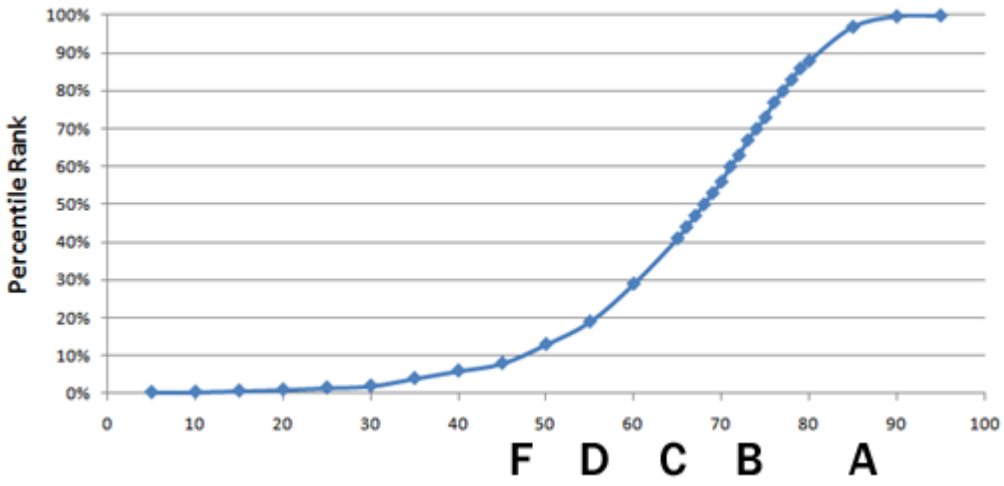


Figura 65: Puntaje SUS en percentiles
Fuente: (usability.gov, n.d.)

De acuerdo con la Figura 66 el puntaje SUS del proyecto fue de 69.25 lo que ubica en el rango percentil aproximado al 60%.

Finalmente, los puntajes SUS clasifican la escala de usabilidad por letras desde F a A, siendo A la clasificación mejor imaginable y F la peor, por lo cual ubica al Sistema de Información, Seguimiento y Control de la Hacienda Ganadera La Vega en la

clasificación C, mismo que se encuentra por encima del promedio de los sistemas en modo de producción.

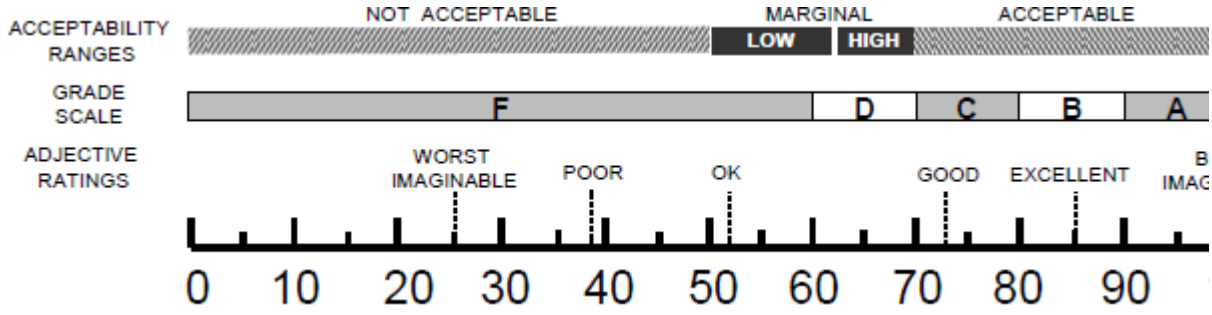


Figura 66: Escala de usabilidad.
Fuente: (usability.gov, n.d.)

En base a los resultados obtenidos se define los impactos; los cuales pueden ser positivos o negativos en un ámbito determinado, el desarrollo del Sistema de Información, Seguimiento y Control de la Hacienda Ganadera La Vega generó los siguientes impactos:

Impacto educativo.

El presente tema de estudió aportó positivamente en al ámbito educativo, permitiendo modelar los procesos en los que se lleva la información en la Hacienda Ganadera en base a una serie de procedimientos informáticos, que están formados por diferentes módulos y funcionalidades que controlan los procesos que realiza la Hacienda, con el fin de complementar en el conocimiento de los usuarios al manipular el sistema además, de ser un aporte cómo herramienta de estudio a las personas interesadas en el tema de investigación.

Impacto tecnológico.

El sistema mejoró el registro de información de la hacienda, permitiendo que tengan indicadores precisos e individuales sobre el control y seguimiento del ganado. Además, automatizó los procedimientos: registros de fechas de nacimientos, altas y bajas, potreros, situación actual y sanidad del ganado.

El sistema permitió disminuir el tiempo de espera para obtener un registro individual del ganado, búsquedas específicas, actualización de la información sin necesidad de dar de baja dicho registro. También mejoró en los indicadores de sanidad permitiendo

llevar un registro por fechas y dosis de tratamiento, siendo útil para posteriores tratamiento y permitiendo elegir el más conveniente en casos similares.

El sistema optimizó el tiempo de tomar decisiones con respecto a elegir el potrero adecuado de acuerdo al hato solicitado, permitiendo ver características específicas de dicho potrero, además de llevar un historial adecuado del potrero como por ejemplo: fechas de riego, días de recuperación, producto aplicado y estado de las cercas.

Impacto económico.

El sistema ayudó a precisar el procedimiento estadístico y matemático para conocer con exactitud la cantidad de dosis para un tratamiento específico ya sea del ganado como de los potreros, historiales de producción de leche, totales diarios y mensuales de la producción de leche.

El sistema benefició económicamente a la hacienda ganadera ya que incrementó la productividad en ella, permitiéndole tomar mejores decisiones a corto plazo, de igual manera de llevar el proceso de seguimiento y control de forma automática, generando de esta manera un ambiente amigable con los usuarios que hacen uso de este software.

Impacto ambiental.

El proceso para la generación, guardado y búsqueda de registro específicos, se lo realizaba manualmente, además la tarea obtener un análisis de producción llevaba mucho tiempo, por lo que la implementación del sistema permitió reducir el uso de útiles de oficina y disminuir los tiempos de respuesta.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

- Angular permitió obtener una visión más amplia en el manejo de una estructura en base a componentes, que básicamente se estructura del componente, el selector que parecerá en forma de tag en el código HTML, la vista. El Framework Angular permitió crear pequeños componentes que facilitaron a la capa de presentación que conjuntamente con las otras herramientas utilizadas fortalecieron el desarrollo de un sistema Data Binding mediante templating, directivas y pipes para obtener una interfaz de usuario más amigable.
- La documentación técnica del Framework Angular elaborada permitió al lector interesado empezar a desarrollar su primera aplicación SPA, brindándole los literales o pasos necesarios para comprender la arquitectura iniciar del framework y así poder ir escalando en base a los requerimientos del negocio.
- Al implementar el Sistema de Información, Seguimiento y Control de la Hacienda Ganadera La Vega, ayudó a la asignación de tareas mediante el módulo de usuarios, roles y permisos del sistema. El módulo de sanidad facilitó al veterinario tratante a conocer el historial de medicamentos utilizados, lo cual ayuda a realizar un diagnóstico más completo y poder usar el mismo tratamiento en otro caso similar. El módulo de ganadería permitió obtener el historial del animal desde su nacimiento y el historial de producción del mismo. Lo cual ayudó a generar un historial digital, que reemplazó al historial realizado en papel que anteriormente se manejaba; favoreciendo al impacto económico y ambiental.
- El análisis de la comparativa ayudó a tener una perspectiva clara del framework Angular que permite desarrollar aplicaciones más confiables y eficientes en relación a procesamiento de peticiones y manejo de datos, mientras que el Framework React el tiempo de respuesta la curva de aprendizaje es mucho en relación a Angular.

Recomendaciones

- Es recomendable hacer uso del sitio oficial del framework el cual provee de información bastante útil y didáctica para conocer la situación actual del funcionamiento, diseño y actualizaciones de Angular, para posteriormente consultar eBooks que ayuden a complementar inquietudes en la utilización de framework.
- En cuanto a la documentación técnica es recomendable hacer uso de la herramienta Git, la cual se integra en la creación de un nuevo proyecto en Angular con el fin de supervisar y mantener un historial de cada etapa del desarrollo del software que sirva al equipo de desarrollo disponer de manera inmediata los cambios realizados en el código y poder mantener actualizado el documento.
- Es recomendable socializar con los usuarios que harán uso del sistema, dando a conocer las ventajas que conllevan el registro de la información en un Software y los resultados que llegarían a obtener. En el despliegue a producción se considera hacer realizarlo en un VPS (Virtual Private Server) ya que se acopla adecuadamente, además es más sencilla la configuración de despliegue o producción del Frontend y Backend. En la fase de desarrollo, es recomendable que se desarrolle el entorno de API Rest como una aplicación separada, diferente del Frontend y Backend, con el fin de facilitar el mantenimiento del código.
- Es recomendable hacer uso de Angular ya que incluye los aspectos que pueda necesitar para crear una aplicación del lado cliente que puede ser una pequeña aplicación que a futuro pueda ir escalando gracias a que incluye la generación de vistas, el uso de Data Binding, las rutas, la organización de componentes en módulos.

REFERENCIAS

Acens. (2018). *Framework para el desarrollo ágil de aplicaciones* (Vol. 1). Retrieved from https://www.acens.com/wp-content/images/2014/03/frameworks-white-paper-acens-.pdf

Acosta, V. (2019). Beneficios de la utilización de Frontend con Angular. Retrieved January 4, 2020, from https://revistadigital.inesem.es/informatica-y-tics/frontend-con-angular-todo-lo-que-debes-saber-sobre-esta-herramienta/

Agrocalidad. (2012). *GUÍA DE BUENAS PRÁCTICAS PECUARIAS*. Retrieved from http://www.agrocalidad.gob.ec/wp-content/uploads/downloads/2013/10/Guia de Buenas Practicas Pecuarias en Leche - editada.pdf

Aguilera, R. (2016). ¿Por qué Angular 2? - Adictos al trabajo. Retrieved November 14, 2019, from https://www.adictosaltrabajo.com/2016/02/15/por-que-angular-2/

Alvarez, M. (2016). Qué es una SPA. Retrieved November 14, 2019, from https://desarrolloweb.com/articulos/que-es-una-spa.html

Alvarez, M., & Basalo, A. (2018). *Introducción : Manual de Angular*. Retrieved from http://desarrolloweb.com/manuales/manual-angular-2.html

Angular. (2018). Architecture Overview of Angular. Retrieved November 14, 2019, from https://v2.angular.io/docs/ts/latest/guide/

Borrelli, P. (2019). Angular vs. React vs. Vue: una comparación de rendimiento - Blog de LogRocket. Retrieved January 24, 2020, from https://blog.logrocket.com/angular-vs-react-vs-vue-a-performance-comparison/

Budiman, E., Puspitasari, N., Wati, M., Haviluddin, & Rahim, R. (2019). Model Framework for Development of Biodiversity Information Systems. *Journal of Physics: Conference Series*, *1230*, 012012. https://doi.org/10.1088/1742-6596/1230/1/012012

Burró, J. D. (2014). Características y arquitectura de un SPA - formandomeformandome. Retrieved November 14, 2019, from http://www.formandome.es/javascript/spa-arquitectura-caracteristicas/

Camacho, F. (2016). Metodologías de Desarrollo. Retrieved November 14, 2019, from

http://diferenciasmetodologias.blogspot.com/

CampusMVP. (2019). ¿Es Angular 2, Angular 4 o simplemente Angular? Retrieved November 14, 2019, from 03/04/2017 website: https://www.campusmvp.es/recursos/post/es-angular-2-angular-4-o-simplemente-angular.aspx

Carvajal, J. U. (2017). Aplicación CRUD con Angular 4. Retrieved November 14, 2019, from https://joseucarvajal.wordpress.com/2017/09/03/angular-4-crud-application/

Cuervo, V. (2019). Tests en Postman – Arquitecto IT. Retrieved December 21, 2019, from http://www.arquitectoit.com/postman/tests-postman/

De La Fuente, J. (2016). *Integración de Single-Page Application en Liferay*. Retrieved from https://upcommons.upc.edu/bitstream/handle/2117/89976/118457.pdf

Domic, E. (2015). Importancia de los registros de enfermería. *Medwave*, *4*(4), 1–20. https://doi.org/10.5867/medwave.2004.04.2793

Elrom, E. E. (2020). Angular 9 vs React 16 — what should you pick? a 2020 showdown. Retrieved April 20, 2020, from https://medium.com/blockchain-developer/angular-9-vs-react-16-a-2020-showdown-2b0b8aa6c8e9

Farina, A. (2015). Navegadores Web - Linea del tiempo. Retrieved November 14, 2019, from https://prezi.com/p7jaih8mz8ll/navegadores-web-linea-del-tiempo/

García, J. M. (2016). ¿Por qué Angular 2? - SlashMobility | Soluciones mobile. Retrieved November 13, 2019, from http://slashmobility.com/blog/2016/10/por-que-angular-2/

Gonzáles, E. (2017). *Control de Rendimiento Deportivo para Competidores de Triatlón* (UNIVERSIDAD MAYOR DE SAN ANDRÉS). Retrieved from https://repositorio.umsa.bo/bitstream/handle/123456789/16685/T-3354.pdf?sequence=1&isAllowed=y

Granda, J. M. (2014). *Sistema para el manejo de fincas ganaderas - simafig mediante herramientas JEE y metodología U.W.E.* (Universidad de Fuerzas Armadas ESPE). Retrieved from http://repositorio.espe.edu.ec/handle/21000/8509

Grijalva, J. (2016). Las pymes procesan 800.000 litros de leche. Retrieved November

14, 2019, from Entrevista website: https://www.eltelegrafo.com.ec/noticias/economia/4/las-pymes-procesan-800-000-litros-de-leche

Hernandez, U. (2015). MVC (Model, View, Controller) explicado. Retrieved November 14, 2019, from Pag Web website: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado

IBM Knowledge Center. (2019). Patrón de diseño de modelo-vista-controlador. Retrieved November 14, 2019, from https://www.ibm.com/support/knowledgecenter/es/SSZLC2_8.0.0/com.ibm.commerce.developer.doc/concepts/csdmvcdespat.htm

ImaginaGroup. (2019). Angular 8: Te contamos todas sus novedades. Retrieved December 18, 2019, from Imagina Formación website: https://www.imaginaformacion.com/angular-8-novedades/

Jdonsan. (2016). Angular2: ¿Y si nos damos un tiempo? Retrieved November 14, 2019, from https://elabismodenull.wordpress.com/2016/01/13/angular2-y-si-nos-damos-un-tiempo/

Jonsson, D. (2014). La producción lechera en Ecuador genera $ 1.600 millones en ventas anuales (Infografía). In *El Telegrafo*. Retrieved from https://www.eltelegrafo.com.ec/noticias/economia/4/la-produccion-lechera-en-ecuador-genera-1-600-millones-en-ventas-anuales-infografia%0Ahttp://www.telegrafo.com.ec/economia/item/la-produccion-lechera-en-ecuador-genera-1-600-millones-en-ventas-anuales-inf

López, J. M. (2018). Arquitectura en proyectos Angular. Retrieved November 15, 2019, from http://juanlopez.com.ar/arquitectura-en-proyectos-angular/

Medina, M. J. (2017). *UNIVERSIDAD TÉCNICA DEL NORTE BIBLIOTECA UNIVERSITARIA* (Universidad Técnica del Norte). Retrieved from http://repositorio.utn.edu.ec/bitstream/123456789/6613/1/04 ISC 435 TRABAJO DE GRADO.pdf

Menéndez, R., & Barzanallana, A. (2016). Historia del desarrollo de aplicaciones Web. Universidad de Murcia. Retrieved November 13, 2019, from

https://www.um.es/docencia/barzana/DIVULGACION/INFORMATICA/Historia-desarrollo-aplicaciones-web.html%0Ahttp://www.um.es/docencia/barzana/DIVULGACION/INFORMATICA/Historia-desarrollo-aplicaciones-web.html

Minar, I. (2017). ¿Por qué Angular 4 es sólo Angular? Retrieved November 14, 2019, from https://platzi.com/blog/angular/

Munguía, E. (2016). Comparación React JS vs Angular. Retrieved November 14, 2019, from http://www.enrique7mc.com/2016/01/comparacion-react-js-vs-angular-2/#prettyPhoto

Navarro, A., Fernández, J., & Morales, J. (2013). *Revisión de metodologías ágiles para el desarrollo de software A review of agile methodologies for software development*. Retrieved from https://www.redalyc.org/pdf/4962/496250736004.pdf

Node.js-Foundation. (2019). Documentación | Node.js. Retrieved November 14, 2019, from https://nodejs.org/es/docs/

Nonaka, I., & Takeuchi, H. (2019). Modelo original de Scrum para desarrollo de software - Scrum Manager BoK. Retrieved January 21, 2020, from http://www.scrummanager.net/bok/index.php?title=Modelo_original_de_Scrum_para_desarrollo_de_software

OK Hosting. (2016). Metodologías del Desarrollo de Software. Retrieved November 14, 2019, from OK hosting website: https://okhosting.com/blog/metodologias-del-desarrollo-de-software/

Oriol, E. (2016). Introducción a Angular Dependency Injection. Retrieved November 14, 2019, from http://blog.enriqueoriol.com/2016/07/introduccion-angular-2-parte-iv-dependency-injection.html

Oriol, E. (2019a). Introducción a Angular Módulo, Componente, Template y Metadatos. Retrieved November 14, 2019, from 30/06/2016 website: http://blog.enriqueoriol.com/2016/06/introduccion-a-angular-2-parte-i-componente.html

Oriol, E. (2019b). Introducción a Angular Módulos. Retrieved November 15, 2019,

from http://blog.enriqueoriol.com/2016/06/introduccion-a-angular-2-parte-i-componente.html

Parra Olivares, J. (2002). Análisis exploratorio y análisis confirmatorio de datos. Retrieved January 23, 2020, from https://www.redalyc.org/articulo.oa?id=12211106

Quijano, J. (2015). Hello World en TypeScript, el lenguaje en el que se construirá Angular 2. Retrieved November 14, 2019, from Genbeta website: https://www.genbetadev.com/javascript/hello-world-en-typescript-el-lenguaje-en-el-que-se-construira-angular-2

Redondo, F. (2017). Postman: gestiona y construye tus APIs rápidamente - Paradigma. Retrieved November 14, 2019, from Paradigmadigital website: https://www.paradigmadigital.com/dev/postman-gestiona-construye-tus-apis-rapidamente/

Resabala, C., & Recalde, P. (2017). Evaluación de la estabilidad del acaricida amitraz con solventes de baja toxicidad para uso veterinario. Retrieved November 14, 2019, from http://www.ghbook.ir/index.php?name=های ر ساندە و ف ر هنگ ذوی ن&option=com_dbook&task=readonline&book_id=13650&page=73&chkhash=ED9C9491B4&Itemid=218&lang=fa&tmpl=component

Rodrigues, M. (2012). Las TIC como herramienta para la superación de asimetrías. In *TIC y agricultura*. Retrieved from http://www.cepal.org/socinfo/noticias/paginas/3/44733/newsletter18.pdf

Santos, W. D., & Serrano, J. (2017). *Desarrollo de una Api Rest con sus Aplicaciones Web Y Móvil para la Venta De Ropa Online de la Empresa Roosman* (UNIVERSIDAD CENTRAL DEL ECUADOR). Retrieved from http://www.dspace.uce.edu.ec/bitstream/25000/9668/1/T-UCE-0011-312.pdf

Srivastav, A. (2019). Benchmarking Angular, React y Vue para pequeñas aplicaciones web. Retrieved January 24, 2020, from https://blog.bitsrc.io/benchmarking-angular-react-and-vue-for-small-web-applications-e3cbd62d6565

usability.gov. (n.d.). System Usability Scale (SUS) | Usability.gov. Retrieved January 24, 2020, from https://www.usability.gov/how-to-and-tools/methods/system-

usability-scale.html

Valerio, J. C. (2016). Angular 2: historia, características y métodos de instalación. Retrieved November 13, 2019, from https://medium.com/@jc.valerio.b/angular-2-historia-características-y-métodos-de-instalación-11492ea67e2b#.cu0eckdm3

Valiñas, D. G. (2019). Angular 8 ya está aquí: ¿Qué trae nuevo? | campusMVP.es. Retrieved December 21, 2019, from https://www.campusmvp.es/recursos/post/angular-8-ya-esta-aqui-que-trae-nuevo.aspx

Wolrd Economic Forum. (2016). Global Competitiveness Report 2015-2016 - Reports - World Economic Forum. Retrieved November 13, 2019, from WEforum.org website: http://reports.weforum.org/global-competitiveness-report-2015-2016/economies/#indexId=GCI&economy=ECU

GLOSARIO DE TÉRMINOS

[1] **GCI:** Índice de Competitividad Global.

[2] **TypeScript:** Lenguaje de programación libre y de código abierto, súper conjunto de JavaScript, basados en clases.

[3] **Cross Platform:** Programa o dispositivo que puede utilizarse sin inconvenientes en distintas plataformas de hardware y sistemas operativos.

[4] **Hato:** Conjunto de animales de ganado mayor o menor.

[5] **Google Trends:** Herramienta que representa con cuánta frecuencia se realiza una búsqueda de un término particular en varias regiones del mundo.

[6] **Web components:** Conjunto de APIs Web que permiten crear etiquetas HTML personalizables.

[7] **ECMAScript:** Estándar que define cómo ser interpretado el lenguaje en cada una de las tecnologías.

[8] **Data Binding:** Coordina partes de una plantilla con partes de un Component.

[9] **SEMVER:** Semantic Versioning se relaciona al contexto de añadir significado a los números de las versiones.

[10] **Routing:** es el encargado de reconocer cuál es la ruta que el usuario desea mostrar, presentando la pantalla correcta en cada momento.

[11] **IBM:** Empresa líder en la investigación, desarrollo y fabricación de las tecnologías de la información más avanzadas del sector.

[12] **API REST:** Es un servicio que puede usarse desde cualquier dispositivo que entienda el protocolo HTTP, además interactúa directamente con la base de datos.

[13] **IntelliSense:** Permiten obtener información acerca del código que utiliza, realiza un seguimiento de los parámetros que escribe y a agrega llamadas a propiedades y a métodos.

[14] **Hándicap:** condición en la cual se advierte una desventaja de una cosa en relación con otra.

[15] **UI:** User Interfaz o Interfaz de Usuario es el medio que permite la interacción de un usuario con un sistema computacional.

[16] **Guideline:** Son pautas y documentación sobre un tema para desarrollar muestras.

[17] **Bootstrapping:** Hace referencia a empezar algo sin recursos o con muy pocos recursos.

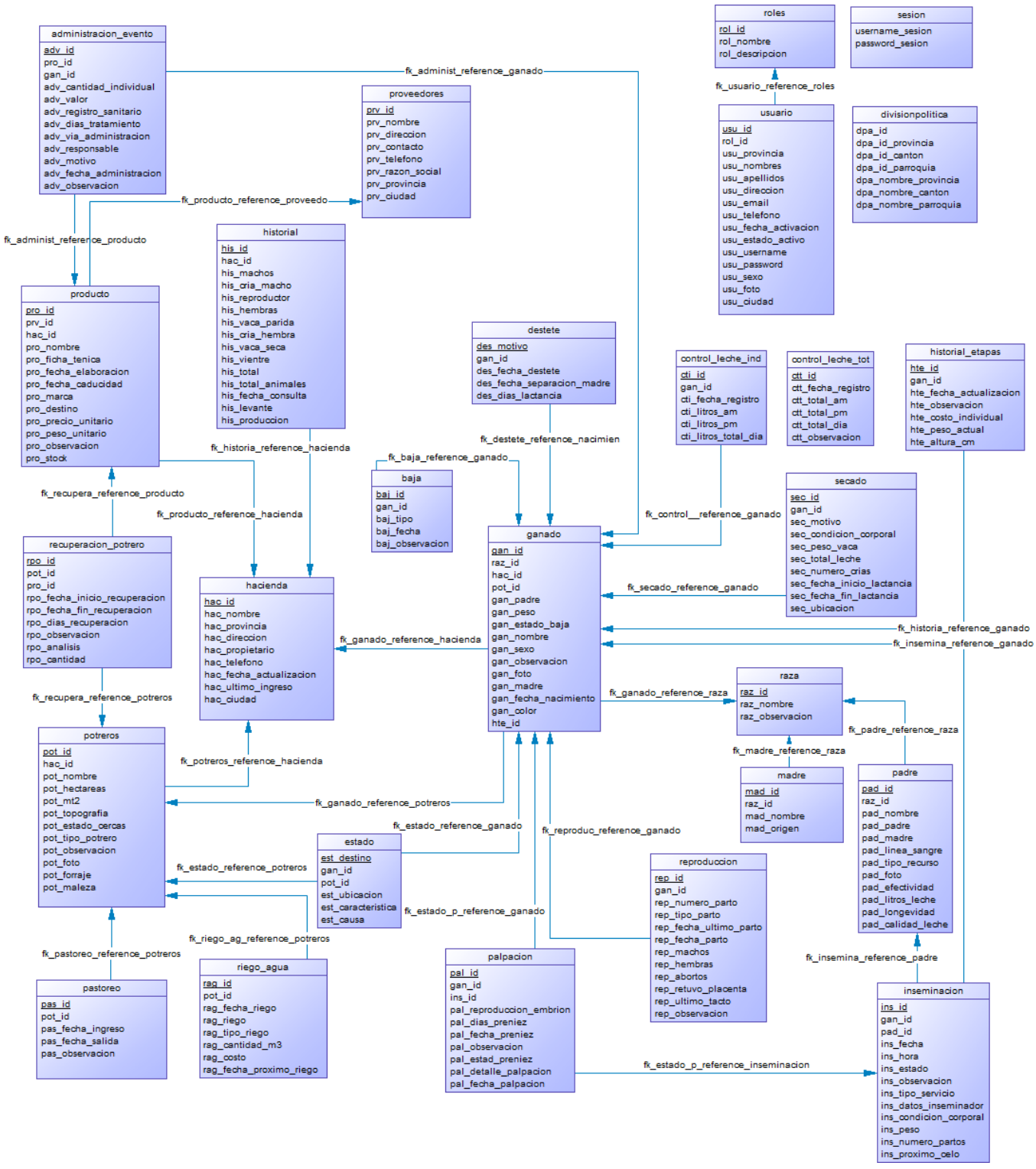
ANEXOS

Los anexos descritos puede encontrarlos en la carpeta Anexos que se encuentra dentro del CD del presente proyecto.

Anexo A: Manual técnico del Sistema de información, seguimiento y control de la hacienda ganadera “La Vega” (en CD).

Anexo B: Documentación técnica del Framework Angular (en CD).

Anexo C: Diseño y descripción de las tablas generadas para el desarrollo del sistema.



Anexo D: Información sobre las herramientas y configuración del mismo desde el repositorio GitHub (en CD).

Anexo E: Formato de encuesta de usabilidad del software Sistema de información, seguimiento y control de la hacienda ganadera “La Vega”.

UNIVERSIDAD TÉCNICA DEL NORTE

Usabilidad del Sistema de Información, seguimiento y control de la Hacienda La Vega

1. Creo que usaría este sistema frecuentemente.
   - Total desacuerdo.
   - En desacuerdo.
   - Ni acuerdo, ni desacuerdo.
   - De acuerdo.
   - Total acuerdo.
2. Encuentro este sistema innecesariamente.
   - Total desacuerdo.
   - En desacuerdo.
   - Ni acuerdo, ni desacuerdo.
   - De acuerdo.
   - Total acuerdo.
3. Creo que el sistema fue fácil de usar.
   - Total desacuerdo.
   - En desacuerdo.
   - Ni acuerdo, ni desacuerdo.
   - De acuerdo.
   - Total acuerdo.
4. Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar este sistema.
   - Total desacuerdo.
   - En desacuerdo.

- Ni acuerdo, ni desacuerdo.
- De acuerdo.
- Total acuerdo.

5. Las funciones de este sistema están bien integradas.
   - Total desacuerdo.
   - En desacuerdo.
   - Ni acuerdo, ni desacuerdo.
   - De acuerdo.
   - Total acuerdo.
6. Creo que el sistema es muy inconsistente.
   - Total desacuerdo.
   - En desacuerdo.
   - Ni acuerdo, ni desacuerdo.
   - De acuerdo.
   - Total acuerdo.
7. Imagino que la mayoría de la gente aprendería a usar este sistema en forma muy rápida.
   - Total desacuerdo.
   - En desacuerdo.
   - Ni acuerdo, ni desacuerdo.
   - De acuerdo.
   - Total acuerdo.
8. Encuentro que el sistema es muy difícil de usar.
   - Total desacuerdo.
   - En desacuerdo.
   - Ni acuerdo, ni desacuerdo.
   - De acuerdo.
   - Total acuerdo.
9. Me siento confiado al usar este sistema.
   - Total desacuerdo.
   - En desacuerdo.
   - Ni acuerdo, ni desacuerdo.
   - De acuerdo.
   - Total acuerdo.

10. Necesité aprender muchas cosas antes de ser capaz de usar este sistema.
    - Total desacuerdo.
    - En desacuerdo.
    - Ni acuerdo.
    - Ni desacuerdo.
    - De acuerdo.

      Total acuerdo.