EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Colima

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

SISTEMA WEB DE APOYO A LA TOMA DE DECISIONES PARA LA COMERCIALIZACIÓN DE PLANTAS ORNAMENTALES

TESIS

QUE PARA OBTENER EL GRADO DE
MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:
ING. HÉCTOR ADÁN MORALES LUGO

DIRECTOR DE TESIS:
D. en C. NICANDRO FARÍAS MENDOZA

CO-DIRECTORA:
D. en C. PATRICIA ELIZABETH FIGUEROA MILLÁN

VILLA DE ALVAREZ, COLIMA, 14 DE AGOSTO DE 2020

Villa de Álvarez, Colima, **4/Septiembre/2020**
Oficio No. **DEPI 1.2.11/115/2020**

ALUMNO MORALES LUGO HÉCTOR ADÁN
PASANTE DE LA MAESTRÍA EN SISTEMAS COMPUTACIONALES
PRESENTE

La División de Estudios de Posgrado e Investigación de acuerdo al procedimiento para la obtención del Título de Maestría de los Institutos Tecnológicos y habiendo cumplido con todas las indicaciones que la comisión revisora hizo a su trabajo profesional denominado **"SISTEMA WEB DE APOYO A LA TOMA DE DECISIONES PARA LA COMERCIALIZACIÓN DE PLANTAS ORNAMENTALES"**, por la opción de tesis, que para obtener el grado de Maestro en Sistemas Computacionales será presentado por Usted, tiene a bien concederle la **AUTORIZACIÓN** de impresión de la tesis citada.

Sin otro particular por el momento, aprovecho la ocasión para enviarle un cordial y afectuoso saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®

RAMONA EVELIA CHÁVEZ VALDEZ
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

SEP - TecNM
INSTITUTO TECNOLÓGICO
De Colima
DIVISIÓN DE ESTUDIOS DE
POSGRADO E INVESTIGACIÓN

RECV/cas

C.p. Archivo.

AGRADECIMIENTOS

En primer lugar, deseo expresar mi agradecimiento a la Dra. Patricia Elizabeth Figueroa Millán, quien con sus conocimientos, apoyo y profesionalismo me guio a la largo del desarrollo de esta investigación para lograr los resultados esperados. No hubiese podido obtener estos resultados de no haber sido por su ayuda incondicional.

Asimismo, quiero agradecer a mi comité de revisores el Dr. Nicandro Farias Mendoza, la Mtra. Ramona Evelia Chávez Valdez y al Dr. Jesús Alberto Verduzco Ramírez por brindarme su apoyo y sugerencias en la revisión de esta Tesis. Gracias por la confianza ofrecida desde el comienzo de esta nueva etapa.

También quiero agradecer al Tecnológico Nacional de México campus Instituto Tecnológico de Colima y al Consejo Nacional de Ciencia y Tecnología (CONACYT) por todos los recursos y herramientas que fueron necesarios para llevar a cabo el proceso de investigación.

A todos, muchas gracias.

EPÍGRAFE

“Las máquinas deben trabajar. La gente debe pensar.”

Richard Hamming (1915-1998)
Matemático, escritor y profesor estadounidense.

RESUMEN

La horticultura ornamental en México es una industria en crecimiento que requiere de la inclusión de diversas tecnologías para automatizar los procesos de producción y comercialización, a fin de incrementar su rentabilidad. Para esto, el análisis de los datos es un factor clave para generar estrategias de producción y comercialización, permitiendo la obtención de conocimiento para el soporte a la toma de decisiones; no obstante, implica un tiempo exhaustivo de procesamiento de información, lo cual afecta principalmente a la productividad de las empresas, debido a la carencia de un sistema de apoyo a la toma de decisiones que implemente herramientas dinámicas de inteligencia de negocios. Por lo tanto, para el desarrollo de esta investigación se tomó como caso de estudio la Sociedad de Producción Rural Ornamentales de Colima (ORNACOL) de R.L de C.V, la cual cuenta con un sistema para la comercialización de plantas ornamentales; sin embargo, necesita de un sistema para optimizar el análisis de los datos históricos almacenados sobre sus ventas y que permita apoyar el proceso de toma de decisiones; por esto, se desarrolló e implementó un sistema web de soporte a la toma de decisiones para la comercialización de plantas ornamentales. Este sistema permite la creación de herramientas dinámicas de inteligencia de negocias y la ejecución de consultas asíncronas a la base de datos, logrando efectuar decisiones eficientes, eficaces y oportunas; además, admite el análisis de la información histórica de la comercialización de plantas ornamentales mediante tablas, gráficas y reportes. Está desarrollado utilizando la metodología PUA, el lenguaje de programación Python y el framework Django, empleando un enfoque innovador al aplicar el algoritmo DFS como mecanismo de búsqueda para determinar la relación existente entre las tablas de la base de datos, mejorando el aprovechamiento de la información histórica almacenada, así como el apoyo a la toma de decisiones para la comercialización de la empresa, reduciendo el tiempo de extracción, procesamiento, análisis y presentación de información. Las características innovadoras de este sistema web permitirá en un futuro próximo incrementar la productividad y competitividad de la empresa.

Palabras clave: *sistema DSS, inteligencia de negocios, análisis de datos, algoritmo DFS.*

ABSTRACT

Ornamental horticulture in Mexico is a growing industry that requires the inclusion of various technologies to automate production and marketing processes, to increase its profitability. For this, data analysis is a key factor to generate production and marketing strategies, allowing obtaining knowledge to support decision-making; However, it involves exhaustive information processing time, which mainly affects the productivity of companies, due to the lack of a decision support system that implements dynamic business intelligence tools. Therefore, for the development of this research, the Sociedad de Producción Rural Ornamentales de Colima (ORNACOL) of R.L de C.V was taken as a case study, which has a system for the commercialization of ornamental plants; however, you need a system to optimize the analysis of the historical data stored on your sales and to support the decision-making process; For this reason, a web system to support decision-making was developed and implemented for the commercialization of ornamental plants. This system allows the creation of dynamic business intelligence tools and the execution of asynchronous queries to the database, achieving efficient, effective and timely decisions; In addition, it admits the analysis of the historical information of the commercialization of ornamental plants through tables, graphs and reports. It is developed using the AUP methodology, the Python programming language and the Django framework, employing an innovative approach by applying the DFS algorithm as a search mechanism to determine the relationship between the database tables, improving the use of information stored history, as well as support for decision-making for the commercialization of the company, reducing the time of extraction, processing, analysis and presentation of information. The innovative characteristics of this web system will allow to increase the productivity and competitiveness of the company in the near future.

Key words: *DSS system, business intelligence, data analysis, DFS algorithm.*

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1. INTRODUCCIÓN

La horticultura es un segmento de la agricultura relacionado no solo con la cultura vegetal, conocida como horticultura alimentaria; sino también, con un gran número de especies como plantas para el embellecimiento de espacios físicos, a lo cual se le conoce como horticultura ornamental. La horticultura involucra diferentes áreas altamente relacionadas que ofrecen grandes oportunidades para educación, recursos, industrias profesionales, tecnología y empleo (Duarte de Oliveira Paiva, 2018). En cuanto a lo anterior, la tecnología ha demostrado ser una herramienta clave para mejorar la capacidad productiva, la competitividad y la comercialización de productos, incluyendo los productos de la horticultura ornamental.

Sin embargo, para que la horticultura ornamental en México pueda ser considerada como una industria ornamental con alcances internacionales de competitividad, se requiere de grandes pasos para solventar los problemas de hardware, conocimiento, infraestructura y cooperación, ya que se observa la inclusión de la tecnología en tres niveles de estratificación: bajo, medio y alto; lo cual, afecta negativamente cuestiones como el análisis, priorización, toma de decisiones y diseño de futuros sistemas (García Mejía et al., 2018). Además, la focalización es un factor clave para asegurar el valor implícito de la información, como elemento principal para la toma de decisiones; es decir, es fundamental que se determine el conjunto de variedades estratégicas de las flores y plantas de ornato con la finalidad de orientar la generación de bases mínimas de información y así comprender el funcionamiento de la oferta y las necesidades específicas de la demanda (COEPPLANTS, 2012)

Por lo tanto, la optimización de los procesos y gestión de la información de las empresas que comercializan de manera electrónica genera una gran cantidad de información, que, si se utiliza correctamente, puede mejorar la toma de decisiones tanto en la comercialización de empresa a empresa y en la comercialización de empresa a clientes. El análisis de datos históricos es un factor clave para generar estrategias de comercialización y producción, permitiendo la obtención de conocimiento para el soporte a la toma de decisiones. No obstante, para las habilidades de un humano es un proceso complejo, tedioso, ineficaz e ineficiente; lo cual, afecta la productividad de las empresas. Ante esto, surgen distintas estrategias para utilizar los datos de estas y convertirlos en información útil para la toma de decisiones, tal es el caso de los Sistemas de Apoyo a la Toma de Decisiones (DSS, por sus siglas en inglés) que cuentan con características de herramientas de Inteligencia de Negocios (BI, por sus siglas en inglés).

Los sistemas DSS son considerados como sistemas de información basados en ordenadores que combinan modelos y datos, que tienen la finalidad de resolver problemas semiestructurados con una amplia implicación del usuario, permitiendo otorgar herramientas que posibilitan la toma de decisiones analítica y cuantitativa con base en la información resultante (Power, 2011). Por otro lado, las herramientas BI combinan la obtención y almacenamiento de datos con herramientas analíticas que presentan información compleja y competitiva a los tomadores de decisiones (Salgueiro et al., 2012). Éstas se utilizan en la generación, tratamiento y comunicación de la información, proporcionando una visión

estratégica del negocio para transformar grandes cantidades de datos en información de calidad; asimismo, proveen recursos como análisis, pronóstico, monitoreo, control y optimización para gestionar la calidad de los procesos y tomar decisiones oportunas (Silva et al., 2016).

Los DSS ayudan a mejorar la toma de decisiones de cada una de las partes que intervienen en una iniciativa de comercio electrónico (Ruiz G. et al., 2009, p. 1). Aunado a esto, el uso de las características de herramientas de inteligencia de negocios permite desarrollar un sistema eficiente para la toma de decisiones. Por lo tanto, en esta investigación se propone el desarrollo de un sistema web de apoyo a la toma de decisiones para la comercialización de plantas ornamentales que integra la inteligencia de negocios para crear una herramienta de soporte a la toma de decisiones innovadora.

1.1 La naturaleza del problema

Actualmente, para incrementar la rentabilidad de una empresa, es necesario almacenar toda la información relacionada con sus procesos de comercialización (clientes, proveedores, productos o ventas, trazabilidad, etc.), con la finalidad de extraer, procesar y analizar la información para una toma de decisiones eficiente y eficaz. Sin embargo, para almacenar la información de los procesos de comercialización se requiere de un sistema gestor que permita automatizar los procesos y transacciones de comercialización que generalmente se realizan de manera manual, apoyando los procesos de toma de decisiones con respecto a la comercialización de productos, en este caso plantas ornamentales, ya que analizar grandes cantidades de información de comercialización requiere de un tiempo significativo de procesamiento y análisis de los datos. Por lo tanto, la utilización de sistemas de apoyo a la toma de decisiones con características de herramientas de inteligencia de negocios beneficia la toma de decisiones oportunas y la reducción de los tiempos de análisis de la información.

1.2 El contexto del problema

En la Sociedad de Producción Rural Ornamentales de Colima (ORNACOL), se lleva a cabo un proceso de producción y comercialización de plantas ornamentales en donde se registra toda la información en una BDR externa al sistema web que gestiona la comercialización, así como algunos requerimientos de la trazabilidad hacia delante de las plantas ornamentales, apegándose al manual de Trazabilidad de Productos Hortofrutícolas para consumo en fresco de los Estados Unidos Mexicanos (SAGARPA, 2018).

Sin embargo, para esta sociedad de producción rural es fundamental conocer el estado de la comercialización de plantas ornamentas, es decir, información sobre las ventas en cierta fecha, productos que menos se venden, productos que se venden más, clientes potenciales, historial de ventas máximas y mínimas por vivero, productividad por año, entre otras. Todo esto para generar las mejores estrategias de comercialización y producción; por lo cual, el procesamiento y análisis de los datos históricos son un punto clave en la generación de esas estrategias, ya que permite extraer conocimiento de los datos históricos de comercialización para el soporte a la toma de decisiones.

Una de las problemáticas principales, es que para un ser humano el análisis de grandes volúmenes de datos en muy poco tiempo es un proceso complicado de realizar; además, de que reduce la productividad y el retraso en los procesos de la sociedad de producción rural por la cantidad de tiempo requerida para llevar estos procesos, debido a que no se tiene definido un sistema DSS, el cual permita procesar, analizar y visualizar datos en tiempo causi-real y apoye a la sociedad de producción rural a tomar decisiones eficientes y eficaces. Por lo cual, surge la necesidad de un sistema web que permita realizar el procesamiento de los datos de comercialización eficiente, oportuna y eficazmente con la finalidad de disminuir los tiempos de procesamiento y análisis de la información. Para esto es necesario que este sistema web se comunique con el sistema de comercialización de ORNACOL, y que permita a través de diversas herramientas integradas y dinámicas de inteligencia de negocios obtener, procesar y analizar la información requerida por la sociedad de producción rural, sirviendo de apoyo al proceso de decisiones sobre la comercialización de sus productos y por consiguiente, incrementar la competitividad de esta sociedad.

1.3 Revisión de la literatura

Hasta donde se tiene conocimiento, existen varias investigaciones relacionadas a la toma de decisiones a través de la utilización de sistemas DSS, las cuales emplean distintas perspectivas: algunas describen el desarrollo e implementación de diversos sistemas BI para apoyar a la toma de decisiones; otras explican los beneficios de la minería de datos como soporte para la toma de decisiones empresariales y cómo ayudan a éstas a sobresalir y posicionarse en el mercado; otras explican la implementación de sistemas DSS en las empresas. Estas soluciones, no necesariamente han sido aplicadas al sector hortícola ornamental, pero consideran puntos claves en la importancia de la implementación de este tipo de sistemas en las empresas.

Un trabajo interesante es el que se mencionan Rupnik & Kukar (2007), el cual propone un enfoque para la integración de un sistema de soporte de decisiones utilizando herramientas de minería de datos, donde se opta por dividir roles entre el experto en la minería de datos y el usuario de negocios para facilitar el soporte de decisiones. Se introduce un sistema de soporte para toma de decisiones llamado DMDSS que apoya a las empresas en este proceso. Este sistema puede realizar modelos o formularios que el experto en minería de datos analiza y utiliza para la toma decisiones conforme a los resultados obtenidos. Además, utiliza la base de datos Oracle y fue desarrollado en la plataforma J2EE. Al analizar la propuesta de estos autores se concluye que algunas de las desventajas que presenta su enfoque es que requiere de un experto de minería de datos, no muestra gráficos para la visualización de datos y sólo permite el análisis de ciertas áreas de interés.

Villegas (2009) estudia la factibilidad en la implementación de un sistema denominado Datamart de apoyo a la toma de decisiones para la obtención de información relevante en el departamento de procesos comerciales de Easy S.A., proponiendo un sistema de apoyo a la toma de decisiones para el manejo de productos y tiendas en una cadena de productos al por menor a partir de datos almacenados en una base de datos para ésta, permitiendo gestionar grupos de tiendas y pronóstico de demanda de productos. En la propuesta de los autores se

describe la aplicación de algoritmos de minería de datos a un conjunto de datos almacenados en un gestor de base de datos, pero carece de un software para la visualización de datos y utiliza varias herramientas para el análisis que por su complejidad solo un experto puedo operar.

Por otro lado, Hovad et al. (2015) describen una solución para la mejora y el soporte de los procesos de toma de decisiones en un tiempo razonable. Proponen e implementan un procedimiento adecuado basado en los métodos y tecnologías apropiadas para crear una aplicación de minería de contenido web centrada en el análisis de tendencias. El sistema web se desarrolló en Python como lenguaje back-end, Pandas para el análisis y minería de datos, JavaScript para front-end, MySQL como base de datos y algunas herramientas útiles para el desarrollo de interfaces que permiten reflejar los cambios en tiempo real. Sin embargo, la solución descrita es entonces una herramienta que permite el acceso fácil a las personas y en la cual se visualiza los datos en tiempo real, esto le da un punto a favor en cuestiones de toma de decisiones; no obstante, es un sistema que aplica la minería web, que analiza el contenido de las páginas web y no de la base de datos, siendo esto una limitante en contraste con el sistema propuesto en esta investigación.

Bozkir & Sezer (2013) establecen la importancia de los sistemas de apoyo a la toma de decisiones (DSS) y de la desventaja que tienen los DSS que se construyen como aplicaciones de escritorio, ya que están diseñados para expertos en minería de datos. En el documento se describe la implementación de ASMINER, una herramienta basada en web que permite ahorrar tiempo en las fases de exploración e informe de los procesos de toma de decisiones basados en minería de datos utilizando una metodología de árboles de decisión, que es un enfoque de minería de datos para la clasificación y la predicción. ASMINER permite a los tomadores de decisiones y también a los trabajadores del conocimiento, explorar e informar con tres técnicas de extracción de datos (árboles de decisión, agrupación y extracción de reglas de asociación). Sin embargo, por lo observado en su descripción, este sistema no está completo en su totalidad y no está automatizado.

Una aplicación interesante y que utiliza varias técnicas de minería de datos es Ensmo, una plataforma web desarrollada por Kinsley (2016) con el objetivo de facilitar el análisis de datos y proporcionar al usuario interfaces amigables que le permitan utilizar herramientas de minería de datos, análisis de datos (gráficas, comparación, pronostico, predicción), estadística y manipulación de datos sin ser un experto en el tema. Ensmo permite cargar un conjunto de datos para generar gráficos, realizar operaciones de datos, hacer predicciones y pronósticos, todo con una simple interfaz de usuario. Es importante mencionar que el sistema está desarrollado en Python para back-end, Pandas para el análisis de datos y Materialize CSS y JavaScript para front-end. Aunque el sistema tiene similitud con el sistema propuesto en este trabajo de investigación, éste requiere de un cierto conocimiento por parte de los tomadores de decisiones, además no está dedicado en su totalidad a la comercialización y no visualiza los datos de manera asíncrona, los datos deben ser proporcionados mediante un archivo y no directamente de una base de datos que se actualiza continuamente.

Por otro lado, Zimmermann (2006) describe un sistema BI para escritorio que ayuda a tomar decisiones a partir de reportes utilizando un componente de Delphi y un cubo de datos para

generar la información extraída de la base de datos. Su propuesta permite la visualización de los datos mediante tablas, gráficas y reportes; sin embargo, las tecnologías utilizadas son para la actualidad obsoletas y requiere de conocimiento básico de SQL para realizar un cambio de base de datos, consultas, etc. de algoritmo para automatizar el proceso de consultas dinámicas.

Damasceno et al. (2018) describe la implementación de un sistema BI mediante un Data Mart, un Data Warehouse y un Cubo de Datos, utilizando la herramienta Pentaho. Éste permite visualizar la información de una base de datos dinámicamente mediante tablas y gráficas; sin embargo, se desarrolló como una aplicación para escritorio que no puede ser utilizada por dispositivos remotos y que, además, requiere de conocimiento de las herramientas, el lenguaje SQL y bases de datos para su configuración. Por otro lado, no muestra un resultado preciso sobre el potencial de la herramienta, los beneficios o su eficiencia.

Como se puede observar las soluciones existentes para el desarrollo de un sistema web de apoyo a la toma de decisiones, presentan las siguientes limitaciones:

- Desarrollados o implementados como aplicaciones de escritorio; por lo cual, para las demandas de la actualidad, se puede considerar como una ventaja y área de oportunidad, el desarrollo de una alternativa web para el acceso remoto al sistema y a la información analizada por éste.
- Implementados con tecnologías obsoletas, interfaces o dashboards no muy amigables para el usuario; por lo tanto, la utilización de tecnologías de vanguardia y de acceso a abierto posibilita el acceso ubicuo al sistema.
- Diseñados con métodos distintos para las consultas dinámicas hacia la base de datos remota; utilizar algoritmos de minería de datos que se enfocan particularmente en descubrir patrones ocultos que ayuden a generar estrategias de comercialización implicando ser manejados por un experto, sin contar con una alternativa para los usuarios no expertos en el tema.

Por lo tanto, en esta investigación se propone el desarrollo y aplicación de un sistema web que utiliza características de las herramientas de inteligencia de negocios para el soporte en la toma de decisiones sobre la comercialización de plantas ornamentales. Este permite resolver problemas semiestructurados, tomando los datos históricos de comercialización generados por el sistema de comercialización de ORNACOL descrito en la sección 1.2. El contexto del problema, con la finalidad de proporcionar sugerencias o soluciones complejas dirigidas a estos datos. Este sistema es desarrollado por fases utilizando la metodología de Proceso Unificado Ágil (PUA) e implementado como un sistema web basado en lenguajes de programación de uso libre como Python y Django para el procesamiento y análisis eficiente de datos. Utiliza el estilo arquitectural REST para la construcción de WEB APIs y la ejecución de peticiones asíncronas al sistema de comercialización previamente mencionado, permitiendo aprovechar la información generada por el sistema de comercialización y apoyar en la toma de decisiones para la generación de mejores estrategias de comercialización.

1.4 Planteamiento de la solución del problema

Considerando el contexto del problema, descrito en las secciones anteriores, en esta tesis se describe el desarrollo e implementación de un sistema basado en web que facilite el proceso de toma de decisiones en la comercialización de plantas ornamentales en la sociedad de producción rural ORNACOL, el cual sea capaz de procesar y analizar un conjunto de datos históricos extraídos de una BDR utilizando herramientas BI para visualizar la información mediante tablas y gráficas. Además de simplificar el uso a los usuarios con interfaces amigables. El sistema descrito, se fundamenta en la arquitectura de un sistema DSS, integrando herramientas de inteligencia de negocios; por lo tanto, en la Figura 1, se muestra el diseño conceptual para la solución del problema de la investigación.

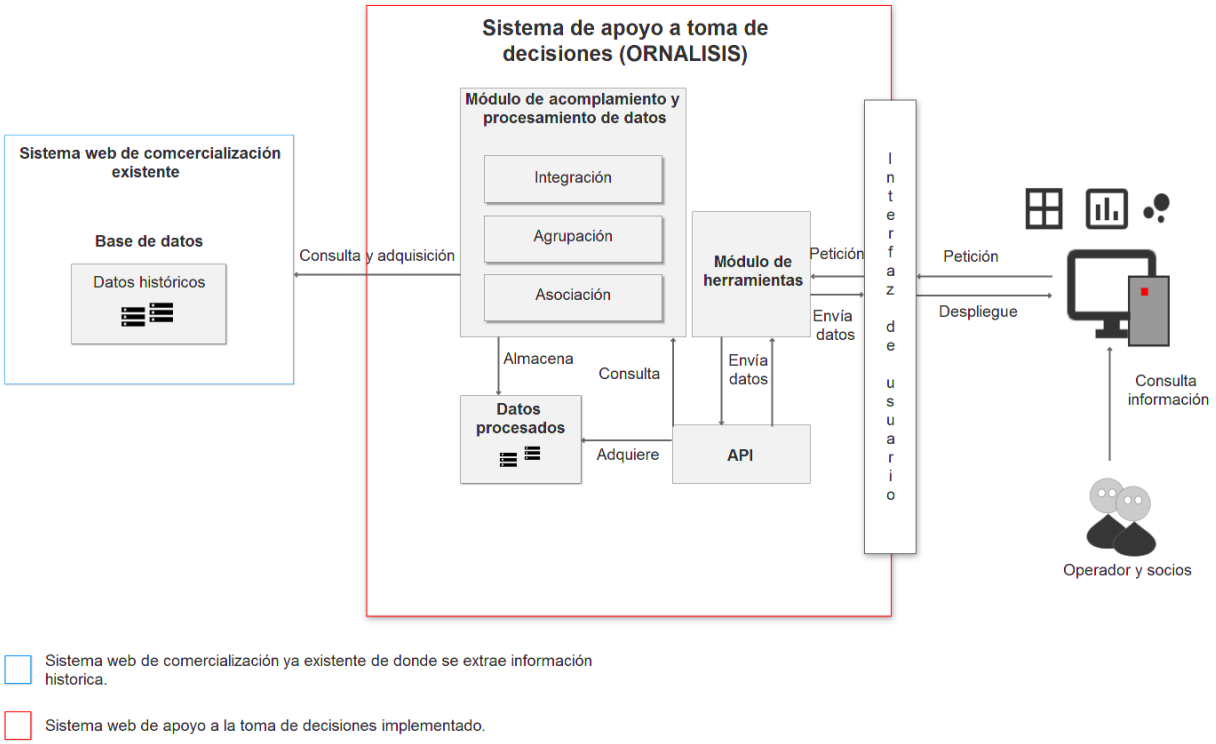

Figura 1. Modelo conceptual de la solución al problema. Fuente: autoría propia.

Como se puede observar, en la Figura 1 el sistema propuesto y denominado ORNALISIS está constituido por una interfaz de usuario, un módulo de acoplamiento y procesamiento de datos, una interfaz de aplicaciones (API) y un módulo de herramientas integradas y dinámicas.

A continuación, se describe cada uno de los elementos necesarios para comprender la propuesta de solución ilustrada en el modelo conceptual.

1.4.1 Desarrollo del Back-end

En el back-end se encuentra la lógica del sistema, utilizando el lenguaje Python, el framework Django y la herramienta para el análisis y modelado de datos, el sistema obtiene los datos almacenados en la BDR histórica, los limpia, procesa y envía al front-end para la visualización.

1.4.2 Desarrollo del Front-end

En el front-end se encuentra la parte visual del sistema, la consulta y adquisición de conocimiento de los datos previamente procesados mediante gráficas, tablas, comparativos y reportes periódicos.

La consulta y adquisición de los datos puede ser solicitada en cualquier página web debido a su diseño a manera de aplicación de programación de aplicaciones (API, por sus siglas en ingles) y el despliegue de los datos obtenidos se grafica a través de JavaScript para una presentación directa en el sistema web.

1.4.3 Arquitectura de comunicación

El sistema se fundamenta en una arquitectura cliente-servidor y, por lo tanto, ORNALISIS se comunica con el sistema de comercialización de ORNACOL siguiendo esta arquitectura, realizando los siguientes pasos para cada petición:

- Paso 1: el cliente realiza una petición Ajax al servidor la cual está compuesta de una consulta de datos específicos y parámetros para el despliegue a través del módulo de herramientas.
- Paso 2: a través de la API se comunica la interfaz de usuario y el servidor y realiza la petición al módulo de procesamiento y acoplamiento de datos (MPA).
- Paso 3: el servidor se encarga de buscar los datos solicitados en la base de datos que contiene datos históricos de la comercialización.
- Paso 4: el MPA se encarga de integrar, agrupar y asociar la información resultante de la consulta.
- Paso 5: la API regresa los datos serializados en formato JSON al módulo de herramientas.
- Paso 6: los datos son enviados al cliente para ser mostrados a través de gráficas o tablas.

1.5 Motivación

La realización de este trabajo de investigación surge de una necesidad de un sector productivo que aporta recursos económicos y laborales a la sociedad, pero que requiere de la inclusión de tecnología para fortalecer sus procesos, surgiendo con esto la motivación principal del desarrollo del sistema propuesto en esta investigación, ya que permitirá aprovechar el potencial de los sistemas de apoyo a la toma de decisiones en conjunto con las herramientas de inteligencia de negocios para el análisis de grandes conjuntos de datos y las ventajas que esta tiene en el sector productivo, mejorando los procesos de comercialización en el sector de

la horticultura ornamental e incrementando la competitividad de los productores de plantas ornamentales, gracias a la optimización de los procesos de análisis sobre la comercialización de éstas , con una toma de decisiones oportuna.

1.6 Justificación

Actualmente, la sociedad de producción rural ORNACOL, carece de un sistema capaz de llevar acabó el análisis y procesamiento de los datos, y que sirva de apoyo para tomar decisiones sobre la comercialización de plantas ornamentales.

En ORNACOL se toman decisiones con base en los datos históricos de comercialización almacenados en la base de datos del servidor de manera no automatizada, cada vez que se requiera un cálculo o gráfico, debe extraerse información y ésta toma tiempo en ser analizada; debido a esto, la información histórica no está siendo debidamente aprovechada para apoyar en la toma de decisiones y está limitando ORNACOL a sobresalir y competir con las demás empresas; además de llevar un proceso poco eficiente de análisis de datos.

1.7 Objetivos

A continuación, se presenta el objetivo general y objetivos específicos necesarios para el desarrollo de la investigación.

1.7.1 Objetivo general

- Implementar un sistema web para el procesamiento y tratamiento de la información de una base de datos histórica de ventas, con la finalidad de aportar una herramienta de apoyo a la toma de decisiones sobre la comercialización de plantas ornamentales.

1.7.2 Objetivos específicos

- Estudiar las técnicas de inteligencia de negocios aplicadas a sistemas de apoyo a toma de decisiones (DSS).
- Seleccionar los métodos de búsqueda apropiadas para la construcción de consultas dinámicas.
- Analizar y procesar los datos históricos de la comercialización de plantas ornamentales en la sociedad de producción rural ORNACOL.
- Desarrollar e implementar un sistema web con interfaces ergonómicas para la interpretación de la información histórica.
- Desarrollar herramientas integradas que cumplan con los requisitos funcionales de la sociedad de producción rural ORNACOL.
- Desarrollar herramientas dinámicas que permitan al operador del sistema construir sus propias herramientas para realizar análisis específicos de información.
- Registrar información histórica en el sistema web.

- Implantar el sistema web.
- Medir el impacto del sistema mediante la evaluación del tiempo de procesamiento antes y después de su implementación.
- Redactar la documentación del proyecto.

1.8 Hipótesis

La implementación y puesta en marcha de un sistema web de apoyo a la toma de decisiones en la sociedad de producción rural ORNACOL permitirá automatizar los tiempos de procesamiento y análisis de datos para la mejora en los procesos de toma de decisiones relacionados con la comercialización de plantas ornamentales.

- **Variables independientes:** La implementación y puesta en marcha de un sistema web.
- **Variables dependientes:** automatizar los tiempos de procesamiento y análisis de datos

1.9 Métodos y herramientas

A continuación, se explican con detalle, los métodos y herramientas utilizadas para llevar a cabo el desarrollo de esta investigación:

1.9.1 Participantes

ORNACOL es una Sociedad de Producción Rural dedicada a la producción y comercialización de plantas ornamentales en el municipio de Coquimatlán, Colima, la cual se tomó como caso de estudio de esta investigación porque uno de sus objetivos principales es apoyar a los productores colocando la producción en diferentes mercados, innovar los procesos para incrementar la productividad y calidad, y en consecuencia la creación de empleos directos e indirectos asegurando la rentabilidad, entre otras cosas.

Es una de las empresas que intenta utilizar el uso de la tecnología para mejorar sus procesos de producción y comercialización y que desea aumentar la exportación de sus productos y mejorar la calidad de éstos.

1.9.2 Enfoque de la investigación

La metodología empleada para llevar a cabo el desarrollo de esta investigación se fundamentó en una investigación mixta, utilizando el enfoque cualitativo para hacer un estudio sobre el impacto de la implementación en ORNACOL, y un cuantitativo para precisar la misma y dar respuesta exacta al problema. Para el desarrollo de la investigación se toma en cuenta un enfoque mixto, utilizando un diseño exploratorio secuencial (DEXPLOS) el cual implica una fase de análisis cualitativo para hacer un estudio subjetivo para generar una hipótesis que se aproxime a lo que se desea obtener seguida de otra de recolección y análisis de datos cuantitativos para precisar la misma y dar respuesta exacta al problema.

En este caso, es útil usar un diseño exploratorio secuencial de tres etapas:

1. Recabar datos cualitativos y analizarlos (obtener categorías y temas, así como segmentos específicos de contenido que los respalden e ilustren).
2. Utilizar los resultados para construir un instrumento cuantitativo (los temas o categorías emergentes pueden concebirse como las variables y los segmentos de contenido que ejemplifican las categorías pueden adaptarse como ítems y escalas, o generarse reactivos para cada categoría). De forma alternativa, se buscan instrumentos que puedan ser modificados para que concuerden con los temas y frases encontradas durante la etapa cualitativa.
3. Administrar el instrumento a una muestra probabilística de una población para validarlo.

Para esto, se requirió realizar una investigación documental y una investigación de campo, las cuales se describen a continuación:

- **Investigación de campo:** se visitó el centro de acopio y comercialización de plantas ornamentales de ORNACOL en donde se realiza la toma de decisiones para la comercialización de plantas ornamentales, permitiendo dar a conocer el proceso actual de toma de decisiones, los tiempos de análisis de la información, el modelo de negocio con la que actualmente se trabaja, la estructura de la base de datos y el personal que operaría el sistema y con el cual se mantendría comunicación para definir los requisitos funcionales y no funcionales necesarios para el desarrollo del sistema.
- **Investigación documental:** se eligió el tema a investigar, se realizó la revisión de la literatura sobre el tema de estudio donde se generaron fichas bibliográficas de apoyo para la redacción de la revisión de la literatura, se establecieron objetivos y se planteó la hipótesis de investigación que sirvió como base para el desarrollo de esta investigación.

1.9.3 Recolección de datos

Para el caso del desarrollo de esta investigación, existen datos almacenados en una BDR y datos en hojas de cálculo con información históricos de la comercialización de plantas ornamentales; además, se aplicó una entrevista semiestructurada como técnica para el levantamiento de los requisitos funcionales y no funcionales y conocer la necesidad que surge de la problemática.

1.9.4 Análisis de datos.

Para analizar los datos, acorde al método mixto definido en esta investigación, se utilizó la estadística descriptiva para cuantificar los resultados de la investigación junto con una evaluación del impacto del desarrollo tecnológico para analizar los resultados de la investigación de manera cualitativa.

1.9.5 Metodología del desarrollo de software

Para el desarrollo del sistema web, se hace uso de la metodología PUA, la cual establece cuatro fases y siete disciplinas en las cuales se lleva a cabo el desarrollo del sistema iterativo incremental y por la cual fue seleccionada para el desarrollo del sistema.

1.10 Beneficios esperados

Con la realización de la presente investigación se buscan beneficios para ORNACOL y para el sector productivo horticultura ornamental, ya que permite el análisis de información, proporciona interfaces amigables y proporciona al productor información valiosa para la toma de decisiones, estos beneficios son los siguientes:

- Mejora el aprovechamiento de la información histórica almacenada en la base de datos.
- Mejora el apoyo a la toma de decisiones para la comercialización, en este caso de plantas ornamentales.
- Reduce el tiempo de análisis de grandes cantidades de información con la utilización de herramientas de inteligencia de negocios, mediante herramientas integradas o prediseñadas, así como herramientas dinámicas que el mismo operador del sistema puede crear para atender necesidades específicas de análisis de información.
- Incrementa la competitividad de los viveristas asociados a ORNACOL, ya que permite a través del análisis de información oportuno, eficiente y eficaz dar soporte a la toma de decisiones sobre la comercialización de plantas ornamentales mediante una visión del pasado para dar oportunidad de generar estrategias en el futuro.

1.11 Organización de la tesis

El presente documento está dividido en seis capítulos, el primer capítulo de introducción y cinco más que describen los pasos realizados para resolver el problema a investigar; además, se incluye el apartado de referencias bibliográficas y los apéndices.

En el segundo capítulo se incluye el marco histórico, conceptual y teórico donde se describen los antecedentes, el contexto del problema y la integración de la teoría del problema de investigación.

En el tercer capítulo se describe detalladamente la metodología, técnicas y herramientas utilizadas en el desarrollo de la investigación; en el cuarto capítulo se describe el desarrollo del sistema siguiendo la metodología PUA.

Además, en el quinto capítulo se muestran los resultados obtenidos y finalmente en el sexto capítulo las conclusiones, trabajo futuro.

CAPÍTULO 2. ESTADO DEL CAMPO DEL CONOCIMIENTO

En este capítulo se aborda el marco histórico, contextual y teórico que ayudarán a fundamentar y comprender la investigación, siendo éstos una descripción a fondo del origen del problema de estudio, referenciando trabajos relacionados al objeto de estudio, así como una narración descriptiva de la teoría fundamental para la compresión de la investigación.

2.1 El marco histórico

Para comprender el problema de la investigación, es necesario conocer el origen de éste y cómo es que se han empleado nuevas tecnologías para dar solución al problema con el paso del tiempo.

Antes de la llegada de nuevas tecnologías, el proceso de la comercialización era un encuentro entre compradores y vendedores en un día y lugar determinado, toda la información era registrada en papel o en sistemas diseñados para almacenar pocos datos debido a la limitante de aquellos tiempos.

Alrededor de 1975, con el nacimiento de nuevas empresas como Microsoft que ofrecían **sistemas informáticos** permitieron incrementar la eficacia en la realización de las tareas, ahorrar tiempo en el desarrollo de las actividades y almacenar la mayor cantidad de información en el menor espacio posible, lo cual aumentó en las organizaciones el interés en los sistemas de información (O´BRIEN & MARAKAS, 1985), ya que éstos permitían simplificar las actividades de la empresa y reducir la burocracia, así como organizar, procesar y transformar la información; por lo cual, las empresas fueron migrando su información a estas herramientas.

Por otro lado, entre 1960 y 1990 surgen los **sistemas de apoyo a las decisiones** (DSS), los cuales son creados para resolver problemas semiestructurados y no estructurados que involucran al usuario, que según Trefil (2005) los DSS pueden ser clasificados por su tipo de problema:

- **Estructurado**: problemas de relaciones de sistemas conocidos que no son generalmente el foco de un DSS.
- **Semiestructurados**: el área más común de los DSS, para acuerdos generales sobre datos representativos del sistema o evaluación del sistema. Requieren conocimiento humano para las decisiones finales.
- **Mal estructurados**: donde la complejidad del problema es alta y la mayoría de las veces no se obtiene una respuesta.
- **No estructurados**: no hay consenso en la interpretación de la información, por lo general, involucra grupos de diferentes talentos y experiencia para evaluar las soluciones candidatas y las soluciones son continuamente monitoreadas y documentadas para uso futuro.

Cabe mencionar que no existe una taxonomía universal para los DSS; sin embargo, según la taxonomía de interacción, Haettenschwiler (1999) distingue entre:

- ***Activo***. Proporcionar sugerencias o soluciones estables a problemas complejos.
- ***Pasivo***. No está diseñado para determinar explícitamente una solución para quienes toman decisiones.
- ***Cooperativo***. es más complicado ya que requiere más interacción entre el DSS y los tomadores de decisiones, iterando y validando hasta generar una solución óptima.

En otra perspectiva, y acorde con la taxonomía de uso, Power (2011) establece cinco categorías para los DSS, las cuales son:

- ***Dirigidos por la comunicación***. Proporcionan información a los grupos que trabajan en tareas compartidas.
- ***Dirigidos por datos***. Enfatizan la recuperación de datos internos o adicionales en tiempo real (o históricos).
- ***Dirigidos por documentos***. Integran las tecnologías almacenadas y de procesamiento recopiladas para ayudar al tomador de decisiones con la recuperación de información.
- ***Dirigidos por conocimiento***. Obtienen recomendaciones específicas para los responsables de la toma de decisiones a partir de información dirigida por expertos.
- ***Dirigidos por modelos***. Proporcionan una visión de los modelos matemáticos sobre los fenómenos percibidos.

Es entonces cuando los sistemas como datawarehouse, de **inteligencia de negocios**, OLAP y potentes herramientas analísticas basadas en web empezaron a popularizarse, permitiendo a las empresas reducir el tiempo de análisis de la información, apoyar a la toma de decisiones y por ende incrementar su productividad; a la vez, el proceso de la comercialización se volvió más complejo. Por lo tanto, las empresas tenían que preocuparse no solo por el registro de los datos y por la automatización de los proceso de comercialización, sino de brindar un mejor servicio a los clientes, expandir su mercado y lo más importante, el análisis de los datos de comercialización, ya que con el paso del tiempo se almacenaba más información que implicaba más tiempo de análisis para la toma de decisiones a futuro y una mayor inversión para la empresa para adquirir un mejor equipo y mantener su información integra y segura, además de personal para realizar el análisis de grandes cantidades de información.

Con la llegada de la **Web 2.0** en el año 2004, definida como la web social, que facilitaba el compartir información y permitía a los usuarios interactuar entre sí, el término de comercialización empezó a evolucionar. La web 2.0 permitió crear medios por los cuales las empresas podían compartir su negocio en línea y por ende expandir su mercado. Poco a poco la competitividad entre las empresas por obtener un nivel de calidad y productividad se volvió más exigente, **sistemas web** compuestos por un cliente y un servidor que los usuarios podían utilizar accediendo a Internet a través de su navegador empezaron a popularizarse.

Posteriormente con el avance tecnológico, el comercio evolucionó y se convirtió en lo que hoy día conocemos como el comercio electrónico, que ya existía de desde los años 70s, pero se popularizó en los 80s. El problema fue que con mayor accesibilidad para los usuarios, el comercio electrónico de negocio a cliente aumentó y las empresas tenían que lidiar con el problema que se supone que ya se había resuelto con el avance tecnológico, el del análisis de grandes cantidades de información en tiempos cortos; lo cual, tornó más compleja esta tarea, ya que cada vez eran más usuarios registrados en el sistema, más transacciones realizadas cada segundo y más información almacenada que estaba siendo desaprovechada.

En el año 2006, surge la **Web 3.0** conocida como la web semántica, la web inteligente que innova la manera en que los datos son almacenados y procesados en aplicaciones web y que permite realizar consultas asíncronas. A partir de estos años y en adelante, numerosas empresas de comercio electrónico empiezan a surgir, la empresa le empieza a tomar confianza a la nueva forma de comercializar y una oportunidad de expandir sus mercados y que más personas de todo el mundo compren sus productos. Así, las herramientas analíticas se tornan una necesidad, requiriendo realizar análisis de información en un tiempo corto y obtener un resultados que ayuden a tomar decisiones sobre las inversiones a futuro, que predigan las ventas en un cierto año, que indique lo que no se vende, lo que se está vendiendo más, las ventas en un periodo de tiempo, la conformidad de los usuarios con el servicio, el año en el que se vendió más, entre otras; es decir, toda la información necesaria para que la empresa sea más productiva y pueda competir con las demás.

Por ende, los sistemas de información se han popularizado con el avance tecnológico a tal grado que ahora se requiere de sistemas analíticos potentes capaces de realizar un análisis de información en poco tiempo y que ayuden a la toma de decisiones a futuro. Además, la mayoría de las empresas que utilizan los DSS son empresas grandes las cuales se ven en la necesidad de utilizarlos debido a su tamaño y la cantidad de información que se genera diariamente; sin embargo, es importante que las microempresas, empresas pequeñas y medianas (MIPYMES) que estén interesadas en aumentar su productividad, mejorar la toma de decisiones e inclusive sus ventas, incluyan este tipo de tecnologías, ya que como lo menciona Casanova, (1994) los DSS tienen la habilidad de dar solución a problemas complejos, facilitan el análisis cuantitativo en periodos de tiempo, facilitan la comunicación entre usuarios, incrementan el control y ejecución general de los gastos, ahorran costos y ayudan a tomar decisiones objetivas.

2.2 El marco contextual

En esta sección, se muestra una breve introducción del problema de investigación, se contextualiza el lugar y ambiente donde se encuentra el problema de investigación y se presentan algunos de los trabajos relacionados mencionando métodos, técnicas y resultados.

2.2.1 Contexto mexicano

Según el censo económico publicado en el 2014 por INEGI (2014), en México el 95.4% de las empresas son microempresas, el 3.6% son empresas pequeñas y el 0.8% son empresas

medianas todas ellas denominadas MIPYMES que aportan el 35.9% de la producción bruta en el país, representando una parte importante en la economía del país, pero se estima que son más ineficientes que empresas grandes debido a la falta de control de la información.

Los DSS son sistemas que ayudan a tomar decisiones y si bien empresas grandes lo utilizan, es importante que las MIPYMES también lo implementen. No obstante, las MIPYMES no participan en el uso de tecnologías de información (TI); lo cual, las estanca en innovación de sus productos, adquisición de clientes y en la optimización de procesos, perdiendo competitividad frente a las demás empresas.

Uno de los principales inconvenientes por los cuales las empresas no participan en el uso de TI es la resistencia al cambio; es decir, las empresas simplemente no quieren actualizarse y puede deberse según Valenzuela Rodríguez (2003) a diferentes aspectos:

- El temor al uso de nuevas tecnologías en algunos trabajadores.
- Errores en el uso de la nueva tecnología.
- El cambio de cultura y comportamiento.
- La escasa participación de los usuarios finales en el levantamiento de requerimientos, diseño y desarrollo de aplicaciones.
- La mala definición de los requerimientos para el desarrollo de la aplicación.
- Hardware y software insuficiente.
- El costo de operar con nuevas tecnologías.

2.2.2 Contexto estatal

En la ciudad de Colima solo empresas de gobierno o empresas importantes dedicadas al comercio electrónico hacen uso de DSS para mantener un control sobre la información y poder dar soporte a iniciativas estratégicas de manera oportuna. La mayoría de las empresas se enfocan en brindar un buen servicio y de sumar clientes a su lista, dejando atrás el control y análisis de los datos, tareas de suma importancia que pueden ser realizadas a través de sistemas informáticos de bajo costo especializados en el análisis.

Actualmente, en la sociedad de producción rural ORNACOL para la cual está destinada esta investigación, se presenta un tipo de problema muy peculiar; actualmente el proceso de comercialización en la sociedad de producción rural ORNACOL se lleva a cabo a través de un sistema web de comercialización donde los datos resultantes de las transacciones realizadas en la comercialización son almacenados en una base de datos histórica. Cada cierto periodo se generan reportes de diferentes especificaciones en hojas de cálculo, lo cual implica extracción, limpieza de los datos y traspaso de la información a hojas de cálculo que le permita organizar la información, generar tablas y gráficas para presentarlas en una próxima reunión, a todos los empleados y socios de ésta. Por lo cual, a la administradora y analista de esta sociedad, le toma un tiempo significativo generar un solo reporte, ya que es mucha información y además se encuentra en lugares diferentes de manera desordenada. Al no implementar un

DSS la sociedad de producción rural ORNACOL no es capaz de tomar decisiones oportunas, ya que no se tiene automatizado el proceso de procesamiento y análisis de los datos, perdiendo con esto competitividad ante las demás empresas de la región.

2.2.3 Trabajos relacionados

Existen DSS que se adaptan a las necesidades y áreas de las empresas; estos sistemas, si se desarrollan por empresas especialistas de consultoría y desarrollo de servicios para proyectos DSS, pueden ser costosos y difíciles de implementar ya que algunos incluyen herramientas de inteligencia de negocios que no son tan accesibles para todas las empresas, siendo este una de las principales inconvenientes en la mayoría de los sistemas de este tipo para las MIPYMES que quieran empezar a invertir en sistemas de apoyo a la toma de decisiones o que requieran automatizar sus procesos.

Actualmente, existen numerosas empresas dedicadas al desarrollo empresarial de DSS, e inclusive personas especialistas en el tema; además, cientos de programas desarrollados por empresas dedicadas al análisis de datos que proveen sistemas para el apoyo a la toma de decisiones a nivel organizacional implementado sistemas analíticos. A continuación, se presentan algunos trabajos que hablan del desarrollo de DSS para resolver diferentes problemas organizacionales.

En el sistema de escritorio independiente que da soporte a la toma de decisiones de ERGO (2000), diseñado para simplificar cualquier tipo de decisión compleja, se utiliza el motor ebestmatch, el cual se encuentra en el sistema de evaluación TEC advisor, que divide las decisiones complejas en pasos simples que ayudan a documentar cada paso del proceso de decisión. Algunas de las herramientas que provee este sistema van desde comparar miles de soluciones potenciales en un solo modelo de decisión, establecer las que mejor se acomoden al usuario, evaluar fortalezas y debilidades con simulación, análisis de sensibilidad, clasificación y más; además, generar informes utilizando gráficos y cuadros interactivos para la generación de análisis precisos y así mejorar la toma de decisiones.

En la propuesta de Tarifa et al. (2013) se desarrolló una serie de subsistemas de apoyo a la toma de decisiones para los procesos productivos vinculados a la industria química, de alimentos y a cuencas hidrográficas pertenecientes a varias áreas disciplinares. El proyecto se coordinó con el trabajo de tres grupos de investigadores, en los que se encuentran especialistas en ingeniería de procesos y en tecnología de alimentos, especialistas en gestión de cuencas hidrográficas y especialistas en informática. En su propuesta lograron implementar cada uno de estos subsistemas y se mejoró la toma de decisiones a nivel producción, gubernamental y privada, generando presentaciones, cursos, publicaciones en congresos tanto nacionales como internacionales.

El sistema desarrollado en Santillán & Mendoza (2007), es un sistema web de apoyo a ejecutivos para la toma eficiente de decisiones de compras. Para esto, utilizaron muestras no

probabilísticas para comprobar la peor y mejor oferta del mercado por parte de los proveedores. Para el desarrollo del sistema mencionan que se desarrolló mediante el ciclo de vida clásico de desarrollo de software que consta de siete fases y para llevar a cabo la codificación utilizaron el lenguaje PHP con algunas librerías para al análisis de datos. Como resultado obtuvieron un sistema web que permite a los ejecutivos tomar decisiones mediante la generación de gráficas y reportes utilizando técnicas para tomar la información de la base de datos a crear.

Por otro lado, Dobaev et al. (2016) han desarrollado la estructura de un DSS que está conformado por tres niveles: adquisición de datos, generación y análisis de datos. El sistema permite analizar los datos y comparar los resultados de análisis obtenidos con diferentes métodos, así como generar recomendaciones para un operador de sistemas de medición de potencia. Los datos son recolectados de bases de datos relacionales almacenadas en AIMSCEM que provee acceso libre a las bases de datos, utilizando SQL para obtener los resultados; además, los datos se obtienen por medio de APIs del mismo sistema y de dispositivos de medición. Como resultados se obtuvo una estructura eficiente que se esperan que reduzca en gran medida el tiempo requerido para identificar y eliminar las pérdidas de energía.

2.3 El marco teórico

En esta sección se introduce el soporte conceptual de la teoría utilizada para el planteamiento del problema, se especifica qué, para qué y cómo se emplea en el sistema.

2.3.1 Sistema de apoyo a la toma de decisiones

Los DSS ayudan a mejorar la toma de decisiones de cada una de las partes que intervienen en una iniciativa de comercio electrónico (Ruiz G. et al., 2009) . El objetivo principal de los DSS es ayudar a la toma de decisiones a través de una integración de conocimiento experto y modelos matemáticos (Trefil, 2005). Éstos surgen para la resolución de problemas semiestructurados y no estructurados, no cuentan con una taxonomía o diseño universal y es por eso por lo que varios autores los definen por su interacción con el usuario y por su usabilidad.

2.3.1.1. Características principales

Las características principales de los DSS son proporcionar acceso rápido y fácil a la información, realizar cálculos rápidos, facilidad de utilización, flexibilidad, capacidad de representación gráfica de los datos, entre otras. Estos son sistemas interactivos utilizados por los gerentes y los cuales pueden generar informes complejos, procesar gran cantidad de datos, ofrecer posibilidad de consultar datos, integrar modelos y bases de datos. Por lo tanto, se deben de considerarse estas características cuando se quiere construir un sistema de soporte de decisiones

2.3.1.2. Desarrollo de un DSS

El desarrollo o arquitectura de un DSS suele ser iterativo para respaldar las decisiones hasta que se confié en una solución precisa. Según Marakas (1999) la arquitectura lógica de un DSS consta de cuatro componentes fundamentales: una base de conocimientos o base de datos, un modelo matemático, interfaz de usuario y los propios usuarios, como se muestra en la Figura 2.

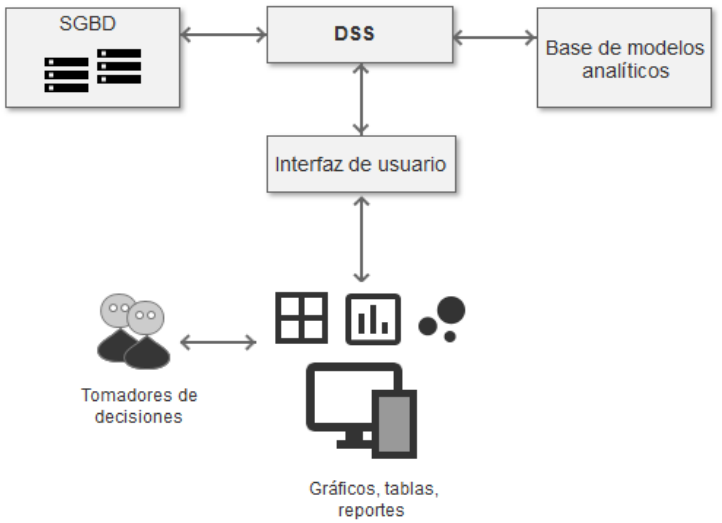

Figura 2. Arquitectura lógica de un sistema de apoyo a la toma de decisiones. Fuente: autoría propia.

Los DSS modernos utilizan una combinación de metodologías clásicas y metodologías de aprendizaje de máquina. Las **metodologías** para el desarrollo varían según su paradigma, modelo, objetivo o problema a resolver, se categorizan dirigidas al conocimiento, comunicación, datos y modelo, algunos ejemplos de metodologías para el desarrollo de un DSS más comunes son las definidas por Marakas (1999):

- **Metodología por fases**: sigue el ciclo de vida del software, definición del problema, análisis, diseño, implementación y pruebas.
- **Análisis ROMC**: esta metodología es descriptiva, es una metodología basada en procesos y se basa en cuatro entidades: Representaciones (R), operaciones (O), ayudas de memoria (M) y control (C). Con esta metodología, los analistas pueden definir diferentes representaciones para los usuarios como, gráficas, tablas, listas, menús, etc.
- **Desarrollo evolutivo**: los pasos del ciclo de vida del software se combinan en un solo paso que se repite iterativamente. Es apropiado para el desarrollo de un DSS debido a que el sistema es evaluado constantemente.
- **Prototipado**: se desarrolla en un corto periodo de tiempo, permitiendo interacción rápida con los componentes del sistema que pueden ser adaptados a las necesidades del usuario. Reduce el riesgo del desarrollo y el costo.
- **Desarrollo del usuario final**: Los usuarios finales pueden desarrollar su propio sistema de apoyo a decisiones usando las herramientas, a esto se le llama

"Selfsourcing" determinación mejorada de los requisitos, mayor participación en la toma de decisiones y mayor velocidad de desarrollo del DSS.

Para el desarrollo de este sistema se da solución a un tipo de problema semiestructurado dirigido a datos en la que el usuario toma una decisión final. Se utilizó la *metodología por fases* desarrollando una arquitectura cliente servidor siguiendo la estructura lógica mostrada en la Figura 2, que dio como resultado una arquitectura física como se muestra en la Figura 3.

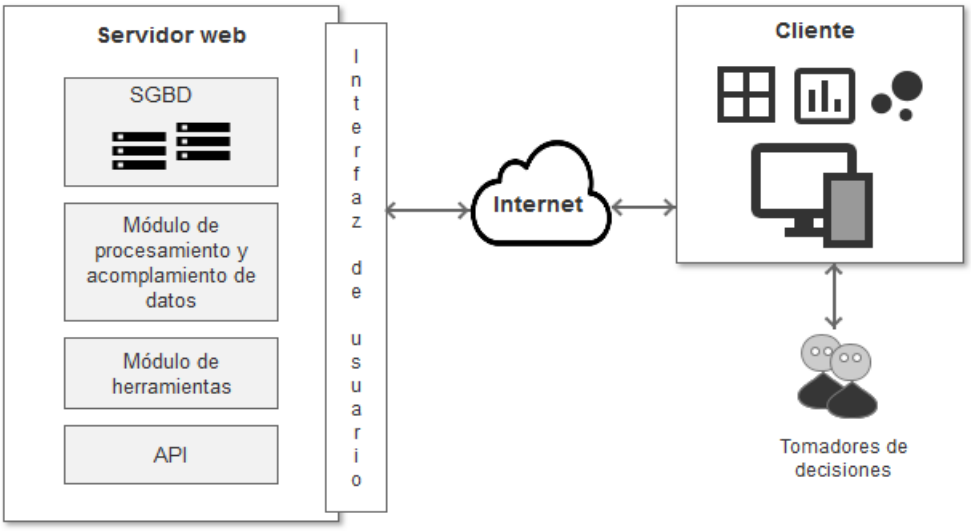

Figura 3. Arquitectura cliente-servidor de ORNALISIS. Fuente: autoría propia.

2.3.2 Lenguaje Python

Python es un lenguaje orientado a objetos licenciado, por la fundación de software de Python lo que lo hace libre de uso y distribución (Python Software Foundation, 2020). Este lenguaje permite desarrollar aplicaciones eficientes y robustas y actualmente existe una comunidad muy amplia y módulos desarrollados por terceros que pueden ser utilizados en el desarrollo de sistemas.

El uso de este lenguaje en la codificación del sistema es esencial ya que se busca un software de código libre, que permita integrar herramientas potentes para el análisis de datos, que sea versátil, escalable y multiplataforma.

2.3.3 Django Framework

Django es un framework web Python de alto nivel que fomenta el desarrollo rápido y el diseño limpio y pragmático. Creado por desarrolladores experimentados, para desarrollar una aplicación web robusta sin necesidad ponerle mucho esfuerzo; además, es gratis y de código abierto (Ramesh et al., 2018).

Django utiliza la Modelo Vista Template (MVT) siendo el *modelo* el que hace referencia al acceso de la capa de datos, el *template* es la capa de presentación, la que decide como se muestran los datos en una plantilla HTML y la *vista* se refiere a la lógica que accede al modelo y envía los datos a la plantilla.

Para facilitar el acceso a los datos y la serialización de estos para presentarlos en los templates es necesario utilizar un framework llamado Django REST framework, un paquete de herramientas poderoso y flexible que permite construir APIs web. Una API es una interfaz de programación de aplicaciones que permite ahorrar tiempo a los desarrolladores construyendo un conjunto de rutinas que acceden a los datos de un determinado software (Freeman, 2019) sumando a eso el uso de AJAX para llamadas asíncronas al sistema.

REST es una arquitectura orientada a los servicios (SOA), una arquitectura de software flexible que puede transportar cualquier tipo de dato (XML, JSON, imágenes, documentos, texto, etc.) por el protocolo HTTP, permite utilizar los métodos de HTTP (GET, POST, PUT, DELETE, PATCH) y utiliza los códigos de respuesta (404, 200, 204, 409) (Blancarte, 2017). El método natural de envió y respuesta de REST es JSON, que tiene un mayor desempeño por ser menos pesados y más rápidos en su procesamiento; en contra parte, SOAP es una arquitectura también SOA, que solo permite transportar datos por XML, es más robusta que REST y permite agregar metadatos y más cosas que JSON no permite; por lo cual, lo convierte en un protocolo más pesado en tamaño y en procesamiento y no es lo que el sistema web propuesto en esta investigación para la toma de decisiones necesita.

La utilización de estas herramientas en el sistema web descrito en esta investigación son de gran importancia, principalmente por su eficiencia, escalabilidad, facilidad acceso a los datos, por su desempeño y; además, por su licencia de código libre. El desarrollo de herramientas dinámicas planteadas en este trabajo de investigación se ven más beneficiadas por el uso de estas tecnologías, ya que requieren obtener datos precisos en el menor tiempo posible, además, de una alta seguridad en el manejo de la información y permisos para la obtención de estos.

2.3.4 Materialize CSS

Materialize CSS es un framework moderno de estilos de código libre para el desarrollo front-end basado en Material Design de Google. Este framewok agiliza y facilita el desarrollo de aplicaciones web y se enfoca en mejorar la experiencia de usuario al utilizar la plataforma.

Para el sistema web a desarrollar es importante mantener un diseño limpio y eficiente en el que el usuario se sienta cómodo trabajando y no le cueste trabajo acostumbrarse al diseño, además de agilizar el desarrollo del sistema y ahorrar tiempo y dinero, así como dar una visión formal al sistema web.

2.3.5 Gestor de base de datos MySQL

MySQL es un sistema de gestión de base de datos (SGBD) relacional que está desarrollado bajo licencia publica y comercial, considerada como una de las herramientas de código libre más utilizadas en el desarrollo y puesta en producción de sistemas web, además, funciona con múltiples plataformas, aunque se encuentra optimizado para GNU/Linux y la mayoría de los hostings web lo utilizan como SGBD principal.

La elección de este SGDB en el sistema web es de gran importancia, principalmente por su rendimiento y escalabilidad, ser de código libre e incluida aplicaciones web, ser el más utilizado en entornos web y ser de fácil uso y configuración.

2.3.6 Algoritmo DFS

El algoritmo de Búsqueda en Profundidad (DFS, por sus siglas en inglés) es de utilidad para determinar la conectividad informática, recorre cada ruta de cada nodo dentro de un conjunto de nodos unidos por aristas (grafo o árbol) hasta más no poder. Si esto se traslada a una base de datos (BD), el algoritmo DFS permite determinar las relaciones entre las tablas de cualquier BD relacional, a partir de la creación de un árbol o grafo equivalente al esquema de la BD y con esto, construir las consultas SQL de forma dinámica sin tener que utilizar varios lenguajes de datos (Chandel & Sood, 2014).

Este algoritmo fue elegido debido a que se necesitaba un método de búsqueda que permita determinar la existencia de relaciones entre tablas de una base de datos, para poder realizar consultas de manera dinámica a la BD histórica de comercialización de plantas ornamentales de ORNACOL y permitir al usuario crear sus propias herramientas, solicitando información específica a la base de datos y sin tener que ser un experto en el lenguaje SQL para la construcción de éstas.

CAPÍTULO 3. MÉTODOS EMPLEADOS

En el Capítulo 1 Introducción se describe la metodología de investigación que sustenta el desarrollo tecnológico descrito en esta tesis con la finalidad de dar soporte a la metodología de desarrollo de software para el desarrollo del sistema. Por lo tanto, en este capítulo se detalla la metodología empleada; la cual, se fundamenta en una metodología para el desarrollo de sistemas DSS, así como en una metodología ágil de desarrollo de software inmersa en esta misma, como se describe a continuación.

3.1 Metodología de desarrollo para el DSS

La arquitectura del sistema propuesto sigue la arquitectura de un DSS y, debido a la utilización de una metodología por fases para el desarrollo del software, se diseñó de la siguiente manera:

- **Solución a problemas semiestructurado:** requieren del conocimiento humano para lograr las decisiones finales.
- **Especificación de la taxonomía de interacción activa**: proporciona sugerencias o soluciones estables a problemas complejos.
- **Especificación de la taxonomía de uso dirigida a datos:** enfatiza la recuperación de datos de una base de datos histórica.
- **Metodología:** en este caso se decidió utilizar una metodología por fases llamada Proceso Unificado Ágil para el desarrollo del software, la cual se describe a continuación.

3.2 Metodología de desarrollo de software

La investigación se sustentó en la metodología de Proceso Unificado Ágil, que establece 4 fases y siete disciplinas (ver Figura 4) en las cuales se lleva a cabo el desarrollo del sistema iterativo incremental. A continuación Jacobson et al. (1999) describen detalladamente cada fase de esta metodología:

- **Incepción:** la fase de incepción o iniciación se enfoca en la iniciación del proyecto, la parte más importante para el desarrollo del software, ya que en esta etapa se describe y define con detalle el modelo de negocio y se da una solución viable para el problema descrito, además del levantamiento de los requisitos funcionales y no funcionales para el desarrollo del software.
- **Elaboración:** en esta fase y parte de la primera se analiza y diseña el prototipo, y se lleva a cabo la disciplina de implementación, todo lo anterior establecido en la incepción es transformado en código. También se realizan algunas pruebas unitarias.
- **Construcción:** esta fase se enfoca de lleno en el desarrollo del software, implementando algoritmos y realizando pruebas para asegurar y validar los bloques de código; además, se gestiona la configuración para asegurar su funcionamiento durante la vida útil del sistema.

- **Transición:** en esta fase se realizan pruebas de validación, aceptación, alfa y beta y se les da seguimiento a las disciplinas de despliegue, configuración, gestión. Esta es la fase final del desarrollo del software.

Dicha metodología se apegó a la planificación de las actividades en un cronograma de actividades para cumplir con el periodo establecido para el desarrollo del proyecto (ver capítulo 4 fase de iniciación). En esta capítulo se muestra lo realizado en cada disciplina de manera general.

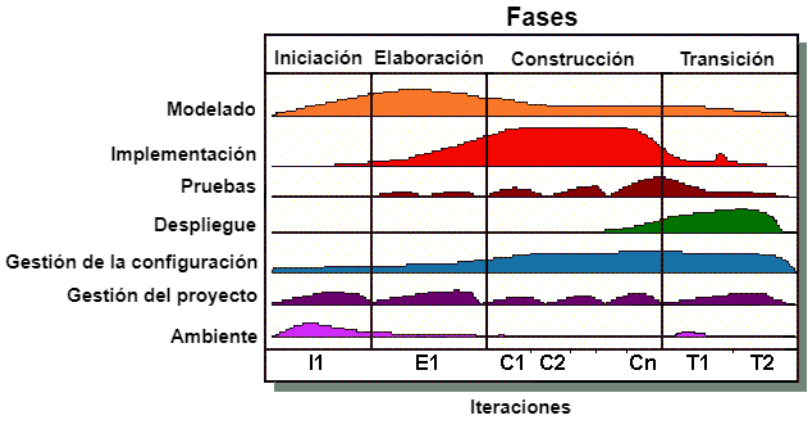

Figura 4. Metodología Proceso unificado Ágil (PUA). Fuente: (Ambler, 2014)

- **Modelado:** Se identificó la solución viable para abordar el dominio del problema utilizando herramientas de prototipado a partir de los requisitos funcionales y no funcionales determinados y dio como resultado un modelo conceptual (véase Figura 1) en el que se describen dos módulos dentro del sistema ORNALISIS: el módulo de acoplamiento y procesamiento de datos, que se encarga de acoplar los datos solicitados la API y procesarlos en el formato deseado para ser consultados por la API en el back-end y el módulo de herramientas, que se encarga de enviar y recibir la información para ser representada en el front-end.
- **Implementación:** se transformó el modelo conceptual construido en el modelado del sistema a código. Para el desarrollo del sistema web se utilizó el lenguaje Python junto con el framework Django y un kit de herramientas potente y flexible llamado Django REST framework para la construcción de la API que permite realizar consultas asíncronas, serializar los datos tanto en el back-end para enviarlos al front-end y establecer políticas de autenticación para el acceso seguro a los datos. Además de realizar una serie de pruebas unitarias para comprobar el correcto funcionamiento de las unidades de código escritas.
- **Pruebas:** se verificó que los requisitos funcionales y no funcionales cumplieran con lo establecido y se realizaron pruebas para encontrar defectos y garantizar la calidad del sistema siguiendo el diagrama del ciclo de vida de las pruebas orientadas a objetos

(FLOOT) adaptado a las pruebas realizadas durante el ciclo de vida del sistema desarrollado para este trabajo de investigación que se muestra en la Figura 5.

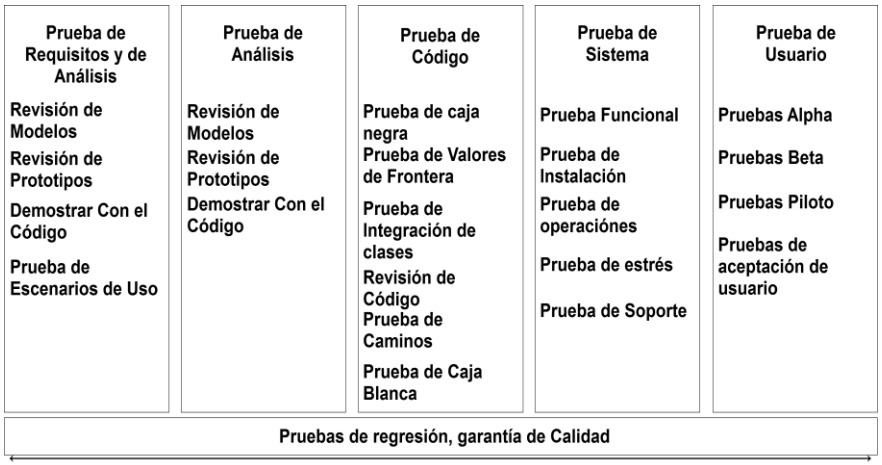

Figura 5. Diagrama FLOOT. Fuente: adaptado de (Ambler, 2014)

- **Despliegue:** Se planificó la fecha de entrega y se estableció un plan para que el sistema estuviera disponible para los usuarios. El sistema web está alojado en un servidor de desarrollo de manera temporal que posteriormente será alojado en la nube y podrá servir como herramienta de análisis y toma de decisiones para ORNACOL.
- **Gestión de la configuración y gestión del proyecto:** se gestionó la configuración (CM) y la gestión del proyecto desde la iniciación del proyecto, se desarrollaron planes de acceso a la información y estrategias de gestión de cambios a lo largo del desarrollo del sistema, asegurando la calidad y eficiencia del desarrollo durante su ciclo de vida.
- **Ambiente:** se instaló la estación de trabajo, se realizaron las plantillas guía para el diseño y se configuro todo lo necesario para la codificación del sistema durante la iniciación y elaboración del proyecto.

La metodología se explica de manera detallada en el capítulo 4 del desarrollo de la tesis.

CAPÍTULO 4. DESARROLLO DE LA TESIS

Este capítulo comprende la parte del desarrollo descrito en esta tesis. Se analiza y se explica detalladamente la planificación del desarrollo tecnológico que sustenta los resultados de esta investigación.

En la Tabla 1 se muestra el resumen de las actividades realizadas en cada una de las fases y dicsiplinas durante el desarrollo del sistema.

Tabla 1. Descripción de actividades realizadas en cada diciplina de la metodología PUA. Fuente: autoría propia.

Disciplina	**Fase**			
	Iniciación	**Elaboración**	**Construcción**	**Transición**
Modelado	- Análisis de proceso de negocio y alcance de proyecto. - Levantamiento de requisitos funcionales y no funcionales. - Descripción de casos de uso. - Determinación de riesgos.	- Modificación de prototipos para el panel de herramientas integradas, dinámicas y detalle de herramienta.	- Creación de diagramas de casos de uso. - Modelo de requisitos. - Diagramas de secuencia. - Diagramas de despliegue. - Diagrama de clases.	Se elaboró el manual de usuario.
Implementación	- Creación de primeros prototipos.	- Creación de arquitectura cliente-servidor o modelo conceptual del funcionamiento del sistema.	- Desarrollo del módulo de herramientas integradas y dinámicas.	Implementaciones finales al código.
Pruebas	- Pruebas de requisitos.	- Modificación de pruebas. - Pruebas de análisis. - Reporte de defectos.	- Pruebas funcionales, caja negra, caja blanca, valores de frontera, integración de clases, código, caminos.	- Pruebas de funcionalidad y operación, instalación y soporte, estrés, beta, piloto y aceptación de usuario.
Despliegue	Se definieron las actividades para determinar las fechas de inicio y entrega del sistema.	- Modificación de plan de despliegue. - Pruebas en varias plataformas para despliegue en la nube.	- Pruebas de despliegue del sistema en el servidor. -Documentación del sistema.	-Terminación de documentación del sistema. - Actividades para despliegue del sistema final en el servidor.

Gestión de la configuración	- Instalación de entorno de trabajo y herramientas necesarias.	Se verificó que todo fuera acorde a lo planeado en la fase de iniciación. - Nuevas herramientas para el desarrollo del sistema.	Se verificó que todo fuera acorde a lo planeado en la fase de elaboración.	Se verificó que todo fuera acorde a lo planeado en la fase de construcción.
Gestión del proyecto	- Factibilidad financiera, tecnológica, operacional y política. - Calendario de actividades.	- Verificación de diagrama de actividades. - Diseño de plan de mitigación de riesgos.	- Mitigación de riesgos mediante el plan de mitigación de riesgos.	Actividades de adiestramiento para los operadores del sistema.
Medio ambiente	- Configuración de un solo equipo de trabajo. - configuración de estación de trabajo.	- Instalación de herramientas necesarias en el entorno de trabajo.	- Búsqueda de herramientas y algoritmos para el sistema. -Configuración del entorno para el despliegue del sistema.	Finalización del desarrollo del proyecto y reajustes del entorno de trabajo.

A continuación, se describe en detalle el desarrollo del sistema propuesto enmarcándolo en la metodología de desarrollo especificada para esto.

4.1 Fase de iniciación

La fase de iniciación comprende el análisis del proceso de negocio para determinar cuál es la problemática que la empresa ORNACOL necesita resolver y establecer una solución que resuelva el problema y mejore el proceso. A continuación, se detalla el trabajo realizado por diciplinas en la fase de iniciación.

4.1.1 Modelo

El objetivo de la disciplina de modelo en la fase de iniciación es entender el negocio de la organización, el problema que se quiere resolver e identificar la solución viable para resolver el problema. Por lo tanto, en esta disciplina se estableció el alcance y se tomó nota de los casos de uso y la descripción de los requisitos funcionales y no funcionales.

4.1.1.1 Alcance del proyecto

En la sociedad de producción rural ORNACOL se tiene un sistema de comercialización que registra clientes, productos, ventas, entre otras cuestiones en una base de datos relacional, de la cual se requiere un sistema que apoye a la toma de decisiones, que sea capaz de establecer conexión a la base de datos remota, extraer información, procesarla y desplegarla

al usuario. Un sistema en el que reduzca el procesamiento y análisis de la información, ya que actualmente todo se realiza manualmente en hojas de cálculo.

4.1.1.2 Proceso de negocio

Para identificar el proceso de negocio que se lleva actualmente en la sociedad de producción rural ORNACOL, se diseñó un diagrama de flujo de datos (Véase Figura 6).

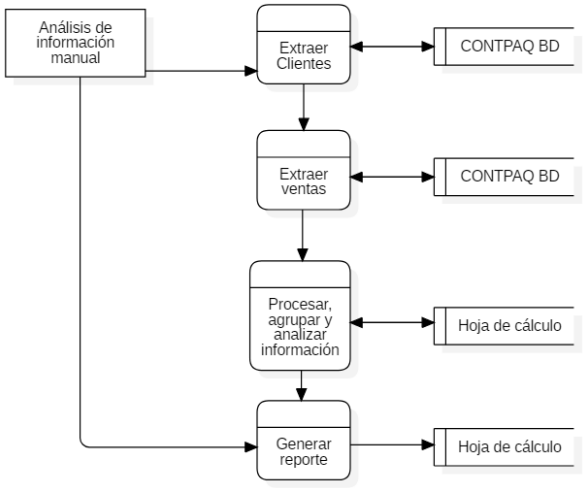

Figura 6. Proceso de negocio anterior de análisis de información en ORNACOL. Fuente: autoría propia.

Como se observa en la Figura 6, el proceso de la sociedad de producción rural ORNACOL es totalmente manual, ya que utilizan un programa de contabilidad para registrar las órdenes y de él extraen la información de los clientes, ventas, etc., para generar un reporte semanal, mensual o anual en hojas de cálculo. El proceso es tedioso, ya que requiere extraer información de distintas fuentes de información, aplicar fórmulas para realizar cálculos y crear gráficas a partir de la información recolectada. Como se describe, el proceso de adquisición y procesamiento de datos no es el óptimo, en cuanto a toma de decisiones oportunas se refiere. Por lo cual, se requiere optimizar el procesamiento de información, considerando que, con la implementación del sistema de comercialización, ya no se trabaja desde un programa de contabilidad si no desde el sistema de comercialización que almacena toda la información en la nube, como se muestra en la Figura 7.

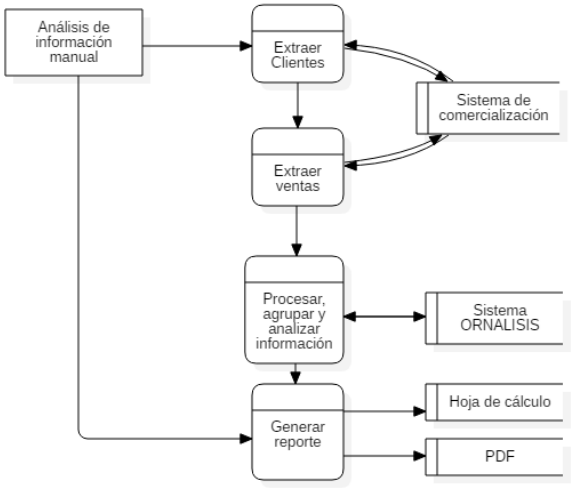

Figura 7. Proceso de negocio propuesto de análisis de información en ORNACOL. Fuente: autoría propia.

Como se puede observar en la Figura 7, ahora los datos se extraen del sistema de comercialización y el procesamiento, agrupación y análisis de información se realiza en el sistema web DSS (ORNALISIS) desarrollado para esta investigación. Así, por último, los reportes son generados a partir de la información resultante del sistema ORNALISIS en hojas de cálculo o PDF.

4.1.1.3 Levantamiento de requisitos

Se realizó el levantamiento de los requisitos funcionales y no funcionales para el desarrollo del sistema, los cuales fueron propuestos por el usuario que operará el sistema y estos mismo cumplen con el proceso de negocio que llevan cotidianamente en ORNACOL, añadiendo las funcionalidades que requieren (véase Figura 8).

Generar reportes semanales, menuales y anuales en PDF y Excel
Admitir distintos tipos de usuarios (Operador, Viverista, Invitado)
Permitir filtrar información por ventas totales, ventas por producto, productos menos vendidos y mas vendidos, ventas por socio, ventas por especie, ventas por periodo, ventas por estado, ventas pendientes, surtidas y canceladas.
Representar informacion en tablas y gráficas.
Extraer informacion de la base de datos de comercialización y manipularla.
Manipular la informacion de forma dinámica.
El sistema debe tener interfaces amigables.
Las interfaces del sistema deben ser responsivas.
El sistema debe ser desarrollado a manera de API.
El sistema debe ser compatible con todo tipo de navegador.
El sistema debe responder en tiempo causireal a las peticiónes

Figura 8. Levantamiento de los requisitos funcionales y no funcionales. Fuente: autoría propia.

4.1.1.4 Descripción de casos de uso

A continuación, se definen formalmente los siguientes casos de uso, los cuales describen cada una de las actividades que el usuario y sistema realizan para cumplir con los requisitos funcionales y no funcionales.

Tabla 2. Descripción formal del caso de uso CU-1. Fuente: autoría propia.

Nombre	**Administrar usuarios / CU-1**	
Actores	Operador.	
Descripción	El operador administra (registra, edita, elimina) la información de los usuarios.	
Precondición	El usuario que administra es operador.	
Flujo principal	Acciones del usuario	Acciones del sistema
	El operador se redirige a el apartado de administración de trabajadores.	El sistema muestra el panel de administración de usuarios.
	El operador realiza la acción de editar, registrar o borrar un usuario.	El sistema guarda los cambios realizados por el operador.
		El sistema muestra una alerta del estado de los cambios guardados.
Postcondiciones	Si los permisos del usuario son correctos, puede administrar los usuarios.	

En la Tabla 2 se muestra el caso de uso para registrar, editar y eliminar usuarios de tipo viverista e invitado. Este caso de uso puede ser utilizado por el operador.

Tabla 3. Descripción formal del caso de uso CU-2. Fuente: autoría propia.

Nombre	**Crear herramienta / CU-2**	
Actores	Operador, viverista, invitado	
Descripción	El operador, viverista o invitado, crea una herramienta.	
Precondición	El usuario es operador, viverista o invitado.	
Flujo principal	Acciones del usuario	Acciones del sistema
	El usuario presiona el botón de crear una nueva herramienta.	El sistema muestra un modal para ingresar el nombre de la herramienta.
	El operador crea una nueva herramienta.	El sistema despliega la herramienta recién creada, para configuración.
Postcondiciones	Si no existe una herramienta con el mismo nombre, el usuario crea una herramienta.	

En la Tabla 3, se muestra el caso de uso para crear una herramienta dinámica. El caso de uso CU-2 puede ser utilizado por todos los usuarios incluidos en el sistema con diferentes roles.

Tabla 4. Descripción formal del caso de uso CU-3. Fuente: autoría propia.

Nombre	**Visualizar herramienta / CU-3**	
Actores	Operador, viverista, invitado.	
Descripción	El operador, viverista o invitado, visualiza una herramienta.	
Precondición	El usuario es operador, viverista o invitado y tiene permisos para visualizar la herramienta o es su propia herramienta.	
Flujo principal	Acciones del usuario	Acciones del sistema
	El usuario presiona el botón de visualizar herramienta.	El sistema despliega la herramienta seleccionada.
	El usuario interactúa con la herramienta.	El sistema muestra los cambios que el usuario realiza en la visualización.
Postcondiciones	Si el usuario tiene los permisos; entonces, el usuario puede visualizar la herramienta.	

En la Tabla 4, se muestra el caso de uso para visualizar herramientas tanto integradas como dinámicas. Este caso puede por todos los usuarios incluidos en el sistema con diferentes niveles de acceso.

Tabla 5. Descripción formal del caso de uso CU-4. Fuente: autoría propia.

Nombre	**Editar herramienta / CU-4**	
Actores	Operador, viverista, invitado.	
Descripción	El operador, viverista o invitado, edita una herramienta.	
Precondición	El usuario es operador, viverista o invitado y es el creador de la herramienta.	
Flujo principal	Acciones del usuario	Acciones del sistema
	El usuario presiona el botón de visualizar herramienta.	El sistema despliega la herramienta seleccionada.
	El usuario edita la herramienta.	El sistema guarda los cambios realizados en la base de datos de manera asíncrona.
		El sistema muestra una alerta de la información guardada correctamente.
Postcondiciones	Si el usuario es el creador de la herramienta; entonces, el usuario puede editar la herramienta.	

En la Tabla 5 se muestra el caso de uso para editar la información de una herramienta dinámica, en el cual se incluyen todos los usuarios del sistema.

Tabla 6. Descripción formal del caso de uso CU-5. Fuente: autoría propia.

Nombre	**Compartir herramienta / CU-5**

Actores	Operador, viverista, invitado.	
Descripción	El operador, viverista o invitado, comparte su herramienta con los demás usuarios.	
Precondición	El usuario es operador, viverista o invitado y es el creador de la herramienta.	
Flujo principal	Acciones del usuario	Acciones del sistema
	El usuario se dirige al panel de herramientas.	El sistema despliega el panel de herramientas.
	El usuario selecciona una herramienta y presiona en opciones.	El sistema despliega las opciones de la herramienta seleccionada.
	El usuario marca la opción de compartir herramienta.	El sistema guarda los cambios marcados.
	El usuario puede ver un icono que indica si la herramienta esta compartida o no.	El sistema muestra una alerta de que la herramienta se compartió con éxito.
Postcondiciones	Si el usuario es el creador de la herramienta; entonces, el usuario puede compartir la herramienta.	

En la Tabla 6, se muestra el caso de uso para compartir una herramienta dinámica con los demás usuarios. El caso de uso CU-5 puede ser utilizado por todos los usuarios.

Tabla 7. Descripción formal del caso de uso CU-6. Fuente: autoría propia.

Nombre	**Buscar herramienta / CU-6**	
Actores	Operador, viverista, invitado.	
Descripción	El operador, viverista o invitado, busca una herramienta.	
Precondición	El usuario es operador, viverista o invitado y está activo en el sistema.	
Flujo principal	Acciones del usuario	Acciones del sistema
	El usuario se dirige al panel de herramientas.	El sistema despliega el panel de herramientas.
	El usuario busca una herramienta por el título, por el tipo de herramienta, por si esta compartida o no y por usuario.	El sistema filtra el resultado y lo despliega al usuario.
Postcondiciones	Si el usuario está activo en el sistema; Entonces, el usuario puede buscar la herramienta.	

En la Tabla 7, se muestra el caso de uso para buscar herramientas tanto integradas como dinámicas en su respectivo apartado.

Tabla 8. Descripción formal del caso de uso CU-7. Fuente: autoría propia.

Nombre	**Generar reporte / CU-7**
Actores	Operador, viverista, invitado.

Descripción	El operador, viverista o invitado, genera un reporte del análisis final.	
Precondición	El usuario es operador, viverista o invitado y tiene permisos para visualizar la herramienta o es creador de la herramienta.	
Flujo principal	Acciones del usuario	Acciones del sistema
	El usuario presiona el botón de visualizar herramienta.	El sistema despliega la herramienta seleccionada.
	El usuario presiona el botón de generar reporte formal.	El sistema muestra un modal para configurar el tipo de reporte deseado y las opciones disponibles.
	El usuario selecciona el tipo de reporte y formato.	El sistema obtiene la información del resultado de la herramienta, la procesa y genera un reporte en Excel o PDF.
		El sistema muestra una alerta de reporte generado correctamente.
Postcondiciones	Si el usuario tiene permisos para visualizar o es creador de la herramienta; Entonces, el usuario puede generar un reporte.	

En la Tabla 8, se muestra el caso de uso para generar reportes tanto de herramientas integradas como dinámicas. El caso de uso CU-7 puede ser utilizado por todos los tipos de usuarios.

4.1.2 Implementación

Se crearon los prototipos de interfaz de usuario necesario para tomar la idea del funcionamiento del sistema. Estos prototipos se ajustan en las siguientes fases conforme al cambio en los requisitos en cada iteración (Véase Figura 9 y Figura 10).

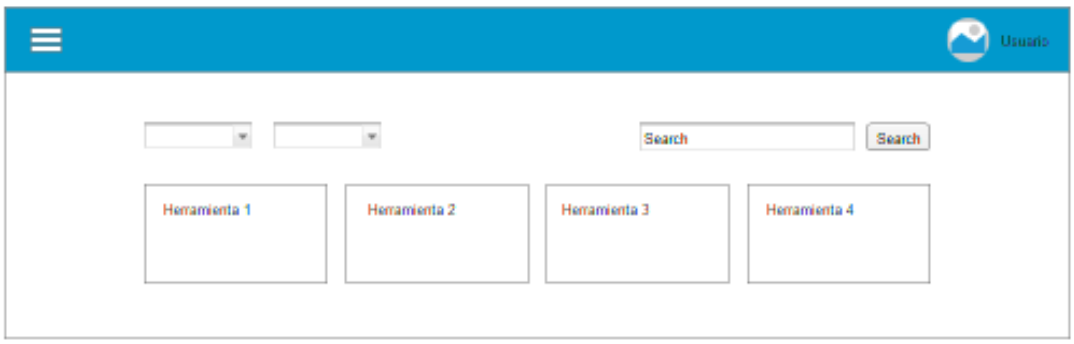

Figura 9. Prototipo de panel de herramientas. Fuente: autoría propia.

Como se puede observar en la Figura 9, se diseñaron los prototipos para la interfaz de usuario, donde se muestra el panel de herramientas integradas y dinámicas que puede ser consultado por todos los usuarios.

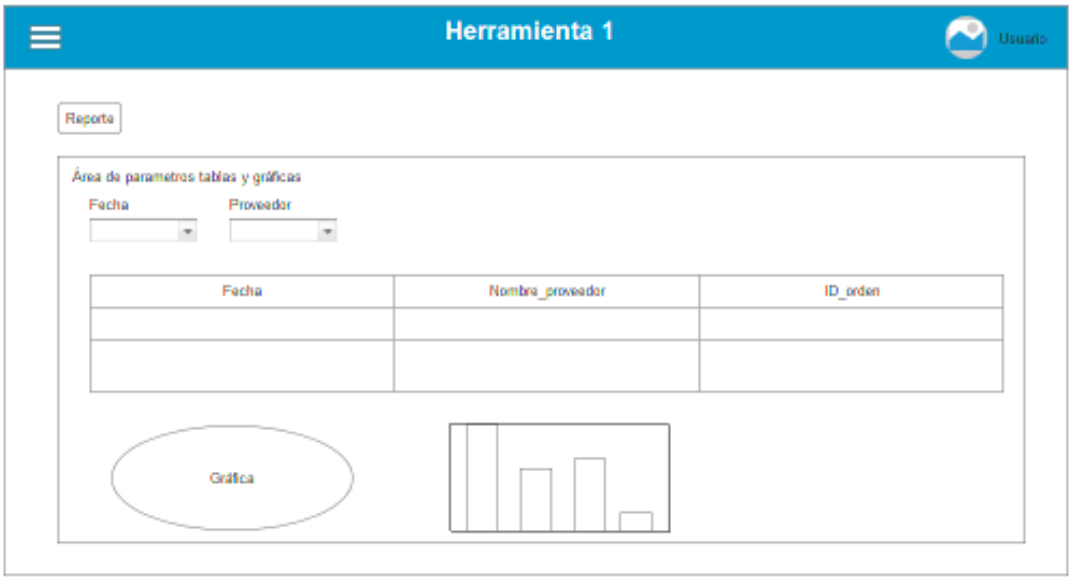

Figura 10. Prototipo de detalle de herramienta. Fuente: autoría propia.

La Figura 10, se muestra el diseño del detalle de una herramienta integrada y dinámica en forma de dashboard con parámetros, tablas y gráficas. El usuario puede analizar la información en cada herramienta y generar reportes periódicos.

4.1.3 Pruebas

Siguiendo el diagrama de pruebas FLOOT de la Figura 5, se llevaron a cabo las pruebas de requisitos descritas en la Tabla 9.

Tabla 9. Pruebas de requisitos efectuadas en el sistema. Fuente: autoría propia.

Prueba	Actividades
Revisión de modelos	Se realizó una revisión y recorrido formal por los modelos, para disminuir los riesgos a futuro.
Revisión de prototipos	Se realizó una revisión de los primeros prototipos para verificar que cumpla con los requisitos.
Demostrar con el código	Se demostró que se podía construir un software con base en el modelo conceptual construido, y se validó que cumpla con los requisitos.
Pruebas de escenarios de uso	Se tomaron los escenarios de cada caso de uso y se realizaron pruebas de recorrido para verificar el funcionamiento previo a codificación.

Además, se revisó lo que se elaboró en la gestión de la configuración y en la gestión del proyecto para que todo esté en orden y se revisó el modelo de requisitos y la propuesta de la arquitectura para poder continuar con la siguiente etapa.

4.1.4 Despliegue

Se definieron las actividades para determinar las fechas de inicio y entrega del sistema y no surjan dudas sobre los entregables a futuro. Además, se realizó el plan de despliegue del

sistema en el que define la plataforma en la que hospedará el sistema web, el dominio para redireccionar la IP donde está alojado el sistema y la base de datos que guardará las tablas generadas por el sistema.

4.1.5 Gestión de configuración

Para empezar con el desarrollo del software se instaló un entorno con las herramientas necesarias para desarrollar el sistema y además se crearon repositorios, se utilizaron herramientas para planeación de actividades en la nube y se definió un horario de trabajo.

En este caso, no se necesitó una gran curva de aprendizaje ni capacitar a ninguna persona para empezar el desarrollo. En lo que se tomó más tiempo fue en desarrollar un algoritmo para la lectura de la base de datos, el cual se explica en la disciplina de implementación, tomando en cuenta que la base de datos puede ser distinta y que las consultas se realizan de forma dinámica.

4.1.6 Gestión de proyectos

En esta disciplina, se determinó la factibilidad financiera, tecnológica, operacional y política del sistema, el calendario de actividades, se determinaron los riesgos.

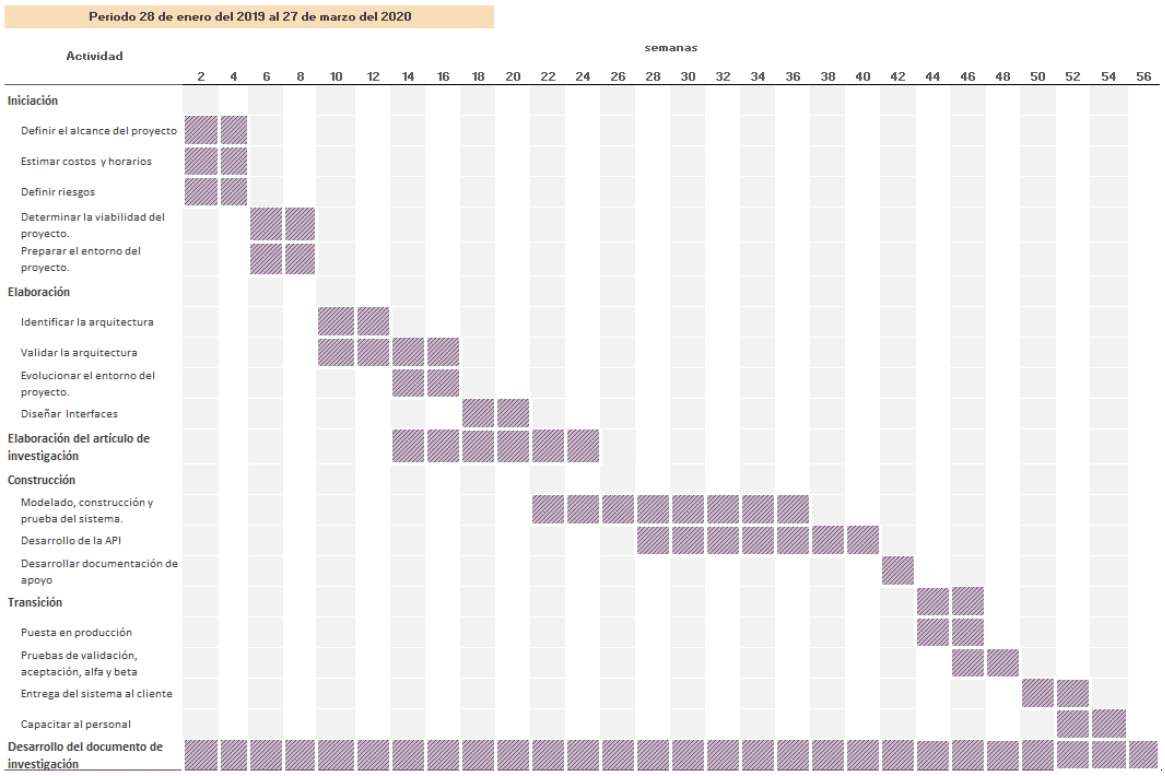

Figura 11. Cronograma de actividades. Fuente: autoría propia.

En la Figura 11 se muestra el calendario de actividades iniciando desde el 28 de enero del 2019 al 27 de marzo del 2020.

A continuación, se muestra el listado de riesgos potenciales que pueden afectar el desarrollo del sistema. Los riesgos se definen en la iniciación del proyecto y se van modificando, agregando y eliminado a lo largo del ciclo de vida del desarrollo del proyecto.

- **Nuevas versiones del framework Django**: para desarrollar la plataforma web se utiliza la versión 2.1.5 del framework Django; el cuál, puede ser un riesgo potencial a futuro ya que el sistema a desarrollar quedaría obsoleto, pero puede ser actualizado fácilmente.
- **Sistema para personas no expertas en el área:** este riesgo es muy importante ya que el sistema debe ser desarrollado de tal manera que todo tipo de usuario pueda operarlo sin necesidad de tener conocimiento en SQL, HTML, Django, etc. Que le permita al usuario realizar cualquier tipo de operación sin complicaciones. Es muy importante considerar este riesgo en las siguientes interacciones para mitigarlo.
- **Cambio en los requisitos funcionales y no funcionales:** estos requisitos se definen en la etapa de iniciación por lo que al paso del tiempo conforme vayan siendo implementados, pueden cambiar y afectar enteramente al desarrollo del proyecto.
- **No utilización de base de datos:** el desarrollo del sistema se base en la utilización de una base de datos para extraer información, si esta base de datos no es utilizada por los usuarios o es modificada, puede afectar el desarrollo del sistema por completo ya que no habría datos que consultar.
- **Rendimiento y despliegue:** que el sistema presente fallas en algún momento por la cantidad de datos consultados y que presente problemas de despliegue en la plataforma seleccionada.

4.1.7 Medio ambiente

En esta diciplina se estableció el lugar de trabajo. Debido a que una sola persona trabaja en el desarrollo del sistema, solamente se establece un equipo de trabajo y la estación de trabajo permanecerá igual a lo largo del desarrollo del sistema.

4.2 Fase de elaboración

En esta fase se aseguró que la arquitectura para el sistema a desarrollar satisfaga los requisitos establecidos por los stakeholders; así mismo, se acotó la lista de pruebas a desarrollar y se crearon nuevos prototipos para las interfaces de usuario (UI, por sus siglas Ingles) del panel de herramientas y detalle de herramientas.

4.2.1 Modelo

Se modificaron los prototipos para el panel de herramientas y el detalle de una herramienta. Estos prototipos se irán ajustando en las demás fases (véase la Figura 12, Figura 13 y Figura 14 y Figura 15).

Figura 12. Prototipo final para el panel de herramientas integradas. Fuente: autoría propia.

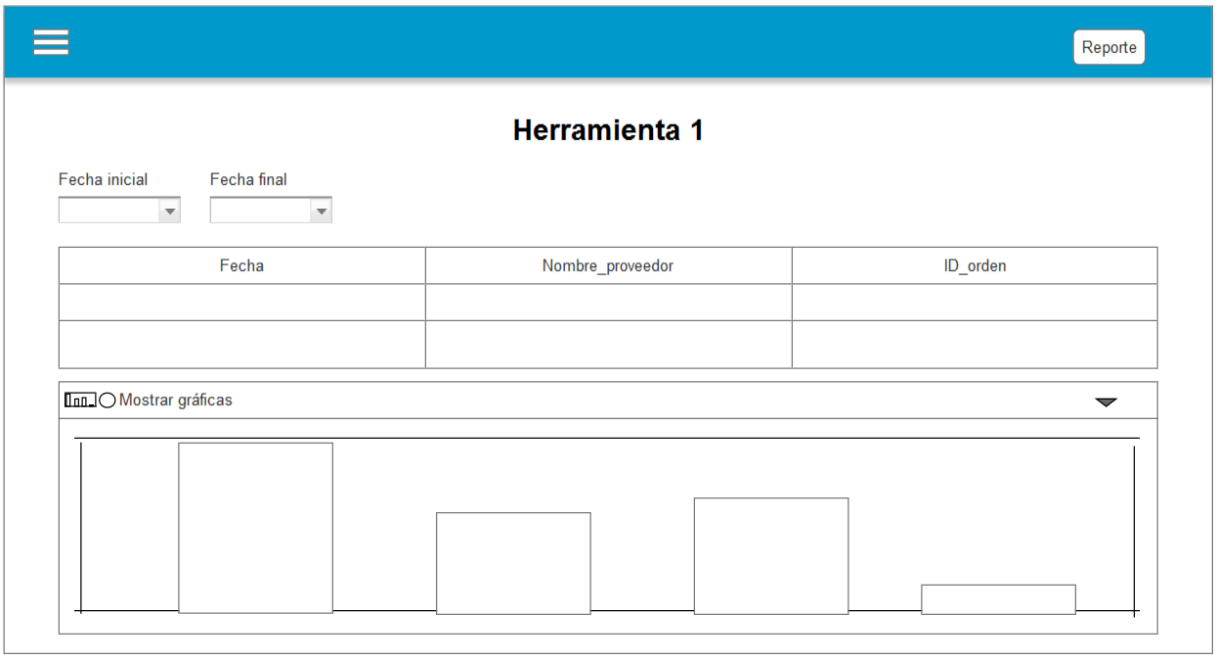

Figura 13. Prototipo final para el detalle de una herramienta integrada. Fuente: autoría propia.

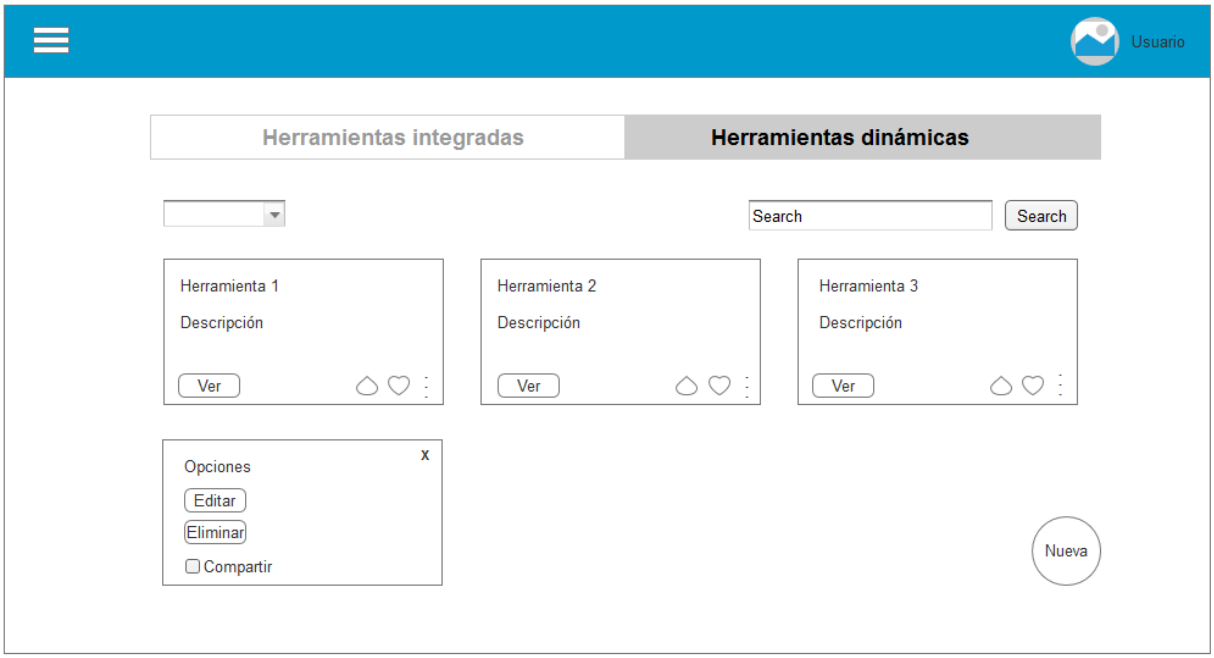

Figura 14. Prototipo final para el panel de herramientas dinámicas. Fuente: autoría propia.

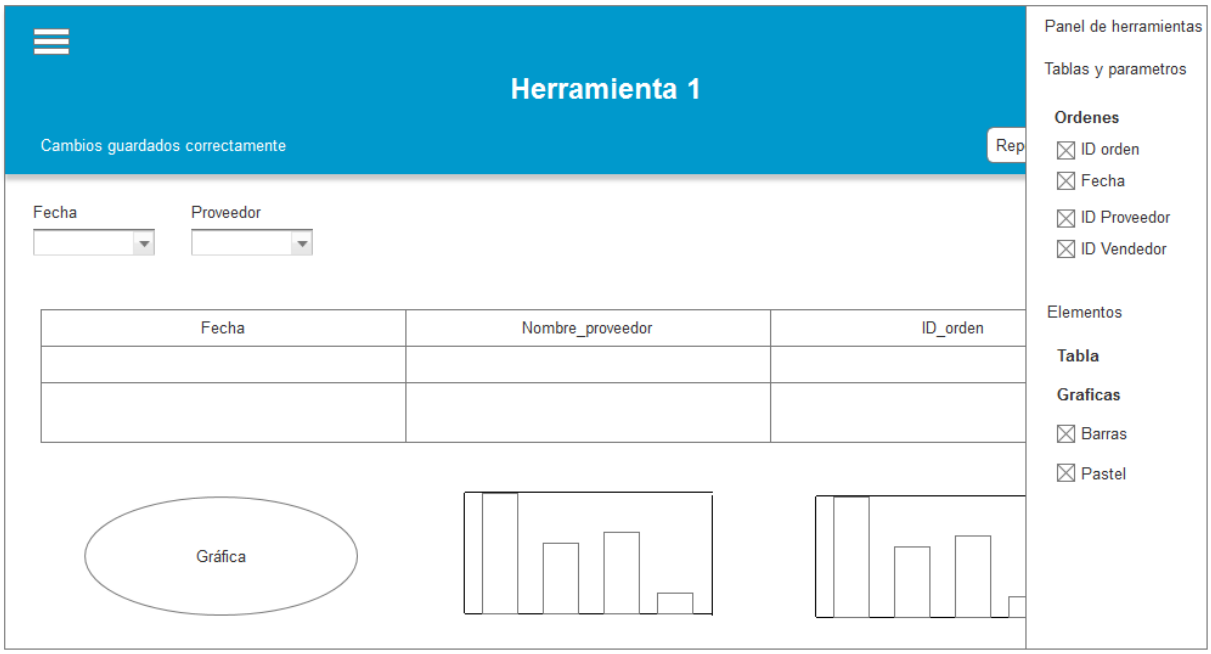

Figura 15. Prototipo final para el detalle de una herramienta dinámica. Fuente: autoría propia.

Como se puede observar en la Figura 14, se modificó el panel de herramientas, se añadieron más elementos para identificar una herramienta y saber si una herramienta es creada por el usuario que ha iniciado sesión o esta compartida por alguien más; además, se añadió una opción de editar herramienta y compartir herramienta como opciones de éstas. En la Figura 15 se muestra ahora un panel de herramienta que se despliega en la parte derecha, el cual contiene las tablas y parámetros que podrán ser utilizados en para filtrar información, así mismo los elementos como tablas y gráficas para visualización. Por otro lado, se añadieron

patrones de diseño de UI como el guardado automático para indicar al usuario que los cambios se guardaron correctamente, debido al funcionamiento asíncrono que tendrá esta interfaz.

4.2.2 Implementación

La arquitectura física del sistema se fundamenta en una arquitectura cliente-servidor, la cual ilustra el funcionamiento del sistema mediante un modelo conceptual mostrado en la Figura 1 del Capítulo 1 Introducción, éste explica cómo el cliente realiza una petición y cómo el sistema se encarga de procesar los datos enviados por el cliente.

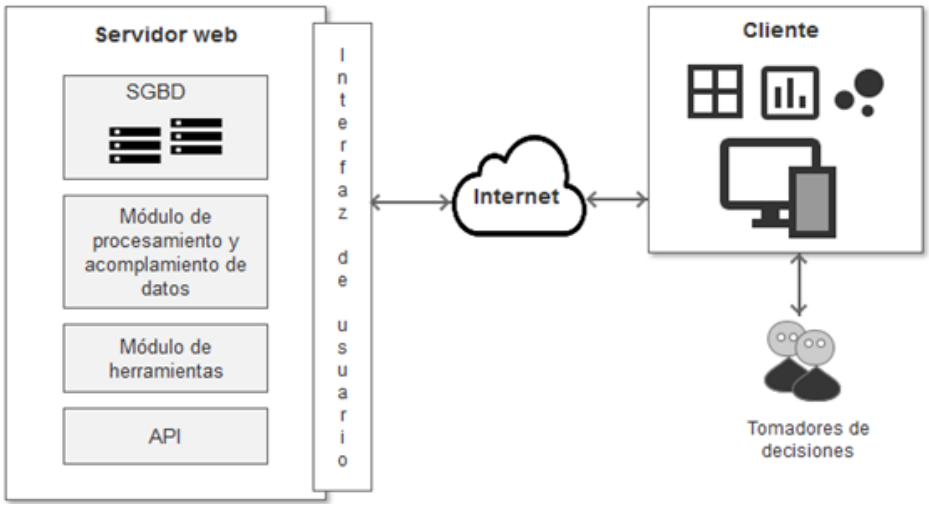

Figura 16. Arquitectura cliente-servidor de ORNALISIS. Fuente: autoría propia.

Como se puede observar en la Figura 16, la arquitectura especifica varios módulos: una interfaz de aplicaciones (API) y un sistema gestor de base de datos (SGDB), un módulo de procesamiento y acoplamiento de datos, y el módulo de herramientas dinámicas. La funcionalidad de estos módulos se describe a continuación:

- **SGBD**: contiene la información histórica de la comercialización de plantas ornamentales; además, se especificó un base de datos propia para el sistema DSS.
- **API**: permite consultar información de manera asíncrona a la base de datos desde cualquier dispositivo utilizando el formato JSON para solicitar y desplegar información.
- **Módulo de procesamiento y acoplamiento de datos:** se encarga de acoplar y procesar los datos recibidos y enviados a través de la interfaz de aplicaciones API, la información es convertida a formato JSON y regresada al cliente para su posterior despliegue en forma de tabla, gráficas y reportes.
- **Módulo de herramientas:** se encarga de enviar y recibir la información en formato JSON a través de la API para ser representada en el front-end al usuario mediante tablas, gráficas, reportes etc.
    - **Las herramientas dinámicas** extraen información de la base de datos histórica mediante parámetros, permitiendo la creación de consultas dinámicas a la base de datos con la implementación del algoritmo DFS para la obtención de

relaciones entre tablas. Este algoritmo DFS se implementó debido que el sistema permite vincular cualquier BD relacional, ya que el sistema está dirigido a usuarios no expertos en la materia; por lo tanto, se necesitaba un método para la construcción de consultas dinámicas que permita convertir cualquier BD relacional a grafo y después determinar mediante DFS si se puede ejecutar la consulta o no, sin requerir un conocimiento sobre la estructura de la base de datos por el usuario del sistema.

El sistema construye dinámicamente una consulta SQL ejecutable por MySQL, llevando a cabo los siguientes pasos:

1. Obtener la información que el usuario solicita en el cliente o navegador web.
2. Procesar la información y solicitar la información interna de la BD seleccionada.
3. Construir las cláusulas SELECT y WHERE a partir de la información solicitada por el usuario.
4. Construir la cláusula FROM, que contiene las tablas utilizadas en la consulta y aplicar el algoritmo DFS diseñado para determinar la existencia de relaciones entre las tablas.
5. Juntar todas las cláusulas y ejecutar la consulta SQL.

- **Las herramientas integradas**, consultan información a la base de datos de comercialización y permiten visualizar información específica mediante tablas, gráficas y reportes personalizados tal cual y como el operador los requiera. Estas herramientas no cuentan con la función de una dinámica ya que están dirigidas a consultas muy específicas, es decir, son herramientas que en este caso en particular la sociedad de producción rural ORNACOL requiere mensualmente para la extracción de información. Por ejemplo, ventas por especie, ventas totales por año, entre otras.

4.2.3 Pruebas

Siguiendo el diagrama de pruebas FLOOT de la Figura 5 Especificado en el Capítulo 3 en la sección de metodología de desarrollo de software, en la diciplina de pruebas de la fase de iniciación, se llevaron a cabo las pruebas de análisis, las cuales se describen en la Tabla 10.

Tabla 10. Pruebas de análisis hechas en el sistema. Fuente: autoría propia.

Prueba	Actividades
Revisión de modelos	Se realizaron las últimas revisiones de los modelos para reducir riesgos.
Revisión de prototipos	Se realizaron los nuevos prototipos para verificar que cumplieran con los requisitos.
Demostrar con el código	Se demostró se podía continuar con el desarrollo del software.

Además, se modificó la lista de pruebas a utilizar y se elaboró un reporte de defectos para indicar si el sistema estaba siendo desarrollado de la manera en la que se planteó.

4.2.4 Despliegue

Se siguió modificando el plan de despliegue y se realizaron pruebas de varias plataformas para determinar cuál era el mejor servidor para alojar el sistema en la nube.

4.2.5 Gestión de configuración

Se verificó que todo fuera acorde a lo planeado en la fase de iniciación. Se añadieron nuevas herramientas para el desarrollo del sistema.

4.2.6 Gestión de proyectos

Se verificó que todo fuera acorde al cronograma de actividades de la fase de iniciación; además, se diseñó un plan para la mitigación de cada riesgo, como se muestra a continuación.

- **Nuevas versiones del framework Django**: se seleccionó la versión 2.1.5 de Django, actualmente la versión estable y reciente del framework que permite actualizaciones futuras; además, se utilizaron librerías que no den problemas a futuro.
- **Sistema para personas no expertas en el área:** el sistema será desarrollado a manera de herramientas, cada herramienta contendrá un requisito funcional especifico y contendrá filtros, tablas y gráficas, para consultar información.
- **Cambio en los requisitos funcionales y no funcionales:** las reuniones mensuales permiten determinar los cambios en el sistema y permitirá al desarrollador acotarse a los cambios aceptados en esa reunión sin afectar el desarrollo.
- **No utilización de base de datos:** se tendrá que insertar información histórica de la sociedad de producción rural ORNACOL a la base de datos para realizar pruebas de las distintas herramientas.
- **Rendimiento y despliegue:** el equipo de trabajo tiene el conocimiento para el despliegue del framework Django en la nube por lo tanto no dará problemas de despliegue; por otro lado, se implementarán estrategias de visualización de información para probar cualquier computadora y verificar que rendimiento sea el correcto.

4.2.7 Medio ambiente

Al estar a punto de empezar con el desarrollo del software, se instalaron las herramientas necesarias en la computadora de desarrollo y se decidió por guardar los cambios del sistema en la nube utilizando OneDrive, como el sistema se planteó en Django y estará en desarrollo, es fácil correr el servidor en la línea de comandos de Windows e iniciar a trabar en cualquier equipo que tenga OneDrive vinculado y las herramientas de desarrollo instaladas.

4.3 Fase de Construcción

En las fases anteriores se realizaron los modelos necesarios para entender el funcionamiento del negocio y crear una arquitectura que resuelva el problema de éste. En esta fase, se construyó el sistema al punto de estar listo para las pruebas de software y para presentar la primera versión del sistema en funcionamiento.

4.3.1 Modelo

En el modelo de la fase de construcción se empieza la construcción del sistema donde se realizó el análisis de lo modelado hasta ese momento y se trabajó junto con los stakeholders para definir los casos de uso y requerimientos finales y diagramas de secuencia, clases, requerimientos, despliegue; además, se diseñaron diagramas de secuencia, despliegue, clases y diagrama de forma libre que describe por capas el funcionamiento del sistema.

4.3.1.1 Diagrama de casos de uso

Para entender los casos de usos necesarios para el desarrollo del sistema, se modelan los requisitos funcionales de manera general en el diagrama de casos de uso de la Figura 17. Diagrama de casos de uso. Fuente: autoría propia.

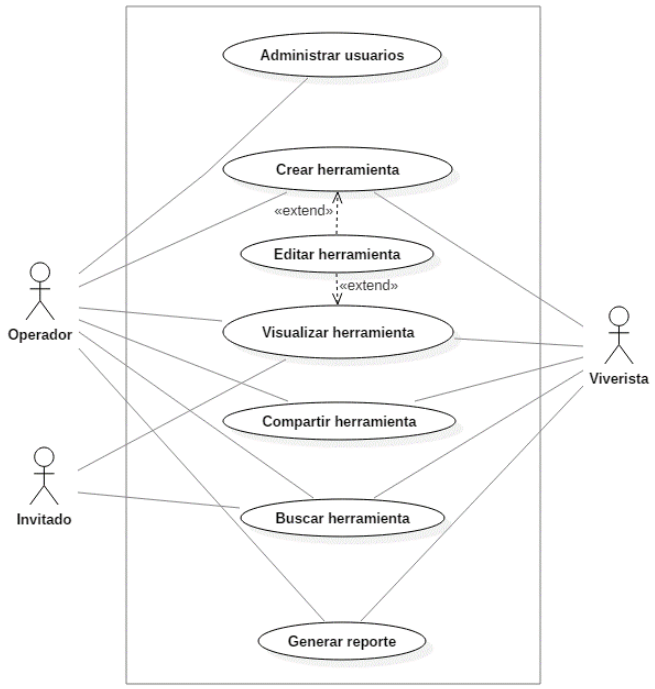

Figura 17. Diagrama de casos de uso. Fuente: autoría propia.

En la Figura 17, se muestra el diagrama de casos de uso a partir de los casos de uso descritos en la fase de iniciación, donde se puede observar que habrá tres tipos de usuarios operador, viverista e invitado que podrán interactuar con siete casos de uso dependiendo de sus permisos. En general, estos son los casos de uso utilizados para el desarrollo del sistema.

4.3.1.2 Modelo de requisitos

Después de levantar los requisitos se realizó un modelo de requisitos funcionales y no funcionales siguiendo la especificación de requisitos IEEE 830 (IEEE, 2008)

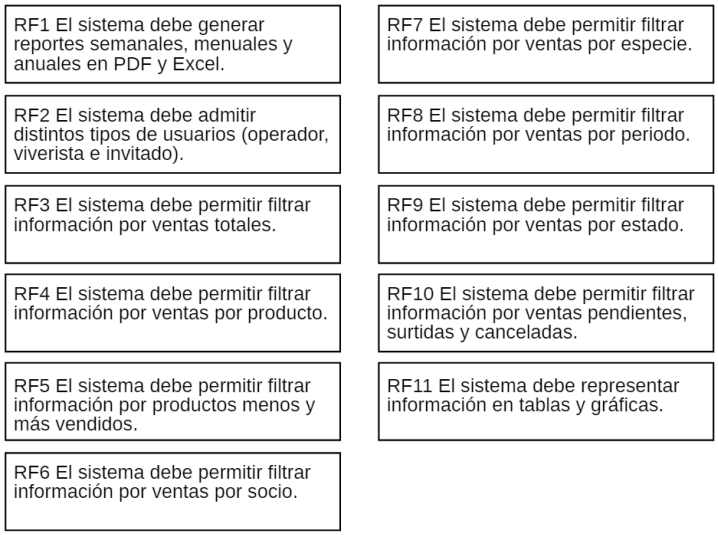

Figura 18. Requisitos funcionales. Fuente: autoría propia.

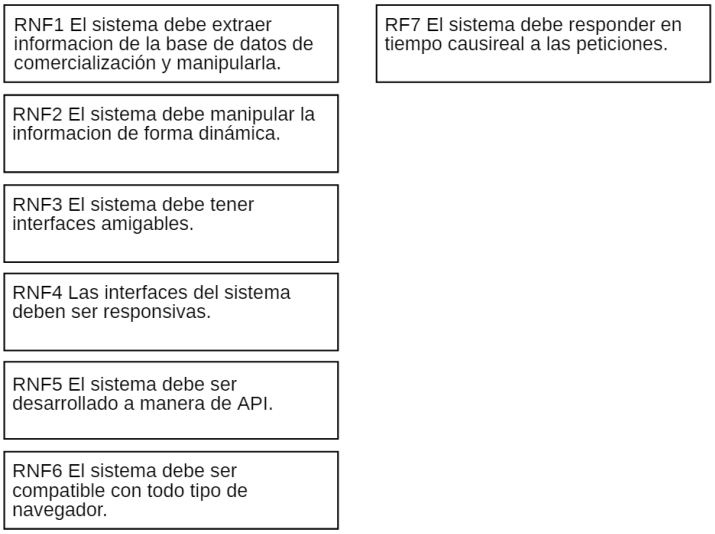

Figura 19. Requisitos no funcionales. Fuente: autoría propia.

Los requisitos que se observan en la Figura 18 y la Figura 19 son los que fueron utilizados para el desarrollo del sistema, mismos que se fueron cumpliendo en cada iteración.

4.3.1.3 Diagramas de secuencias

Los diagramas de secuencia modelan el flujo de las actividades realizadas por los actores y el sistema a través del tiempo de una manera visual. Este tipo de diagrama representa los detalles de los casos de uso del sistema; además, de planificar y comprender la funcionalidad detallada de un escenario actual o futuro. A continuación, se muestran diagramas de secuencia diseñados para cada caso de uso de la Figura 17.

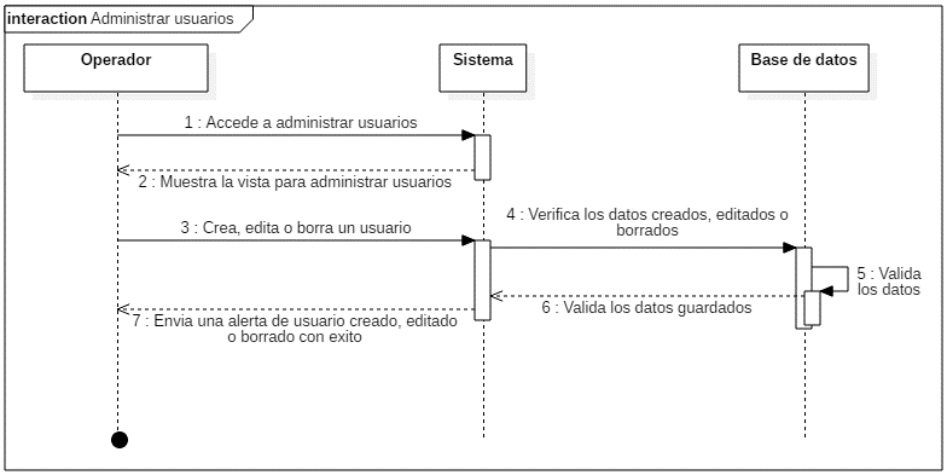

Figura 20. Diagrama de secuencia para administrar usuarios. Fuente: autoría propia.

El diagrama de secuencia de la Figura 20, describe el CU-1 y flujo de la interacción entre el usuario, sistema y base de datos para administrar usuarios.

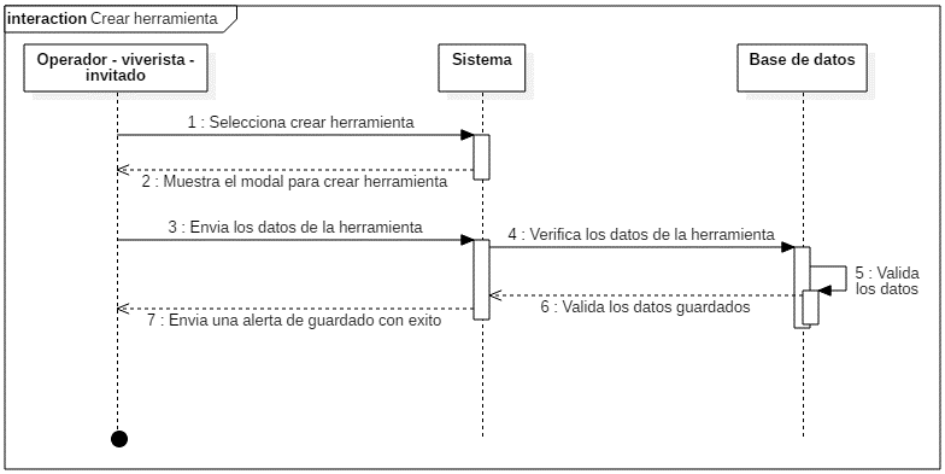

Figura 21. Diagrama de secuencia para crear una herramienta dinámica. Fuente: autoría propia.

El diagrama de secuencia de la Figura 21, describe el CU-2 y flujo de la interacción entre el usuario, sistema y base de datos para crear una herramienta dinámica.

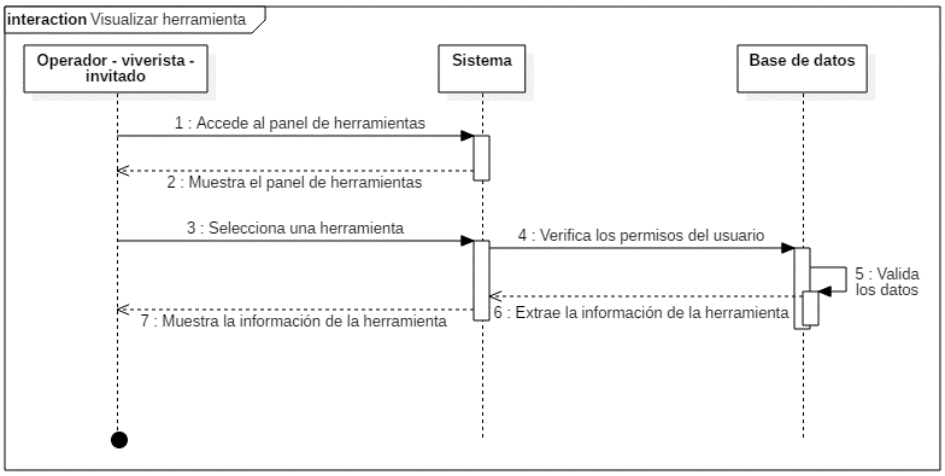

Figura 22. Diagrama de secuencia para visualizar herramientas integradas y dinámicas. Fuente: autoría propia.

El diagrama de secuencia de la Figura 22, describe el CU-4 y flujo de la interacción entre el usuario, sistema y base de datos para visualizar herramientas integradas y dinámicas.

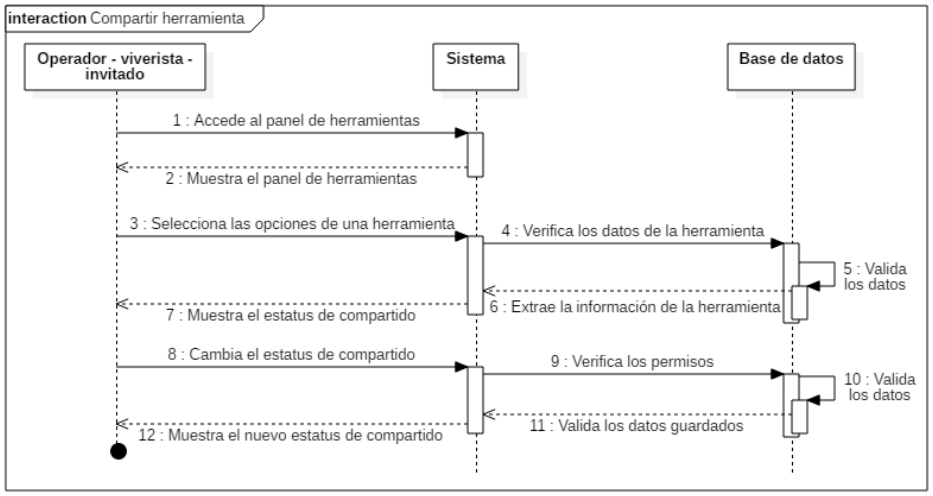

Figura 23. Diagrama de secuencia para compartir herramienta dinámica. Fuente: autoría propia.

El diagrama de secuencia de la Figura 23, describe el CU-5 y flujo de la interacción entre el usuario, sistema y base de datos para compartir permisos de herramientas dinámicas entre los distintos usuarios.

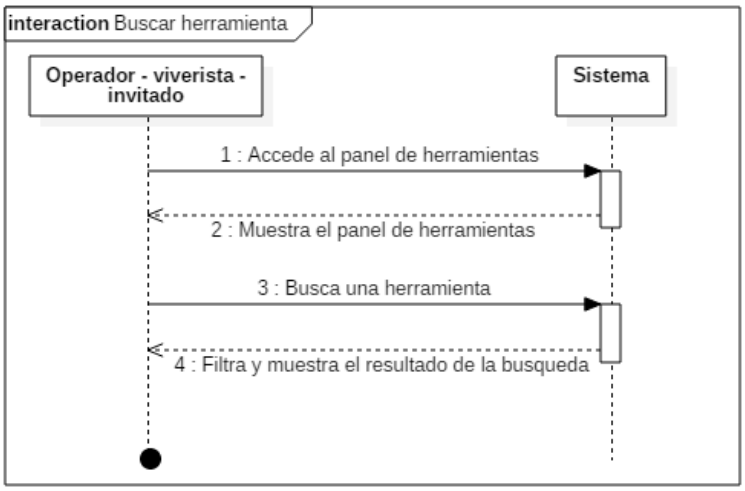

Figura 24. Diagrama de secuencia para buscar herramienta integrada y dinámica. Fuente: autoría propia.

El diagrama de secuencia de la Figura 24, describe el CU-6 y flujo de la interacción entre el usuario y sistema para filtrar herramientas integradas y dinámicas.

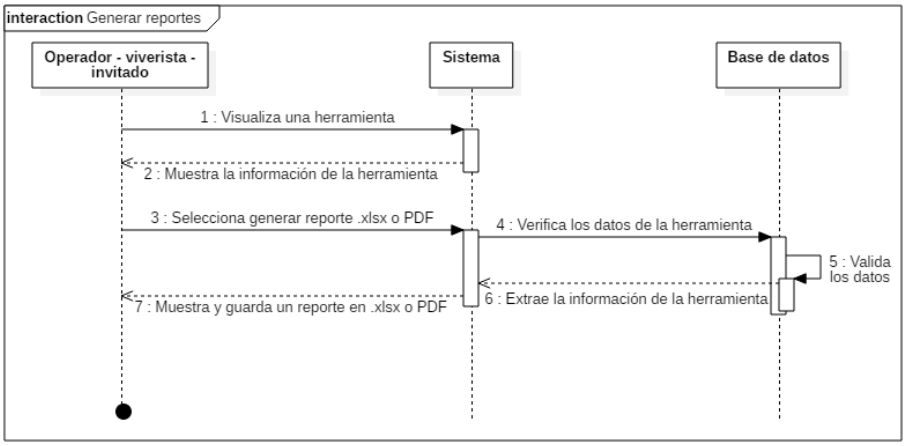

Figura 25. Diagrama de secuencia para generar reportes de herramientas integradas y dinámicas. Fuente: autoría propia. Fuente: autoría propia.

El diagrama de secuencia de la Figura 25, describe el CU-7 y flujo de la interacción entre el usuario, sistema y base de datos para generar reportes en herramientas integradas y dinámicas.

4.3.1.4 Diagrama de despliegue

El diagrama de despliegue se diseñó para modelar la arquitectura en tiempo causi-real del sistema, mostrando los nodos involucrados en el funcionamiento del sistema. El diagrama de despliegue de la Figura 26 muestra el nodo correspondiente al servidor de aplicaciones, el nodo correspondiente a la base de datos del sistema y el nodo correspondiente al dispositivo del usuario.

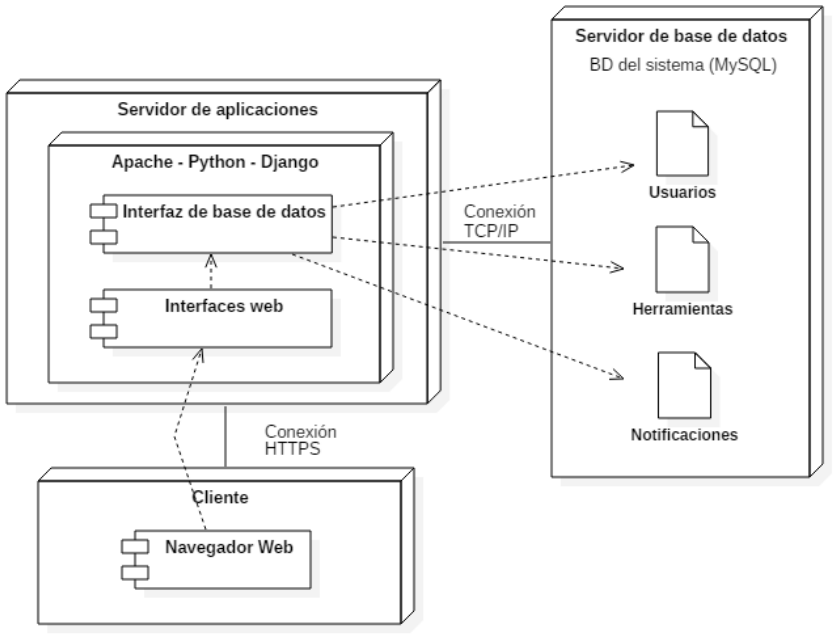

Figura 26. Diagrama de despliegue del sistema. Fuente: autoría propia.

4.3.1.5 Diagrama de clases

Se diseñó el diagrama de clases que muestra el esquema de la base de datos que el sistema utiliza para la creación de herramientas dinámicas y administración de usuarios (véase Figura 27).

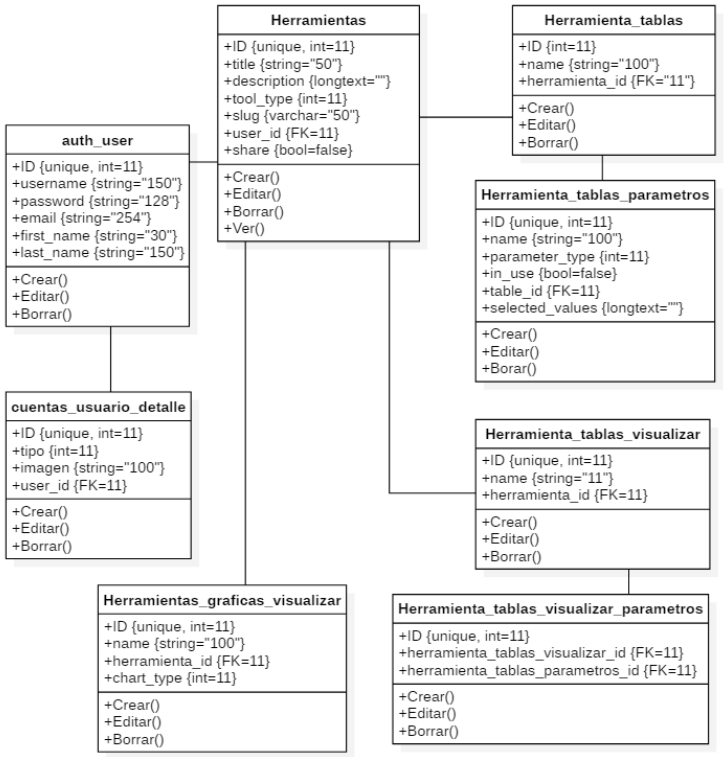

Figura 27. Diagrama de clases del sistema ORNALISIS. Fuente: autoría propia.

Como se puede observar en la Figura 27, el sistema trabaja con seis clases esenciales para el funcionamiento del sistema, cada una de las clases tiene las mismas operaciones de creación, edición, borrado y lectura.

4.3.2 Implementación

Una vez realizado el modelado y haberse aprobado la arquitectura del sistema, se empezó el desarrollo de éste siguiendo los requisitos funcionales y no funcionales previamente descritos.

Como ya se mencionó, el sistema cuenta con dos tipos de herramientas, las herramientas integradas y las herramientas dinámicas. A continuación, se explica de manera detallada el desarrollo de estas herramientas.

4.3.2.1 Herramientas integradas

Las herramientas integradas cumplen con los requisitos funcionales solicitados por la sociedad de producción rural ORNACOL, incluyendo la herramienta de ventas por proveedor, ventas por especie, ventas por estado, especies menos y más vendidas y ventas totales.

Para las herramientas integradas se hizo uso de la base de datos histórica de comercialización y se utilizaron Web APIs para la consulta de información en formato JSON y presentarla al usuario mediante tablas y gráficas. Además, cada herramienta permite la creación de reportes en formato .PDF y .XLSX para ser presentados a los socios de ORNACOL.

A continuación, se presenta la implementación de estas herramientas mostrando una parte del código de cada herramienta.

La herramienta de ventas por proveedor obtiene las ventas totales por proveedor en una fecha especifica, permitiendo al usuario filtrar informacion y visualizarla mediante tablas y gráficas como se muestra en la Figura 28.

```python
class getVentasPorProveedor(APIView):
    permission_classes = (permissions.IsAuthenticated,)

    def post(self, request, format=None):
        DataDict = dict(request.data)

        fecha_inicial= request.POST["fecha_inicial"]
        fecha_final= request.POST["fecha_final"]

        if fecha_final=="":
            fecha_final= datetime.date.today()
            fecha_final=fecha_final.strftime('%d/%m/%Y')

        fecha_inicial= fecha_inicial.split("/")
        fecha_inicial= fecha_inicial[2]+"/"+fecha_inicial[1]+"/"+fecha_inicial[0]
        fecha_final= fecha_final.split("/")
        fecha_final= fecha_final[2]+"/"+fecha_final[1]+"/"+fecha_final[0]

        cursor = connections['ornacoldb'].cursor()
        query_total= 'SELECT SUM(detalle_pedido.dp_total) AS total_venta, COUNT(*), SUM(detalle_pedido.dp_cantidad) AS total_plantas FRO
        cursor.execute(query_total)
        query_total_result = cursor.fetchall()

        total_venta= query_total_result[0][0]
        total_plantas=  query_total_result[0][2]

        if query_total_result[0][1]==0:
            total_venta=0
            total_plantas=0

        where_clause= "WHERE detalle_pedido.dp_id_planta=planta_viverista.plaviv_id AND planta_viverista.plaviv_id_vivero=vivero.id_vive
        where_clause+=' AND (pedido.p_fecha_pedido BETWEEN "'+fecha_inicial+'" AND "'+fecha_final+'")'

        if 'id_proveedor' in self.request.POST:
            proveedores= request.POST["id_proveedor"]
            where_clause += " AND (viverista.id_viverista"+" = "
            for parameter, values in DataDict.items():
                if parameter== "id_proveedor":
                    cv=0
                    for proveedor in values:
                        cv+=1
                        if cv == len(values):
                            where_clause += proveedor+")"
                        else:
                            where_clause += proveedor +" OR "+ "viverista.id_viverista"+" = "
        else:
            proveedores=0

        query_datos= "SELECT CONCAT(viverista.id_viverista,' - ', viverista.nombre), SUM(detalle_pedido.dp_cantidad) AS Cantidad, ROUND(
        cursor.execute(query_datos)
        query_datos_result = cursor.fetchall()
        print(query_datos_result)
        chart_data=[]
        chart_data_pie=[]
        chart_labels=[]
        chart_data_doughnut=[]

        for datos in query_datos_result:
            chart_data.append(datos[2])
            chart_labels.append(datos[0])
            chart_data_pie.append(datos[4])
            chart_data_doughnut.append(datos[5])
        data={
            "data":query_datos_result,
            "chart_data": chart_data,
            "chart_labels":chart_labels,
            "chart_data_pie":chart_data_pie,
            "chart_data_doughnut":chart_data_doughnut
        }
        return Response(data)
```

Figura 28. Código de herramienta de ventas por proveedor. Fuente: autoría propia.

La herramienta de ventas por especie obtiene las ventas totales por especies y por proveedor en una fecha específica, permitiendo al usuario filtrar información y visualizarla mediante tablas y gráficas como se muestra en la Figura 29

```python
class getVentasPorEspecie(APIView):
    permission_classes = (permissions.IsAuthenticated,)

    def post(self, request, format=None):
        DataDict = dict(request.data)

        fecha_inicial= request.POST["fecha_inicial"]
        fecha_final= request.POST["fecha_final"]
        if fecha_final=="":
            fecha_final= datetime.date.today()
            fecha_final=fecha_final.strftime('%d/%m/%Y')

        fecha_inicial= fecha_inicial.split("/")
        fecha_inicial= fecha_inicial[2]+"/"+fecha_inicial[1]+"/"+fecha_inicial[0]
        fecha_final= fecha_final.split("/")
        fecha_final= fecha_final[2]+"/"+fecha_final[1]+"/"+fecha_final[0]

        cursor = connections['ornacoldb'].cursor()

        where_clause= "WHERE detalle_pedido.dp_id_planta= planta_viverista.plaviv_id AND planta_viverista.plaviv_id_planta= planta.id_planta AND p
        where_clause+=' AND (pedido.p_fecha_pedido BETWEEN "'+fecha_inicial+'" AND "'+fecha_final+'")'

        query_viveristas= "SELECT * FROM viverista"
        cursor.execute(query_viveristas)
        query_viveristas_result = cursor.fetchall()

        if 'id_especie' in self.request.POST:
            especies= request.POST["id_especie"]

            where_clause += " AND (planta.id_planta"+" = "
            for parameter, values in DataDict.items():
                if parameter== "id_especie":
                    cv=0
                    for especie in values:
                        cv+=1
                        if cv == len(values):
                            where_clause += especie+")"
                        else:
                            where_clause += especie +" OR "+ "planta.id_planta"+" = "
        else:
            especies=0

        tablas_por_proveedor={}
        cv=0

        for viverista in query_viveristas_result:
            query_datos= "SELECT CONCAT(viverista.id_viverista,' - ', viverista.nombre), CONCAT(detalle_pedido.dp_id_planta,' - ', planta.nombre),
            cursor.execute(query_datos)
            # print(query_datos)
            query_datos_result = cursor.fetchall()
            chart_data=[]
            chart_labels=[]
            for datos in query_datos_result:
                chart_data.append(datos[0])
                chart_labels.append(datos[1])

            tablas_por_proveedor[cv]={
                'id':viverista[0],
                'name':viverista[1],
                'data': query_datos_result,
                'chart_data': chart_data,
                'chart_labels':chart_labels
            }

            cv+=1

        return Response(tablas_por_proveedor)
```

Figura 29. Código de herramienta de ventas por especie. Fuente: autoría propia.

La herramienta de ventas por estado obtiene las ventas totales por estado en una fecha especifica, permitiendo al usuario filtrar información y visualizarla mediante tablas y gráficas como se muestra en la Figura 30.

```python
class getVentasEstados(APIView):
    permission_classes = (permissions.IsAuthenticated,)

    def post(self, request, format=None):

        fecha_inicial= request.POST["fecha_inicial"]
        fecha_final= request.POST["fecha_final"]

        if fecha_final=="":
            fecha_final= datetime.date.today()
            fecha_final=fecha_final.strftime('%d/%m/%Y')

        fecha_inicial= fecha_inicial.split("/")
        fecha_inicial= fecha_inicial[2]+"/"+fecha_inicial[1]+"/"+fecha_inicial[0]
        fecha_final= fecha_final.split("/")
        fecha_final= fecha_final[2]+"/"+fecha_final[1]+"/"+fecha_final[0]

        cursor = connections['ornacoldb'].cursor()
        # Extraer el rango de años de los datos historicos
        query_estados= 'SELECT estados.key_estado as id, estados.nombreEstado as nom
        cursor.execute(query_estados)
        columns = [col[0] for col in cursor.description]
        data_estados= [dict(zip(columns, row)) for row in cursor.fetchall()]

        query_estados= 'SELECT estados.nombreEstado as nombre, SUM(detalle_pedido.dp
        cursor.execute(query_estados)
        query_estados_result= cursor.fetchall()

        data={
            'estados': data_estados,
            'estados_table': query_estados_result,
        }

        return Response(data)
```

Figura 30. Código de herramienta de ventas por estado. Fuente: autoría propia.

La herramienta de clasificacion de especies menos y mas vendidas, obtiene las ventas por especies menos y mas vendidas en una fecha específica, permitiendo al usuario filtrar información y visualizarla mediante tablas y gráficas como se muestra en la Figura 31.

```
class getEspeciesMenosYMas(APIView):
    permission_classes = (permissions.IsAuthenticated,)

    def post(self, request, format=None):
        DataDict = dict(request.data)

        fecha_inicial= request.POST["fecha_inicial"]
        fecha_final= request.POST["fecha_final"]

        if fecha_final=="":
            fecha_final= datetime.date.today()
            fecha_final=fecha_final.strftime('%d/%m/%Y')

        fecha_inicial= fecha_inicial.split("/")
        fecha_inicial= fecha_inicial[2]+"/"+fecha_inicial[1]+"/"+fecha_inicial[0]
        fecha_final= fecha_final.split("/")
        fecha_final= fecha_final[2]+"/"+fecha_final[1]+"/"+fecha_final[0]

        cursor = connections['ornacoldb'].cursor()

        query_mas_vendidos= 'SELECT planta.id_planta, planta.nombre, planta.presentacion, SUM(detalle_pedido.
        cursor.execute(query_mas_vendidos)
        query_total_mas_vendidos = cursor.fetchall()

        query_menos_vendidos= 'SELECT planta.id_planta, planta.nombre, planta.presentacion, SUM(detalle_pedid
        cursor.execute(query_menos_vendidos)
        query_total_menos_vendidos = cursor.fetchall()

        chart_data_menos=[]
        chart_labels_menos=[]
        chart_data_mas=[]
        chart_labels_mas=[]

        for datos in query_total_menos_vendidos:
            chart_data_menos.append(datos[3])
            chart_labels_menos.append(datos[1]+" - "+datos[2])
        for datos in query_total_mas_vendidos:
            chart_data_mas.append(datos[3])
            chart_labels_mas.append(datos[1]+" - "+datos[2])

        data={
            "data":{
                "menos": query_total_menos_vendidos,
                "mas": query_total_mas_vendidos
            },

            "chart_data_menos": chart_data_menos,
            "chart_data_mas":chart_data_mas,
            "chart_labels_menos":chart_labels_menos,
            "chart_labels_mas":chart_labels_mas
        }
        return Response(data)
```

Figura 31. Código de herramienta de clasificación de las 10 especies menos y más vendidas. Fuente: autoría propia.

La herramienta de ventas totales obtiene las ventas totales por año y permite visualizar las ventas año con año y mes por mes, desplegando al usuario tablas y gráficas su debido análisis como se muestra en la Figura 32

```
def getDataVentas(order):
    cursor = connections['ornacoldb'].cursor()
    # Extraer el rango de años de los datos historicos
    query_years= "SELECT YEAR(pedido.p_fecha_pedido) FROM pedido GROUP BY YEA
    cursor.execute(query_years)
    query_years_result = cursor.fetchall()

    data={}

    cv=0

    for year in query_years_result:
        data_months=[]
        for month in range(1, 13):
            query_totals= 'SELECT SUM(detalle_pedido.dp_cantidad), SUM(detall
            cursor.execute(query_totals)
            query_totals_result = cursor.fetchall()
            if len(query_totals_result)==0:
                data_months.append(("0","0"))
            else:
                data_months.append((query_totals_result[0][0], query_totals_r


        query_totals= 'SELECT SUM(detalle_pedido.dp_cantidad), SUM(detalle_pe
        cursor.execute(query_totals)
        query_totals_result = cursor.fetchall()
        if len(query_totals_result)==0:
            total_plantas=0
            total_ventas=0
        else:
            total_plantas=query_totals_result[0][0]
            total_ventas=query_totals_result[0][1]

        data[cv]={
            'year':year[0],
            'data': data_months,
            'total_plantas': total_plantas,
            'total_ventas': total_ventas
        }
        cv+=1

    return data
```

Figura 32. Código de herramienta de ventas totales. Fuente: autoría propia.

La herramienta comparativo realiza un comparativo anual de las ventas por año y por mes. Esta herramienta permite al usuario comparar información y visualizarla mediante tablas y gráficas como se muestra en la Figura 33.

```
class getVentasTotalesComparativo(APIView):
    permission_classes = (permissions.IsAuthenticated,)

    def post(self, request, format=None):
        cursor = connections['ornacoldb'].cursor()

        query_years= "SELECT YEAR(pedido.p_fecha_pedido)
        cursor.execute(query_years)
        query_years_result = cursor.fetchall()
        data={}
        cv=0
```

Figura 33. Código de comparativo anual de ventas totales. Fuente: autoría propia.

La opción para generar reportes, permite al usuario generar reportes en formato .PDF y .XLSX apartir de ciertos parámetros, como se muestra en la Figura 34 y Figura 35

```
# Construir Excel
output = io.BytesIO()
# Feed a buffer to workbook
workbook = xlsxwriter.Workbook(output)

file_name= request.data['file_name']

if 'incluir_desc' in request.POST:
    subject= request.data['descripcion']
else:
    subject=""

fecha_splitted= request.data['fecha'].split("/");

workbook.set_properties({
    'title':    request.data['titulo'],
    'subject':  subject,
    'author':   'ORNACOL',
    'manager':   request.user.username,
    'company':  'ORNACOL',
    'category': 'ORNALISIS',
    'keywords': 'Ventas anuales',
    'created':  datetime.date(int(fecha_splitted[2]),int(fecha_splitted[1]),int(fecha_splitted[0])),
    'comments': 'Reporte generado desde ORNALISIS'
})

# Escribir el titulo del excel
titulo_style=workbook.add_format({'bold': True, 'font_size':20, 'align': 'center'})
subtitle_style=workbook.add_format({'bold': True, 'font_size':14, 'align': 'center', 'bg_color':'#CC6600' })
formato_cadena = workbook.add_format({'align':'center'})
formato_numero = workbook.add_format({'num_format': '0.00'})
formato_numero_total = workbook.add_format({'bold': True, 'num_format': '0', 'font_color':'#008000'})
formato_moneda = workbook.add_format({'num_format': '$#,##0.00', 'align':'center'})
formato_porcentaje = workbook.add_format({'num_format': '0.00%'})
formato_porcentaje_total = workbook.add_format({'bold': True, 'num_format': '0.00%', 'font_color':'#008000'})
formato_moneda_total = workbook.add_format({'bold': True, 'num_format': '$#,##0.00', 'font_color':'#008000'})
formato_moneda_total_centered = workbook.add_format({'bold': True, 'num_format': '$#,##0.00', 'align': 'center', 'font_color':'#008000'})

worksheet_especies = workbook.add_worksheet("Especies")
worksheet_especies.merge_range('A1:H1', str(request.data['titulo']), titulo_style)
worksheet_especies.write(1, 0, "")


worksheet_especies.set_column(0, 0, 10)
worksheet_especies.set_column(1, 1, 12)
worksheet_especies.set_column(2, 2, 20)
worksheet_especies.set_column(3, 3, 50)
worksheet_especies.set_column(4, 4, 10)
worksheet_especies.set_column(5, 5, 20)
worksheet_especies.set_column(6, 6, 20)
worksheet_especies.set_column(7, 7, 20)

row= 2
for key, value in tablas_por_proveedor.items():
    header = ["Proveedor","Código de proveedor","Código de planta", 'Descripción', "Cantidad", "Venta","Precio de compra", "Compra"]
    no_data = ["","","", '', "0", "0","$0", "$0"]
    # Llenar la fila de el header
    header_style = workbook.add_format({'bold': True, 'align': 'center', 'bg_color':'#FCD5B4'})
    header_style.set_text_wrap()
    header_style.set_align('vcenter')
    proveedor_style = workbook.add_format({'bold': True, 'align': 'center', 'font_size':16, 'font_color':'#0398E5'})
    proveedor_style.set_align('vcenter')
    proveedor_style.set_rotation(90)
    proveedor_style.set_text_wrap()
```

Figura 34. Código para generar documento en .PDF. Fuente: autoría propia.

```
function generarPDF(formData){
    var doc = new jsPDF("p","mm","letter");

    var finalY = doc.previousAutoTable.finalY

    var pageHeight = doc.internal.pageSize.height || doc.internal.pageSize.getHeight();
    var pageWidth = doc.internal.pageSize.width || doc.internal.pageSize.getWidth();

    // añadir logo
    var logo = new Image();
    logo.src = "/media/images/data_analisis.png";
    doc.addImage(logo, 'JPG',   0, 0, pageWidth, (pageHeight / 2) - 30 );

    let brand = "ORNALISIS";
    doc.setTextColor("#568c5a"); doc.setFontSize(18);
    doc.text(brand, (pageWidth / 2) +5,  12, 'center');

    logo.src = "/media/images/ORNALISISLOGO.png";
    doc.addImage(logo, 'JPG', (pageWidth / 2) -25, 7, 10, 5 );

    let title = formData["titulo"];
    doc.setTextColor("dodgerblue"); doc.setFontSize(20);
    doc.text(title, pageWidth / 2,  (pageHeight / 2) - 20, 'center');

    let periodo_txt = "Periodo:";
    doc.setTextColor("black"); doc.setFontSize(15);
    doc.text(periodo_txt, pageWidth / 2,  (pageHeight / 2), 'center');

    var fecha_inicial= $('input[name="fecha_inicial"]').val();
    var fecha_final= $('input[name="fecha_final"]').val()

    let fechas = "Del "+fecha_inicial+" al "+fecha_final;
    doc.setTextColor("black"); doc.setFontSize(15);
    doc.text(fechas, pageWidth / 2,  (pageHeight / 2) + 10, 'center');

    var lMargin=35; //left margin in mm
    var rMargin=35; //right margin in mm
    var pdfInMM=210;  // width of A4 in mm

    let socios_txt = "Socios:";
    doc.setTextColor("black"); doc.setFontSize(15);
    doc.text(socios_txt, pageWidth / 2,  (pageHeight / 2) + 25, 'center');

    //console.log(socios_c0)
    var lines_socios =doc.splitTextToSize(socios_c0, (pdfInMM-lMargin-rMargin));
    doc.setTextColor("gray"); doc.setFontSize(13);
    doc.text(lines_socios, pageWidth/2, (pageHeight / 2) + 35,  'center');

    if(formData['incluir_desc'] == "on"){
        let desc_txt = "Descripción:";
        doc.setTextColor("black"); doc.setFontSize(15);
        doc.text(desc_txt, pageWidth / 2,  (pageHeight / 2) + 75, 'center');
        let desc = formData['descripcion'];
        doc.setTextColor("gray"); doc.setFontSize(13);
        var lines_desc =doc.splitTextToSize(desc, (pdfInMM-lMargin-rMargin));
        doc.text(lines_desc, pageWidth/2, (pageHeight / 2) + 85,  'center');
    }

    today ="Generado el: "+formData['fecha'];
```

Figura 35. Código para la generación de reportes en formato .XLSX. Fuente: autoría propia.

4.3.2.2 Herramientas dinámicas

Debido a que se decidió implementar una alternativa que permita conectar cualquier base de datos relacional y realizar un sistema que por medio de herramientas HTML el usuario no experto pueda operar el sistema, se utilizó el algoritmo DFS capaz de encontrar las relaciones contenidas en la base de datos para construir consultas dinámicas complejas de *n* tablas y filtrar información de forma asíncrona utilizando Web APIs.

Se realizó una evaluación cualitativa del método para la determinación de relaciones para las herramientas dinámicas comparados con otros métodos tradicionales o de mayor complejidad como se observa en la Tabla 11, que cumplan con el objetivo de determinar la existencia de relaciones entre tablas de una BDR y construir consultas dinámicas.

Tabla 11. Tabla comparativa de métodos para la determinación de relaciones en BDRs. Fuente: autoría propia.

Método	Ventajas	Desventajas
Consultas SQL normales (Dobson, 2017)	Permiten crear consultas SQL rápidas y específicas, utilizando tablas y relaciones conocidas.	Se requiere conocer el esquema entidad-relación para construir varias consultas SQL para extraer la información solicitada.
Expresiones de tabla comunes (CTEs) recursivas (Kołodziejski, 2019)	Permiten crear consultas recursivas evaluando los resultados previos, por lo que pueden utilizarse para búsqueda de caminos en un grafo.	Se requiere conocer el esquema ER y es necesario generar una vista con la información de las relaciones entre las tablas para determinar los caminos.
Procesamiento de lenguaje natural, búsqueda de objetos (Yin et al., 2005)	Permite encontrar las mejores rutas de unión entre relaciones y predecir relaciones entre tablas no relacionadas.	Complejidad alta de implementación. Se requiere conocer el esquema ER.
DFS y métodos utilizados en esta investigación	- DFS Permite determinar relaciones en una BDR sin conocer sus tablas y relaciones. - Permite trabajar con múltiples BDRs. - Permite la creación de consultas SQL sin conocimiento sobre el lenguaje.	- Se requiere la utilización de BDRs para generar el grafo. - Las relaciones deben de estar definidas.

4.3.2.2.1 Construcción de consultas dinámicas

El sistema se encarga de procesar las peticiones que le son enviadas a través de la Web API mediante un conjunto de direcciones URL o endpoints que se encargan de recibir y enviar datos en formato JSON. Dependiendo de los elementos añadidos en la herramienta, se sigue un proceso que involucra algoritmos que construyen dinámicamente una consulta SQL entendible por el gestor de base de datos MySQL. A continuación, se presenta un diagrama de flujo (ver Figura 36) con la intención de visualizar el proceso de construcción de las consultas dinámicas.

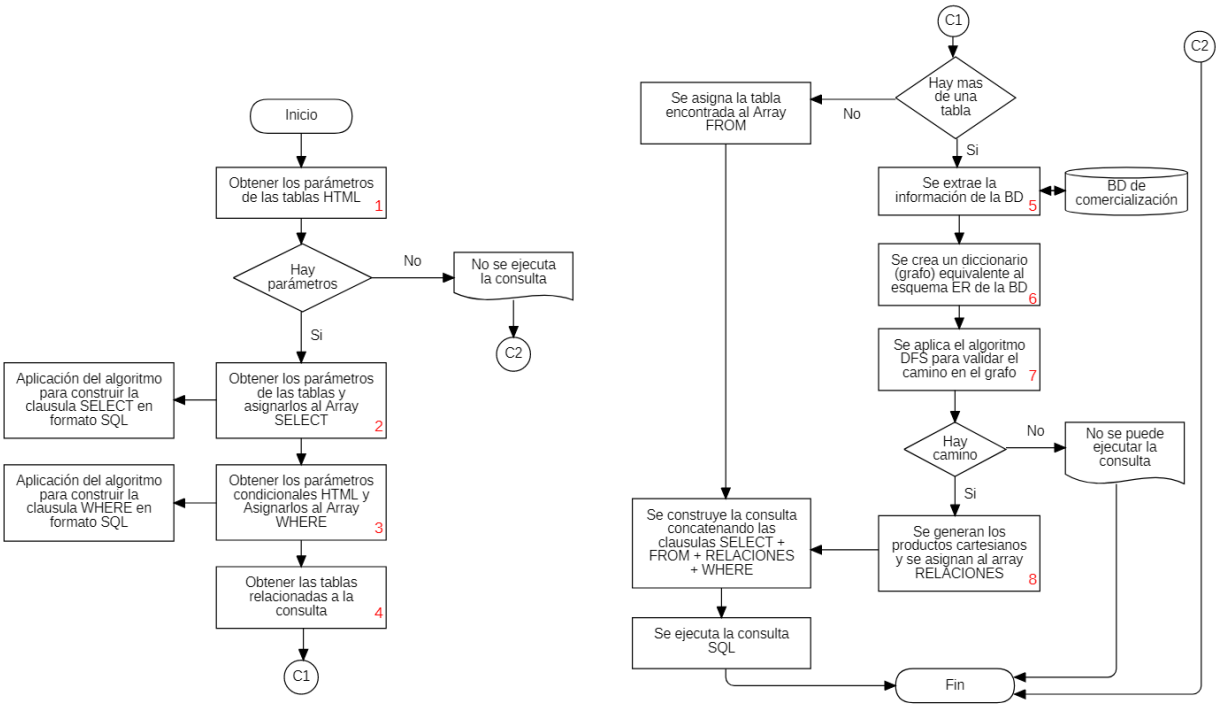

Figura 36. Diagrama de flujo para la creación de consultas dinámicas. Fuente: autoría propia.

Como se puede observar en el diagrama de flujo de la Figura 36, se lleva a cabo todo un proceso para la creación de consultas dinámicas, empezando por entender que elementos HTML equivalen a las instrucciones de SQL.

Equivalencia de elementos HTML a instrucciones SQL

A continuación, se muestra la explicación de lo que representan los elementos HTML de la interfaz de usuario, como tablas y gráficas, con sus respectivos parámetros a instrucciones SQL, esto debido a que se construyen consultas dinámicas utilizando elementos HTML que el usuario añade a la herramienta:

- **Tabla HTML:** equivale a la instrucción **SELECT** de SQL**,** la cual determina las columnas que se seleccionaran de cierta tabla. Las tablas aceptan solo **parámetros**, es decir, se deben añadir parámetros a la tabla para generar la instrucción SELECT, de lo contrario el resultado de la consulta será nulo.

  Cada parámetro trae información de la tabla a la que pertenece, el nombre de la columna y sus valores. Si se añaden tres parámetros(columnas), la variable SELECT almacenará el nombre de las tres columnas, asignándose a una variable de tipo Array de la siguiente manera:

  SELECT = “nombre_proveedor, planta, precio_venta”

- **Gráficas**: estas gráficas toman los parámetros de la tabla seleccionada y despliegan el resultado. Las gráficas aceptan parámetros y funcionan igual que las tablas, al modificar un valor, se realiza una consulta que filtra la información

solicitada y despliega los resultados en la gráfica.

- **Parámetros:** equivalen a la instrucción **WHERE,** la cual determina los filtros para la información, si el usuario modifica la información, ésta se actualiza y genera una nueva consulta. Si se añade el parámetro fecha y se selecciona un valor para el mismo, la variable WHERE, almacenará la columna y el valor seleccionado, asignándose como se ejemplifica a continuación:

  WHERE = "fecha BETWEEN '2018-01-01' AND '2018-12-01'"

  Los parámetros, como ya se mencionó anteriormente, pueden ser de varios tipos y estos equivalen a elementos y controles HTML (select, select multiple, input para texto, input para fecha, input para checkbox, etc.).

A partir de los parámetros y tablas, el sistema crea la instrucción **FROM** a través de la utilización del Array FROM, asignándose como se ejemplifica a continuación:

FROM = "proveedores, productos, ordenes"

Una vez entendido este proceso, se obtienen los parámetros de las tablas HTML (paso 1) que equivalen a la cláusula SELECT, si no hay parámetros la consulta no se puede ejecutar ya que no hay nada que seleccionar. Si hay parámetros estos se asignan al array de la cláusula SELECT mediante la ejecución del algoritmo para construir la cláusula SELECT en formato SQL (paso 2), de igual manera se obtienen los parámetros condicionales HTML y se asignan al array de la cláusula WHERE ejecutando el algoritmo para la construcción de la cláusula WHERE (paso 3).

En la Figura 37 se muestra una parte de la codificación de la construcción de las consultas dinámicas, específicamente la parte de la construcción de la cláusula SELECT y FROM para añadirlo a la consulta final a ejecutarse por la base de datos.

```
# Se obtienen los parametros de las tablas añadidas en la herramienta
tablas= Herramienta_tablas_visualizar.objects.filter(herramienta__pk= int(request.POST['id_herramienta']))

# Se inicializa diccionario de tablas detectadas por cada tabla con un array vacío
from_clause_dict= [ [] for i in range(tablas.all().count())]
select_clause_dict=[ [] for i in range(tablas.all().count())]

# Construir la clausula SELECT y FROM
i=0 # Contador tablas
for tabla in tablas.all():
    id_tabla=tabla.pk
    cp=0 # Contador de parametros
    table_parameters= tabla.parameters.all()
    for parametro in table_parameters:
        cp+=1
        if not parametro.table.name in from_clause_array:
            from_clause_array.append(parametro.table.name)
        # Si el nombre de la tabla no esta en el diccionario, se añade.
        if not parametro.table.name in from_clause_dict[i]:
            from_clause_dict[i].append(parametro.table.name)
        if cp == tabla.parameters.all().count():
            if parametro.parameter_type == 4:
                select_clause += "DATE_FORMAT("+parametro.table.name+"."+parametro.name+",'%Y/%m/%d')"
            else:
                select_clause += parametro.table.name+"."+parametro.name
        else:
            if parametro.parameter_type == 4:
                select_clause += "DATE_FORMAT("+parametro.table.name+"."+parametro.name+",'%Y/%m/%d'), "
            else:
                select_clause += parametro.table.name+"."+parametro.name+", "
    select_clause_dict[i]=select_clause
    select_clause="SELECT "
    i+=1
```

Figura 37. Código para la construcción de la cláusula SELECT y FROM. Fuente: autoría propia.

En la Figura 38 se muestra la codificación de la construcción de la cláusula WHERE para la consulta final a ejecutarse.

```
# Construir cláusula WHERE
cp=0 # Contador de parametros
for parameter, values in DataDict.items():
    if not parameter == "id_herramienta":
        cp+=1
        # Obtener parametros para la cláusula WHERE
        parameter_splitted= parameter.split("/")
        instance_parametro= get_object_or_404(Herramienta_tablas_parametros, pk= int(parameter_splitted[0]))
        if not instance_parametro.table.name in from_clause_array:
            from_clause_array.append(instance_parametro.table.name)
        if not instance_parametro.table.name in from_clause_array_charts:
            from_clause_array_charts.append(instance_parametro.table.name)
        for i in range(len(from_clause_dict)):
            if not instance_parametro.table.name in from_clause_dict[i]:
                from_clause_dict[i].append(instance_parametro.table.name)
        for i in range(len(from_clause_charts_dict)):
            if not instance_parametro.table.name in from_clause_charts_dict[i]:
                from_clause_charts_dict[i].append(instance_parametro.table.name)
        if cp>1:
            where_clause += " AND ("
        if  instance_parametro.parameter_type == 4 or instance_parametro.parameter_type == 6:
            where_clause += instance_parametro.table.name+"."+parameter_splitted[1]+" BETWEEN "
            cv=0 # Contador de valores
            for value in values:
                cv+=1
                if cv == len(values):
                    where_clause += "\""+value+"\")"
                else:
                    where_clause += "\""+value+"\""+" AND "
        elif instance_parametro.parameter_type == 2:
            where_clause += instance_parametro.table.name+"."+parameter_splitted[1]+" = "
            cv=0
            for value in values:
                cv+=1
                if cv == len(values):
                    where_clause += "\""+value+"\")"
                else:
                    where_clause += "\""+value+"\""+" OR "+ instance_parametro.table.name+"."+parameter_splitted[1]+" = "
```

Figura 38. Código para la construcción de la cláusula WHERE. Fuente: autoría propia.

Hasta aquí se obtendría una consulta construida sin relaciones. Si se ejecuta esta consulta, es muy probable que los resultados resulten erróneos o se consulte información innecesaria para el análisis; para ello, se requiere un mecanismo para determinar las relaciones entre las tablas encontradas en la construcción de la consulta para complementar la consulta final a ejecutarse en la base de datos, pero no sin antes conocer la información interna de la base de datos para poder extraer las relaciones, que serán necesarias para la conversión de la base de datos a grafo.

Como se puede observar, a partir de cada elemento **HTML** que el usuario añada a la herramienta, se genera una instrucción SQL que va concatenándose en una cláusula **SELECT** o clausula **WHERE** y a partir de ahí se genera la cláusula **FROM** que son las tablas que se necesitan para que la consulta se ejecute correctamente (paso 4).

Es importante mencionar que si después de la construcción de la cláusula **SELECT**, **FROM** y **WHERE** no se detectaron más de una tabla para la consulta, la consulta final es el resultante de la concatenación de esas tres clausulas, debido a que la consulta con una sola tabla no requiere relaciones y por lo tanto puede ejecutarse tal cual. Si existen más de una tabla se continua con el paso 5 mostrado a continuación.

Extracción de información de la BD

Independientemente del lenguaje que se utilice es necesario, después de establecer conexión con la base de datos, realizar una consulta SQL a la BD para extraer la información interna (tablas y relaciones) como se ilustra en la Figura 39, debido a que se desconoce a priori la estructura de ésta.

```
"SELECT TABLE_NAME, COLUMN_NAME,
CONSTRAINT_NAME, REFERENCED_TABLE_NAME,
REFERENCED_COLUMN_NAME FROM
INFORMATION_SCHEMA.KEY_COLUMN_USAGE WHERE
TABLE_SCHEMA='BASEDEDATOS'"
```

Figura 39. Consulta SQL para extracción de tablas y relaciones de una base de datos. Fuente: autoría propia.

Como se observa en la Figura 39, se crea una consulta SQL que será de utilidad para determinar el esquema ER de la BD, donde se selecciona información como: nombres de tablas y columnas, llaves, referencias a tablas y columnas asociadas.

Creación de diccionarios equivalentes a la BD

Como se mencionó anteriormente es necesario conocer la información interna de la BD, y a partir de ésta, crear una representación de la BDR mediante un grafo (paso 6), ya que éste es requerido por el algoritmo DFS para buscar la ruta (relación) entre dos nodos (tablas) en el grafo (representación de la BDR).

Un grafo es conjunto de nodos unidos por aristas que permiten representar relaciones entre sí (Véase Figura 40), justamente lo que se necesita para representar la estructura de una BDR.

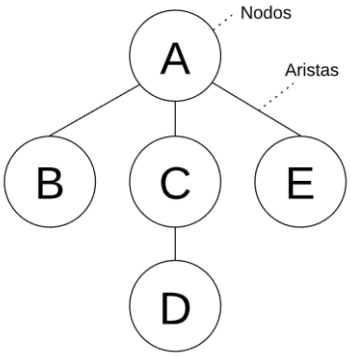

Figura 40. Representación de un grafo con nodos y aristas. Fuente: autoría propia.

Como se muestra en la Figura 40, el grafo cuenta con Nodos y aristas, donde cada nodo será reemplazado por una tabla de la base de datos y cada artista representará la relación entre las tablas.

Se implementó una lista de adyacencia utilizando un diccionario de Python, en donde la llave representa el nodo y el valor es un arreglo que hace referencia a los otros nodos a los que está conectado, permitiendo determinar las relaciones entre éstos.

Para comprender, el proceso anterior considere una BDR conformada por 5 tablas: pedidos, clientes, detalle pedido, vendedores y productos (Véase Figura 41).

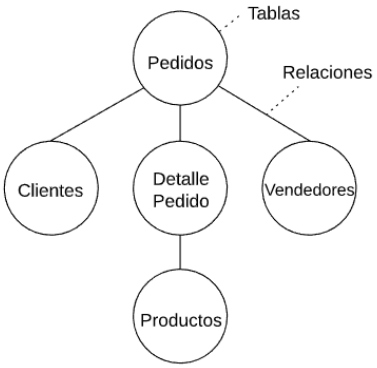

Figura 41. Representación de una base de datos en grafo. Fuente: autoría propia.

El grafo de la BDR quedaría representado mediante una lista de adyacencia en Python utilizando diccionarios como se ilustra en la Figura 42.

```
grafo = {
    "Clientes":["Pedidos"],
    "Pedidos":["Clientes","DetallePedido",
               "Vendedores"],
    "DetallePedido":["Pedidos","Productos"],
    "Productos":["DetallePedido"],
    "Vendedores":["Pedidos"]
   }
```

Figura 42. Representación de grafo en diccionario Python. Fuente: autoría propia.

El diccionario presentado en la Figura 42 sólo sirve como un ejemplo de la representación del grafo implementando una lista de adyacencia mediante diccionarios en Python, estableciendo etiquetas numéricas (Véase Figura 43).

```
grafo_num = {
    "1":["2"],
    "2":["1","3","5"],
    "3":["2","4"],
    "4":["3"],
    "5":["2"]
   }
```

Figura 43. Representación de grafo numérico en diccionario Python. Fuente: autoría propia.

Como se observa en la Figura 43, se le asignó un número a cada tabla, debido a que es necesario pasar un grafo numérico a la implementación del algoritmo DFS presentado en la sección de Resultados, ya que esta implementación en específico requiere etiquetas numéricas para funcionar correctamente.

En la Figura 44 se muestra la función para mapear la base de datos a grafos, mismos que son necesarios para el funcionamiento del algoritmo DFS.

```python
# Mapear la base de datos relacional a grafos
def get_grafos():
    graph= {}
    graph1={}
    graph2={}
    graph2_inverse={}
    queryt= "SELECT TABLE_NAME, COLUMN_NAME, CONSTRAINT_NAME, REFERENCED_TABLE_NAME, REFERENCED_COLUMN_NAME FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE WHERE TABLE_SCHEMA='"
    connection= connections[settings.ACTIVEBDALIAS].cursor()
    connection.execute(queryt)
    relationst = connection.fetchall()
    relationst_primary= relationst
    for relationt in relationst:
        list_values= []
        if not str(relationt[2])=="PRIMARY":
            if str(relationt[0]) in graph1:
                if type(graph1.get(relationt[0])) is set:
                    set_vals= graph1.get(relationt[0])
                    list_values.append(relationt[3])
                    set_vals.update(set(list_values) )
                    graph1.update({str(relationt[0]): set_vals})
                else:
                    list_values.append(graph1.get(relationt[0]))
                    list_values.append(relationt[3])
                    graph1.update({str(relationt[0]): set(list_values)})
            else:
                graph1.update({str(relationt[0]) : str(relationt[3])})
        else:
            list_values1= []
            for relationt_p in relationst_primary:
                if str(relationt[0]) == str(relationt_p[3]):
                    if not str(relationt[0]) == str(relationt_p[0]):
                        list_values1.append(str(relationt_p[0]))
            if not len(list_values1)==0:
                if len(list_values1)>1:
                    graph1.update({str(relationt[0]): set(list_values1)})
                else:
                    graph1.update({str(relationt[0]): str(list_values1[0])})
    # Igualar nombre de tabla a un numero para facilitar la busqueda dfs
    connection.execute("SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_SCHEMA='"+settings.ACTIVEBD+"'  ")
    tables = connection.fetchall()
    count=0
    for table in tables:
        graph2.update({str(table[0]): count})
        count += 1
    count=0
    for table in tables:
        graph2_inverse.update({str(count): str(table[0])})
        count += 1
    # Iterar los valores del grafo con nombres en string de las tablas para
    # crear un grafo con solo un caracter '1' '2' etc
    for key, values in graph1.items():
        list_values= []
        if type(values) is set:
            for value in values:
                list_values.append(str(graph2.get(value)))
            graph.update({str(graph2.get(key)): set(list_values)})
        else:
            graph.update({str(graph2.get(key)): {str(graph2.get(values))}})
    return graph, graph1, graph2, graph2_inverse, relationst
```

Figura 44. Código de la función para mapear la base de datos relacional a grafo. Fuente: autoría propia.

Aplicación del algoritmo DFS y determinación de relaciones

Una vez obtenido el grafo numérico, el algoritmo DFS va a buscar en cada nodo la existencia de relaciones y para verificar si existe algún camino por el cual se pueda relacionar la información (paso 7). La metodología para la determinación de las relaciones entre las tablas de una base de datos se describe en el diagrama de flujo de la Figura 45 y Figura 46.

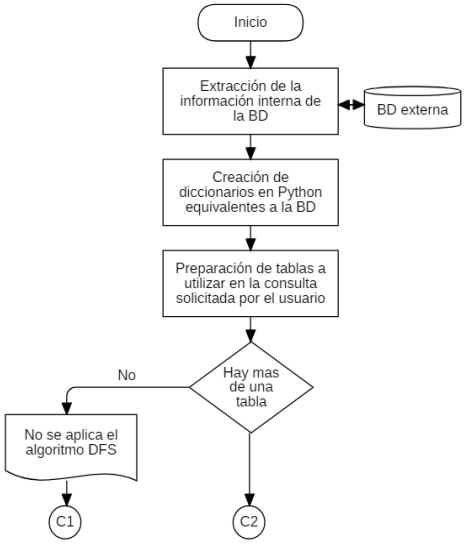

Figura 45. Diagrama de flujo para la búsqueda y validación de relaciones en BDRs con DFS. Fuente: autoría propia.

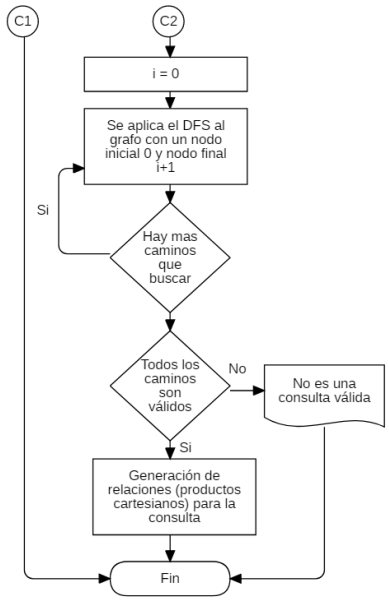

Figura 46. Continuación del diagrama de flujo para la búsqueda y validación de relaciones en BDRs con DFS. Fuente: autoría propia.

Como se puede observar en la Figura 45 y Figura 46, el algoritmo DFS se ejecutará siempre y cuando exista más de una tabla en la consulta solicitada por el usuario, determinando la existencia de caminos entre las tablas y validando la ejecución de la consulta final. Si alguno de los caminos no es válido, termina el proceso y la consulta no se ejecuta.

Para verificar si existe algún camino por el cual se pueda relacionar la información, el algoritmo DFS toma como valores de entrada los siguientes parámetros: 1) el grafo con etiquetas numéricas equivalente a la base de datos; 2) el nodo inicial (por el cual empezará la búsqueda) y 3) el nodo final o de destino. Esto se logra en Python implementando una función que recibe

3 parámetros importantes, un grafo, un punto inicial y uno final (véase Figura 47 y Figura 48. El algoritmo siempre toma dos tablas del array de tablas que utiliza la consulta y les asigna un punto inicial y un punto final. No importa el orden y no tienen un peso, solamente importa obtener si existe relación entre esas dos tablas.

```
# utilizando Depth-First Search para encontrar el camino entre dos tablas y
# crear las relaciones
paths_list= []
for i in range(0, len(tables_in_number)-1):
    paths_list.append(find_path(graph_numeros, str(tables_in_number[0]), str(tables_in_number[i+1])))
```

Figura 47. Código de función que pide tres parámetros para encontrar el camino entre dos tablas.

```
# Función para encontrar el camino entre dos tablas
def find_path(graph, start, end, path=[]):
    path = path + [start]
    if start == end:
        return path
    if start not in graph:
        return None
    for node in graph[start]:
        if node not in path:
            newpath = find_path(graph, node, end, path)
            if newpath: return newpath
    return None
```

Figura 48. Algoritmo DFS para encontrar el camino entre dos puntos. Fuente (Python Software Foundation, 2019).

Una vez obtenidos los caminos válidos, el algoritmo DFS selecciona la ruta más corta con lo que se determinarán las relaciones entre las tablas involucradas de la consulta final. Para ello, solo hace falta crear los productos cartesianos correspondientes a cada tabla (paso 8). En SQL un producto cartesiano o unión cruzada es la unión de información de dos tablas donde las columnas de una tabla se unen con otra (w3resource, 2020). Para esto, un ciclo en Python construye los productos cartesianos requeridos, regresando un array de relaciones válidas para ejecutar la consulta del ejemplo abordado. El array de las relaciones resultantes se muestra en la Figura 49.

```
relaciones = [
    "DetallePedido.IDproducto=Productos.IDproducto",
    "Pedidos.IDpedido=DetallePedido.IDpedido",
    "Clientes.IDcliente=Pedidos.IDcliente"
]
```

Figura 49. Array de relaciones válidas para la consulta final. Fuente: autoría propia.

Lo anterior se traduce de la siguiente manera:

1. La tabla **1** se relaciona con la **2**:
   Clientes.IDcliente = **Pedidos**.IDcliente
2. La tabla **2** se relaciona con la **3**:
   Pedidos.IDpedido=**DetallePedido**.IDpedido
3. La tabla **3** se relaciona con la **4**: **DetallePedido**.IDproducto=**Productos**.IDproducto

Posteriormente, los tres productos cartesianos resultantes se integran a la consulta final del sistema web de apoyo a la toma de decisiones y se añaden a la cláusula WHERE para terminar la construcción de la consulta dinámica como se muestra en la Figura 50.

```
SELECT nombreCliente, nombreProducto
FROM Clientes, Pedidos, DetallePedido, Productos
WHERE Clientes.IDcliente = Pedidos.IDcliente AND
Pedidos.IDpedido = DetallePedido.IDpedido AND
DetallePedido.IDproducto = Productos.IDproducto
```

Figura 50. Uniones cruzadas agregadas a la consulta SLQ. Fuente: autoría propia.

El código del proceso para construir la cláusula WHERE y el diagrama de flujo de la Figura 45 y Figura 46 para determinar las relaciones entre las tablas, se muestra en la Figura 47.

```
# Determinar relaciones entre tablas
where_relationship_tables_dict= [ [] for i in range(tablas.all().count())]
# Si la longitud no es mayor a 1, significa que es solo 1 tabla, por lo tanto no se buscan relaciones
c_fca=0 # Contador para agregar un AND o no
c_fca_c=0 # Contador para agregar un AND o no
# Iterar por las tablas
for j in range(len(from_clause_dict)):
    if len(from_clause_dict[j]) > 1:
        # Obtener grafos y diccionarios
        graph_numeros, graph_nombres, dict_tabla_numero, dict_numero_tabla, relations = get_grafos()
        tables_in_number=[]
        for element in from_clause_dict[j]:
            if element in graph_nombres:
                tables_in_number.append(dict_tabla_numero.get(element))

        # Utilizando Depth-First Search para encontrar el camino entre dos tablas y crear las relaciones
        paths_list= []
        for i in range(0, len(tables_in_number)-1):
            paths_list.append(find_path(graph_numeros, str(tables_in_number[0]), str(tables_in_number[i+1])))

        paths_list_names=[]
        # Crear diccionario de nombres de las tablas relacionadas
        for path in paths_list:
            for i in range(0, len(path)):
                paths_list_names.append(dict_numero_tabla.get(path[i]))
        # Extraer relaciones
        for relation in relations:
            if relation[2] != "PRIMARY":
                if relation[0] in paths_list_names:
                    if relation[3] in paths_list_names:
                        first_table= relation[0]
                        second_table= relation[3]
                        relation= " "+first_table+"."+relation[1]+"="+second_table+"."+relation[4]
                        if not relation in where_relationship_array_tables:
                            where_relationship_array_tables.append(relation)
                        if not first_table in from_clause_dict[j]:
                            from_clause_dict[j].append(first_table)
                        if not second_table in from_clause_dict[j]:
                            from_clause_dict[j].append(second_table)
        where_relationship_tables_dict[j] = where_relationship_array_tables
    else:
        where_relationship_tables_dict[j]= ""
```

Figura 51. Código para la determinación de relaciones en bases de datos relacionales. Fuente: autoría propia.

Finalmente se verifica que los datos extraídos coincidan con la consulta construida por el usuario y se despliega la información en la vista del cliente mediante tablas y gráficas.

4.3.3 Pruebas

Siguiendo el diagrama de pruebas FLOOT de la Figura 5 en el capítulo 3 en la sección de metodología de desarrollo de software, en la diciplina de pruebas de la fase de iniciación, se llevaron a cabo las pruebas de código, las cuales se describen en la Tabla 12 Estas pruebas validaron el funcionamiento de las funciones del sistema, para verificar que todo vaya conforme a lo planeado.

Tabla 12. Pruebas de Código hechas en el sistema. Fuente: autoría propia.

Prueba	Actividades
Pruebas de caja negra	Mediante una serie de entradas se pusieron a prueba las siguientes funciones del sistema: **Módulo de herramientas dinámicas**: • Validar API utilizando la aplicación Postman para realizar peticiones GET y POST • Validar funciones del front-end mediante Ajax enviando valores en JSON de los parámetros, tablas y gráficas. • Validar opciones de configuración de los reportes .PDF y .xlsx **Módulo de herramientas integradas:** • Validar funciones del front-end mediante Ajax, enviando valores en JSON de los parámetros, regresando un array con los datos extraídos y desplegados correctamente en el cliente. • Validar opciones de configuración de los reportes .PDF y .xlsx **Módulo de procesamiento y acoplamiento de datos:** • Se validaron las entradas enviadas por el cliente para extraer la construcción de consultas dinámicas mediante el algoritmo de construcción de consultas y el algoritmo DFS para determinación de relaciones.
Pruebas de caja blanca	Se valido cada línea de código programada y se documentó para futuras actualizaciones.
Prueba de valores de frontera	Se realizaron algunas pruebas con las entradas validando que los valores no se desbordaran y causaran problemas. El sistema puede soportar los parámetros necesarios y regresar hasta 300 mil datos para ser desplegados en tablas gráficas.
Prueba de integración de clases	Se valido que las clases e instancias funcionen según lo definido.
Revisión de código	En este caso el código fue revisando constantemente hasta la etapa final.
Prueba de caminos	Se validaron las rutas lógicas del código.

4.3.4 Despliegue

Se hicieron pruebas de despliegue del sistema en preproducción en el servidor final y se realizaron los últimos ajustes; Además, se inició con la escritura de la documentación del sistema.

Figura 52. Página principal de ORNALISIS. Fuente: autoría propia.

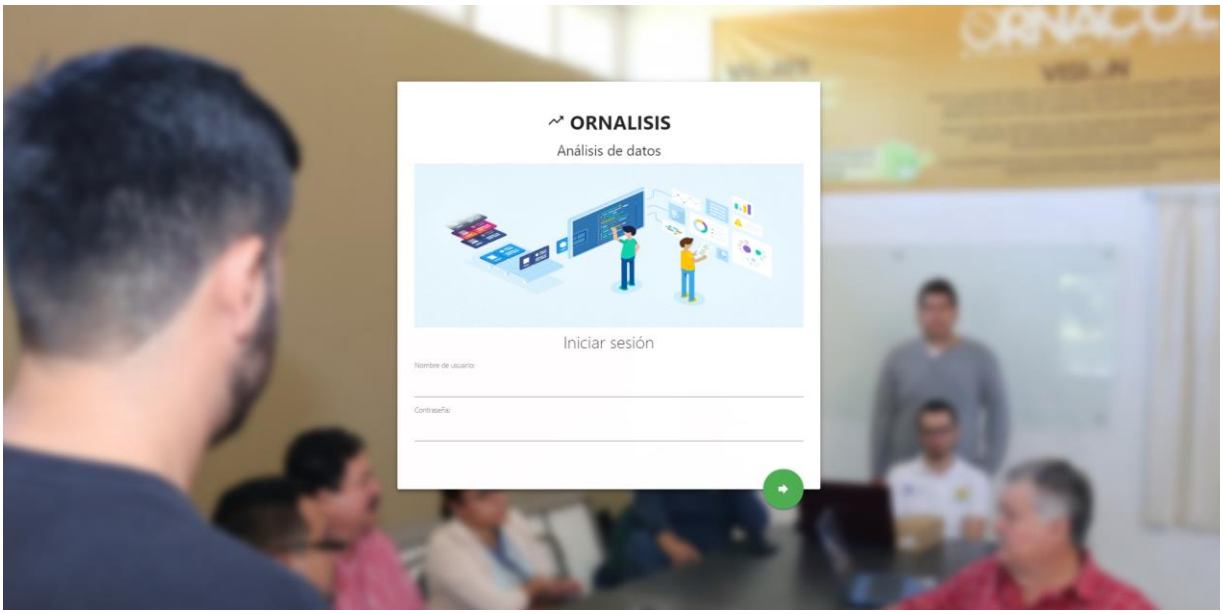

Figura 53. Inicio de sesión de ORNALISIS. Fuente: autoría propia.

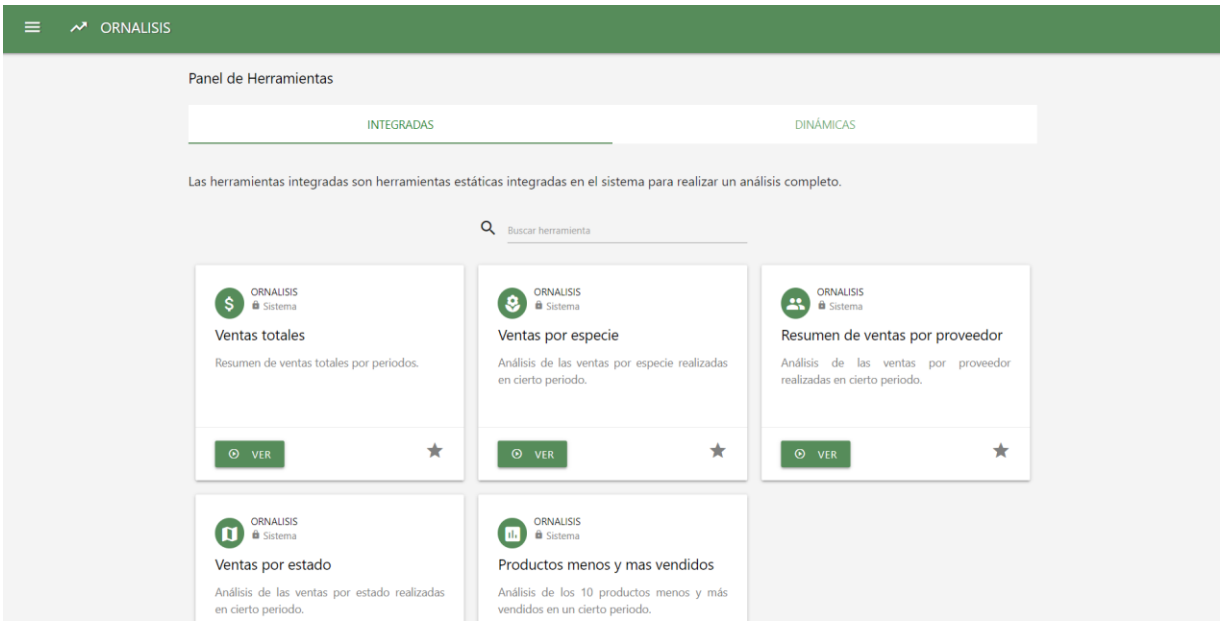

Figura 54. Panel de Herramientas de ORNALISIS. Fuente: autoría propia.

En la Figura 52, Figura 53 y Figura 54 se muestran los prototipos de las vista finales codificadas en HTML, desplegadas en el servidor final tal cual se planeó.

4.3.5 Gestión de configuración

Se verificó que todo fuera acorde a lo planeado en la fase de elaboración.

4.3.6 Gestión de proyectos

Se verificó que todo fuera acorde a lo planeado en la fase de elaboración; además, se mitigaron los riesgos del listado de riesgos implementando el plan de la etapa anterior y se comunicó a los operadores del sistema.

- **Nuevas versiones del framework Django**: mitigado.
- **Sistema para personas no expertas en el área:** mitigado.
- **Cambio en los requisitos funcionales y no funcionales:** mitigado.
- **No utilización de base de datos:** mitigado.
- **Rendimiento y despliegue:** mitigado.

4.3.7 Medio ambiente

Se realizó la búsqueda de herramientas y algoritmos que pudieran resolver los requisitos no funcionales del sistema. Además, se configuró el entorno para el despliegue del sistema en su etapa final.

4.4 Fase de transición

Se establecieron las pautas para la puesta en marcha del sistema, en las cuales se incluye el despliegue del sistema en el servidor, actualización de componentes, pruebas preliminares a la entrega, capacitación del personal y manual de usuario.

4.4.1 Modelo

Se elaboró el manual de usuario, mismo que puede consultarse en el apéndice 1.

4.4.2 Implementación

Se realizaron las Implementaciones finales al código.

4.4.3 Pruebas

Siguiendo el diagrama de pruebas FLOOT de la Figura 5 en el capítulo 3 en la sección de metodología de desarrollo de software, en la diciplina de pruebas de la fase de iniciación, se llevaron a cabo las pruebas de usuario para verificar el funcionamiento del sistema en producción, las cuales se describen en la Tabla 13 y Tabla 14.

Tabla 13. Pruebas de análisis hechas en el sistema. Fuente: autoría propia.

Prueba	Actividades
Pruebas funcionales y de operación	Diseño de las pruebas que permitan detectar errores en el despliegue del sistema web. Realización de las pruebas funcionales previas a la entrega: 1. Verificación del funcionamiento del sistema en el cliente del operador. 2. Conexión de los clientes con la base de datos en el servidor. Verificación de la extracción de información mediante consultas dinámicas.
Pruebas de instalación y soporte	Se validó la instalación del software, la instalación de las dependencias y bases de datos en el servidor.
Prueba de estrés	Se validó que el sistema funcione como se espera en grandes volúmenes de transacciones, usuarios, carga, peticiones y de extracción de información utilizando bases de datos en la nube con más de 100 mil datos.

Tabla 14. Pruebas de usuario hechas en el sistema. Fuente: autoría propia.

Prueba	Actividades
Prueba beta	Se realizaron las últimas pruebas para validará la versión final del software.
Pruebas piloto	Se validó el funcionamiento realizando pruebas directamente con el usuario final, en donde se tomaron nota de las ultimas observaciones y se pusieron a pruebas los módulos del sistema. A continuación, se describe el plan de las pruebas piloto: • Presentación del sistema al operador. • Inicio de sesión.

	• Navegar libremente por el sistema. • Ir al panel de herramientas integradas y visualizar una herramienta. • Ir al panel de herramientas dinámicas, crear una herramienta y configurar las tablas. • Entender panel de elementos de las herramientas dinámicas. • Entender los permisos del sistema. • Generar reportes .PDF y .xlsx. • Cerrar sesión.
Prueba de aceptación de usuario	Se realizó mediante el cuestionario la escala de usabilidad del sistema (pro sus siglas en ingles SUS) y el modelo TAM, donde los usuarios dieron visto bueno al sistema y entendieron todo lo explicado.

4.4.4 Despliegue

Se llevaron a cabo ciertas actividades para verificar el correcto despliegue del sistema en el servidor, las cuales se muestran a continuación:

1. Despliegue del sistema en el servidor ***Namecheap***.
2. Actualización de los componentes en el servidor.
3. Creación de la base de datos en el servidor.
4. Carga de información de ventas en el servidor.
5. Vinculación de base de datos de comercialización con el sistema web.

4.4.5 Gestión de configuración

Se verificó que todo fuera acorde a lo planeado en la fase de construcción.

4.4.6 Gestión de proyecto

Se establecieron las actividades de adiestramiento a realizar para los operadores del sistema, las cuales se describen en la Tabla 15.

Tabla 15. Actividades de adiestramiento para los operadores. Fuente: autoría propia.

Tema	Actividades	Usuarios involucrados
Introducción al software	• Presentación de ultimas actualizaciones al software. • Presentación de los módulos del software. • Funcionamiento general del software.	Todos los miembros de la sociedad de producción rural ORNACOL.
Utilización de autenticación y permisos.	• Iniciar sesión. • Registro de usuarios. • Opciones de usuarios.	Todos los miembros de la sociedad de producción rural ORNACOL.

Utilización de Módulo herramientas integradas	• Utilización de parámetros. • Visualización de tablas y gráficas.	Operador del sistema.
Utilización de Módulo de herramientas dinámicas	• Utilización de panel de elementos. • Utilización de parámetros. • Visualización de tablas y gráficas.	Operador del sistema.
Generación de reportes	• Generar reportes en PDF. • Generar reportes .xlsx. • Configuración de reportes.	Operador del sistema.
Utilización de administración.	• Panel de administración de tablas de la base de datos.	Operador del sistema.

4.4.7 Medio ambiente

Finalizar desarrollo del proyecto y reacomodar el entorno de trabajo y las herramientas utilizadas para otros proyectos.

CAPÍTULO 5. RESULTADOS OBTENIDOS

En este capítulo se realiza la evaluación del diseño DSS y el del desempeño del sistema web desarrollado tomando tiempos de respuesta en la realización de consultas siguiendo la arquitectura cliente servidor; además, se compara el tiempo de procesamiento de datos manuales con la utilización del sistema para la generación de reportes periódicos.

5.1 Resultados del desarrollo del sistema

En esta sección se muestran los resultados del sistema web, dividido en herramientas integradas, dinámicas y generación de reportes.

5.1.1 Herramientas integradas

Las herramientas integradas conforman una parte del sistema web, un módulo de herramientas que cumplen los requisitos funcionales de ORNACOL. El usuario puede visualizar información de ventas en un periodo de tiempo por proveedor, especie, estado, totales, especies menos y más vendidas, un comparativo de ventas anuales y mensuales y generación de reportes periódicos.

- **Herramienta de ventas por proveedor**: su objetivo es visualizar información de las ventas por proveedor o socio en un periodo de tiempo. La vista cuenta con una tabla y dos gráficas y se desarrolló tal cual se presentaba en hojas de cálculo (véase Figura 55). Puede generarse un reporte específicamente para esta herramienta.

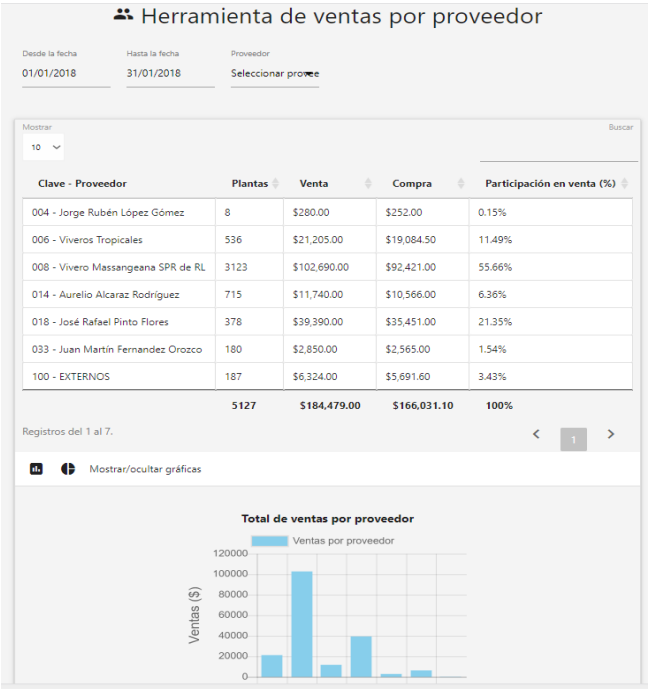

Figura 55. Vista de herramienta de ventas por proveedor. Fuente: autoría propia.

- **Herramienta de ventas por especie:** su objetivo es visualizar información de ventas por especie de plantas en la cual se incluye un resumen de ventas por especie por proveedor y cada uno cuenta con una tabla y grafica para facilitar el análisis al usuario. Puede generarse un reporte específicamente para esta herramienta.

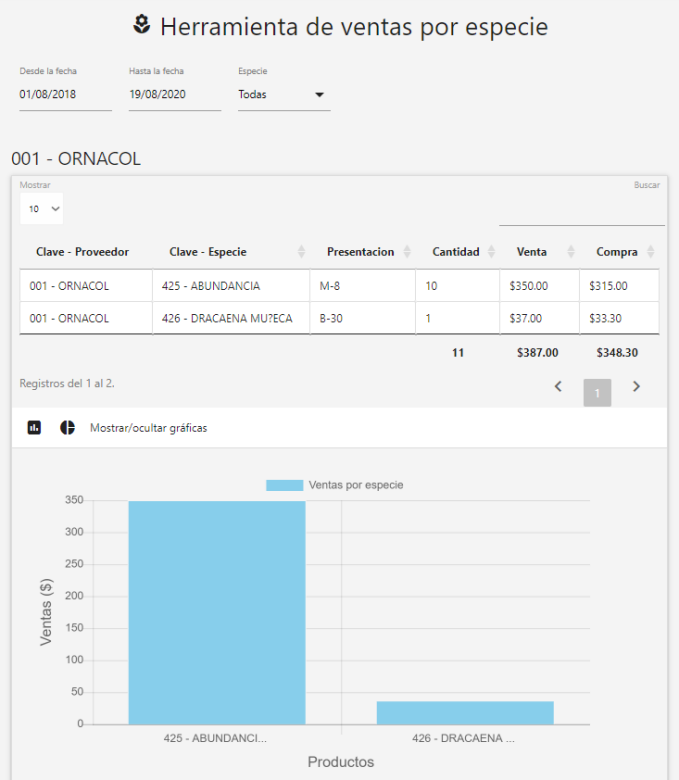

Figura 56. Visa de herramienta de ventas por especie. Fuente: autoría propia.

- **Herramienta de ventas por estado:** su objetivo es desplegar el total de ventas por estado mostrando un mapa con una paleta de colores en azul que indica con color más intenso el estado con más ventas, de igual manera se muestra una gráfica y una tabla para facilitar su análisis. Puede generarse un reporte específico para esta herramienta.

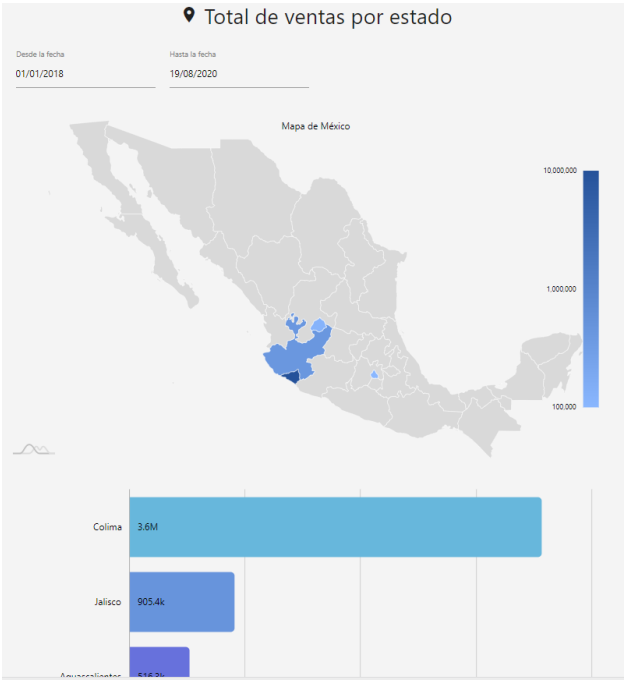

Figura 57. Vista de herramienta de ventas por estado. Fuente: autoría propia.

- **Herramienta de clasificación de las 10 especies menos y más vendidas:** su objetivo es desplegar las especies menos y más vendidas con su total en un periodo de tiempo.

Figura 58. Vista de clasificación de especies menos y más vendidas. Fuente: autoría propia.

- **Herramienta de ventas totales:** su objetivo es visualizar las ventas totales por año y mes, mostrando una tabla y gráficas para facilitar el análisis. Puede generarse un reporte específico para esta herramienta.

Figura 59. Vista de herramientas de ventas totales. Fuente: autoría propia.

- **Herramienta de comparativo de ventas totales:** su objetivo es visualizar un comparativo de las ventas totales por año y por mes. Puede incluirse en el reporte de ventas totales.

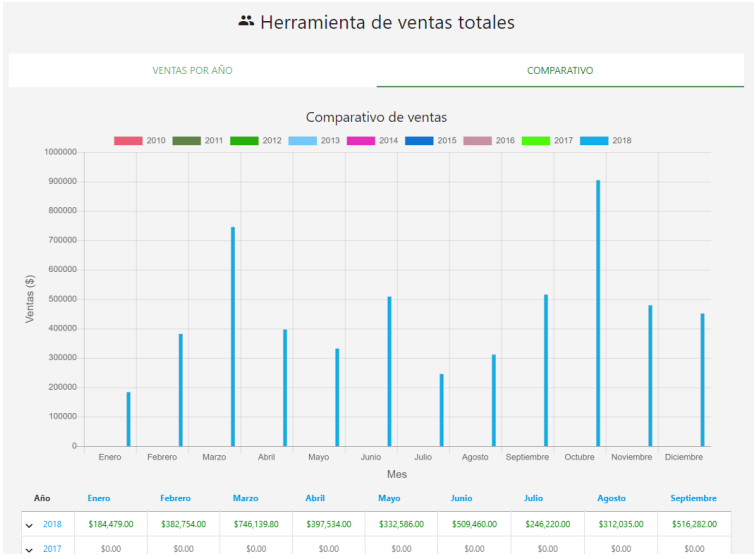

Figura 60.Vista de comparativo de ventas totales. Fuente: autoría propia.

- **Herramienta de generación de reportes:** su objetivo es generar documento en formato .PDF y .XLSX, facilitando varias opciones de configuración al usuario para generar un reporte a su necesidad.

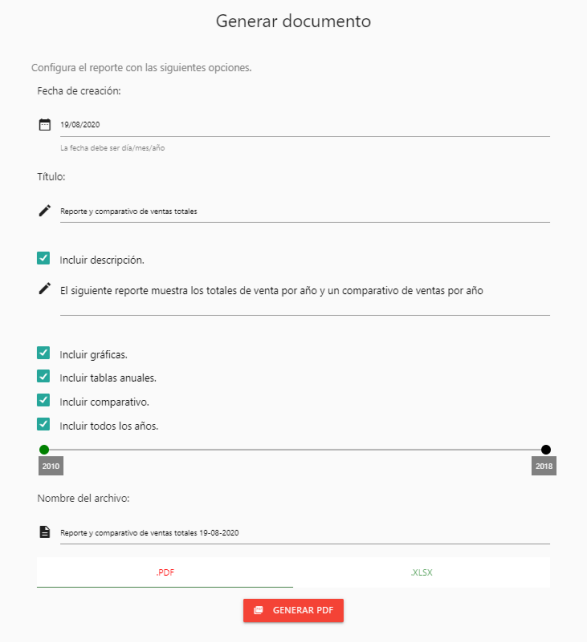

Figura 61. Vista para generar reportes de ventas en .PDF y .XLSX. Fuente: autoría propia.

5.1.2 Herramientas dinámicas

Las herramientas dinámicas conforman una parte del sistema web, un módulo de herramientas que permiten al usuario realizar consultas especificas a la base de datos. El usuario puede visualizar información de cualquier tipo, aplicar filtros para obtener lo que necesita y añadir cuantas tablas y gráficas necesite para apoyar a la toma de decisiones.

- **Panel de herramientas dinámicas:** su objetivo es la gestión de herramientas dinámicas permitiendo crear, editar, visualizar y borrar una herramienta; además, puedes compartir la herramienta con los demás usuarios para que tengan acceso a la información. Figura 62.

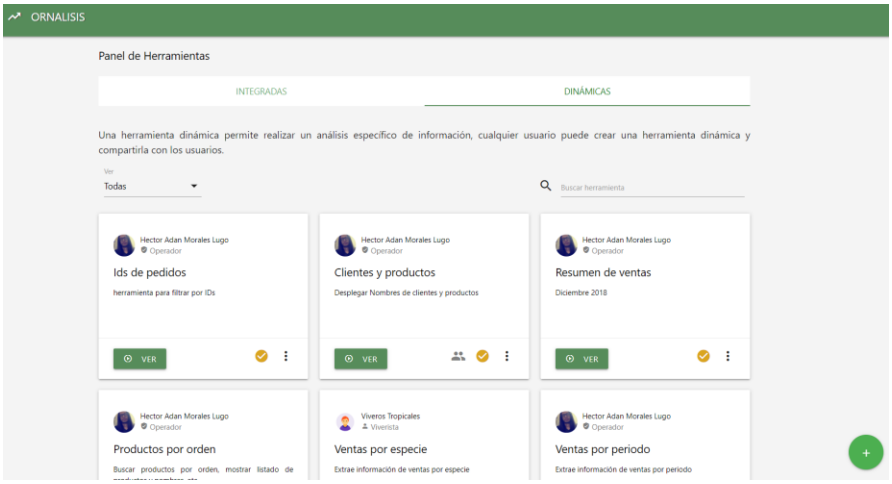

Figura 62. Panel de herramientas dinámicas. Fuente: autoría propia.

- **Detalle de una herramienta dinámica:** al crear una herramienta, se despliega el modal para seleccionar las tablas de la base de datos seleccionada que se van a utilizar para el análisis (Figura 63).

Figura 63. Modal principal de selección de tablas. Fuente: autoría propia.

- Después de ello las tablas y sus campos se despliegan dentro del detalle de la herramienta en un panel a la derecha en forma de parámetros; además los elementos de visualización como gráficas y tablas y la configuración de los parámetros (Figura 64).

Figura 64. Panel lateral de elementos. Fuente: autoría propia.

- En la vista de detalle de una herramienta dinámica de la Figura 65 puedes arrastrar los elementos a la vista utilizando la función *drag and drop* del navegador y eliminar cualquier elemento utilizando el clic izquierdo del ratón. En el detalle puedes añadir

tablas y a las tabas añadirles parámetros al igual que removerlos, pues añadir filtros y puedes añadir distintos tipos de gráficas para la visualización.

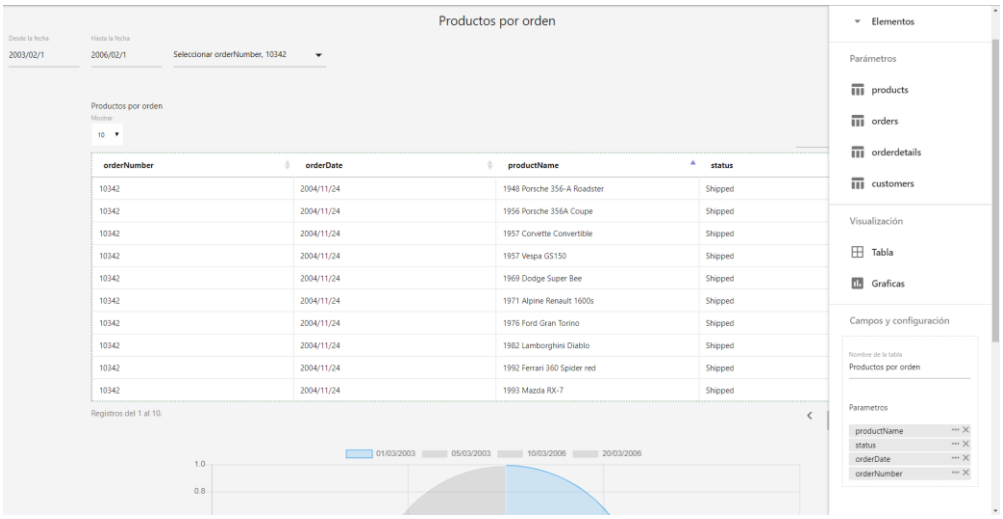

Figura 65. Detalle de una herramienta dinámica. Fuente: autoría propia.

5.1.3 Tiempos de respuesta de las consultas SQL

Debido al diseño de la Web API utilizando Django REST, permite al sistema tener un mejor rendimiento y no interrumpir las interacciones del usuario con el sistema, pero es importante analizar el tiempo que tarda realizar una consulta compleja (más de dos tablas) donde resulte una gran cantidad de datos, ya que el sistema responderá dependiendo el tipo de conexión y equipo que se tenga como se muestra en la Tabla 16.

Tabla 16. Tiempos de respuesta de consultas realizadas por el usuario al servidor con equipos con distintas características. Fuente: autoría propia.

Equipo	Conexión	Tipo de consulta	Tiempo promedio
Computadora de escritorio, procesador Intel Core i7 7700K, 8GB RAM	Por cable, 10 MB de velocidad	Simple (2 tabla) 2K datos	30 ms
		Compleja (más de dos tablas) 100K datos	100 ms
Laptop, procesador Intel Celeron, 4GB RAM	Wifi, 10 MB	Simple (2 tabla) 2K datos	250 ms – 450 ms
		Compleja (más de dos tablas) 100K datos	250 ms - 1.2 s
Computadora de escritorio, Intel Core i7 860, 4GB de RAM	Por cable, 10 MB de velocidad	Simple (1 tabla) 2K datos	250 ms – 450 ms
		Compleja (más de dos tablas) 100K datos	250 ms – 800ms

En la Tabla 16 se muestran los tiempos de respuesta con diferentes equipos, los cuales se miden desde que el usuario genera una consulta al servidor, éste la procesa, ejecuta los algoritmos de generación de grafos y búsqueda en profundidad y construcción de consultas, extrae información de la consulta generada y hasta el regreso y presentación de un resultado al cliente.

5.2 Discusión de resultados.

En esta sección se discuten los resultados obtenidos señalando la importancia, beneficios e impacto; además, se compara con otros desarrollos y se plantean algunas consideraciones.

- **Construcción de consultas SQL dinámicas:** se implementó una estrategia para que el usuario puede realizar consultas SQL dinámicamente a la base de datos, que pueda operarse sin necesidad de conocimientos avanzados en el área, totalmente transparente para los usuarios. Este tipo de herramientas suelen ser de tipo inteligencia de negocios y son muy útiles para realizar un análisis especifico de información,
- **Determinación de relaciones en BDRs utilizando el algoritmo DFS**: en esta investigación se utilizó el algoritmo DFS para determinar relaciones entre tablas de una base de datos relacional, permitiendo al usuario construir consultas SQL dinámicas sin conocimiento previo de la base de datos relacional. La aplicación de este algoritmo podría ayudar a desarrollar sistemas de inteligencia de negocios para cualquier empresa que busque gestionar su información y requieran un sistema capaz de utilizar cualquier base de datos relacional y ejecutar consultas dinámicas complejas. Considerando, además, que las bases de datos relacionales son las bases de datos mayormente utilizadas para el desarrollo de sistemas de información, proporcionando interoperabilidad también con los sistemas ya existentes.
- **Bases de datos relacionales y no relacionales:** existen un tipo especial de bases de datos, conocidas como bases de datos gráficas (GDB), las cuales usan los nodos de un grafo para almacenar entidades de datos como por ejemplo Amazon Neptuno, Neo4j, OrientDB, Microsoft SQL y Oracle, entre otras. En un GDB es más rápido consultar las relaciones porque se almacenan permanentemente en la BD, pero debido a que la mayoría de las empresas ya cuentan con bases de datos relacionales SQL o les es más fácil utilizarlas, se decidió aplicar una estrategia aplicable a las BD relacionales y a partir de ello desarrollar un módulo de herramientas dinámicas que permite consultar de diferentes conjuntos de datos sin necesidad de desarrollar un sistema específico para cada BD.

A continuación, para efectuar la discusión de resultados con otros autores, se presenta una tabla en el cual se comparan las funcionalidades de los trabajos que más se acercan al desarrollo de esta investigación.

Tabla 17. comparación de ORNALISIS con otros trabajos. Fuente: autoría propia.

Funcionalidad	Trabajos relacionados			
	ORNALISIS	**(Damasceno et al., 2018)**	**Ensmo (Kinsley (2016)**	**(Hristidis & Papakonstantinou, 2002)**
Presenta un método para la construcción de consultas dinámicas.	SI	SI	NO	SI
Utiliza herramientas de inteligencia de negocios para el apoyo a la toma de decisiones	SI	SI	NO	NO
Presenta un método para la búsqueda de relaciones.	SI	NO	NO	SI
Implementa un sistema de apoyo a la toma de decisiones	SI	SI	SI	SI
Permite la conexión cualquier base de datos relacional remota sin conocimiento de su estructura.	SI	NO	NO	NO
Presenta un método para la construcción de consultas SQL utilizando procesamiento de lenguaje natural	NO	NO	NO	SI
Utiliza técnicas de minería de datos para el análisis de la información	NO	NO	SI	NO

Como se puede observar en la Tabla 17, los trabajos considerados en esta discusión cuentan con ciertas funcionalidades parecidas al sistema ORNALISIS, que es el desarrollo de un sistema de apoyo a la toma de decisiones; sin embargo, carecen de funcionalidades como la implementación herramientas de inteligencia de negocios para la toma de decisiones, de un método para establecer conexión con cualquier base de datos relacional y trabajar sobre ella sin conocer su estructura o no permiten la manipulación de información asíncrona a la base de datos, pero si demuestran algunas ideas consideradas a trabajo futuro como el desarrollo de un método para la construcción de consultas SQL utilizando procesamiento de lenguaje natural y la utilización de técnicas de minería de datos para la predicción de resultados, esto para optimizar el procesamiento de las consultas dinámicas y mitigar el riesgo de un conocimiento básico en el lenguaje SQL.

5.3 Impacto de la implementación

Con el objetivo de comprobar la hipótesis establecida para la presente investigación se evaluaron los tiempos en las actividades de extracción, procesamiento y análisis en la

sociedad de producción rural ORNACOL para la generación de reportes semanales, mensuales y anuales, se estimaba un promedio de 1 a 3 horas para realizar el procesamiento manual de la información dependiendo de la cantidad de información necesaria para el reporte. Con la implementación del sistema este tiempo de procesamiento se redujo significativamente, como se puede observar en la Figura 66.

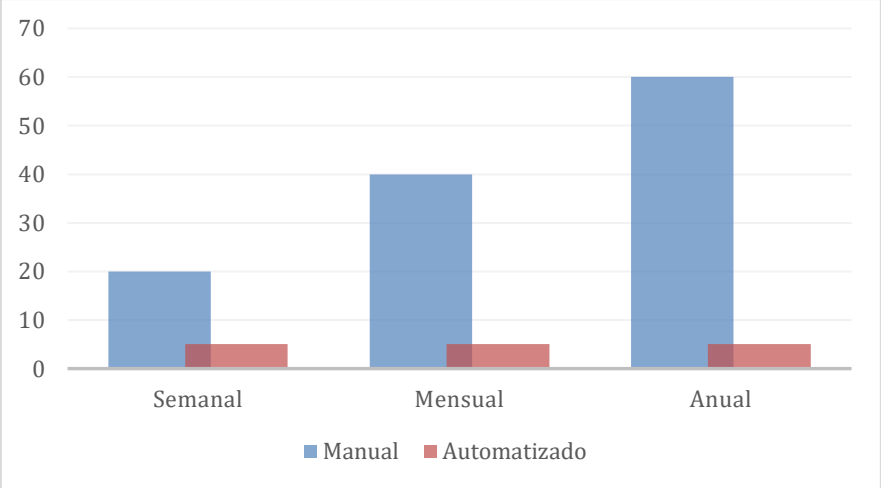

Figura 66. Tiempo de procesamiento de los datos para generar reportes periódicos. Fuente: autoría propia.

Con el sistema implementado, como se ilustra en la Figura 66, se logró automatizar el proceso manual de recolección y agrupación de los datos, logrando reducir el tiempo de procesamiento máximo de los datos (donde el que procesa los datos es una persona) de 20 minutos a 1 hora a máximo 5 minutos o menos de forma automatizada, lo cual abarca desde la creación y configuración de los parámetros para la herramienta, hasta la consulta, procesamiento, acoplamiento, despliegue y generación de reportes (en las herramientas dinámicas), mejorando el análisis y la toma de decisiones en tiempos más cortos. Se mejoró el tiempo de análisis y procesamiento de la información en un promedio de 84.7%, logrando un ahorro significativo de tiempo y el acceso eficaz, oportuno y eficiente a la información, apoyando así el proceso de toma de decisiones.

CAPÍTULO 6. CONCLUSIONES Y TRABAJO FUTURO

Este capítulo comprende la parte de la conclusión, se analiza y se explica en resumen el funcionamiento del desarrollo tecnológico que sustenta los resultados de esta investigación y se dan recomendaciones para un trabajo futuro.

6.1 Conclusiones

En este documento se presenta la implementación de un sistema web que da soporte a la toma de decisiones, automatizando y optimizando el proceso manual de recolección y agrupación de la información sobre la comercialización de plantas ornamentales.

El sistema permite hacer uso de dos tipos de herramientas: las herramientas integradas, las cuales cumplen con los requisitos funcionales que requiere la empresa y permiten generar reportes completos con gráficas para ser presentados formalmente y las herramientas dinámicas, que permiten al usuario realizar un tipo de consulta específica a cualquier base de datos vinculada. Las herramientas dinámicas trabajan con consultas dinámicas complejas que por medio de parámetros se obtienen datos de la BD, se procesan, serializan y se representan en el front-end por medio de tablas, gráficas y reportes. Debido a que las herramientas dinámicas pueden establecer conexión a cualquier BD relacional, un algoritmo crea un grafo equivalente al ER de la BD y el algoritmo DFS, a partir del grafo generado, permite optimizar la búsqueda de las relaciones entre dos o más tablas para crear las consultas dinámicas.

Las herramientas dinámicas pueden ser creadas por todos los socios de la empresa para poder realizar sus propios análisis periódicos. Así pues, el sistema web ORNALISIS contribuye a incrementar la cantidad, calidad, rentabilidad y sustentabilidad de la producción y comercialización de plantas ornamentales, apoyando a la toma de decisiones, reduciendo el tiempo de procesamiento y análisis de grandes cantidades de información e incrementando la productividad de la empresa. Fundamentado así, el desarrollo de este trabajo de investigación.

La herramienta propuesta beneficia a los productores de plantas ornamentales en los siguientes aspectos:

- Permite el aprovechamiento de la información histórica almacenada en bases de datos relacionales.
- Reduce el tiempo de análisis de grandes cantidades de información con la utilización de herramientas de inteligencia de negocios, mediante herramientas integradas o prediseñadas, así como herramientas dinámicas para atender necesidades específicas de análisis de información.
- Reducir tiempo y complejidad en el proceso de extracción y procesamiento de la información con la generación de reportes periódicos.
- Mejora el apoyo a la toma de decisiones para la comercialización.
- Incrementa la competitividad de los productores, ya que permite a través del análisis de información oportuno, eficiente y eficaz dar soporte a la toma de decisiones sobre

la comercialización de plantas ornamentales mediante una visión del pasado para dar oportunidad de generar estrategias en el futuro.

6.2 Trabajo futuro

El uso constante de este sistema de apoyo a la toma de decisiones permitirá a la empresa ORNACOL automatizar y optimizar el proceso de recolección de la información. Este sistema cuenta con dos tipos de herramientas siendo una de ellas las herramientas dinámicas que permiten la extracción y análisis de información de cualquier base de datos relacional y las herramientas integradas que atienden necesidades específicas de ORNACOL y establecidas en los requisitos funcionales. Sin embargo, como trabajo futuro se espera mejorar el funcionamiento de las herramientas dinámicas y mitigar los riesgos de requerir un mínimo nivel de conocimiento en bases de datos para interpretar las tablas seleccionadas y utilizar un método más eficiente para combinar filas de tablas relacionadas.

Finalmente, se plantean algunas recomendaciones para continuar con las mejoras del sistema a futuro:

- Implementar una mejora al sistema que permita utilizar distintos tipos de bases de datos, que no sean estrictamente bases de datos relacionales.
- Desarrollar una interfaz que permita al usuario interpretar las tablas de la base de datos, seleccionada en las herramientas dinámicas, mostrando ventanas modales o emergentes que permitan dar un tutorial por el sistema y explicar cada una de las funciones de las herramientas.
- Utilizar la cláusula JOIN para que las consultas sean más rápidas, ya que para realizar consultas dinámicas, el sistema utiliza uniones cruzadas (productos cartesianos) para combinar filas de tablas relacionadas.
- Si se requiere integrar una herramienta que no se contempla en las funcionalidades del sistema y que no se puede ofrecer mediante las consultas dinámicas, el desarrollador deberá tener conocimiento sobre las siguientes tecnologías: Python, el framework Django, jQuery, Ajax, DataTables y ChaJS ya que las herramientas integradas utilizan esas tecnologías para la extracción y despliegue de la información.
- Automatizar la generación de reportes periódicos de ventas estableciendo una fecha y enviarlos a los socios por correo electrónico.

REFERENCIAS BIBLIOGRÁFICAS

Ambler, S. W. (2014). The Agile Unified Process. *The Agile Unified Process (AUP)*. http://www.ambysoft.com/scottAmbler.html

Blancarte, O. (2017). *SOAP vs REST ¿Cuál es mejor? México*. Oscar Blancarte Software Architect

Bozkir, A. S., & Sezer, E. A. (2013). A new web based data mining exploration and reporting tool for decision makers. *Artificial Intelligence Research*, *2*(3). https://doi.org/10.5430/air.v2n3p70

Casanova, R. M. (1994). *DSS una nueva metodología*.

Chandel, A., & Sood, M. (2014). *Searching and Optimization Techniques in Artificial Intelligence: A Comparative Study & Complexity Analysis*. *3*(3), 6.

COEPPLANTS, C. A. C. (2012). *Plan Rector: Comité Sistema Producto Ornamentales Colima*. Colima: COEPPLANTS Colima A.C. http://dev.pue.itesm.mx/sagarpa/estatales/EPT%20COMITE%20SISTEMA%20PRODUCTO%20ORNAMENTALES%20COLIMA/PLAN%20RECTOR%20QUE%20CONTIENE%20PROGRAMA%20DE%20TRABAJO%202012/PR_ORNAMENTALES_COLIMA_2012.pdf

Damasceno, E., Azevedo, A., & Pinto, A. (2018). Business Intelligence—Implantation on Federal Institute of Triângulo Mineiro (IFTM) System: *Proceedings of the 10th International Conference on Computer Supported Education*, 528–535. https://doi.org/10.5220/0006818805280535

Dobaev, A. Z., Maslakov, M. P., & Dedegkaeva, A. A. (2016). Development of decision support system for data analysis of electric power systems. *2016 2nd International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, 1–4. https://doi.org/10.1109/ICIEAM.2016.7911584

Dobson, R. (2017). *List Dependencies for SQL Server Foreign Keys* [Artículo]. mssqltips. https://www.mssqltips.com/sqlservertip/4753/list-dependencies-for-sql-server-foreign-keys/

Duarte de Oliveira Paiva, P. (2018). *Horticulture and Ornamental Horticulture*. *24*(1), 6. https://doi.org/10.14295/oh.v24i1.1169

ERGO. (2000). *ERGO DSS*.

Freeman, J. (2019, agosto 1). *What is an API? Application programming interfaces explained* [Blog]. infoworld.com. https://www.infoworld.com/article/3269878/what-is-an-api-application-programming-interfaces-explained.html

García Mejía, E. E., García Virgen, J., & Chávez Valdez, R. E. (2018). Gestión de la Comercialización de Plantas Ornamentales Utilizando Normas de Trazabilidad hacia delante. *RIIIT. Revista internacional de investigación e innovación tecnológica*, *6*(35), 0–0.

Haettenschwiler., P. (1999). *Neues anwenderfreundliches Konzept der Entscheidungsunterstützung. Gutes Entscheiden in Wirtschaft, Politik und Gesellschaft*.

Hovad, J., Lněnička, M., & Komárková, J. (2015, julio 1). *Real-Time Web Mining Application to Support Decision-Making Process*. https://doi.org/10.1109/DT.2015.7222957

Hristidis, V., & Papakonstantinou, Y. (2002). DISCOVER: Keyword Search in Relational Databases. *Proceedings of the 28th VLDB Conference*, 670–681. https://doi.org/10.1016/B978-155860869-6/50065-2

IEEE. (2008). *Especificación de Requisitos según el estándar de IEEE 830*. https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

INEGI. (2014). *Censos Económicos*.

Jacobson, I., Brooch, G., & Rumbaugh, J. (1999). *Unified Software Development Process* (01 ed.). Addison-Wesley Professional.

Kinsley, H. (2016). *Análisis de datos simplificado en Python y pandas utilizando herramientas de minería de datos* (Versión 1) [Django]. Harrison Kinsley. https://ensmo.com/

Kołodziejski, M. (2019). *Get to Know the Power of SQL Recursive Queries* [Artículo]. learnsql. https://learnsql.com/blog/get-to-know-the-power-of-sql-recursive-queries/

Marakas, G. M. (1999). *G. M. Marakas. Decision support systems in the twenty-first century* (2a ed.). P. Hall.

O´BRIEN, J. A., & MARAKAS, G. M. (1985). *Sistemas de Información Gerencial* (7a ed.). Mc Graw-Hill.

Power, D. J. (2011). *Decision support systems: Concepts and resources for managers*. Greenwood/Quorum Books.

Python Software Foundation. (2019). *Python Patterns—Implementing Graphs*. https://www.python.org/doc/essays/graphs/

Python Software Foundation. (2020, junio 30). *What is Python?* [Docs]. General Python FAQ. https://docs.python.org/3/faq/general.html#what-is-python

Ramesh, B. N., Amballi, A. R., & Mahanta, V. (2018). *DJANGO THE PYTHON WEB FRAMEWORK*. *6*(2), 5.

Ruiz G., A., Hernandez R., L. A., & Giraldo O., W. J. (2009). Aplicación de los sistemas de soporte a la decisión (DSS) en el comercio electrónico: Implementing a decision support system (DSS) in e-business. *Ingeniería e Investigación*, *29*(2), 94–99.

Rupnik, R., & Kukar, M. (2007). *Decision support system to support decision processes with data mining*. https://www.researchgate.net/publication/26596382_Decision_Support_System_to_support_decision_processes_with_Data_Mining

SAGARPA. (2018). *Manual de Trazabilidad de_Productos_Hortofrut colas para consumo en fresco de los Estados Unidos Mexicanos*.

https://www.gob.mx/cms/uploads/attachment/file/120192/Manual_de_Trazabilidad_de_Productos_Hortofrut_colas_para_consumo_en_fresco_de_los_Estados_Unidos_Mexicanos.pdf

Salgueiro, J. L. R., Carrión, G. C., & González, J. L. G. (2012). *Los sistemas de inteligencia de negocio como soporte a los procesos de toma de decisiones en las organizaciones*. 23.

Santillán, Á. G., & Mendoza, M. L. (2007). *Diseño y desarrollo de un sistema Web de apoyo a ejecutivos para la toma eficiente de decisiones de compras*. 39.

Silva, R. A., Silva, F. C. A., & Gomes, C. F. S. (2016). O USO DO BUSINESS INTELLIGENCE (BI) EM SISTEMA DE APOIO À TOMADA DE DECISÃO ESTRATÉGICA. *Revista Gestão Inovação e Tecnologias*, *6*(1), 2780–2798. https://doi.org/10.7198/S2237-0722201600010005

Tarifa, E. E., Martínez, S. L., & Chalabe, S. A. (2013). *Desarrollo de Sistemas de Apoyo para la Toma de Decisiones en Procesos Productivos*. 5.

Trefil, J. S. (2005). *The Encyclopedia of Science and Technology*.

Valenzuela Rodríguez, J. A. (2003). *Las tecnologías de información en las pequeñas y medianas empresas (PYME's)*. https://www.gestiopolis.com/tecnologias-de-informacion-en-las-pymes/

Villegas, J. A. G. (2009). *DESARROLLO DE UN SISTEMA DE APOYO A LA TOMA DE DECISIONES PARA EL MANEJO DE PRODUCTOS Y TIENDAS EN UNA CADENA DE RETAIL A PARTIR DE DATOS TRANSACCIONAL DE VENTAS Y CARCTERÍSTICAS DE TIENDAS*. 158.

w3resource. (2020). *SQL Cross Join*. https://www.w3resource.com/sql/joins/cross-join.php

Yin, X., Han, J., & Yang, J. (2005). Searching for Related Objects in Relational Databases. *12-2005*, 227–236.

Zimmermann, T. R. (2006). *DESENVOLVIMENTO DE UM SISTEMA DE APOIO À DECISÃO BASEADO EM BUSINESS INTELLIGENCE*. 77.

APÉNDICES Y ANEXOS

Apéndice 1. Manual de usuario

El presente manual es una guía que tiene como finalidad facilitar al usuario el uso del Sistema ORNALISIS, mediante la creación de herramientas dinámicas y específicas. El manual contiene una descripción de las opciones del panel de herramientas y está orientado a todo tipo de usuarios del sistema.

Objetivo

Servir como una guía para los usuarios que utilizarán el sistema ORNALISIS, permitiendo que estos realicen las diferentes tareas según los permisos obtenidos.

Perfil de usuarios

Se especifican los perfiles de los usuarios de este manual; los cuales se dividen en tres categorías: Usuarios que involucra al operador, usuarios socios de la empresa ORNACOL e invitados.

Requerimientos de conocimiento de usuarios

- Conocimientos básicos del manejo de computadoras.
- Conocimientos básicos de uso de exploradores web.
- Conocimiento básico de bases de datos y hojas de cálculo.

Página principal

En la página principal, se muestran las funciones de ORNALIS a primera vista, además podrás encontrar un botón color negro en la parte central para redirigirte al inicio de sesión.

Figura 67. Landing page de ORNALISIS. Fuente: autoría propia.

Inicio de sesión

El siguiente paso es iniciar sesión, presionando **INICIAR SESION** de la página anterior. Al presionar el botón, el sistema te redirigirá a la página de inicio de sesión, véase Figura 68 Ingresando el usuario y contraseña correctas, podrás iniciar sesión.

Figura 68. Página de inicio de sesión. Fuente: autoría propia.

Panel de herramientas

El panel de herramientas es la página de inicio del sistema, donde se despliega el panel de herramientas y puedes elegir ver entre herramientas integradas y dinámicas, como se muestra en la Figura 69.

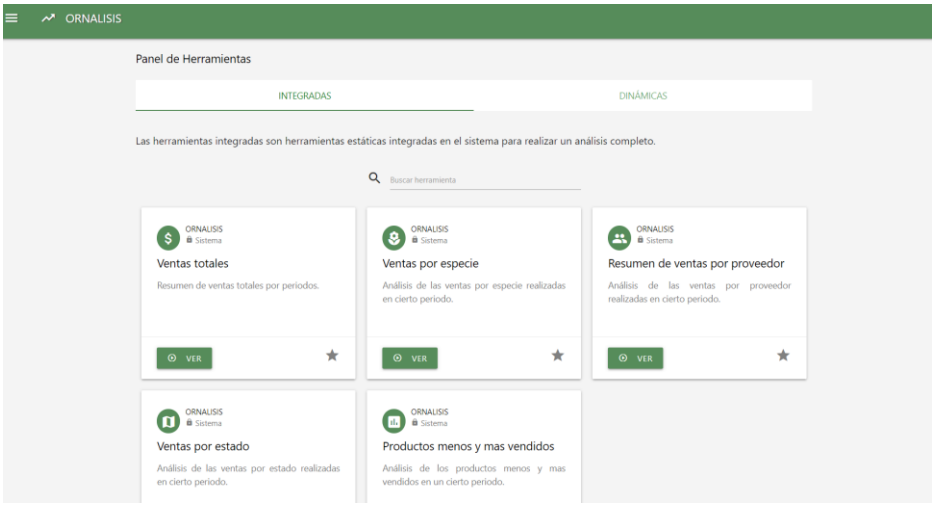

Figura 69. Página de inicio-panel de herramientas. Fuente: autoría propia.

- **Herramientas integradas**: herramientas programadas para resolver una necesidad específica, entre ellas se encuentran ventas totales, ventas por especie, resumen de ventas por proveedor, ventas por estado y productos menos y más vendidos como se muestra en la Figura 70 Estas herramientas no cambian, es decir, no pueden ser modificadas, pero si manipuladas para extraer la información. Cada herramienta se muestra en un recuadro con su título, descripción y el creador de la herramienta; además, un botón para entrar en ella (Figura 71)

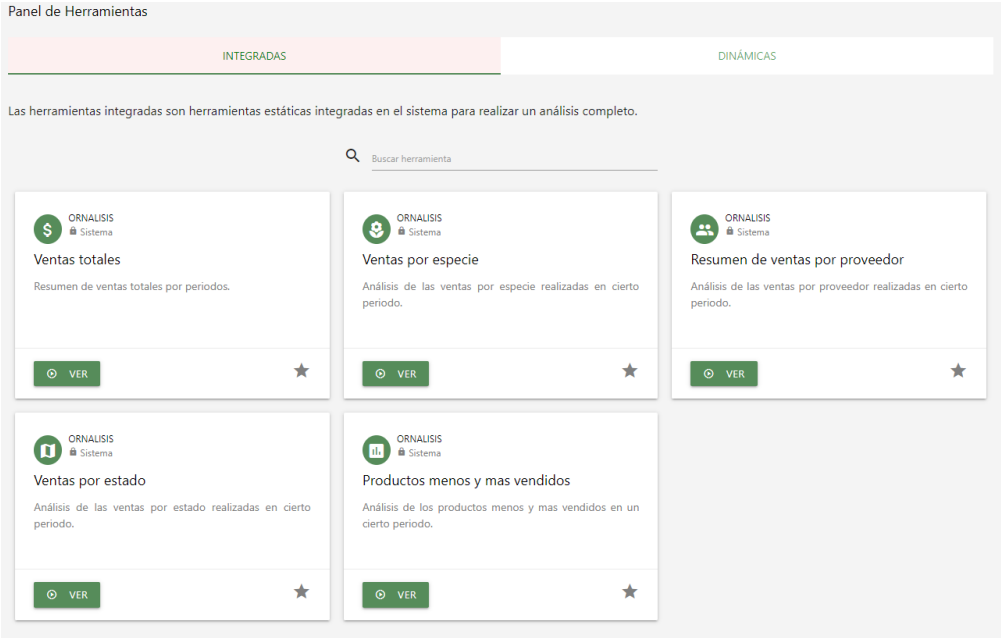

Figura 70. Herramientas integradas. Fuente: autoría propia.

Figura 71. Contenido de una herramienta integrada. Fuente: autoría propia.

- **Herramientas dinámicas:** herramientas que permiten el análisis dinámico de la información a partir de la conexión de cualquier base de datos relacional con relaciones bien establecidas. En este panel puedes crear tus propias herramientas para realizar

un análisis especifico (Figura 72). Cada herramienta dinámica contiene su título, descripción, usuario que creo la herramienta, opciones de permisos y configuración Figura 73.

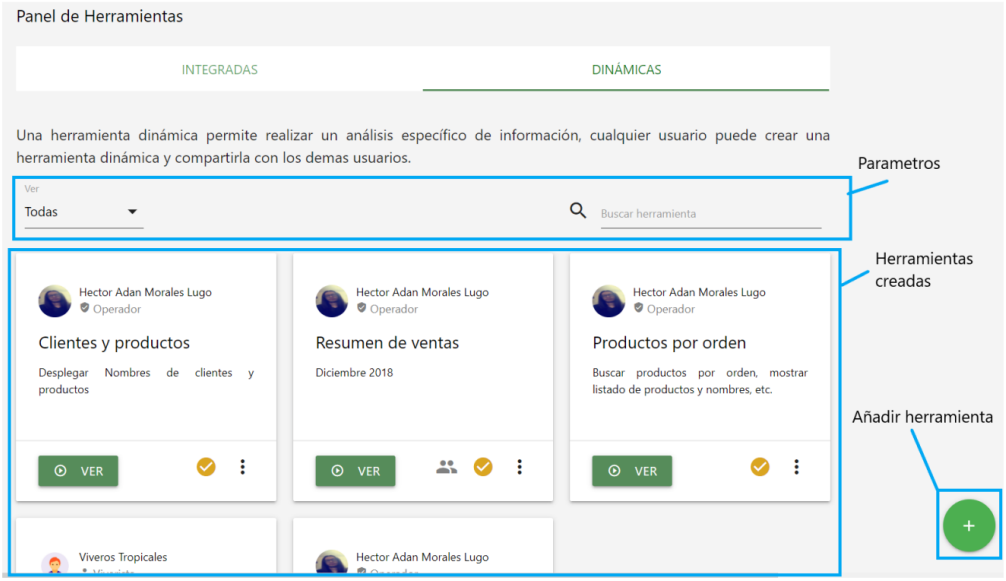

Figura 72. Panel de herramientas dinámicas. Fuente: autoría propia.

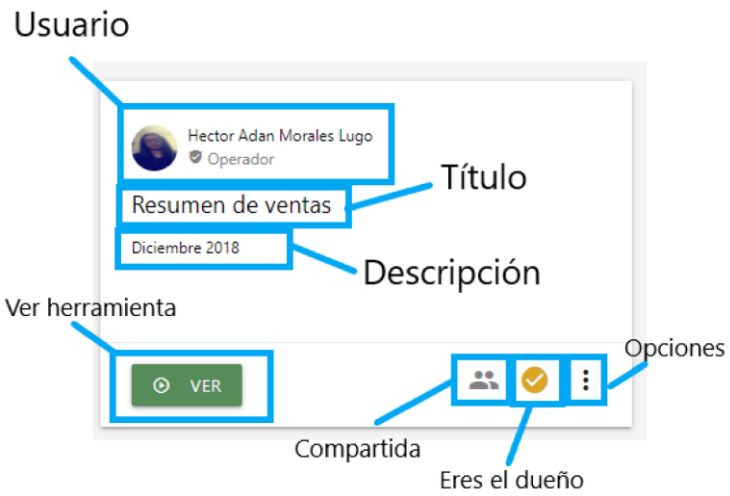

Figura 73. Contenido de una herramienta dinámica. Fuente: autoría propia.

Presionando el botón de opciones, se desplegará el recuadro de opciones de la Figura 74, podrás editar la herramienta, borrarla o compartirla con los demás.

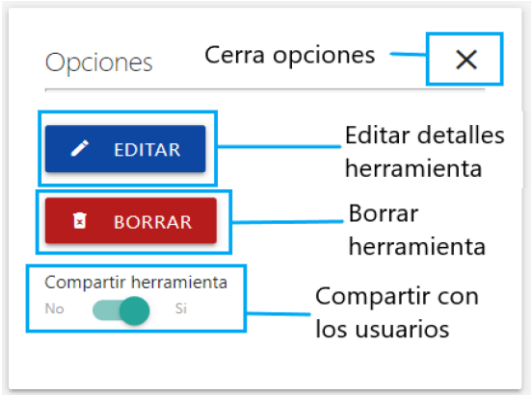

Figura 74. Opciones secundarias de una herramienta dinámica. Fuente: autoría propia.

Menú de opciones

El menú de opciones aparece en la parte izquierda, presionando el icono de la esquina superior izquierda podrás desplegarlo y navegar por el sistema.

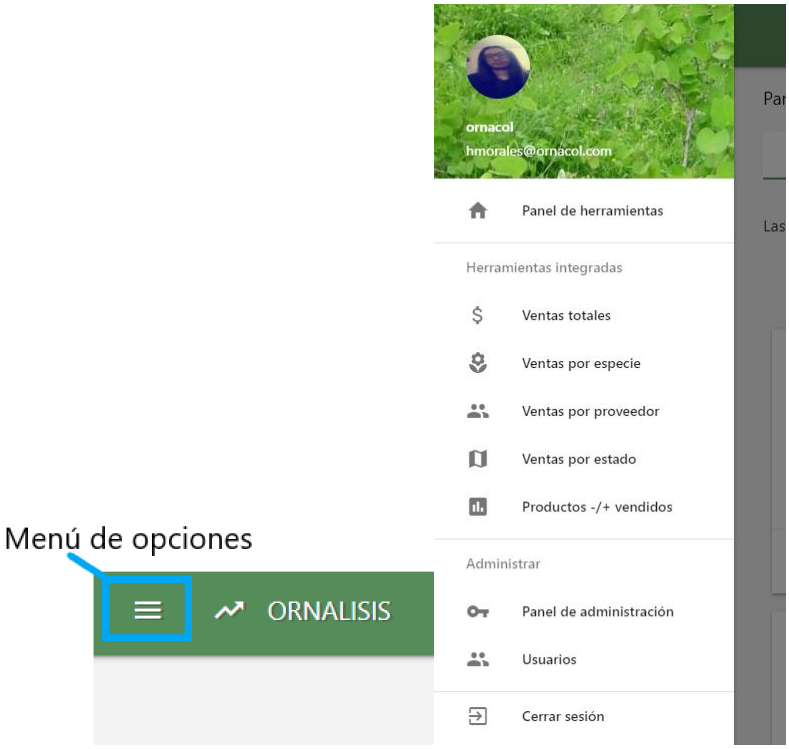

Figura 75. Botón de menú de opciones, menú de opciones. Fuente: autoría propia.

Panel de usuarios

Para navegar al panel de usuario despliegue el menú de opciones anterior y diríjase a la opción: Usuarios. Posteriormente se desplegará la vista de la Figura 76 que permitirá al operador registrar un nuevo usuario o editar la información de uno ya añadido.

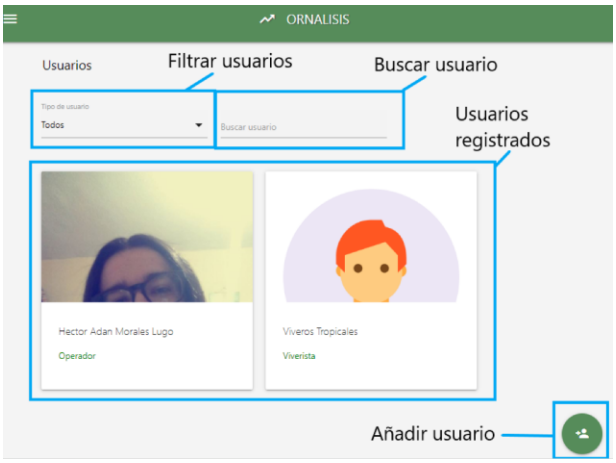

Figura 76. Panel de usuarios. Fuente: autoría propia.

Generación de reportes

Para generar un reporte diríjase a la parte superior de la herramienta integrada o dinámica, presione el icono de descarga y configure las opciones necesarias para descargar el reporte .PDF o .xlsx.

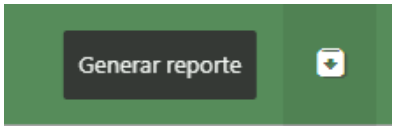

Figura 77. opción para generar reporte

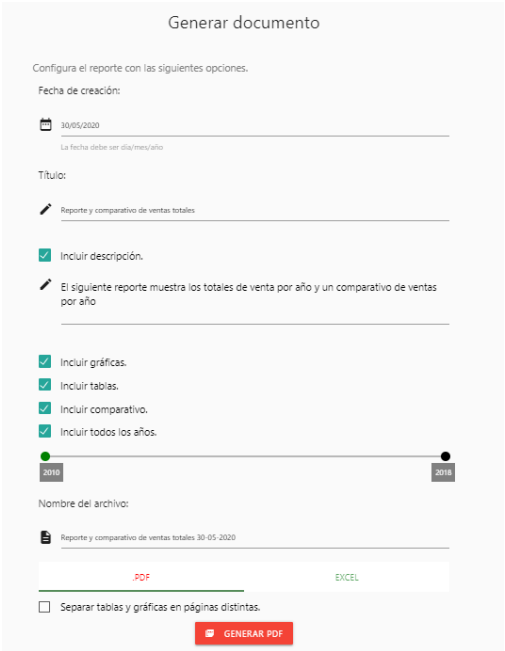

Figura 78. Opciones para generar los reportes. Fuente: autoría propia.

A continuación, se descargar un archivo en el formato especificado que muestra la misma información de la página web en formato reporte u hojas de cálculo.

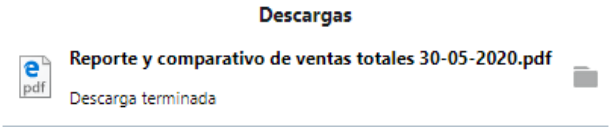

Figura 79. Reporte generado. Fuente: autoría propia.

Cerrar sesión

Para cerrar sesión presione el menú de opciones y seleccione la opción cerrar sesión. automáticamente se cerrará la sesión y te redireccionará a la página principal.

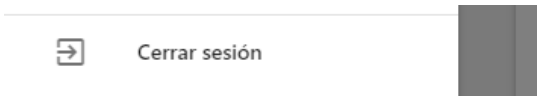

Figura 80. Cerrar sesión. Fuente: autoría propia.