Desarrollo del front end del sistema de gestión de clínicas veterinarias aplicando paradigma de Línea de Producto de Software (LPS) enfocado en un desarrollo co-localizado.

Castro Martin, Julio David y Poveda Solano, José Francisco

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Trabajo de Integración curricular, previo a obtener el título de Ingeniería de Software

Dr. Jácome Guerrero Patricio Santiago

8 de febrero del 2023

Latacunga

CERTIFICADO DE ANÁLISIS
magister

Tesina_Castro-Poveda_16-02-2023-Antiplagio_V2

2% Similitudes

1% Texto entre comillas
< 1% similitudes entre comillas
7% Idioma no reconocido

Nombre del documento: Tesina_Castro-Poveda_16-02-2023-Antiplagio_V2.docx
ID del documento: f02c7a089f35006adb5f3fc949ffc8c08397e01f
Tamaño del documento original: 2,34 Mo

Depositante: JOSÉ LUIS CARRILLO
Fecha de depósito: 16/2/2023
Tipo de carga: interface
fecha de fin de análisis: 16/2/2023

Número de palabras: 5740
Número de caracteres: 39.333

Ubicación de las similitudes en el documento:

Fuentes

Fuente principal detectada

N°	Descripciones	Similitudes	Ubicaciones	Datos adicionales
1	Tesina_Nunez-Camacho_16-02-2023-Antiplagio.docx | Tesina_Nunez-Camac... #8f0507 El documento proviene de mi biblioteca de referencias	1%		Palabras idénticas : 1% (70 palabras)

Fuente con similitudes fortuitas

N°	Descripciones	Similitudes	Ubicaciones	Datos adicionales
1	Tesina_Final_Sanchez_Romo_16-02-2023-Antiplagio.docx | Tesina_Final_San... #258f1b El documento proviene de mi biblioteca de referencias	< 1%		Palabras idénticas : < 1% (16 palabras)

Fuentes mencionadas (sin similitudes detectadas)

Estas fuentes han sido citadas en el documento sin encontrar similitudes.

1 https://sgcv-identityserver.azurewebsites.net/
2 http://localhost:3000/signin-oidc
3 http://localhost:3000

PATRICIO SANTIAGO JACOME GUERRERO

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Certificación

Certifico que el trabajo de integración curricular: "**Desarrollo del front end del sistema de gestión de clinicas veterinarias aplicando paradigma de Línea de Producto de Software (LPS) enfocado en un desarrollo co-localizado**" fue realizado por los señores **Castro Martin, Julio David** y **Poveda Solano, José Francisco**; el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizada en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Latacunga, 16 de enero de 2023

.....

Jácome Guerrero, Patricio Santiago

C. C. **1001689791**

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Responsabilidad de Autoría

Nosotros **Castro Martin, Julio David** y **Poveda Solano, José Francisco**; con cédulas de ciudadanía n° **1805015904** y **1805287362**; declaramos que el contenido, ideas y criterios del trabajo de integración curricular: **Desarrollo del front end del sistema de gestión de clinicas veterinarias aplicando paradigma de Línea de Producto de Software (LPS) enfocado en un desarrollo co-localizado,** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Latacunga, 16 de enero de 2023

Castro Martin, Julio David

C. C. **1805015904**

Poveda Solano, José Francisco

C. C. **1805287362**

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Autorización de Publicación

Nosotros **Castro Martin, Julio David** y **Poveda Solano, José Francisco**; con cédulas de ciudadanía n° **1805015904** y **1805287362**; autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de integración curricular: **Desarrollo del front end del sistema de gestión de clínicas veterinarias aplicando paradigma de Línea de Producto de Software (LPS) enfocado en un desarrollo co-localizado,** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Latacunga, 16 de enero de 2023

.....

Castro Martin, Julio David

C. C. **1805015904**

.....

Poveda Solano, José Francisco

C. C. **1805287362**

Dedicatoria

A mis padres, hermanas, familiares y amigos que me han apoyado y han confiado en mí, ya que han sido una parte importante en mi vida personal y en mi formación profesional. Ayudándome siempre a seguir adelante a pesar de las adversidades.

Julio Castro

Dedicatoria

A mis padres, hermanos, familiares y amigos que me han acompañado en el transcurso de mi vida universitaria, ya que han sido un pilar fundamental en mi formación y mis metas, todo se los debo a ellos.

José Poveda

Agradecimientos

Este trabajo de titulación es el fruto del esfuerzo realizado durante todo el transcurso de la carrera, agradezco a todas las personas que han estado conmigo durante el transcurso de mis estudios universitarios. A los ingenieros que han sido amigos, a mis compañeros del proyecto por el apoyo brindado durante la implementación del sistema, a mi familia que siempre ha confiado en mí y a mi compañero de tesina, qué gracias al esfuerzo común hemos logrado culminar con nuestro proceso de titulación.

Julio Castro

Agradecimientos

Quiero agradecer a todas las personas que han estado presentes en este trabajo de titulación como son: familiares, amigos que me han apoyado en el transcurso de este arduo trabajo.

A mis compañeros de trabajo que hemos comenzado desde los más bajo hasta llegar a cumplir un sueño que lo anhelábamos tanto.

A los ingenieros que han ido aportando sus conocimientos y ayuda a lo largo de mi vida universitaria y en este trabajo de titulación.

Y sobre todo gracias a Dios que me ha dado la oportunidad de poder llegar a este punto de mi vida, junto con toda mi familia y amigos.

José Poveda

Glosario

A

Análisis de dominio: Determinación de la lógica de negocio.

APIs: Interfaz de programación de aplicaciones.

App service: Componente para publicar proyectos en Azure.

Azure: Sistema de computación en la nube de Microsoft.

B

Back end: Parte de un sistema el cual interactúa con la base de datos y con el front end.

Big-bang: Forma de organización de equipos en Desarrollo Global de Software.

C

Core assets: activos base.

CSS: Hoja de estilo en cascada.

D

Database: Base de datos.

Desarrollo co-localizado: Forma de localización dentro del desarrollo global de software.

DGS: Desarrollo global de software.

E

ECOP: Examen clínico orientado a problemas.

F

Features: Funcionalidades del sistema.

FODA: Análisis de dominio orientado a características.

Front end: Parte de un sistema web la cual permite al usuario interaccionar con él.

Folow the sun: Método de desarrollo en DGS.

G

Git: Gestor de versionamiento.

Git Flow: Marco de trabajo de git.

H

HTTP: Protocolo de transferencia de hipertexto.

L

Lean: Marco de trabajo para empresas enfocado en el cliente.

Liquibase: Framework para gestión de base de datos.

LPS: Línea de producto software.

N

Namespace: Palabra reservada del lenguaje Typescript.

Nodejs: Lenguaje para back end de javascript

O

OpenId: Protocolo de autenticación basado en oauth2.0.

P

Pet Grooming: Servicio complete de aseo y cuidado de mascotas.

PuLSE: Ingeniería de línea de producto software

Pull request: Acción para la unión de ramas en git.

R

React: Librería de desarrollo web.

S

Sprint: Es un periodo breve de tiempo fijo en el que un equipo de scrum trabaja para completar una cantidad de trabajo establecida.

Sprint daily: Reunión diaria realizada dentro del sprint.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURA

ÍNDICE DE TABLAS

Resumen

El sistema de gestión para clínicas veterinarias (SGCV) aplicando el paradigma de Línea de Producto Software (LPS), está compuesto por servicios independientes y específicos que trabajan juntos para proporcionar una solución a la digitalización de información dentro de las clínicas veterinarias. Donde cada microservicio se encarga de una tarea específica, como el manejo de consultas, Pet Grooming y gestión de productos, permitiendo una mayor escalabilidad, flexibilidad y fiabilidad en comparación a un sistema monolítico tradicional. Este trabajo tiene como objetivo mejorar la administración de información dentro de la clínica veterinaria, en donde se propone la adopción de una metodología de microservicios, Línea de Producto Software (LPS), Análisis de Dominio Orientado a Características (FODA) y Desarrollo Global de Software (DGS). Para cumplir el objetivo se realizó el sistema utilizando herramientas que se complementan entre sí (Azure DevOps, git, React), que ha permitido el modelado de diagramas y el desarrollo del sistema. También han permitido la administración y la gestión del proyecto de manera que se han simplificado los procesos de distribución de las tareas también ha permitido enlazar el código con cada una de las tareas. Teniendo como resultados el consumo de cada microservicio y su despliegue en un componente gráfico.

Palabras clave: Clínicas Veterinarias, Línea de Producto Software, Desarrollo Global de Software, microservicios, Azure.

Abstract

The management system for veterinary clinics (SGCV) applying the Software Product Line (LPS) paradigm, is made up of independent and specific services that work together to provide a solution to the digitization of information within veterinary clinics. Where each microservice is responsible for a specific task, such as query handling, Pet Grooming and product management, allowing greater scalability, flexibility and reliability compared to a traditional monolithic system. This work aims to improve information management within the veterinary clinic, where the adoption of a microservices methodology, Software Product Line (LPS), Feature-Oriented Domain Analysis (SWOT) and Global Software Development is proposed. (DGS). To meet the objective, the system was made using tools that complement each other (Azure DevOps, git, React), which has allowed the modeling of diagrams and the development of the system. They have also allowed the administration and management of the project in such a way that the task distribution processes have been simplified, it has also allowed the code to be linked to each of the tasks. Having as results the consumption of each microservice and its deployment in a graphical component.

Key words: Veterinary Clinics, Software Product Line, Global Software Development, microservices, Azure.

Capítulo I

Introducción

Antecedentes

“Según datos del Instituto Ecuatoriano de Estadística y Censos (INEC), existen 128 locales que prestan servicios veterinarios en el mercado porteño, que generan un movimiento económico anual que supera los USD 2,3 millones.” (Ponce, 2013) en Porto Viejo, Ecuador. En el Ecuador, las clínicas veterinarias han ido tomando una gran importancia debido a que “seis de cada diez hogares del país tienen mascotas” (La República, 2019). Las empresas de clínicas veterinarias, que ofrecen atención de mascotas, como perros, gatos entre otras, se han vuelto comunes a lo largo del tiempo debido a su amplia gama de servicios que incluyen la venta de productos, cuidado de mascotas, SPA, atención médica y algunas clínicas veterinarias ofrecen todos estos servicios en un solo lugar.

“En Ecuador, la tecnología ha sido una herramienta de trabajo muy esencial e importante para el desarrollo tanto empresarial y personal, con esta tecnología se puede obtener varios productos o servicios que garantice el buen uso.” (Zavala, 2019). En Ecuador, las clínicas veterinarias se llevan a cabo los procesos de registro de manera manual, lo que conlleva a una falta de organización en el orden de los expedientes físicos, una carencia de códigos que permitan la identificación a cada expediente, una información ilegible en los registros debido a la calidad de caligrafía, y la posibilidad de que la información obtenida sea afecta por la pérdida de documentos o inexistencia de información.

Por esta razón, las clínicas veterinarias han decidido implementar un enfoque automatizado para el manejo de sus datos y están analizando en diversas opciones en la implementación de herramientas tecnológicas para mejorar la gestión de sus establecimientos.

Justificación e importancia

En el Plan Nacional del Buen Vivir (2017 - 2021) en el objetivo número 3 se da respaldo a la importancia del cuidado de la fauna del país:

> La comprensión de la ecodependencia, además, se extiende al cuidado y protección de la fauna, constatando la importancia de la vida y la dignidad en su sentido ético amplio, por lo que es preciso precautelar el bienestar animal con normativa, política pública y jurisprudencia expresa, clara y directa. (Secretaria nacional de planificación, 2017)

En Ecuador, la población se ha visto un aumento en la atención y el bienestar de sus mascotas, lo que ha llevado a la creación de nuevos centros especializados en el cuidado de animales domésticos y el aumento de la cantidad de profesionales veterinarios. Las clínicas veterinarias buscan proporcionar un servicio completo y de calidad para mascotas, contando con médicos especialistas en sus respectivas áreas para brindar un servicio más profesional. Como objetivo se está desarrollando un sistema que ayudara en los servicios por las clínicas veterinarias, incluyendo el registro, diagnóstico, tratamiento, SPA, venta de productos e historial clínico. La implementación de este sistema permitirá una reducción en el tiempo de atención al cliente.

La implementación del sistema en las clínicas veterinarias tiene como objetivo mejorar la satisfacción y rapidez en el proceso de registro de datos, lo que resultara en una mayor agilidad en los procesos y un mejor control de la información, evitando errores en los procesos manuales y físicos. El uso de este sistema tecnológico aumentara la productividad y la eficiencia en los procesos que se manejan dentro de las clínicas veterinarias.

Las necesidades de las clínicas veterinarias económicamente es satisfacer las necesidades mediante la reutilización de componentes software para la creación de nuevos

productos, lo que reduce los costos, tiempo para producir sistemas de calidad mediante la aplicación del paradigma LPS.

El propósito social es proveer software que satisfaga las necesidades específicas y generales de las clínicas veterinarias en el Ecuador, a un costo accesible, con el objetivo de mejorar la calidad de los procesos de atención al cliente

Alcance

Teniendo en cuanta la relevancia de las clínicas veterinarias y los servicios que ofrecen, se plantea la creación de un a aplicación web que permita mejorar la organización y almacenamiento de información en las diversas áreas de las clínicas veterinarias, mediante la administración de pacientes, tutores, historial clínico, consultas, carnet de vacunación, SPA, hospedaje, exámenes comentario y un catálogo de productos, permitiendo una mejor gestión y almacenamiento de la información.

Este sistema está realizado con arquitectura microservicios por lo cual será eficaz, fácil de mantener y probar ya que está dividido en las características antes mencionadas, “de manera que los procesos brindan las clínicas veterinarias sean realizados de forma ágil y eficiente, tiene como resultado una reducción de tiempo de consulta del paciente y atención al cliente.” (Cedeno, Catuto, & Rodas-Silva, 2021). Además, se establecerá la implementación de un sistema de autenticación para asegurar y proteger la información que manejan las clínicas veterinarias.

Planteamiento del problema

Las clínicas veterinarias experimentas dificultades en la recolección manual de información, lo que conduce a una fata de organización en los registros físicos, la ausencia de un código que identifique cada expediente y también la información ilegible debido a la calidad de la caligrafía implementada.

Para la gestión administrativa de veterinarias se han desarrollado sistemas que apoyan a los profesionales de la salud, (Cedeno, Catuto, & Rodas-Silva, 2021) mencionan:

> Como es el caso de la clínica San Agustín, ubicada en Santiago de Chile, en el que se requirió el desarrollo de un "Sistema de Gestión para una Clínica Veterinaria" que permita facilitar y gestionar la información de forma más ordenada, en un solo lugar.

En el Ecuador se ha llevado a cabo proyectos con el propósito que buscan mejorar la atención y la gestión de información en las clínicas veterinarias.

> En la Península de Santa Elena, en el año 2017, se llevó a cabo la implementación de un “Sistema en la nube para Control y Manejo de Procesos Clínicos”. Este proyecto facilitó la reducción de los tiempos de respuesta, dando la optimización de los principales procesos del centro médico y mejorando la seguridad, disponibilidad e integridad de la información (Cedeno, Catuto, & Rodas-Silva, 2021)

En el año 2022 en la ciudad de Ambato se realizó el proyecto denominado “Sistema electrónico de monitoreo de mascotas para la gestión de clínicas veterinarias utilizando VOIP e IOT” (León , 2022). El objetivo es mejorar el registro de información de signos vitales de las mascotas, en donde los datos obtenidos serán reflejados en una aplicación web desarrollada en React, lo que facilita la supervisión y el monitoreo en tiempo real el estado de salud de las mascotas.

Actualmente, existen sistemas informáticos para clínicas veterinarias que ofrezcan un servicio integral de gestión de información, datos e historias clínicas para los clientes. La falta de un sistema informático es complicada debido a que estos procesos se gestionan

manualmente y no son muy eficientes, donde pueden existir perdida de documentos e inexactitud al momento de registro y búsqueda de información.

Como resultado es necesario instaurar un sistema que facilite el registro de información de historial médico, tratamiento y exámenes clínicos del paciente.

Formulación del problema a resolver

Como diseñar y desarrollar el front end del sistema de gestión de clínicas veterinarias y consumo de microservicios.

Objetivos

Objetivo general

Diseñar y desarrollar el front end del sistema de gestión de clínicas veterinarias y consumir los microservicios.

Objetivos específicos

- Revisión del estado del arte de LPS, microservicios y DGS.
- Elaborar la LPS basada en un análisis de dominio en clínicas veterinarias.
- Desarrollar el front end del sistema de gestión de clínicas veterinarias en un ámbito DGS.

Hipótesis

La utilización de LPS para desarrollar software permite implementar soluciones que se ajustan a los requerimientos de las clínicas veterinarias.

Señalamiento de variables

Variable independiente: LPS

Variable dependiente: Requisitos de las clínicas veterinarias

Capítulo II

Fundamentación teórica y referencial

Introducción

En este capítulo se repasarán los principios fundamentales de la LPS, igualmente las características y procesos que la sustentan, incluyendo el Análisis de Dominio Orientado a Características (FODA) y la aplicación de la arquitectura de microservicios en el desarrollo de software. Al final del capítulo, se presentará un listado de trabajos relacionados con el proyecto actual.

Marco Teórico

Línea de Producto de Software (LPS)

LPS es un paradigma que se está consolidando en ciertas áreas específicas de la producción de software, caracterizada por la reutilización planificada de core assets (componentes de software) en combinación con features (características), (Espinel, Carrillo, Flores, & Urbieta, 2022) para generar soluciones a través de productos de software.

LPS según (Clements & Northrop, 2001) es un conjunto de sistemas software, que contienen un marco común de características, las cuales satisfacen al segmento de mercado el cual se analiza; estas se desarrollan a partir de un grupo de funcionalidades comunes denominadas core assets (activos base).

Definición de los conceptos fundamentales dentro de LPS:

- Feature: se define como un elemento que puede estar interrelacionado con otras estructuras del sistema, o bien describir los requisitos no funcionales, y se utiliza para describir o diferenciar un producto dentro de la línea.
- Core asset**:** es un elemento de software que se utiliza en el proceso de desarrollo de uno o más productos de una LPS. Puede ser un componente de software, modelo de

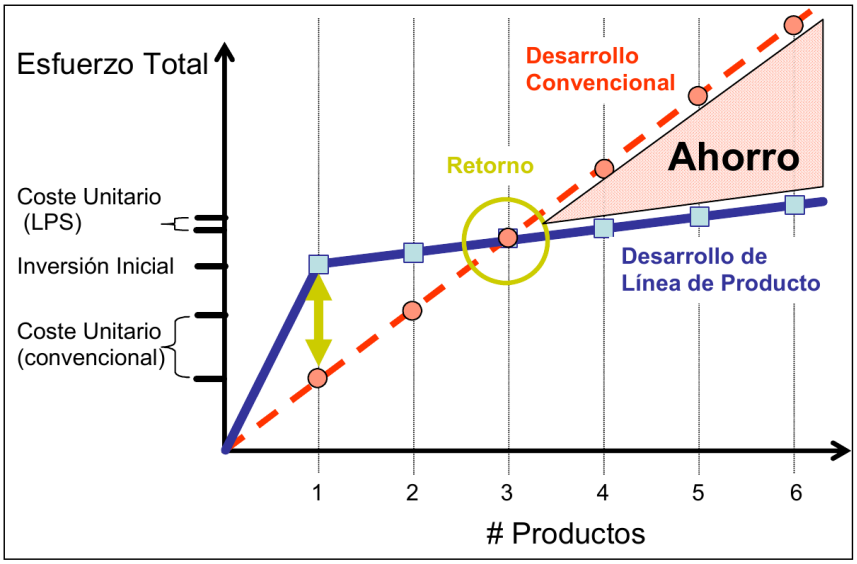
procesos, arquitectura o cualquier otro resultado que surja durante la construcción del producto.

La aplicación de LPS ha llevado al crecimiento del software en distintos segmentos de mercado, con el objetivo de satisfacer las necesidades específicas de cada uno de ellos y reducir el tiempo de desarrollo gracias a la reutilización de componentes.

La implementación de LPS puede aumentar la productividad de los desarrolladores; reducir el esfuerzo y costo en el desarrollo de software, mientras se mantiene la efectividad y funcionalidad de los productos de software. Los estudios de casos han demostrado que la productividad mejora significativamente en contraste con las metodologías convencionales. En la Figura 1, se presenta una gráfica comparativa que muestra la reducción del esfuerzo a largo plazo, lo que con lleva a la reducción en los costos de producción al aplicar LPS en lugar de una metodología convencional.

Figura 1

Comparación de desarrollo convencional vs LPS



Nota. Esta figura muestra una comparativa de esfuerzo total realizado en el desarrollo convencional de software y al utilizar LPS. Tomado de (Días)

La LPS permite al equipo de desarrollo de software utilizar un núcleo común para varios sistemas comunes, lo que resulta en un ahorro de recursos y tiempo a largo plazo. Los componentes de este núcleo (core assets) permiten el desarrollo de múltiples productos a partir del dominio común. La LPS sigue siendo un enfoque novedoso que requiere nuevas arquitecturas y métodos que proporcionen mecanismos para demostrar y representar las características y variabilidades del dominio. Algunos de los métodos disponibles incluyen:

- **Síntesis:** un enfoque amplio para construir sistemas de software que representen instancias de familias de sistemas con descripciones similares. (Software Productivity Consortium, 1993)
- **Abstracción, especificación y traslación orientada a la familia:** un análisis de características comunes del dominio que es importante para: identificar el contexto; describir el dominio; proporcionar un conjunto de términos clave; identificar características y variaciones comunes; cuantificar la variabilidad proporcionando parámetros de variación; e identificar y registrar información útil durante el análisis. (Weiss & Chi, 1999)
- **Ingeniería de Línea de Producto de Software:** un método para construir y utilizar líneas de productos. La estructura general de PuLSE incluye las siguientes etapas: desarrollo, componentes técnicos y componentes de soporte. (Bayer, et al., 1999)
- **Análisis de Dominio Orientado a Características:** un método para soportar la reutilización a nivel arquitectónico y funcional. (Kang, Cohen, Hess, Novak, & Peterson, 1990)

Proceso de desarrollo LPS

El proceso incluye tres actividades que no tienen un orden preestablecido para su ejecución, y que están en constante ejecución y retroalimentación. Según (Clements & Northrop, 2001), las actividades del paradigma LPS son: Ingeniería del dominio (desarrollo de

core assets), Ingeniería de la aplicación (desarrollo de productos) y gestión del proceso de la LPS (Espinosa, 2014).

En La **ingeniería del dominio** se crean los core assets, que son la base a partir de la cual se puede desarrollar un producto específico (Espinel, Carrillo, Flores, & Urbieta, 2022), es la encargada del análisis y desarrollo de todos los componentes que se requieren construir cada uno dentro de la LPS. Esta actividad produce uno o más componentes como: el plan de productos, la lista de productos que forman parte de la línea y la arquitectura de referencia (Clements & Northrop, 2001).

En la **Ingeniería de aplicación** tiene lugar la instanciación de productos de la línea (Espinel, Carrillo, Flores, & Urbieta, 2022) en esta actividad se implementa cada uno de los productos, estos son construidos utilizando los productos resultantes de la ejecución de la Ingeniería de dominio como los requisitos específicos de cada producto.

Un concepto central para la Ingeniería de dominio y aplicación es la **Gestión de Configuración de Software(GCS)** la cual se encarga de manejar la variabilidad de los core assets y los productos de la LPS, esta actividad se vuelve aún más compleja a medida que aumenta el número de combinaciones entre core assets y features (Espinel, Carrillo, Flores, & Urbieta, 2022), esta actividad se encarga de controlar y coordinar las distintas tareas llevadas a cabo durante el desarrollo de los core assets y de los productos de la LPS.

La Figura 2 representa las actividades que se realizan en el paradigma de LPS. Estas actividades son interactivas, están estrechamente relacionadas entre sí y evolucionan continuamente. Los core assets se utilizan para implementar nuevos productos, a menudo estos productos crean nuevas versiones de core assets, por lo cual la gestión de proceso de LPS se encarga de controlar la evolución del core assets, las características del producto y los productos desarrollados en la LPS (Quishpe, 2018).

Figura 2

Actividades esenciales para el desarrollo de LPS

Nota. Se observa las actividades esenciales para LPS. Tomado de (Northrop & Clements, 2012)

Análisis de Dominio Orientado a Características (FODA)

Según (Kang, Cohen, Hess, Novak, & Peterson, 1990) el método FODA permite la reutilización a nivel funcional y arquitectónico. Donde los productos de domino representan la funcionalidad y arquitectura los cuales se pueden adaptar al desarrollo de componentes software. El método FODA permite utilizar una metodología de análisis de requisitos, hasta el mantenimiento del sistema.

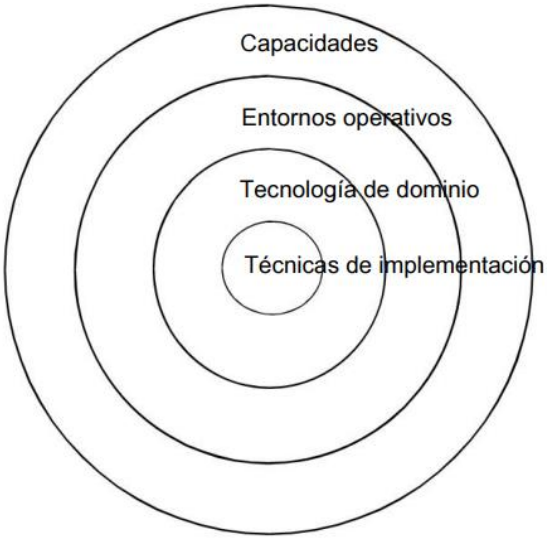
Parametrización de producto

La parametrización es idear elementos genéricos que se puedan adaptar a los valores de parámetros. Debido a que hay diversas maneras de implementar estas técnicas, como subrutinas, genéricos, marcos y procesadores, en donde se enfocan principalmente en la codificación del sistema. En un dominio dado, las aplicaciones comparten ciertas capacidades, pero estas se modelan como características únicas para cada usuario de las clínicas

veterinarias. Además, las aplicaciones pueden ejecutarse en diferentes entornos operativos, interactuar con diferentes dispositivos, y operar en distintos sistemas operativos o hardware. La figura 3 ilustra cómo el desarrollo de una aplicación puede implicar decisiones que varían con el tiempo.

Figura 3

Tipos de decisiones de desarrollo



Nota. En la figura se observa los tipos de decisiones de desarrollo que se presentan al momento de parametrizar los modelos y funciones. Tomado de (Kang, Cohen, Hess, Novak, & Peterson, 1990)

Los modelos de dominio arquitectónicos y funcionales pueden ser parametrizados al utilizar los problemas, soluciones y características como base.

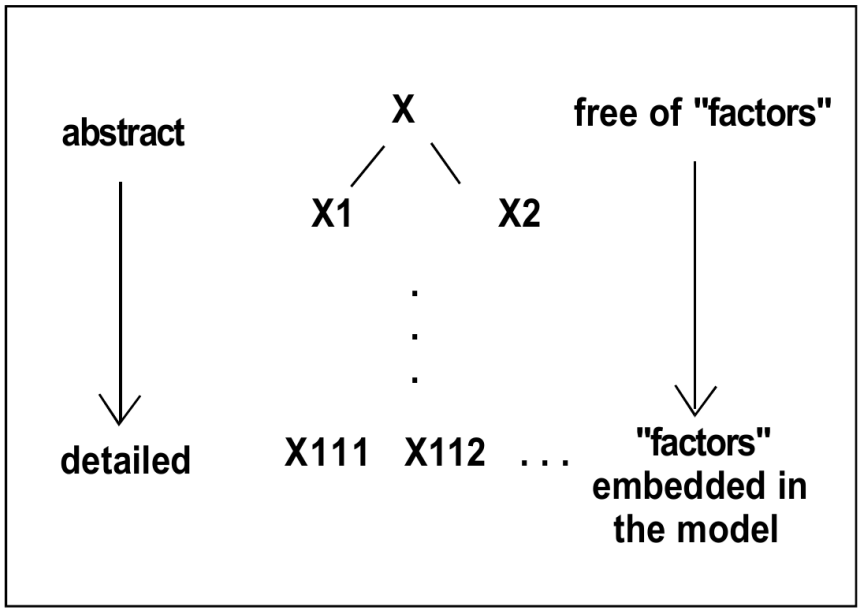
Niveles de abstracción

El método FODA se enfoca en simplificar los niveles de abstracción utilizando diferentes niveles donde una característica de un nivel representa una generalización de los niveles inferiores. Este proceso se lleva a cabo hasta llegar a características cada vez más específicas, lo que permite identificar los componentes esenciales de la aplicación. La Figura 4 muestra los

diferentes niveles de abstracción que conducen a los factores más fundamentales, que se convierten en elementos reutilizables de la LPS.

Figura 4

Modelo de abstracción



Nota. En la figura se observa los niveles de abstracción hasta llegar a los factores más esenciales. (Kang, Cohen, Hess, Novak, & Peterson, 1990)

Los productos derivados del análisis de dominio son distintas representaciones abstractas identificadas durante el proceso de análisis. Estos productos del dominio son organizados y presentados como resultado del análisis. Los modelos obtenidos a través del análisis de dominio se emplean en el desarrollo de los productos de la LPS. Los analistas de requisitos corroboran si el producto solicitado se ajusta a la LPS y, en caso de ser así, eligen las funcionalidades del sistema basándose en las abstracciones obtenidas del análisis de dominio.

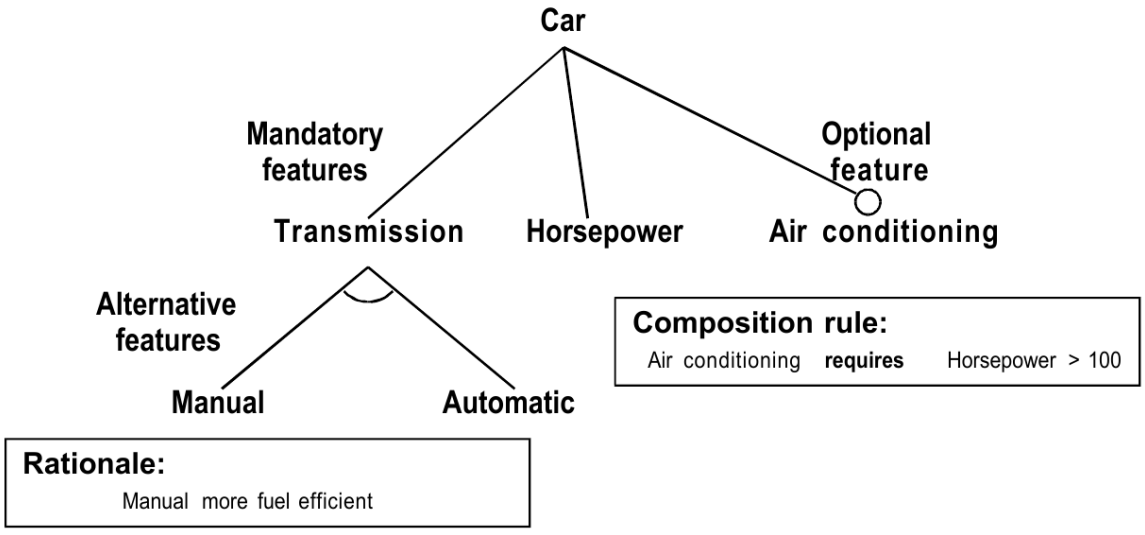
Modelamiento de dominio FODA

En el ámbito del desarrollo de software, se emplean diagramas de flujo de datos que resultan útiles solamente para el equipo de desarrollo, ya que los clientes pueden no entenderlos. Por el contrario, los clientes necesitan comprender los componentes y las capacidades fundamentales de la aplicación, como los servicios que ofrece, el rendimiento y la plataforma de hardware. En este sentido, el análisis FODA se enfoca principalmente en la perspectiva de la funcionalidad, detallando las características y los niveles de abstracción, que pueden ser utilizados para la creación de productos o la parametrización de nuevos modelos.

Las características representan los atributos que resultan esenciales para los usuarios finales, pues ayudan a definir la disponibilidad de funciones que puede ofrecer el sistema. En la Figura 5, se puede observar las decisiones que debe tomar una persona al momento de adquirir un automóvil, presentado en forma de un diagrama de características FODA.

Figura 5

Ejemplo diagrama de características



Nota. Ejemplo de diagrama de características mostrando la composición de un auto. Tomado de (Kang, Cohen, Hess, Novak, & Peterson, 1990)

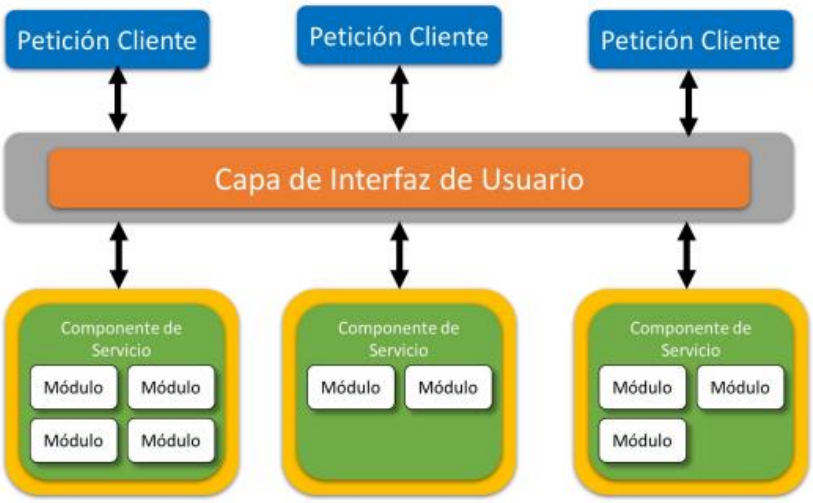
El diagrama de características expone los distintos elementos que conforman el dominio, así como las conexiones que existen entre ellos. Cada característica se presenta mediante un nombre sencillo que se desarrolla en el diccionario del diagrama. Las características pueden ser obligatorias u opcionales, y se indican en el diagrama mediante un círculo pintado o uno sin pintar, tal como se aprecia en la Figura 5. La elección de características opcionales se efectúa con base en los objetivos que el cliente persigue.

Arquitectura de microservicios

La arquitectura de microservicios tiene como objetivo el desarrollo de aplicaciones mediante un conjunto de pequeños servicios, cada uno de ellos ejecutado en su propio proceso y mecanismo de comunicación a través de APIs HTTP. Estos servicios se construyen en torno a las capacidades o funcionalidades de negocio y se despliegan de forma automatizada. La arquitectura de microservicios fomenta el desarrollo y la implementación de aplicaciones que se componen de unidades independientes, modulares, autónomas y autocontenidas, a diferencia de la tradicional arquitectura monolítica, tal como se ilustra en la Figura 6.

Figura 6

Ejemplo básico de arquitectura de Microservicio



Nota. Ejemplo de estructura de arquitectura de un microservicio. Tomado de (Lopez & Maya , 2017)

La arquitectura de microservicios se enfoca en el desarrollo de software mediante componentes, permitiendo la independencia en el desarrollo de cada uno de ellos. Esta estrategia de desarrollo posibilita la creación de grupos de trabajo específicos para cada microservicio, lo que facilita su mantenimiento, reutilización y escalabilidad. Cuando se combina con LPS, esta arquitectura permite representar cada uno de los features como un componente o microservicio del sistema, lo que resulta en una implementación más organizada y rápida de cada uno de los features, y los hace mantenibles, reutilizables y escalables. Además, la arquitectura de microservicios se convierte en un core asset fundamental para la LPS, en el cual los microservicios básicos o componentes base del FODA se convierten en módulos reutilizables de LPS.

Desarrollo Global de Software (DGS)

La globalización, según la Real Academia Española, se define como “la tendencia de los mercados y de las empresas a extenderse, alcanzando una dimensión mundial que sobrepasa las fronteras nacionales” (Piattini, Vizcaíno, & Garcia, 2014). un proyecto DGS depende de cuatro factores las distancias geográficas, temporal y sociocultural, diferencias lingüísticas que podrían causar retrasos en entregas.

En el contexto de DGS, un proyecto puede ser descompuesto en varios subproyectos y sus respectivas tareas pueden ser distribuidas entre diferentes nodos o sitios que colaboran en el proyecto. Existen tres formas de distribución de trabajo: "basado en módulos", donde se divide el proyecto en artefactos independientes que son asignados a diferentes sitios; "basado en fases", donde cada fase del desarrollo es asignada a un sitio o grupo de desarrollo; y "follow the sun", donde los grupos de desarrollo se encuentran distribuidos geográficamente para

garantizar la producción las 24 horas del día, aprovechando la diferencia horaria entre los sitios.

Formación de equipos distribuidos

En el contexto de DGS, la formación de equipos distribuidos es un factor crítico que requiere una cuidadosa consideración. Es esencial identificar los roles y responsabilidades de cada miembro del equipo, para así crear equipos que se complementen entre sí en términos de habilidades y capacidades. En lugar de simplemente asignar tareas a los miembros del equipo, se debe tener en cuenta la formación de equipos basados en características, en la que cada miembro puede contribuir de manera efectiva a la ejecución de un objetivo común, de acuerdo a sus habilidades y destrezas particulares. “También se debe tomar en cuenta de ser factible, los miembros de un mismo equipo pertenezcan a una misma localidad” (Piattini, Vizcaíno, & Garcia, 2014). Para la organización de equipos de características se propone los siguientes métodos:

Reorganización big-bang: Esta práctica implica la organización de equipos de trabajo mediante la agrupación de varios especialistas en un solo equipo, con el fin de asegurar que el equipo tenga conocimientos sobre la mayoría de los procesos o sistemas involucrados. De esta forma, se busca que el equipo esté más completo y capacitado para abordar los diferentes desafíos del proyecto de software.

Expansión gradual de la responsabilidad: Con este método se llevan a cabo pequeñas modificaciones en la composición de los equipos de trabajo, con el objetivo de ir ajustándolos hasta llegar a un equipo basado en características.

Introducción gradual de equipo de características: En este método se seleccionan las características más relevantes del product backlog y solo a ellas se les aplica el cambio de equipo basado en características.

DevOps

DevOps es una cultura que enfatiza la cooperación en equipo y alienta a las empresas a ofrecer funciones de productos de software a través de procesos automatizados (Bijwe & Shankar, 2022).

DevOps está definida por dos palabras: desarrollo (Dev) y operaciones (Ops); es una metodología que describe formas para mejorar los procesos de una idea para pasar del desarrollo a la implementación dentro de un entorno de producción de esta forma generar un valor agregado para el cliente. Disminuyendo tiempos de entrega, esta forma describe como el equipo de desarrollo y el de operaciones están relacionados y además deben contar con una buena comunicación (Cusco, 2022).

DevOps y el ciclo de vida de las aplicaciones

DevOps tiene un efecto en el ciclo de vida de una aplicación a través de múltiples etapas que incluyen la planificación, el desarrollo, la entrega y el uso, en las cuales cada fase está intrínsecamente vinculada con las demás y se influyen mutuamente dentro del proceso.

Planificación: Los equipos describen las características y la funcionalidad de las aplicaciones y los sistemas que se van a crear, realizando el seguimiento al progreso en forma general y pormenorizada, la creación de registros de trabajo pendiente, el seguimiento de errores y visualización del progreso son algunas de las formas en que los equipos DevOps realizan la planificación (Microsoft, n.d.).

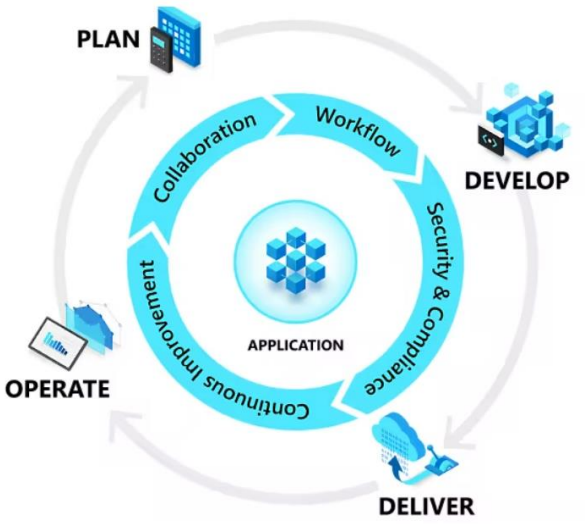
Desarrollo: Incluye todos los aspectos de la programación (escritura, pruebas, revisión e integración), la compilación de ese código que permitan implementar en varios entornos, en donde los equipos buscan innovar la eficiencia sin sacrificar la calidad, la estabilidad o productividad de sistemas (Microsoft, n.d.).

Entrega: Es el proceso de implementar aplicaciones en entornos de producción de un modo constante y confiable. La fase de entrega incluye también la implementación y la configuración de la infraestructura básica totalmente gobernada que constituye esos entornos (Microsoft, n.d.).

Uso: Los equipos trabajan para asegurar la confiabilidad, la alta disponibilidad y el objetivo de ningún tiempo de inactividad del sistema, al tiempo que refuerzan la seguridad. Buscan identificar los problemas antes de que afecten a la experiencia del cliente y mitigarlos rápidamente a medida que surgen (Microsoft, n.d.).

Figura 7

Ciclo de vida de las aplicaciones



Nota. Etapas que presenta el ciclo de vida DevOps. Tomado de (Microsoft, n.d.).

DevOps y la nube

La adopción de soluciones en la nube ha modificado el modo en que los equipos desarrollan e implementan utilizando aplicaciones, ya que la posibilidad de proporcionar y configurar entornos en la nube de diferentes regiones se relaciona con la agilidad en los equipos para desplegar aplicaciones y crear entornos complejos en la nube. De igual manera,

la capacidad de apagar estos recursos cuando no son necesarios ya que brindan una innovación más rápida, con la reducción de costos operativos, una ejecución más eficiente de la infraestructura y la posibilidad de escalar en función de los cambios en las necesidades empresariales.

SAFe 5

SAFe 5 es un marco para agilidad empresarial, el cual integra Lean, Agile y DevOps en un sistema integral, ayudando a las empresas a prosperar ofreciendo productos y servicios innovadores de forma rápida, predecible y de calidad (Scaled Agile Framework, 2021).

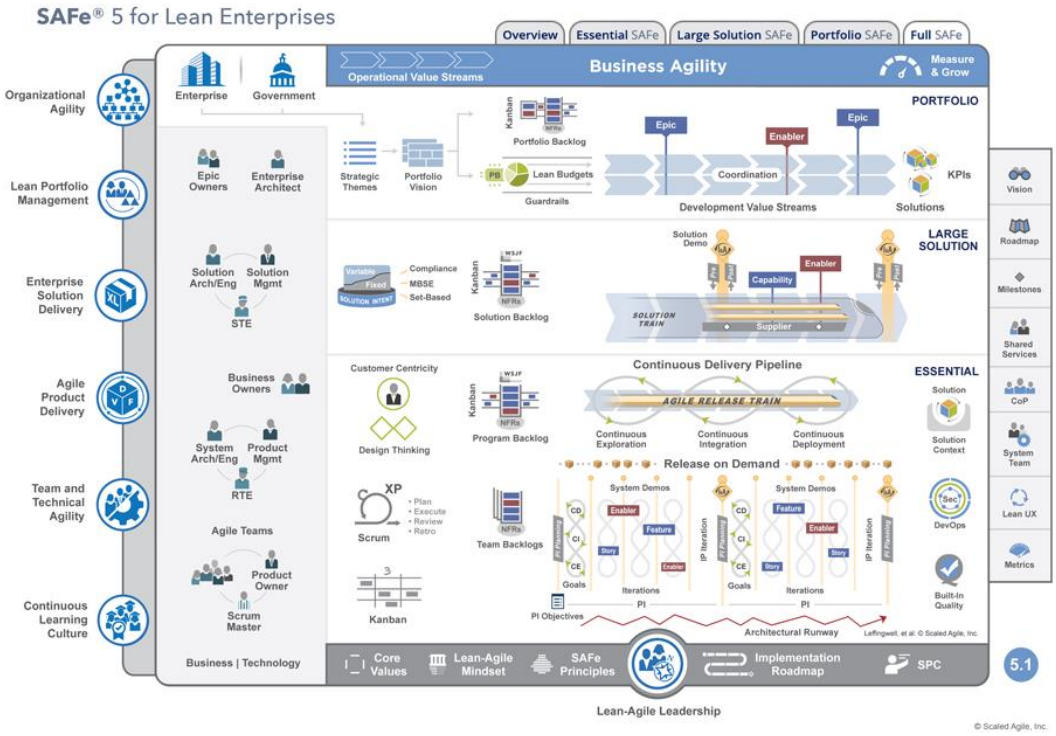
Las empresas tienen la capacidad de ajustar el marco de trabajo SAFe 5 a sus propias necesidades de negocio. Este marco de trabajo está diseñado para ser escalable, lo que significa que puede adaptarse tanto a la gestión de un número reducido de equipos de trabajo, como a la administración de cientos o incluso miles de personas.

El marco SAFe 5 es adaptable y se actualiza de forma constante, ya que permite que se ajuste ágilmente a las necesidades de las clínicas veterinarias. Durante la pandemia de COVID-19, SAFe 5 se vio obligado a adaptarse a las circunstancias actuales, como lo hicieron también muchas empresas, lo que demuestra su capacidad para responder a situaciones cambiantes.

La Figura 8 se puede observar una visión general de los principales componentes que forman parte del marco de SAFe 5, incluyendo sus valores, principios fundamentales, roles, artefactos y elementos de implementación. En donde estos componentes son esenciales para comprender y aplicar correctamente el marco SAFe 5.

Figura 8

Panorama general SAFe 5



Nota. Componente dentro del marco SAFe 5. Tomado de (Scaled Agile Framework, 2021).

Scrum

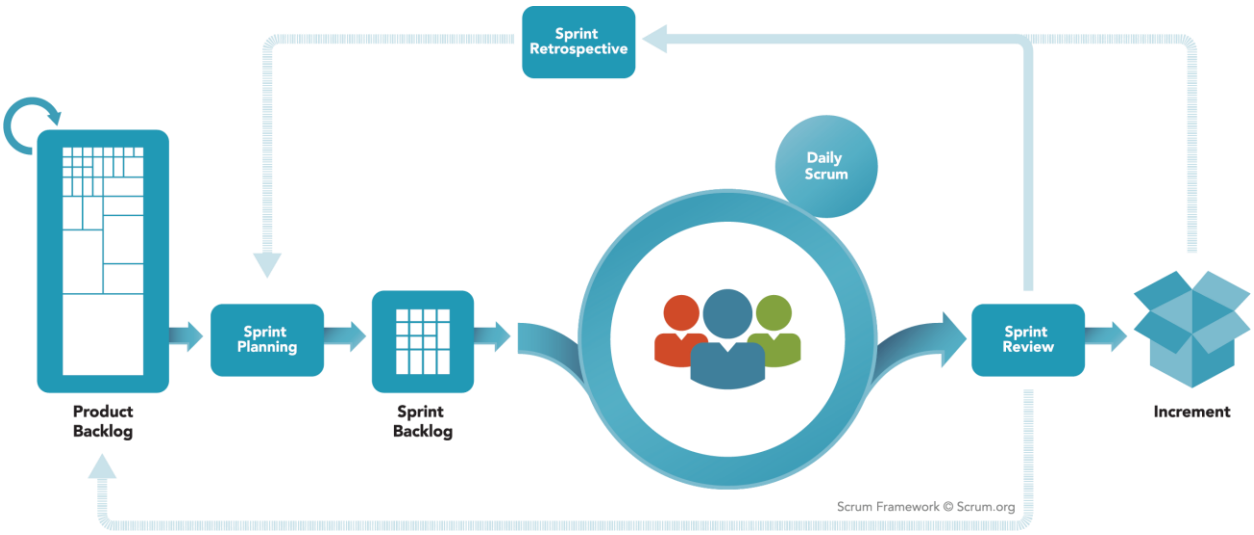
Scrum es una forma de hacer el trabajo en equipo en pequeñas piezas a la vez, con experimentación continua y bucles de retroalimentación en el camino para aprender y mejorar a medida que avanza (Scrum, 2023).

Scrum es una metodología de gestión de proyectos de software que se enfoca en ser rápida y ágil. Su enfoque se basa en la observación, la experiencia y la experimentación. Scrum utiliza un enfoque iterativo en el que el proyecto, este se divide en ciclos cortos llamados sprints. Dentro de cada sprint, se trabaja en incrementos y se lleva a cabo una retroalimentación continua y adaptación a los procesos de entrega. Este enfoque iterativo permite a los equipos de desarrollo de software hacer entregas tempranas y frecuentes, lo que

a su vez permite obtener retroalimentación temprana del cliente y adaptar el proyecto según sea necesario.

Figura 9

Funcionamiento de scrum



Nota. En la figura se muestra los procesos que forman parte de scrum. Tomado de (Scrum, 2023).

Trabajos relacionados

Gestión de Configuración y Línea de Productos para Mejorar el Proceso Experimental en Ingeniería del Software: La Ingeniería del Software (IS) Empírica adopta el método científico a la IS para facilitar la generación de conocimiento. Una de las técnicas empleadas, es la realización de experimentos. Para que el conocimiento obtenido experimentalmente adquiera el nivel de madurez necesario para su posterior uso, es necesario que los experimentos sean replicados. Para adoptar la GCS a experimentación, se comienza realizando un estudio del proceso experimental como transformación de productos; a continuación, se realiza una adopción de conceptos fundamentada en los procesos del desarrollo software y de experimentación; finalmente, se desarrollan un conjunto de instrumentos, que se incorporan a un Plan de Gestión de Configuración de Experimentos (PGCE). Para adoptar la LPS a

experimentación, se comienza realizando un estudio de los conceptos, actividades y fases que fundamentan la LPS; a continuación, se realiza una adopción de los conceptos; finalmente, se desarrollan o adoptan las técnicas, simbología y modelos para dar soporte a las fases de la Línea de Producto para Experimentación (LPE) (Espinosa, 2014).

Caso de Aplicación para Crear Tiendas Virtuales Usando Líneas de Productos de Software: La ingeniería de líneas de productos de software es un paradigma que propone la reutilización planificada, para aprovechar elementos comunes y variables que tienen productos de software que pertenecen a un mismo dominio. El paradigma de las líneas de productos de software puede ser aplicado al contexto de las tiendas virtuales, pues existen elementos que podrían ser capitalizados para elaborar una familia de tiendas virtuales en lugar de crear solo una (Rincón, Rodríguez, Martínez, & Pabón, 2015).

Subconjuntos Mínimos de Corrección para explicar características muertas en Modelos de Líneas de Productos. El caso de los Modelos de Características: Las líneas de productos son un paradigma que permite administrar eficientemente un conjunto de productos con elementos comunes y variables que pertenecen a un dominio en particular. Este paradigma ofrece beneficios como la reutilización, la disminución de errores y la disminución de tiempos y costos de producción. Los beneficios propuestos para las líneas de productos pueden ser extensibles al software, pues en el desarrollo de software es necesario administrar la reutilización y la variabilidad, este paradigma aplicado al software se conoce como Líneas de Productos de Software (Rincón, Giraldo, Mazo, Salinesi, & Díaz, 2013).

Aplicación De La Refactorización De Software Con La Herramienta Foda: La metodología Feature Oriented Análisis de Dominio Orientado a Características (FODA) logrando software exitoso para refactorizar el código y reutilización. Define los siguientes conceptos; Dominio: Conjunto de aplicaciones actuales y futuras que comparten un conjunto de capacidades y datos comunes. Análisis de dominio: El proceso de identificar, recopilar organizar y representar la

información relevante de un dominio basado en el estudio de los sistemas existentes y su desarrollo los conocimientos históricos adquiridos a partir de las expectativas de dominio, la teoría subyacente y la tecnología emergente dentro del dominio. teoría subyacente y la tecnología emergente dentro del dominio. Característica: Aspecto, cualidad o característica prominente o distintiva, visible para el usuario, de un sistema o sistemas informáticos. o característica de un sistema o sistemas de software (Malathi & Sudhakar, 2018).

Metodologías de Desarrollo de Software: En la actualidad la rapidez y el dinamismo en la industria del software han hecho replantear los cimientos sobre los que se sustenta el desarrollo de software tradicional. Estudios recientes y el mismo mercado actual está marcando la tendencia en la ingeniería del software teniendo como características principales atender a las necesidades de rapidez, flexibilidad y variantes externas que hacen de nuestro entorno una ventaja más competitiva al aumentar la productividad y satisfacer las necesidades del cliente en el menor tiempo posible para proporcionar mayor valor al negocio. Ante esta situación, el grado de adaptación de las metodologías tradicionales a estos entornos de trabajo no eran del todo eficientes y no cubrían las necesidades del mercado actual. En la actualidad existen una gran cantidad de metodologías para el desarrollo de software, separadas en dos grandes grupos; las metodologías tradicionales o pesadas y las metodologías agiles (Maida & Pacienzia, 2015).

Aplicación de la metodología de desarrollo ágil Scrum para el desarrollo de un sistema de gestión de empresas: Este proyecto describe el desarrollo de un sistema de gestión de empresas, también conocido como ERP. El cual se desarrolla empleando la metodología ágil Scrum. El proyecto se divide en varias iteraciones en las cuales se obtiene como resultado un prototipo. El sistema está compuesto por una aplicación cliente para uso del usuario y una base de datos centralizada donde se almacenarán todos los registros relativos a la empresa (Urteaga, 2015).

Técnicas de análisis de dominio: organización del conocimiento para la construcción de sistemas software: La comunidad dedicada al diseño de sistemas informáticos ha propuesto una serie de técnicas conocidas como “análisis de dominio”, que tienen como finalidad la captura y adquisición del conocimiento que deben almacenar y procesar las aplicaciones informáticas orientados a una misma función o uso. El análisis de dominio ofrece herramientas para capturar la información crítica sobre las entidades, datos y procesos característicos de un área de actividad, y facilitar la especificación, el diseño y la construcción de los sistemas que soporten dichas actividades. Como disciplina, el análisis de dominio pretende la reutilización intensiva del conocimiento y capitalizar la experiencia adquirida en el diseño y construcción de sistemas informáticos de un mismo tipo (Brun, 2007).

Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web: Actualmente, el proceso de desarrollo de software que realiza la Coordinación General de Tecnologías de la Información y Comunicación (CGTIC) de la Asamblea Nacional del Ecuador (ANE) constituye el empleo de una arquitectura de software tradicional o monolítica que ha sido adoptada del lenguaje de programación utilizado, la plataforma o de la experiencia del personal del área de desarrollo; por el aspecto monolítico, este tipo de aplicaciones empaquetan toda la funcionalidad en una sola y gran unidad ejecutable (un solo archivo o aplicación), lo que ha provocado dificultades en aspectos como mantenimiento, escalabilidad y entregas. El objetivo del presente estudio fue identificar las tecnologías, metodología y arquitectura que utiliza la CGTIC para el desarrollo de aplicaciones web y la correspondiente identificación de las tecnologías existentes para el desarrollo e implementación de microservicios, utilizando como base de la investigación un enfoque cualitativo, con un tipo de investigación descriptiva y diseño documental (López, 2017).

Automatización de los procesos de construcción de software mediante la aplicación de técnicas DEVOPS: Partiendo de una aplicación monolítica con procesos de construcción

manuales, en este trabajo se desarrollan pruebas para verificar su correcto funcionamiento para posteriormente evolucionar dicha aplicación, separarla en pequeños servicios independientes y automatizar su despliegue empleando técnicas DevOps. Se lleva a cabo un ciclo completo de software, centrado en la construcción del entorno de trabajo, gestión de repositorios Git, orquestación de servicios mediante contenedores, testing, integración continua (IC) y, por último, monitorización (Fernández, 2022).

Aplicación basada en arquitectura de microservicios: El principal objetivo de este trabajo es implementar una arquitectura de microservicios con una malla de servicios (service mesh). Hablaremos de las ventajas y desventajas de una arquitectura de microservicios frente a una monolítica, y las ventajas que ofrecen las mallas de servicio a nivel seguridad, conectividad, observabilidad y control. Los servicios de este proyecto han sido implementados con los diferentes lenguajes que predominan hoy en día, como Python, Java, PHP o Spring Boot, entre otros. Además, contienen varias APIs REST y una base de datos SQL. En definitiva, tratamos los principales paradigmas de los desarrolladores DevOps (Navarro & Cabrera, 2020).

Conclusiones

Se logró con éxito definir todos los conceptos relevantes para el proyecto y se examinó un listado de trabajos relacionados con el proyecto.

Capítulo III

Implementación

Introducción

En el presente capítulo se procede a detallar todo el proceso que se lleva a cabo para la implementación del sistema. Se inicia explicando la metodología del proyecto utilizando el marco de trabajo SAFe 5, posteriormente se detalla el diseño del sistema utilizando una metodología orientada a objetos. Finalmente se describe el proceso de desarrollo y cada uno de los sprints realizados en el proyecto.

Metodología

Para la implementación del sistema se utilizó el marco de trabajo ágil SAFe 5 y DGS, se procedió a organizar los diferentes grupos de trabajo basados en conocimientos y experiencia personal de cada uno de los integrantes del equipo de trabajo. Como resultado se obtuvieron 3 equipos de trabajo como se muestra en la Tabla 1, a los cuales se les asignaron sus diferentes responsabilidades basadas en las características del sistema y en la localidad de cada uno de los integrantes de los equipos.

Tabla 1

Asignación de equipos y responsabilidades.

Equipo	Integrantes	Localidad	Responsabilidades
A	- Julio Castro - José Poveda	Ambato	Desarrollo del front end del sistema
B	- Martín Camacho - José Núñez	Ambato – Latacunga	Desarrollo del back end del sistema y despliegue
C	- Paola Romo - Ana Sánchez	Ambato	Desarrollo del back end y seguridades

Nota. En la tabla se muestran los equipos de trabajo y las responsabilidades asignadas a cada uno de ellos.

Cada uno de los equipos definidos en la Tabla 1 cuenta con un especialista en: base de datos, back end, APIs y front end. Para la organización de estos equipos de características se tomó el método de big-bang. En el equipo A encargado del front end del sistema consta de un especialista en consumo APIs, y otro especialista en HTML y CSS. En el equipo B existen especialistas de base de datos, nodejs y despliegue continuo. Finalmente, el equipo C consta de especialistas de Seguridades, back end y nodejs. Dadas estas especialidades se procedió a la asignación de las responsabilidades a cada uno de los equipos como se detalla en la Tabla 1. Al utilizar la metodología ágil Scrum se procedió a asignar roles de scrum a los integrantes del proyecto. Estos roles se encuentran definidos en la Tabla 2.

Tabla 2

Roles de scrum

Rol scrum	Asignado
Scrum máster	Julio Castro
Product owner	Martín Camacho
Desarrolladores back end	Martín Camacho
	José Núñez
	Pao Romo
	Ana Sánchez
Desarrolladores front end	Julio Castro
	José Poveda

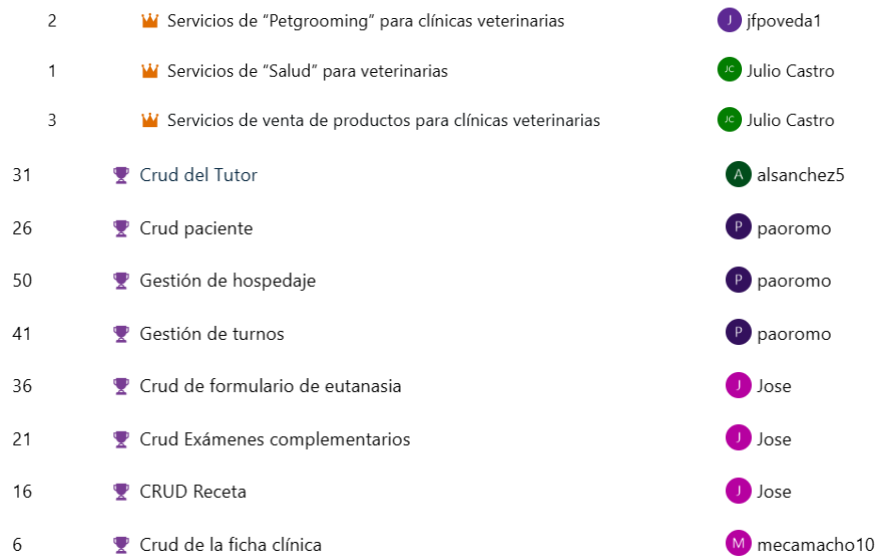
Nota. Se presenta una tabla especificando los roles tomados en el proyecto.

Posteriormente se trasladó a Azure DevOps el cual permite realizar la implementación del sistema de una manera co-localizada. Aquí se pasó cada una de las Épicas, características,

historias de usuario. Se asigno encargados de gestionar cada las épicas y características como se muestra en la Figura 10.

Figura 10

Asignación de responsables dentro de Azure DevOps



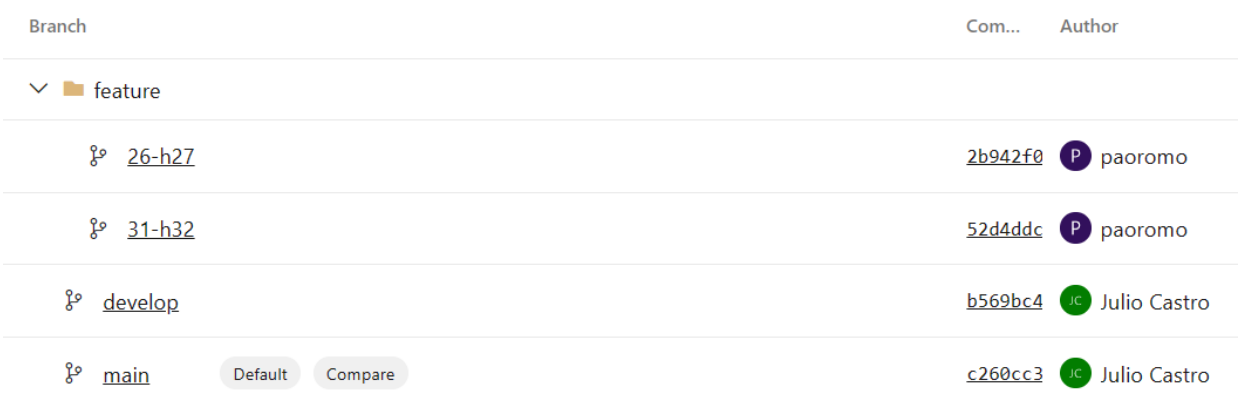
Nota. En la figura se muestra la asignación de responsabilidades de las épicas y las características del sistema.

Para el desarrollo del sistema se realizó un repositorio git a cada uno de los microservicios y proyectos existentes como se muestra en la Tabla 7. En la Figura 8 se puede observar que cada uno de los ítems posee un identificador único, el cual se lo utiliza para la gestión de ramas de los repositorios. Las ramas vienen dadas bajo el marco de trabajo de git Flow, buscando así una mejor relación con las características e historias de usuario del sistema. Para ello se determinó un estándar de nombres para las ramas, git Flow nos otorga por defecto el prefijo 'feature/', posterior a esto se agrega el identificador de la característica a la cual hace referencia y también al de la historia de usuario. En consecuencia, se determinó el formato: 'feature/{id de la caracteristica} – h{id de la historia de usuario}'. La utilización de esta

forma nos da como resultado una mayor organización dentro de las ramas y un enlace directo a los ítems de trabajo, como se muestra en la Figura 11.

Figura 11

Organización de las ramas en el repositorio MicroservicioPacienteTutor



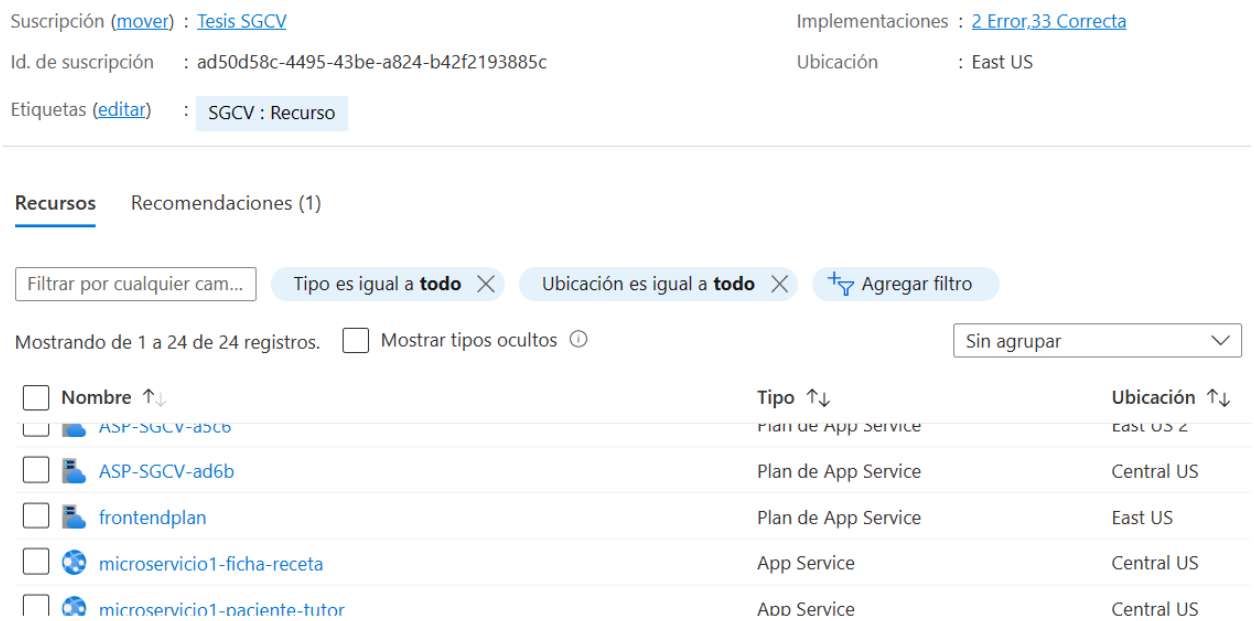
Nota. Ejemplo de ramas en los repositorios.

Terminada una historia de usuario se la evalúa independientemente en su propia rama, luego se procede a realizar un pull request para enviar los cambios hacia la rama develop en ella el coordinador de la característica se encarga de evaluar la integración del nuevo código. Una vez realizado el pull request se prueba el código, finalmente se lo une a la rama main para su posterior publicación. El repositorio de Azure DevOps se encuentra en el Anexo G.

Para la administración de Azure se creó un grupo de recursos en el cual se crearon todos los componentes que se requirieron para el despliegue, entre ellos se encuentran: bases de datos elásticas, app services y plan services. Todos estos servicios de Azure se pueden observar en la Figura 12.

Figura 12

Grupo de recursos de Azure



Nota. Recursos utilizados en azure.

Diseño del sistema

El proceso del diseño del sistema inicia con la toma de requisitos a 4 clínicas veterinarias, este proceso se realizó mediante la utilización de entrevistas a cada una de las clínicas veterinarias. Se procedió a realizar las entrevistas utilizando el formato descrito en el Anexo A, con la autorización de cada clínica se procedió a grabar cada una de las entrevistas para su posterior transcripción.

Tabla 3

Resumen de resultados de la entrevista.

Pregunta	Resultado
¿La veterinaria tiene un sistema de gestión?	En mayoría las clínicas veterinarias no poseen sistemas de gestión, es decir llevan sus procesos internos de una forma manual.

Pregunta	Resultado
¿Qué servicios ofrece su veterinaria?	- Consultas - Vacunación - Cirugía - Tienda de accesorios - Peluquería - Laboratorio - Hotelería
¿Qué proceso lleva para cumplir con sus servicios?	Se brinda atención dependiendo del tipo de consulta a realizarse, en la mayor parte del tiempo se prioriza por orden de llegada a la clínica. En caso de emergencia se procede a estabilizar al paciente y se procede con la revisión. En el caso de una consulta ordinaria se realiza el ECOP de manera convencional tomando los datos del paciente como peso, síntomas, etc.
¿Realiza agendamiento de citas?	Únicamente se realiza en caso de requerir cirugías, caso contrario no se programa citas, ya que, esto genera molestia en las personas que esperan ser atendidas por orden de llegada.
¿Qué roles existen dentro de su veterinaria?	El dueño viene a administras toda la clínica veterinaria, los veterinarios se dedican principalmente al área de salud y Pet Grooming, y existe personas encargadas de la tienda de la veterinaria (petshop).

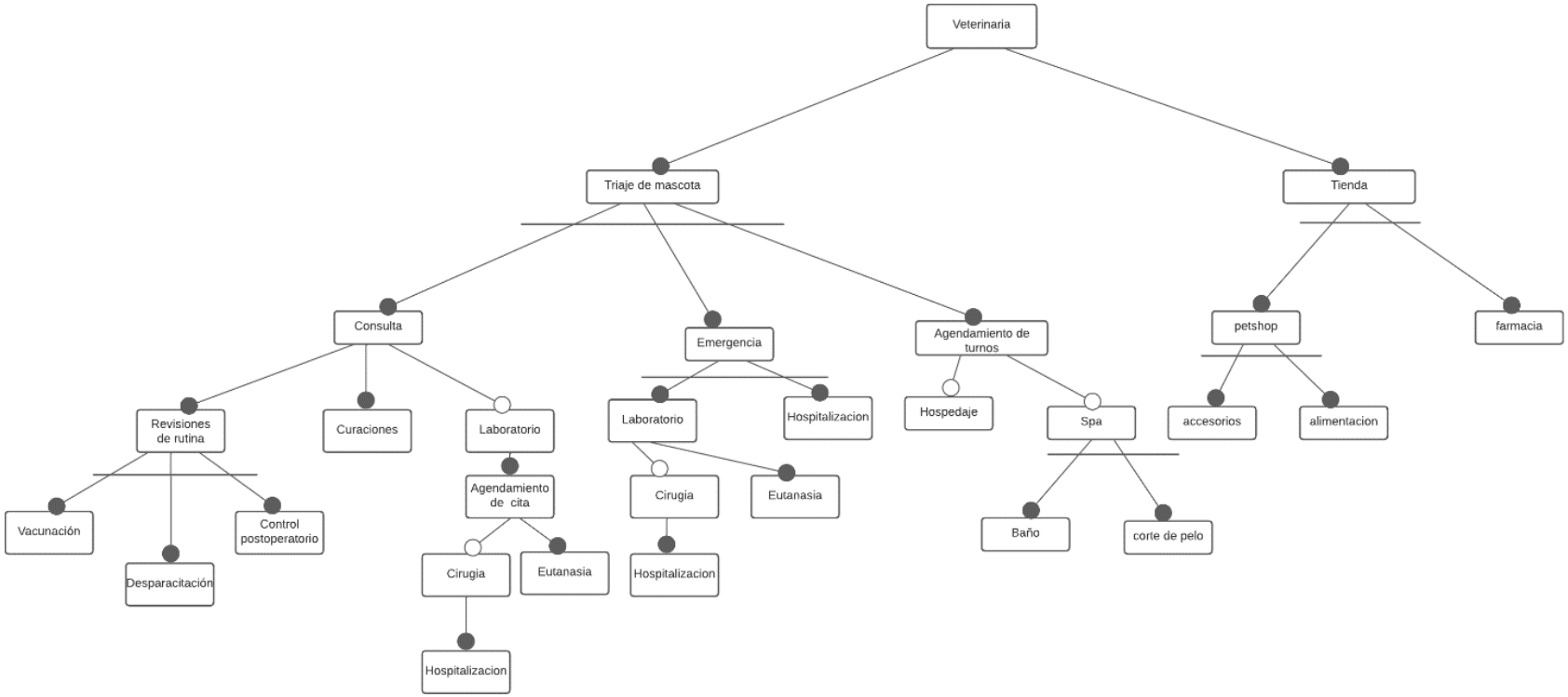
Nota. En la tabla se detalla un resumen de las respuestas obtenidas mediante la entrevista realizada a las diferentes clínicas veterinarias, se expande los resultados de las entrevistas en el Anexo A.

Una vez obtenidos los resultados se procede aplicar el paradigma de LPS el cual se obtiene mediante la utilización del método FODA. A partir de la información obtenida de la toma de requisitos se procedió con el modelado de diagrama de características el cual mostrará el dominio aplicado de las clínicas veterinarias.

Para la obtención del diagrama de características se procedió a realizarlo en base a los conocimientos obtenidos previamente en las entrevistas. Como resultado se llegó a la obtención de la primera versión de este diagrama el cual se encuentra detallado en la Figura 13. En las revisiones de este diagrama de observo que existían algunos procesos que no se encontraban detallados de una manera clara, por lo cual se definió que este diagrama de características es poco eficiente y se procedió a realizar sus respectivas correcciones.

Figura 13

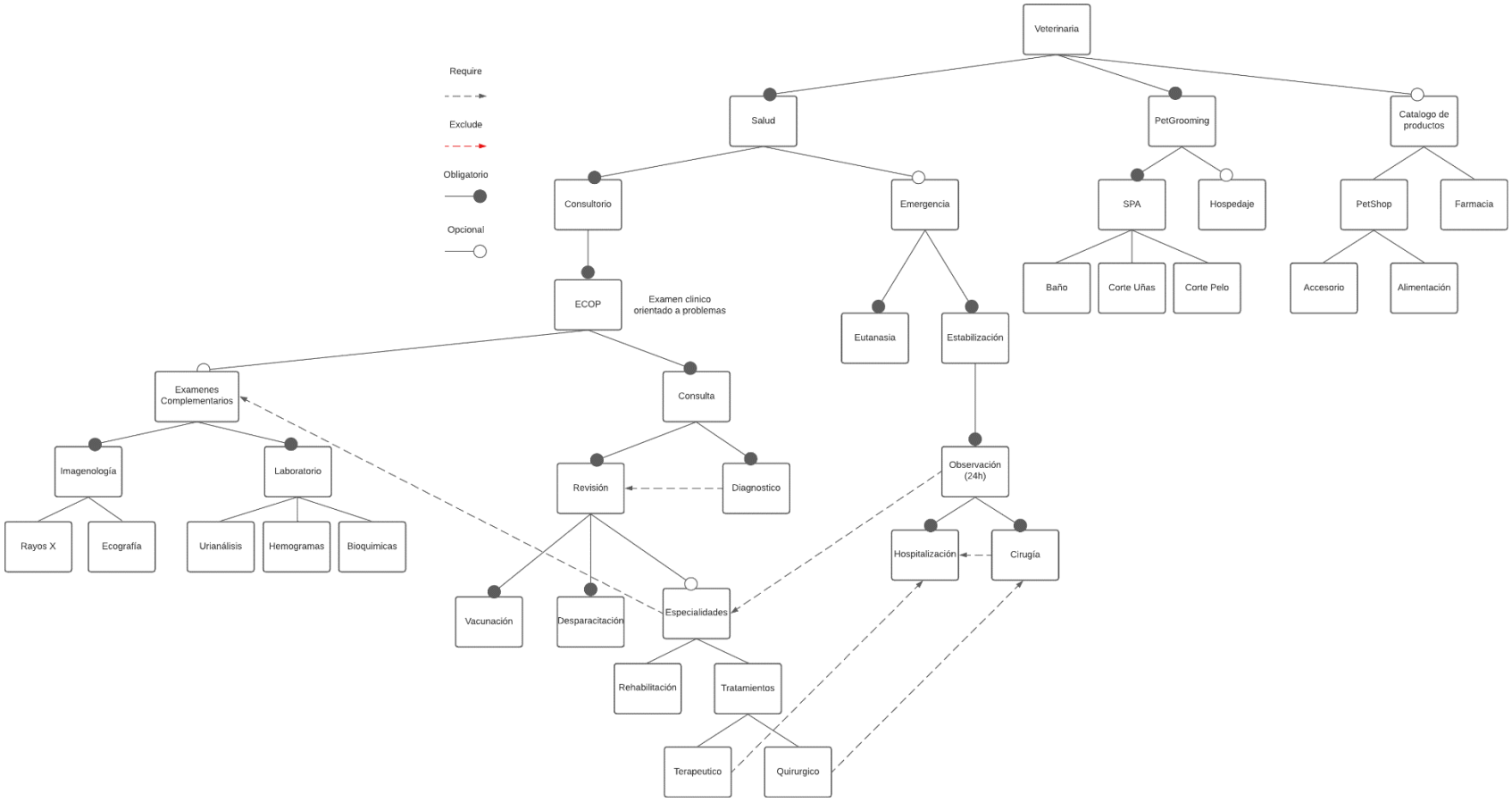
Primera versión del diagrama de características



Nota. Se muestra la primera versión del diagrama de características.

Figura 14

Ultima versión del diagrama de características.



Nota. En la figura se muestra versión oficial del diagrama de características.

Debido a la falta de características dentro del diagrama, ya que existían procesos que no se detallaban correctamente, se realizó uno nuevo con la ayuda de un veterinario. Proporcionando nueva información que ayudó a la obtención de la última versión del diagrama el cual se encuentra en la Figura 14, por lo que al final se definió que el diagrama de características es eficiente, ya que muestran detallados los procesos que se lleva a cabo dentro de una veterinaria. Todas las versiones del diagrama de características se encuentran en el Anexo B

Una vez definido el diagrama de características, da lugar a estructurar el portafolio SAFe 5, donde se detalla los flujos de desarrollo, los cuales ayudaran al proceso de implementación del sistema.

En base al diagrama de características, se determinó que las clínicas veterinarias comparten procesos similares: Salud, Pet Grooming, Gestión de Productos, estos procesos dan lugar a cada una de las épicas dentro del portafolio SAFe 5, que se muestra en la Tabla 4.

Tabla 4

Resumen de épicas del proyecto

Épica	Descripción	Resultados Comerciales
Salud	Las clínicas veterinarias requieren automatizar sus procesos de registro de información de clientes y mascotas, para poder generar el historial clínico	Los clientes pueden acceder al sistema para consultar y programar citas. Disminuye el papeleo realizado por el veterinario. El veterinario puede registrar y consultar las fichas e historial clínico.

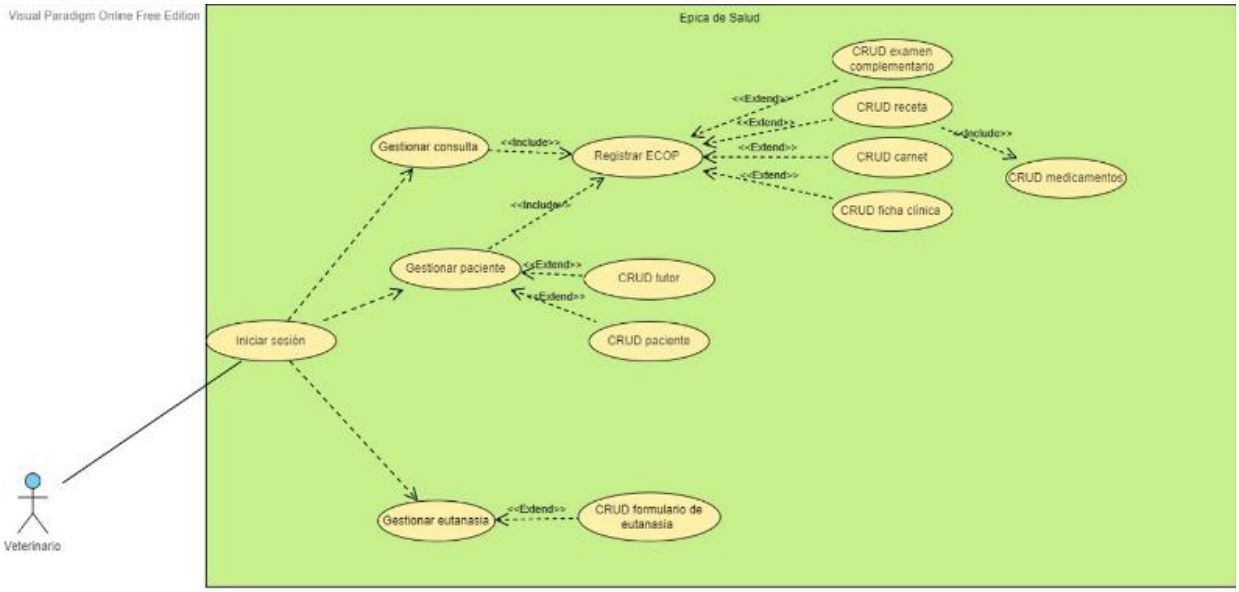
Épica	Descripción	Resultados Comerciales
Pet Gromming	Las clínicas veterinarias tienen como proceso el ingreso y salida de spa y hospedaje.	Los clientes pueden acceder al sistema para consultar y agendar turnos. Agiliza el proceso para agendar turnos. El veterinario puede consultar los turnos programados para cada día.
Gestión de Productos	Las clínicas veterinarias necesitan un control de inventario de los productos que ingresan y salen de su inventario.	La clínica veterinaria puede controlar y tener conocimientos de los productos que entran y salen de la clínica. Se puede saber de manera exacta la disponibilidad de productos de la clínica, y no se tendrá problemas para el despacho de estos.

Nota. Información resumida de las épicas dentro del plan de trabajo SAFe 5 de las clínicas veterinarias.

Cada una de las épicas presenta sus respectivos casos de uso, donde describen el comportamiento del sistema y una descripción textual de las acciones que tienen los actores previstos dentro del sistema, a continuación, se presenta un ejemplo de caso de uso aplicado a la épica. Todos los diagramas de caso de uso se encuentran en el Anexo C.

Figura 15

Diagrama de caso de uso



Nota. Diagrama de casos de uso de la épica de salud, actor involucrado veterinario.

Posteriormente se procede a realizar la tabla de especificación de caso de uso por cada caso de uso especifico dentro del diagrama, cada uno de ellos se encuentran definidos en el Anexo D. En la especificación de caso de uso de la Tabla 5, se detalla de manera específica cada una de las actividades y pasos básicos que se involucran dentro de este proceso. Dando así una visión clara del caso de uso y definiendo las que posteriormente se convierten en las historias de usuario.

Tabla 5

Caso de uso crear ficha clínica

ID	*CasoDeUso-01*
Descripción	*Crear ficha clínica*
Actores	*Veterinario*
Precondiciones	*Debe existir un paciente registrado junto a su tutor registrado.*
Pasos básicos	*1.El usuario debe iniciar sesión en la aplicación*

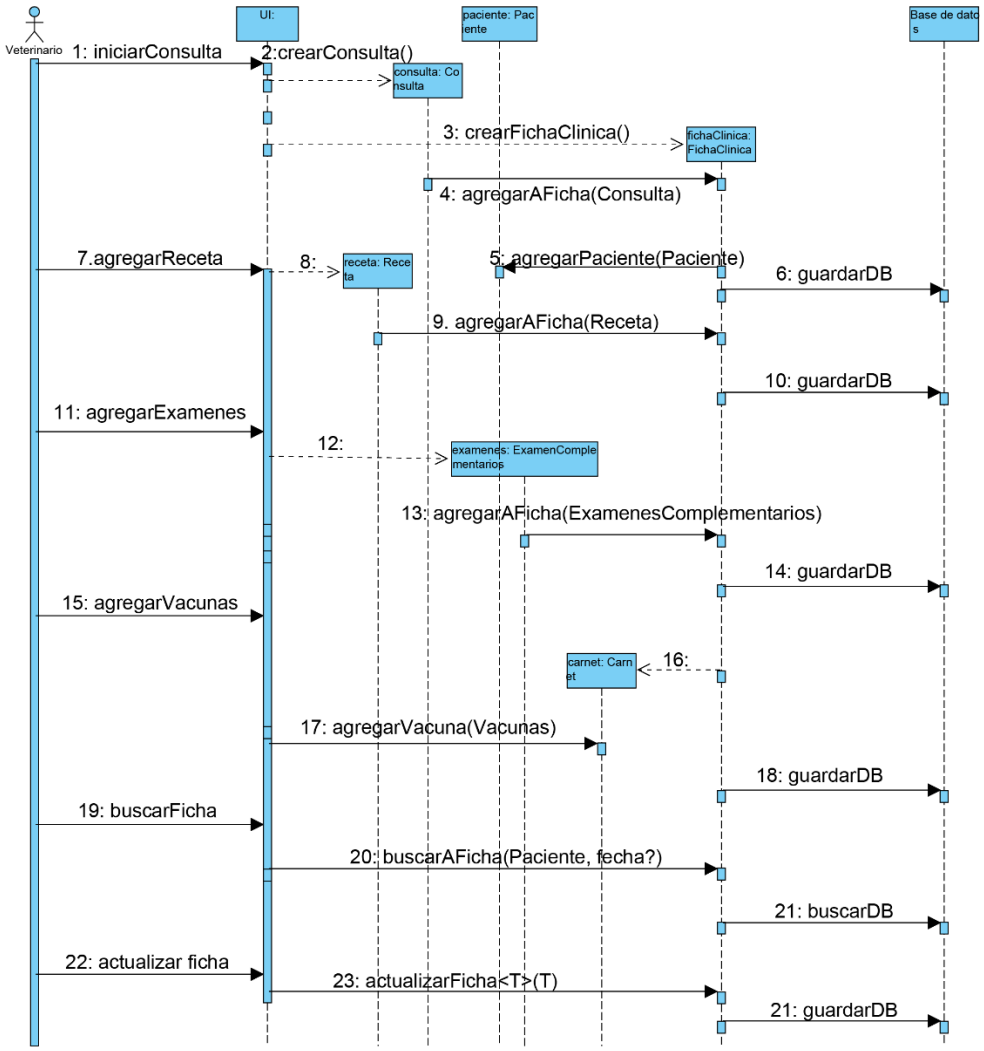
ID	*CasoDeUso-01*
	2.El usuario selecciona la opción de crear una nueva ficha clínica *3. Selecciona los datos del paciente* *4. Selecciona los datos del tutor del paciente* *5.Ingresa los datos solicitados en la ficha clínica* *6.Los datos ingresados son validados* *7.Se guarda la ficha clínica*
Caso de excepción	*Si el tutor del paciente no posee un animal no puede generarse la ficha clínica* *Si el paciente no está asociado a un tutor no se puede generar la ficha clínica*
Validaciones/Reglas de negocio	*Los datos ingresados deben ser validados antes de ser guardados* *Se debe verificar que exista una ficha clínica única para un paciente*
Post Condiciones	*El veterinario puede editar, eliminar o consultar la ficha clínica*

Nota. Especificación de caso de uso sobre crear ficha clínica.

Para cada caso de uso se realiza su respectivo diagrama de secuencia descrito individualmente por capacidad o característica (Anexo E), estos permiten representar el comportamiento del sistema, con el fin de observar la cadena que lleva cada una de las funcionalidades del sistema como se muestra en la Figura 16.

Figura 16

Diagrama de secuencia de ficha clínica.



Nota. Diagrama de secuencia del proceso de consulta de ficha veterinaria.

En el diagrama de la Figura 16 se muestra la secuencia de pasos o actividades a realizar por el sistema para poder realizar una consulta dentro de la clínica veterinaria, este diagrama muestra la interacción del actor (veterinario) con el sistema y la base de datos. Podemos observar los momentos puntuales en los cuales se ingresan o guardan los datos.

En la Tabla 6 da a conocer un resumen del portafolio SAFe 5, todas las tablas generadas para el portafolio se encuentran dentro del Anexo F. Debido a la gran cantidad de

información y procesos dentro de la épica salud, se procede a dividirla en dos capacidades: ECOP, gestión de paciente, ya que estas dos ramas tienen distintos objetivos al momento de la asistencia a un paciente dentro de la clínica veterinaria, las cuales al agruparlas tienen procesos similares dentro de la épica salud, y dan parte a las características del sistema.

Con el diagrama de características se analiza las funcionalidades que presentan dentro de las capacidades, para poder identificar los procesos que requiere la clínica veterinaria.

Tabla 6

Cuadro de características del sistema

Épica	Capacidad	Característica	Hipótesis de Beneficio
Salud	ECOP	Ficha clínica	El veterinario puede registrar las fichas clínicas de sus pacientes de manera rápida.
		Carnet de vacunación	El veterinario puede gestionar la información de las vacunas de sus pacientes.
		Receta medica	Permite al veterinario administrar las recetas de los pacientes.
		Exámenes complementarios	El veterinario puede crear un formulario con los resultados del paciente que va enlazado al historial clínico.
	Gestión de paciente	Paciente	El veterinario puede crear, consultar, editar y eliminar la información del paciente
		Tutor	El veterinario puede registrar un nuevo cliente (tutor) en la clínica veterinaria.

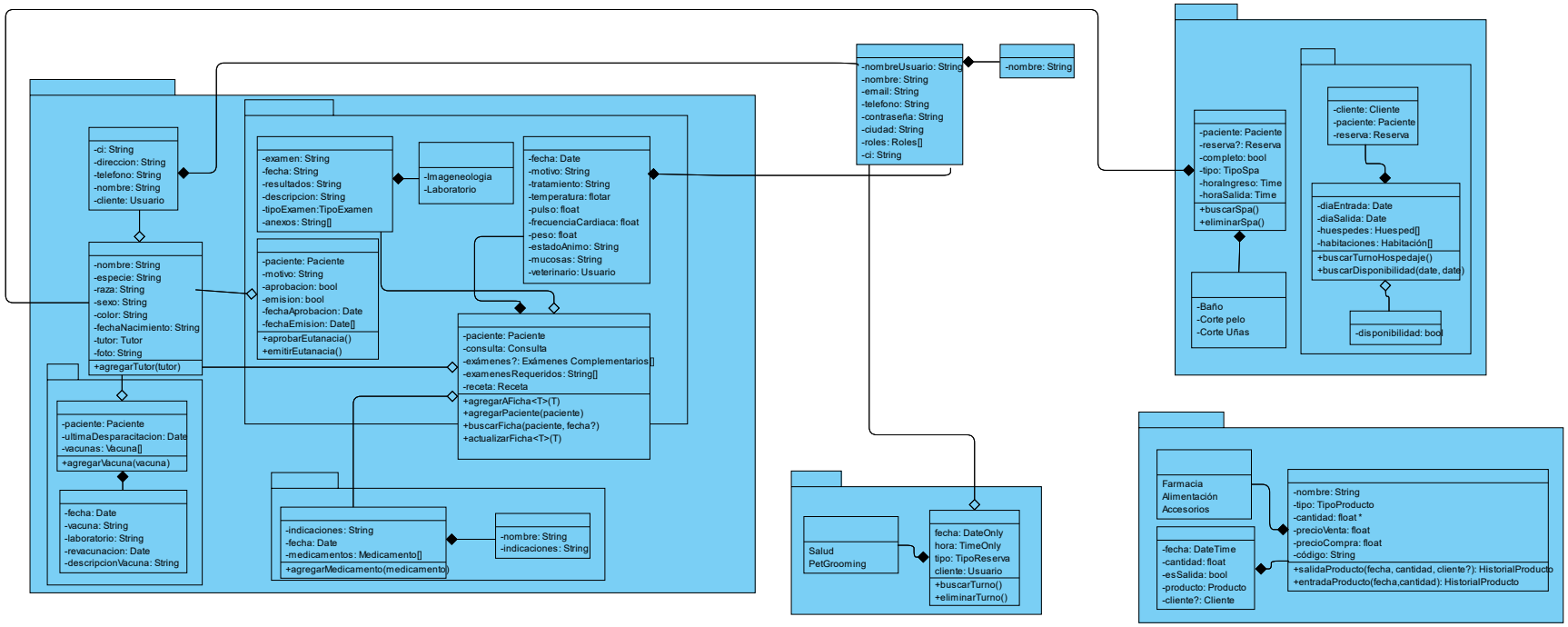
Épica	Capacidad	Característica	Hipótesis de Beneficio
Pet Grooming	N/A	SPA	El veterinario puede agendar una cita para diferentes servicios que requiera su mascota
		Hospedaje	El veterinario puede dar ingreso y salida de la mascota
Gestión de Productos	N/A	Producto	Realizar un registro de los productos tanto de entrada como salida.

Nota. Se presenta las características y capacidades que tiene el sistema.

Una vez realizado todo el proceso anterior en las características y los casos de uso, se procede a realizar el diagrama de clases que ayudan a trazar la estructura del sistema y modelar sus entidades, a continuación, se muestra en la Figura 17 el diagrama de clases.

Figura 17

Diagrama de clase del sistema

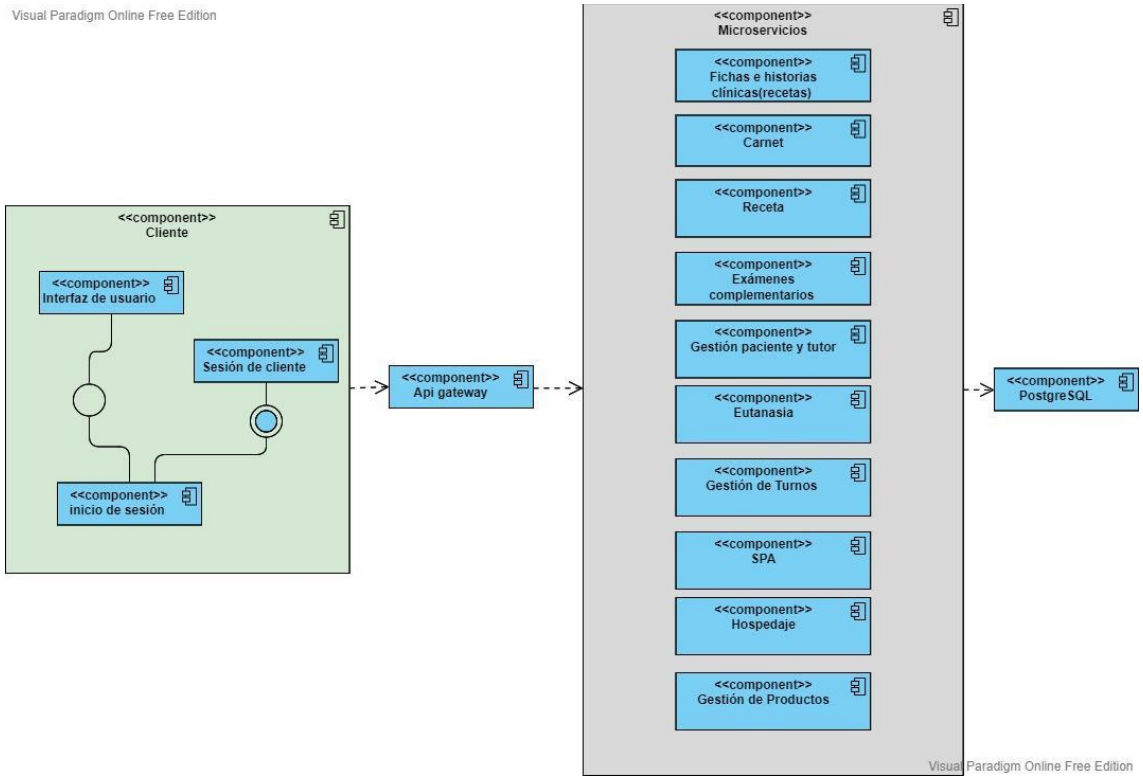


Nota. Diagrama que muestra las clases utilizadas para modelar el sistema.

Para finalizar se procede a realizar el diagrama de despliegue que ayudará a visualizar los componentes individuales del sistema para tener una visión general del sistema y documentar la organización de componentes del sistema y sus relaciones y dependencias mutuas. Como se muestra en la Figura 16 el diagrama de despliegue que se utilizó para el sistema.

Figura 18

Diagrama de componentes del sistema



Nota. Se muestran los componentes del sistema.

Desarrollo del sistema

Para el desarrollo del sistema se creó un proyecto dedicado exclusivamente para la gestión de la base de datos, también se creó un proyecto de front end, un proyecto por cada microservicio a implementar y un proyecto para el servidor de autenticación cada uno de estos desarrollados en un lenguaje determinado por cada grupo que realizó su repositorio. En la

Tabla 7 se encuentra la lista de los repositorios existentes, los integrantes de los equipos de trabajo se encuentran detallados en la Tabla 1.

Tabla 7

Listado de repositorios y microservicios

Repositorio	Equipo	Tecnología
MicroservicioAgendamientoTurnos	C	Node js (express)
MicroservicioCarnetEutanasia	B	Node js (express)
MicroservicioCatalogoProductos	B	Node js (express)
MicroservicioFichaReceta	B	Node js (express)
MicroservicioGestionSpa	C	Node js (express)
MicroservicioHospedaje	C	Node js (express)
MicroservicioPacienteTutor	C	Node js (express)
Database	A, B, C	Liquibase (postgresql)
identityServer	A, C	C# .net (IdentityServer)
FrontEnd	A	JavaScript (React js)

Nota. Se listan todos los repositorios utilizados para el desarrollo del proyecto.

Todos estos subproyectos (repositorios), se encuentran relacionados a funcionalidades del sistema. Todos estos se encuentran divididos en tres sprints los cuales se determinaron por nivel de importancia dentro de las clínicas veterinarias. El primer sprint consta de las características de: Ficha clínica, receta, pacientes y tutores; el segundo consta de: carnet, exámenes complementarios, Pet Grooming. La organización del sprint se muestra en la Tabla 8, donde se detalla cada uno de los encargados por funcionalidades.

Tabla 8

Organización de sprints (sprint planning)

Sprint	Encargado	Tiempo	Funcionalidad / Característica	Encargado	Prioridad
1	Julio Castro	10 días hábiles	Ficha Clínica	Martín Camacho	4
			Receta	José Núñez	3
			Paciente	Paola Romo	4
			Tutor	Ana Sánchez	2
2	José Poveda	10 días hábiles	Carnet	Martín Camacho	3
			Exámenes complementarios	José Núñez	3
			Spa	Ana Sánchez	2
			Hospedaje	Paola Romo	2
3	José Poveda	5 días hábiles	Producto	Martín Camacho	1
			Autenticación	Paola Romo	3

Nota. Se presenta el cronograma de los sprints.

Cada uno de los sprints se realizan de forma consecutiva en el transcurso de un total de 5 semanas, teniendo así al culminar un producto completo con todas las funcionalidades de la LPS.

Para el desarrollo del front end el equipo A, ha utilizado el repositorio ‘FrontEnd’ como se observa en la Tabla 7 para el desarrollo de esta parte del sistema se utilizó la librería React.js. Debido a que esta permite realizar toda la parte web del sistema basándose en componentes lo cual ayuda a reutilizar el código como se puede observar dentro del código de ficha clínica. Aquí se reutilizan los componentes individuales de consulta y receta para generar

la ficha clínica como podemos observar en el fragmento de código del método render del componente ‘FichaPage’ del repositorio:

```
return (
            <>
                <div className="container-fluid">
                    <div className="row">
                        <div className="col">
                            <h1>Ficha clínica {!this.props.isHistorial ? "del
paciente" : ""}</h1>
                        </div>
                    </div>
                </div>
                <NuevaConsultaPage
serviceConsulta={this.props.serviceConsulta} onFicha={true}
idconsulta={this.state.idConsulta}
idficha={this.state.idFicha}></NuevaConsultaPage>
                <CrearRecetaPage serviceReceta={this.props.serviceReceta}
onFicha={true} idFicha={this.state.idFicha}></CrearRecetaPage>
            </>
        )
```

Como se observa en el fragmento de código los componentes reutilizados son NuevaConsultaPage y CrearRecetaPage. El código completo del sistema se encuentra en el Anexo G y más capturas del sistema se encuentran en el manual de usuario Anexo H.

Sprint 1

Tabla 9

Sprint backlog 1

Tarea	Asignado	Estado	Estimación (días)
Ficha clínica			**5**
Crear ficha clínica	Julio Castro	Completa	2
Actualizar ficha clínica	Julio Castro	Completa	2
Buscar ficha clínica	Julio Castro	Completa	1
Receta			**5**
Crear receta	José Poveda	Completa	2
Actualizar receta	José Poveda	Completa	2
Buscar receta	José Poveda	Completa	1
Paciente			**5**
Crear paciente	Julio Castro	Completa	2
Actualizar paciente	Julio Castro	Completa	1
Buscar paciente	Julio Castro	Completa	1
Borrar paciente	Julio Castro	Completa	1
Tutor			**5**
Crear tutor	José Poveda	Completa	2
Actualizar tutor	José Poveda	Completa	1
Buscar tutor	José Poveda	Completa	1
Borrar tutor	José Poveda	Completa	1

Nota. Sprint backlog 1 del equipo de trabajo A

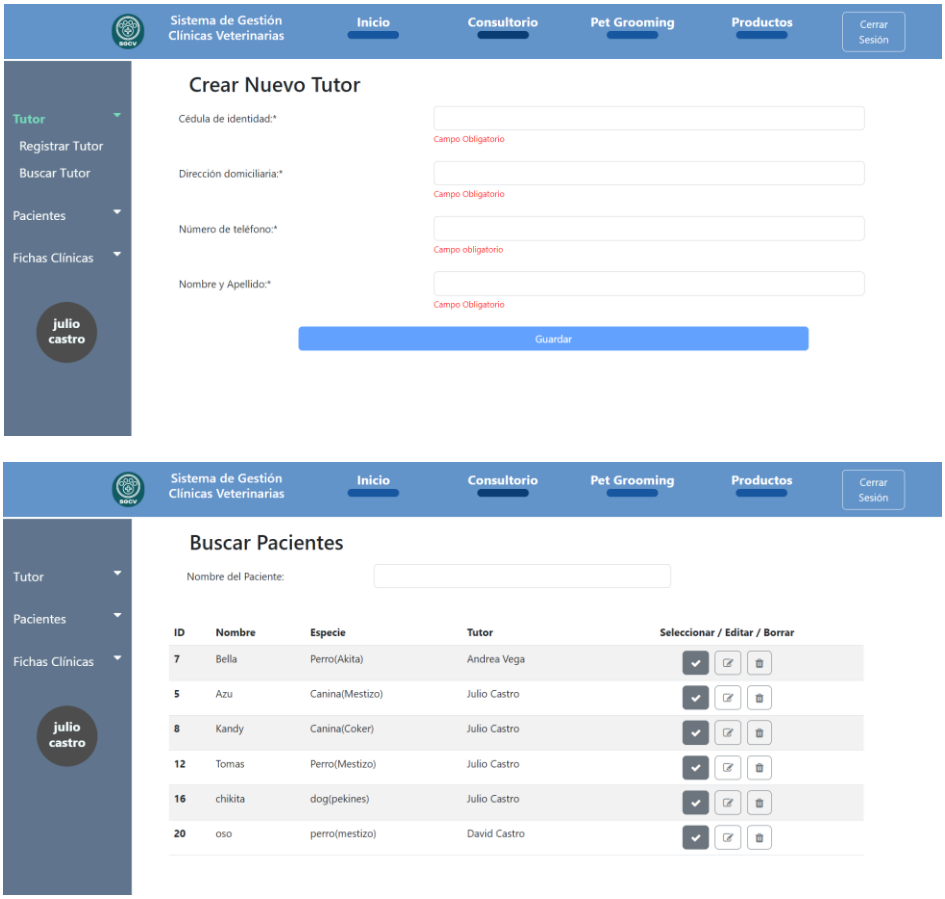
Una vez definidas las actividades para cada uno de los sprints. Se inicio el proceso realizando una reunión principal en conjunto con todos los equipos de trabajo, aquí cada uno de los equipos de trabajo fueron orientados en las actividades a realizarse. Se inicio explicando a los equipos de trabajo las actividades a realizarse y se otorgó los repositorios necesarios para

el desarrollo como se muestran en las Tablas 7 y 9. Se indicio el proceso a llevar para la implementación de cada una de las funcionalidades y características. Este proceso inicia mediante la implementación de la base de datos la cual se realiza en el proyecto Database de liquibase. Aquí se definen todas las tablas que se utilizaran dentro del sistema. Posterior a ello se realiza la implementación de los microservicios correspondientes como se muestran en la Tabla 7.

Se inicio con el desarrollo del sprint el equipo A inicio con el desarrollo de sus actividades dividiéndose las actividades entre los integrantes del equipo. Se inicio con el desarrollo de las características más fundamentales dentro del Sprint 1 las cuales consisten en la gestión de paciente y tutor. Se inicio con la implementación de las clases relacionadas a los modelos. Posteriormente se codificó las clases de tipo ‘services’ las cuales se encargan de realizar las peticiones HTTP hacia el back end recibiendo y enviando datos para el funcionamiento del sistema. Una vez culminados tanto los modelos y los servicios se procedieron a implementar la parte gráfica de las funcionalidades cada una representada como un componente de React. Se puede observar en la Figura 18, los resultados de la implementación de las características de paciente y tutor.

Figura 19

Implementación de paciente y tutor

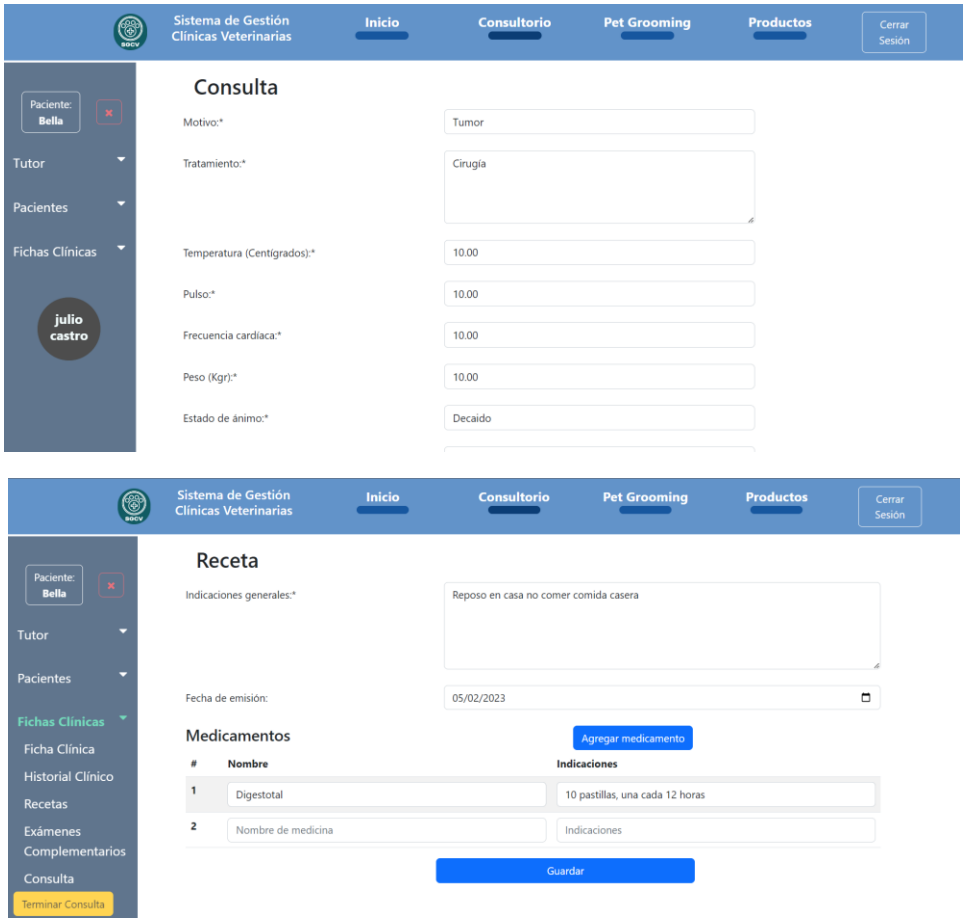


Nota. Capturas de pantalla de las secciones de registro de tutor y búsqueda de paciente.

De igual manera se realizaron las implementaciones de receta y ficha clínica. Una vez terminadas todas las funcionalidades del sprint se procedió a realizar los respectivos pull request sobre la rama develop del repositorio. Aquí se procedió a realizar las diferentes pruebas de funcionalidad. Una vez realizado se hizo el pull request hacia la rama main teniendo listo así todas las funcionalidades correspondientes del sprint.

Figura 20

Implementación de ficha y receta



Nota. Capturas de pantalla de las páginas de consulta y receta.

Durante el desarrollo del sprint se realizó los respectivos sprint daily en el cual se compartía con los integrantes del equipo, todos los avances realizados o impedimentos encontrados en el día anterior.

Sprint 2

Tabla 10

Sprint backlog 2

Tarea	Asignado	Estado	Estimación (dias)
Carnet			**5**
Crear carnet	Julio Castro	Completa	2
Actualizar carnet	Julio Castro	Completa	2
Buscar carnet	Julio Castro	Completa	1
Exámenes complementarios			**5**
Crear examen complementario	José Poveda	Completa	2
Actualizar examen complementario	José Poveda	Completa	2
Buscar examen complementario	José Poveda	Completa	1
Spa			**5**
Crear spa	Julio Castro	Completa	2
Actualizar spa	Julio Castro	Completa	1
Buscar spa	Julio Castro	Completa	1
Borrar spa	Julio Castro	Completa	1
Hospedaje			**5**
Crear hospedaje	José Poveda	Completa	2
Actualizar hospedaje	José Poveda	Completa	1
Buscar hospedaje	José Poveda	Completa	1
Borrar hospedaje	José Poveda	Completa	1

Nota. Sprint backlog 2 del equipo de trabajo A

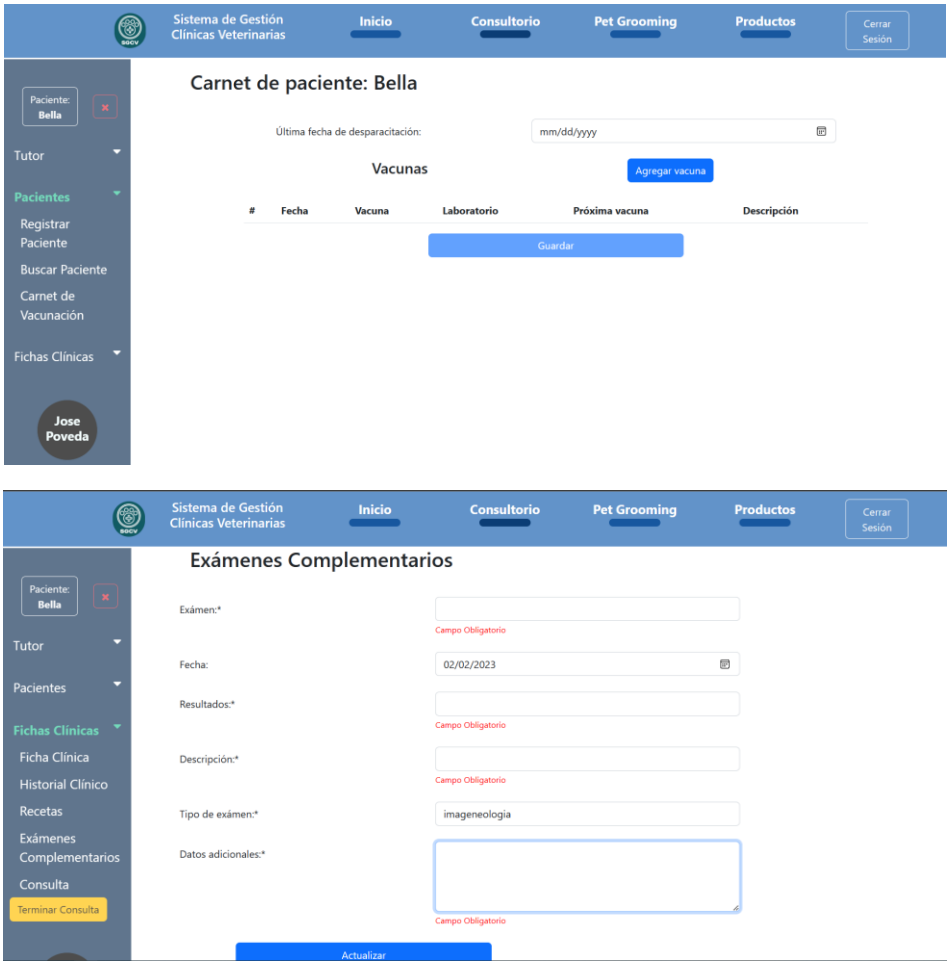
El proceso que se llevó a cabo para la realización del Sprint 2 con el fin de definir las actividades y procesos dentro del sistema, fue juntamente con el encargado del equipo de

trabajo de cada funcionalidad para determinar el procedimiento y orientación de cada una que se implementaran dentro del sistema y asignando la prioridad que tiene cada una de las características. Para luego proceder con la asignación de repositorios para que se pueda trabajar individualmente por cada funcionalidad.

Una vez definidas los procesos y prioridades de cada una de las características, se inició con el desarrollo del Sprint 2 junto con cada uno de los integrantes de los equipos. Como principal objetivo consiste en carnet y exámenes complementarios. Para iniciar estas características se inició asignando al equipo B, para que se proceda a realizar las respectivas funciones de las características mencionadas y proceder con la codificación de las clases de tipo ‘services’, para así poder realizar las respectivas peticiones HTTP hacia el back end con el fin de recibir y enviar información para observar el funcionamiento del sistema con su respectivo componente gráfico realizado en React. Como resultado se puede observar en la Figura 21, los resultados los componentes gráficos de cada una de las características de carnet y exámenes complementarios.

Figura 21

Componte gráfico de carnet y exámenes complementarios

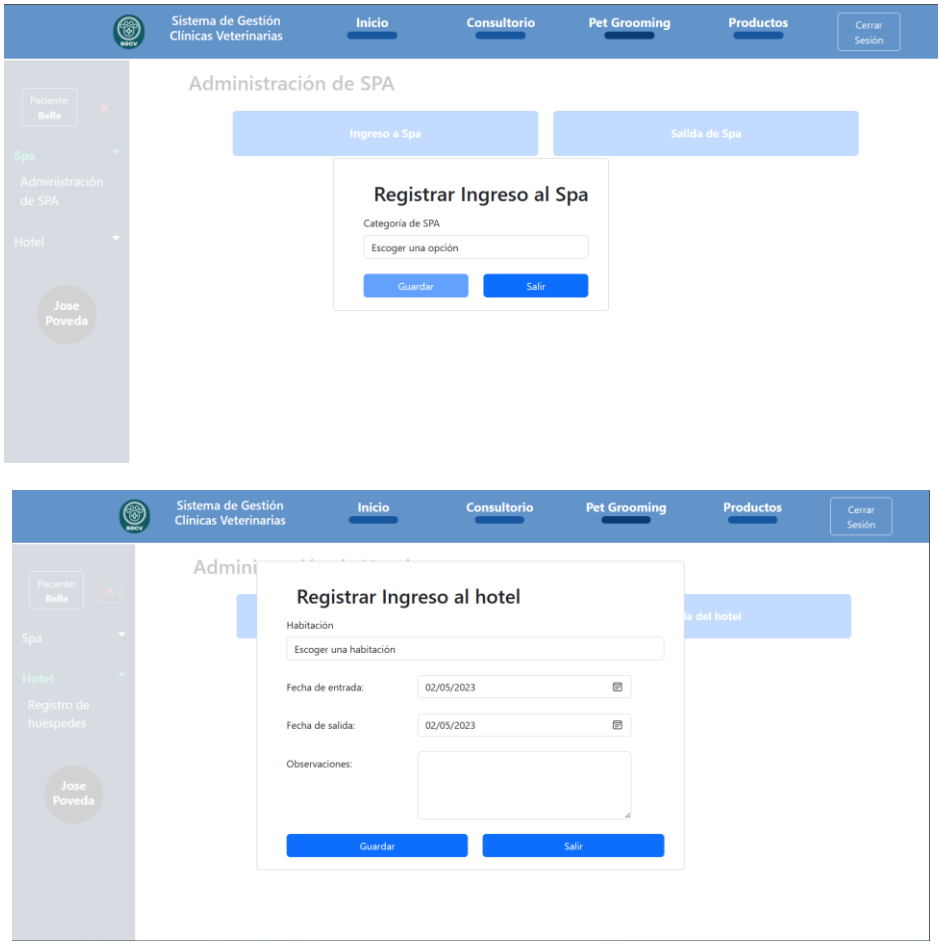


Nota. Visualización de las pantallas de carnet y exámenes complementarios de un paciente.

De igual manera se aplicó el mismo procedimiento para la realización de las funcionalidades de SPA y hospedaje junto con el equipo C para la verificación y validación de peticiones y envió de información de las características, para posteriormente ser mostradas en sus componentes gráficos como se muestra en la Figura 22.

Figura 22

Componente gráfico de SPA y Hotel



Nota. Visualización de pantallas de ingreso de SPA y hotel

Sprint 3

Tabla 11

Sprint backlog 3

Tarea	Asignado	Estado	Estimación (dias)
Gestión de producto			**5**
Crear producto	José Poveda	Completa	2
Actualizar producto	José Poveda	Completa	1
Buscar producto	José Poveda	Completa	1
Historial de producto	José Poveda	Completa	1
Seguridades			**5**
Implementación de openId	Julio Castro	Completa	2
Asignación de autorización a páginas	Julio Castro	Completa	3

Nota. Sprint backlog 3 del equipo de trabajo A

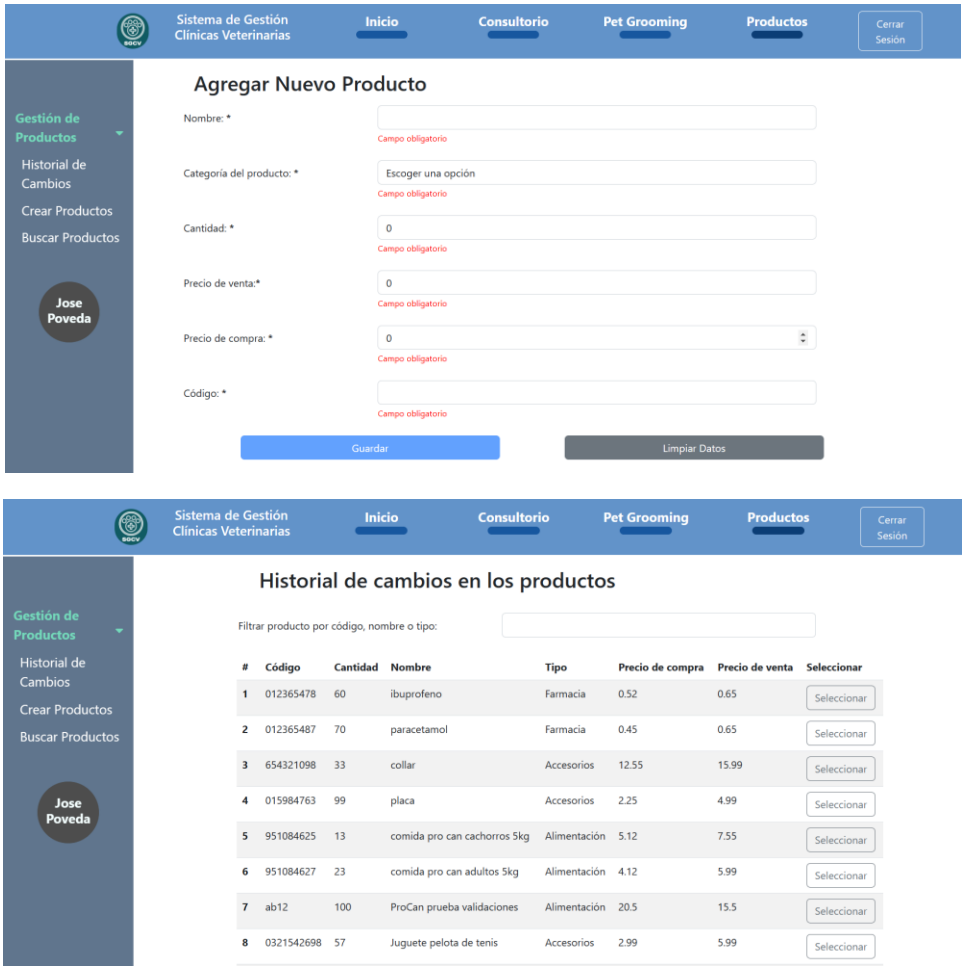
Para la siguiente actividad dentro del Sprint 3 se procedió con la gestión de productos y seguridades del sistema. Realizando una reunión en conjunto con los equipos de trabajo, donde se procedió a la orientación de las actividades, donde se inicia explicando los objetivos que tiene cada una de las características y se otorgó los repositorios necesarios para el cumplimiento de cada una de ellas, con el objetivo de facilitar a las funcionalidades del sistema.

Luego se procede con el desarrollo de las actividades divididas dentro del grupo de trabajo asignado a la gestión de productos aplicando la implementación de las clases relacionadas a los modelos, para posteriormente codificar los servicios HTTP y peticiones dentro del sistema como se muestra en la Figura 23. Una vez finalizado el proceso de gestión de productos se asignó las tareas para la siguiente característica de seguridad del sistema en

donde se verificará la autenticación y registro para el sistema realizando peticiones y envío de datos. Todas estas características tienen su respectivo componente gráfico como se muestra en las siguiente Figura 24.

Figura 23

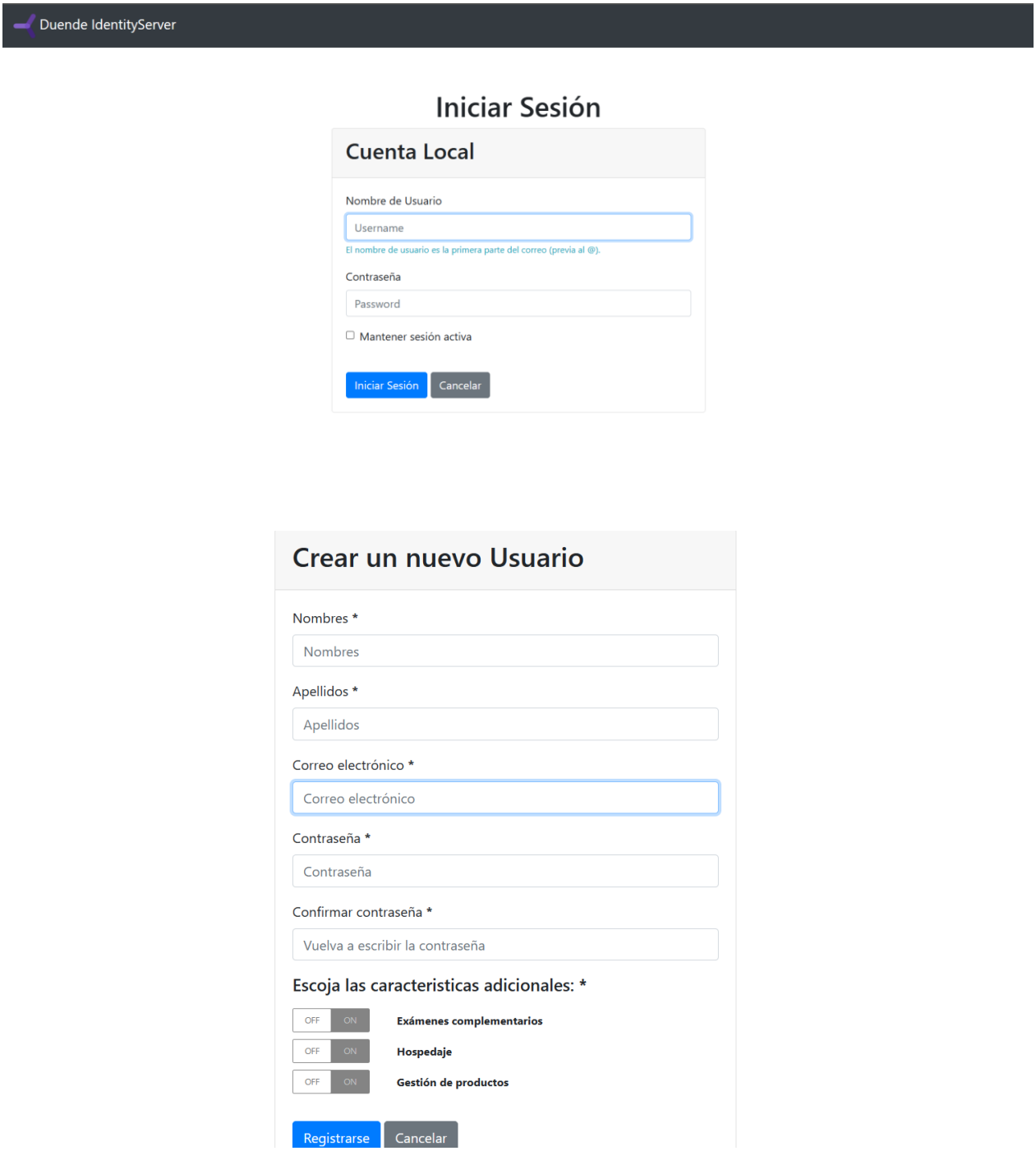
Componente grafico de gestión de productos



Nota. Visualización de pantalla de registro e historial de cambio del producto

Figura 24

Componente grafico de inicio de sesión



Nota. Visualización de inicio de sesión y registro de usuario.

Para conectarse al servidor de autenticación como se observa en la Figura 24 se utilizó el protocolo de openid, el cual nos permite conectar el front end del sistema con el servidor de autenticación. Para llegar a esto se implementó un servicio denominado ‘auth.ts’ el cual contiene un namespace con las configuraciones y funciones que se utilizaran para realizar las conexiones. A continuación, se detalla el fragmento de código donde se encuentra la configuración de la conexión al servidor de autenticación.

```
import * as Oidc from 'oidc-client';

let config: Oidc.UserManagerSettings = {
    authority: "https://sgcv-identityserver.azurewebsites.net/",
    client_id: "sgcv",
    client_secret: "*****-******-*****",
    redirect_uri: "http://localhost:3000/signin-oidc",
    response_type: "code",
    scope: "openid profile",
    post_logout_redirect_uri: "http://localhost:3000",
    mergeClaims: false,
};
```

En el siguiente fragmento se detallan las funciones realizadas para la autenticación y la autorización del sistema.

```
export namespace Authentication {
    export async function LogIn () {
        console.log('Login')
        return await mgr.signinRedirect();
    }
    export async function logout() {
        console.log('Logout')
```

```
        return await mgr.signoutRedirect();
    }
}
```

Validación del sistema

La implementación de validación del sistema incluye la verificación de la funcionalidad, la compatibilidad, la seguridad, la usabilidad y el rendimiento del sistema. Esto implica la realización de pruebas de funcionalidad que cumpla con los estándares y requisitos de calidad.

Para la validación del sistema en un inicio se realizó creando casos de pruebas hipotéticos y en entornos reales, para ello se determinó varios casos detallados en la Tabla 12 presente a continuación.

Tabla 12

Casos de pruebas

Num.	Caso de prueba	Descripción	Precondición	Resultado esperado	Estado
1	Registro de un tutor	Se realiza el ingreso en el sistema de un nuevo tutor	Se ingresa la información básica de una persona: nombres, dirección, número telefónico y cedula de identidad.	El nuevo tutor se refleja en el momento de realizar la búsqueda de tutores existentes.	Aprobado
2	Registro de un paciente	Se registra un nuevo paciente dentro de la clínica veterinaria	Se ingresa la información del paciente como nombre, raza, especie y se registra con un tutor	Se ha guardado el paciente y se lo puede seleccionar para una posterior consulta.	Aprobado

Num.	Caso de prueba	Descripción	Precondición	Resultado esperado	Estado
3	Inicio de una consulta	Se realiza una consulta veterinaria de rutina a un paciente	Se realiza la consulta de un paciente que llega con malestar estomacal.	Se guarda la información de la consulta y se puede generar una receta.	Aprobado
4	Ingreso en spa	Se registra el baño de un perro	Se realiza el ingreso al spa y la salida una hora después	Se guarda la información del registro del spa y se puede registrar la entrada y la salida.	Aprobado
5	Ingreso en hospedaje	Un gato se quedará durante una noche.	Se realiza el ingreso y asignación de jaula y también la salida	Se guarda a información del huésped, la entrada y la salida	Aprobado
6	Ingreso y salida de productos.	Se registra el ingreso de 50 collares para perros de raza grande. Posteriormente se venden 5 collares.	Se registra la compra de 50 collares a $8.50 c/u y se los busca vender a $10.99. Se registra la salida de 5 collares.	se guarda un nuevo producto y se puede acceder al historial de este.	Aprobado

Nota. Se detallan los casos de prueba realizados al sistema.

Una vez realizados los planes de prueba del sistema de la Tabla 12 se determinó que la utilización de LPS, ayuda a realizar un sistema acorde de los requerimientos obtenidos de las clínicas veterinarias.

Conclusiones

En este capítulo se detalló de manera clara y objetiva todo el proceso para el desarrollo y solución del sistema, en donde se observó el uso de herramientas DGS que permitieron a los equipos de trabajo asegurarse que el sistema cumpla con los estándares y requisitos y se pudo entender el proceso e implementación de desarrollo de software involucrado dentro del paradigma de LPS con la creación de componentes reutilizables.

Capítulo IV

Conclusiones y Recomendaciones

Conclusiones:

- Al finalizar este documento se ha concluido el desarrollo tanto del sistema, como la documentación, en donde incluye la elaboración del sistema de gestión para clínicas veterinarias aplicando LPS.
- Un LPS bien diseñado y optimizado puede mejorar la navegación y accesibilidad a la información tanto para clientes y propietarios de mascotas, aumentando su eficiencia operativa al automatizando y centralizando las tareas administrativas liberando tiempo para el personal clínico, y mejorando la comunicación entre personal, los clientes y los propietarios de mascotas, lo que puede aumentar la satisfacción del cliente y la confianza en la clínica ayudando a facilitar la toma de decisiones proporcionando información valiosa y actualizada sobre la salud y bienestar de las mascotas.
- Con la de aplicación de microservicios para el sistema de gestión de clínicas veterinarias permite añadir o eliminar servicios sin afectar el funcionamiento del sistema en su totalidad, lo que facilita la escalabilidad, al ser diseñados los servicios de forma individual los microservicios pueden ser probados y desplegados de manera independiente lo que ayuda a una mayor flexibilidad, ayudando así a la optimización y rapidez al momento de realizar el procesamiento y ejecución de tareas dentro del sistema.
- Para finalizar el desarrollo de front end del sistema de gestión para clínicas veterinarias en el ámbito DGS al ser bien diseñado y optimizado puede mejorar la navegación y accesibilidad a la información para los usuarios y mejorar su experiencia dentro del sistema, mejorando así la seguridad para la protección de datos y la privacidad de los usuarios y reduciendo el riesgo de vulnerabilidades, también ayuda a la flexibilidad que

permite la integración con otros sistemas y tecnologías, lo que puede aumentar la eficiencia y la funcionalidad del sistema en su totalidad.

Recomendaciones:

- Al aplicar LPS permite estandarizar los procesos y componentes utilizados en la creación del sistema, aumentando la eficiencia y reduciendo la cantidad de tiempo y recursos necesarios para el desarrollo de nuevos productos software.
- Trabajar con la arquitectura de microservicios permite desarrollar, probar y desplegar componentes individuales de una aplicación de manera independiente, aumentando así su flexibilidad, debido a que al tener componentes pequeños se puede trabajar de manera más eficiente y requiere menos tiempo para desarrollarlos.
- Con la utilización del diagrama de características FODA, ayuda al enfoque en la toma de requisitos del usuario, permitiendo a los equipos de desarrollo tomar decisiones específicas sobre que característica implementar y en qué orden.
- Planificar el proyecto con anticipación y detallar todas las tareas necesarias para cumplir con los objetivos del sistema.

Bibliografía

Bayer, J., Fleger, O., Knauber, P., Laqua, R., Muthig, D., Schmid, K., . . . DeBaul, J. (1999). *PuLSE: A Methodology to Develop Software Product Lines.*

Bijwe, A., & Shankar, P. (2022). Challenges of Adopting DevOps Culture on the Internet of Things Applications - A Solution Model. *2022 2nd International Conference on Technological Advancements in Computational Sciences (ICTACS).* Tashkent, Uzbekistan: IEEE. doi:https://doi.org/10.1109/ICTACS56270.2022.9988182

Cedeno, A., Catuto, A., & Rodas-Silva, J. (2021, Diciembre 5). El uso de aplicaciones Web para la Gestión de clínicas veterinarias y su incidencia en la mejora de procesos administrativos. *Ecuadorian Science Journal*, 109-120. doi:https://doi.org/10.46480/esj.5.4.174

Clements, P., & Northrop, L. (2001). *Software Product Line: Practices and Patterns.* Addison Wesley.

Cusco, B. (2022). *Desarrollo e implementación de una arquitectura DevOps para un sistema web basado en microservicios en infraestructuras basadas en código.* Quito: Universidad Politécnica Salesiana.

Días, Ó. (n.d.). *LÍNEAS DE PRODUCTO SOFTWARE.* Universidad del País Vasco.

Espinel, G. P., Carrillo, J. L., Flores, M. J., & Urbieta, M. (2022). Software Configuration Management in Software Product Lines: Results of a Systematic Mapping Study. *IEEE Latin America Transactions*, 718-730. doi:https://doi.otg/10.1109/TLA.2022.9693556

Espinosa, E. G. (2014). *GESTIÓN DE CONFIGURACIÓN Y LÍNEA DE PRODUCTOS PARA MEJORAR EL PROCESO EXPERIMENTAL EN INGENIERÍA DEL SOFTWARE.* Madrid: Universidad Politécnica de Madrid.

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., & Peterson, S. A. (1990). *Feature-Oriented Domain Analysis (FODA) Feasibility Study.* Pennsylvania: Carnegie Mellon University.

La República. (2019, Febrero 28). *Seis de cada 10 hogares del país tienen mascota*. Retrieved from proecuador: https://www.proecuador.gob.ec/seis-de-cada-10-hogares-del-pais-tienen-mascota/

León , Y. (2022). *Sistema electrónico de monitoreo de mascotas para la gestión de clínicas veterinarias utilizando VOIP e IOT.* Ambato: Universidad Técnica de Ambato.

Lopez, D., & Maya , E. (2017). *Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web.* Pichincha.

Malathi, S., & Sudhakar, P. (2018). Implementation of Software Refactoring Using FODA Tool. *2018 3rd International Conference on Communication and Electronics Systems*, 839-842. doi:https://doi.org/10.1109/CESYS.2018.8723986

Microsoft. (n.d.). *¿Qué es DevOps?* Retrieved from azure: https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-devops

Northrop, L. M., & Clements, P. C. (2012). *A Framework for Software Product Line Practive, Version 5.0.* Pittsburgh: Software Engineering Institute.

Piattini, M. G., Vizcaíno, A., & Garcia, F. O. (2014). *Desarrollo Global de Software.* España: Grupo Editorial RA-MA.

Ponce, T. (2013, Enero 28). *Servicios a la orden de... las mascotas*. Retrieved from Revista Lideres: https://www.revistalideres.ec/lideres/servicios-orden-mascotas.html

Quishpe, C. E. (2018). *Desarrollo de una aplicación web que mejore la gestión de los productos experimentales en Ingeniería de Software aplicando el paradigma de línea de producto*

software en el Grupo de Investigación GITBIO. Latacunga: Universidad de las Fuerzas Armadas ESPE Extensión Latacunga.

Rincón, L., Giraldo, G., Mazo, R., Salinesi, C., & Díaz, D. (2013). Subconjuntos mínimos de corrección para explicar características muertas en podelos de líneas de productos. El caso de los modelos de características. *2013 8th Computing Colombian Conference*, 1-6. doi:https://doi.org/10.1109/ColumbianCC.2013.6637515

Rincón, L., Rodríguez, G., Martínez, G., & Pabón, M. (2015). Creating virtual stores using software product lines: An application case. *2015 10th Computing Colombian Conference*, 71-78. doi:https://doi.org/10.1109/ColumbianCC.2015.7333414

Scaled Agile Framework. (2021, Julio 22). *SAFe 5 for Lean Enterprises*. Retrieved from scaledagileframework: https://www.scaledagileframework.com/safe-for-lean-enterprises/#:~:text=SAFe%205%20is%20built%20around%20the%20Seven%20Core,new%20competencies%20%28Organizational%20Agility%20and%20Continuous%20Learning%20Culture%29.

Scrum. (2023). *¿Qué es Scrum?* Retrieved from Scrum.org: https://www.scrum.org/resources/what-is-scrum/

Secretaria nacional de planificación. (2017). *Plan nacional del buen vivir 2017 2021.* Quito: Consejo nacional de planificación.

Software Productivity Consortium. (1993). *Reuse-Driven Software Processes Guidebook.* Virginia: Virginia Center of Excellence.

Weiss, D. M., & Chi, T. R. (1999). *Software Product-Line Engineering: A Family-Based Software Development Approach.* Addison Wesley.

Zavala, D. (2019). *Sistema informático enfocado a la web para el agendamiento de citas médicas y control de historia clínica para la clínica veterinaria "Entre Huellas y Bigotes" de la ciudad de Santo Domingo.* Santo Domingo: Universidad regional autónoma de los Andes.

Anexos