Implementación de los microservicios orientados a la Administración de historias y fichas clínicas usando el paradigma de Línea de Producto Software (LPS) enfocado a un desarrollo co-localizado (DGS).

Camacho Moncayo, Martín Enrique y Núñez Amores, José Antonio

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

08 de febrero del 2023

Latacunga

CERTIFICADO DE ANÁLISIS
magister

Tesina_Nunez-Camacho_16-02-2023-Antiplagio

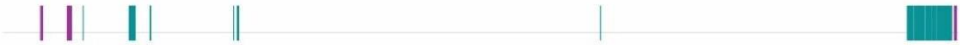
7% Similitudes
0% Texto entre comillas
0% similitudes entre comillas
< 1% Idioma no reconocido

Nombre del documento: Tesina_Nunez-Camacho_16-02-2023-Antiplagio.docx
ID del documento: 8f050775efde773dfe27874f69e953b2fcaec8a3
Tamaño del documento original: 2,96 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: 8334
Número de caracteres: 55.987

Ubicación de las similitudes en el documento:



Fuente principal detectada

Nº	Descripciones	Similitudes	Ubicaciones	Datos adicionales
1	**Tesina_Final_Sanchez_Romo_16-02-2023-Antiplagio.docx** | Tesina_Final_San... #256f1b El documento proviene de mi biblioteca de referencias 1 fuente similar	6%		Palabras idénticas : 6% (431 palabras)

Fuentes con similitudes fortuitas

Nº	Descripciones	Similitudes	Ubicaciones	Datos adicionales
1	**www2.deloitte.com** | Artefactos Scrum: las 3 herramientas clave de gestión https://www2.deloitte.com/es/es/pages/technology/articles/artefactos-scrum.html	< 1%		Palabras idénticas : < 1% (40 palabras)
2	**Documento de otro usuario** #590dfd El documento proviene de otro grupo	< 1%		Palabras idénticas : < 1% (13 palabras)
3	**blog.eclass.com** | ¿Cuáles son los eventos de Scrum? Conoce aquí los Sprint https://blog.eclass.com/cuales-son-los-eventos-de-scrum-conocelos-aqui-0	< 1%		Palabras idénticas : < 1% (18 palabras)
4	**www.obsbusiness.school** | Roles, Eventos y Artefactos en la metodología Scrum | O... https://www.obsbusiness.school/blog/roles-eventos-y-artefactos-en-la-metodologia-scrum	< 1%		Palabras idénticas : < 1% (10 palabras)

Fuentes mencionadas (sin similitudes detectadas)

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

1. HTTPs://microservicio1-ficha-receta.azurewebsites.net
2. HTTPs://microservicio2-carnet-eutanasia.azurewebsites.net
3. HTTPs://microservicio3-catalogo-productos.azurewebsites.net/
4. HTTPs://doi.org/10.1109/ICTACS56270.2022.9988182
5. HTTPs://doi.org/10.46480/esj.5.4.174

Firmado electrónicamente por:
PATRICIO SANTIAGO JACOME GUERRERO

3

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: **"Implementación de los microservicios orientados a la Administración de historias y fichas clínicas usando el paradigma de Línea de Producto Software (LPS) enfocado a un desarrollo co-localizado (DGS)."** fue realizado por los señores **Camacho Moncayo, Martín Enrique y Núñez Amores, José Antonio**; 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, 08 de febrero de 2023

.....

Jácome Guerrero, Patricio Santiago

C. C. 1001689791

4

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Responsabilidad de Autoría

Nosotros **Camacho Moncayo, Martín Enrique; Núñez Amores, José Antonio**; con cédulas de ciudadanía n° **1804290284, 0503926107**; declaramos que el contenido, ideas y criterios del trabajo de integración curricular: **Implementación de los microservicios orientados a la Administración de historias y fichas clínicas usando el paradigma de Línea de Producto Software (LPS) enfocado a un desarrollo co-localizado (DGS)**, 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, 08 de febrero de 2023

.....

Camacho Moncayo, Martín Enrique

C. C. 1804290284

.....

Núñez Amores, José Antonio

C. C. 0503926107

5

Departamento de Ciencias de la Computación

Carrera de Ingeniería de Software

Autorización de Publicación

Nosotros **Camacho Moncayo, Martín Enrique; Núñez Amores, José Antonio**; con cédulas de ciudadanía n° **1804290284, 0503926107**; autorizo/autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de integración curricular: **Implementación de los microservicios orientados a la Administración de historias y fichas clínicas usando el paradigma de Línea de Producto Software (LPS) enfocado a un desarrollo co-localizado (DGS),** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Latacunga, 08 de febrero de 2023

.....

Camacho Moncayo, Martín Enrique

C. C. 1804290284

.....

Núñez Amores, José Antonio

C. C. 0503926107

Dedicatoria

A mis padres, abuelitos, hermana y a toda la gente que creyó en mi durante estos últimos años que me dio aliento para continuar y sobrellevar cualquier obstáculo.

Martín Camacho

El Presente trabajo lo dedico con mucho cariño y con el esfuerzo mismo que esto ha significado a los seres que más amo y estimo:

A MIS PADRES Y HERMANOS, por darme ejemplo de sacrificio, superación, perseverancia y responsabilidad, valores que he puesto en práctica con mucho amor en el trayecto y consecución de mi carrera.

A MIS COMPAÑEROS, con quienes compartimos conocimientos, alegrías y tristezas, apoyándonos cada día para que nuestro sueño se haga realidad, de manera muy especial a mi AMIGO MARTÍN CAMACHO, quien ha sido mi apoyo incondicional en este proceso de profesionalización.

Y a MIS MAESTROS que han aportado significativamente, inculcando con ética y ejemplo la pasión por mi carrera, principalmente al Dr. Santiago Jácome guía y orientador de este trabajo.

José Núñez

Agradecimiento

Quiero agradecer a mis compañeros de proyecto especialmente a José N. por ser un apoyo incondicional y un gran amigo en el transcurso del proyecto.

A Paola R. por ser un pilar fundamental en mi crecimiento personal y profesional en estos últimos años por siempre apoyarme y darme ánimos.

A Joshua N., Johan V., y Santiago C. que se han convertido en mis hermanos y me han apoyado siempre tanto dentro como fuera del escenario y a pesar de las dificultades siempre han estado.

A Génesis Z. por siempre creer en mí y que a pesar de la falta de tiempo siempre se dio un minuto para aconsejarme y escucharme.

A mi madre Cristina por estar ahí y nunca dejarme solo ya que gracias a su amor incondicional he llegado a donde estoy a mi padre José que siempre me dio aliento cuando el mundo se venía abajo, a mi hermana Viviana que me apoyó en todo lo que he hecho hasta ahora.

Y sobre todo a mis abuelitos Carmen y Guillermo gracias a ellos soy quien soy y todos los logros por venir son por y para ellos.

Martín Camacho

Agradecimiento

Obtener mi título ha sido una de mis metas más preciadas…

“Me agradezco a mí mismo por seguir adelante y no decaer.

Por ser valiente, esas veces que quise salir corriendo…

Por seguir intentando día a día sin rendirme.

Por soñar y amar lo que hacía, a pesar de las circunstancias...

Por valorar el esfuerzo de todos quienes nunca me dejaron solo.

Por aprovechar y apreciar las oportunidades

Por ser yo mismo, por amarme y valorarme...

Por todo eso…”

ME AGADEZCO, ME VALORO Y ME FELICITO

José Núñez

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 estilos 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.

Follow 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.

Node.js: 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 TABLAS

ÍNDICE DE FIGURAS

Resumen

El Sistema de Gestión de Clínicas Veterinarias (SGCV) aplicando el paradigma de Línea de Producto Software (LPS), se compone por microservicios independientes y específicos que trabajan juntos para proporcionar una solución a la digitalización de información y automatización de proceos dentro de las clínicas veterinarias cubriendo las tres áreas principales de estas: Salud, Pet Grooming y Catálogo de Productos. Este trabajo tiene como objetivo mostrar el proceso a seguir desde el Análisis de Dominio hasta el despliegue e implementación de los microservicios orientados al área de Salud y Catálogo de Productos. Para cumplir el objetivo se usó una arquitectura orientada a microservicios con una metodología de desarrollo ágil como Scrum con prácticas DevOps para lograr una integración continua y entrega continua (CI/CD) además que las herramientas para el desarrollo están basadas en tecnologías web para back end en este caso se usó Node.js basado en el lenguaje JavaScript. El desarrollo de este proyecto fue colaborativo por lo cual se aplicó Desarrollo Global de Software. El SGCV puede implementarse en cualquier veterinaria sin importar su tamaño y puede escalarse a medida que la veterinaria crece permitiendo una implementación personalizada cubriendo las necesidades que se requieran en las tres áreas principales mencionadas con anterioridad.

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

Abstract

The Veterinary Clinics Management System (SGCV) applying the Software Product Line (SPL) paradigm, is composed of independent and specific microservices that work together to provide a solution to the digitization of information and automation of processes within the veterinary clinics covering the three main areas of these: Health, Pet Grooming and Product Catalog. The objective of this work is to show the process to follow from the Domain Analysis to the deployment and implementation of the microservices oriented to the Health and Product Catalog areas. To meet the objective we used a microservices oriented architecture with an agile development methodology such as Scrum with DevOps practices to achieve continuous integration and continuous delivery (CI/CD) in addition to the development tools are based on web technologies for back end in this case Node.js based on JavaScript language was used. The development of this project was collaborative so Global Software Development was applied. The VMS can be implemented in any veterinarian regardless of its size and can be scaled as the veterinarian grows allowing a customized implementation covering the needs that are required in the three main areas mentioned above.

Keywords: Veterinary Clinics, Sofware Product Line, Global Software Development, microservices.

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 clínicas veterinarias son empresas que brindan sus servicios a macotas como perro, gatos entre otras, que se han incrementado a lo largo de la historia, ya que, en este negocio existen procesos como venta de productos, cuidado de mascotas, SPA, atención médica e incluso existen clínicas veterinarias que tienen todos estos procesos.

"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). La mayoría de las clínicas veterinarias en el Ecuador realizan todo su proceso y registro de forma manual lo cual provoca falta de organización de los expedientes físicos, carencia de código que identifique a cada expediente, la información de los registros en ocasiones no es legible por la calidad de caligrafía utilizada, existe la probabilidad de que puedan afectar en la información obtenida en los registros médicos, productos y pacientes debido a la pérdida o inexistencia de información.

Por ello las clínicas veterinarias han optado por llevar un manejo automatizado de sus datos, y toman alternativas en la implementación de herramientas tecnológicas para el soporte y ayuda de gestión dentro 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 eco dependencia, 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 el Ecuador las personas han brindado una mejor atención y cuidado hacia los animales, lo que ha llevado a generar nuevos centros de especialidades de cuidado para animales domésticos; aumentando así la cantidad de profesionales veterinarios. Dentro de las clínicas veterinarias buscan prestar una atención completa y de calidad para las mascotas, con médicos preparados en su respectiva área para dar un servicio más profesional posible, por lo cual se diseñará e implementará un sistema que brinda ayuda para los servicios que tienen las clínicas veterinarias que albergan acciones tanto como son de registro, diagnóstico, tratamiento, spa, venta de productos e historial médico, obteniendo un resultado la reducción de tiempo al momento de atención hacia el cliente.

El objetivo de la implementación del sistema para las clínicas veterinarias se logrará una satisfacción y rapidez al momento de un registro automatizado de datos dentro de las clínicas veterinarias, obteniendo como resultado la agilidad y rapidez en los procesos, teniendo un mejor control de la información dentro de las clínicas veterinarias, evitando de esta manera los procesos manuales y físicos de la información tengan errores al momento de realizarlos. Con el uso de un sistema tecnológico se podrá aumentar tanto la productividad y agilidad en procesos que manejan dentro de las clínicas veterinarias.

Económicamente el objetivo es cubrir las necesidades de Clínicas Veterinarias a través de la reutilización de componentes software en construcción de nuevos productos

disminuyendo costos, tiempo y recursos en la producción de sistemas de calidad aplicando el paradigma LPS.

Socialmente el objetivo es abarcar diversas Clínicas Veterinarias a nivel nacional con software que cumpla sus necesidades específicas y generales, con un costo accesible, que permita mejorar la calidad en el proceso de atención al cliente

Alcance

Considerando la importancia de las clínicas veterinarias y los servicios que se brindaran se propone desarrollar una aplicación web, para mejorar la gestión y almacenamiento de la información de todas áreas mediante la administración de pacientes, tutores, historial clínico, consultas, carnet de vacunación, spa, hospedaje, exámenes complementarios y catálogo de productos.

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). Adicionalmente se implementará autenticación para proteger y garantizar la seguridad de los datos que la veterinaria posea.

Formulación del problema a resolver

Como desarrollar microservicios orientados a la Administración de historias y fichas clínicas que forman parte del sistema de gestión de clínicas veterinarias.

Objetivo general

Implementar los microservicios orientados a la Administración de historias y fichas clínicas usando el paradigma de Línea de Producto Software (LPS) enfocado a un desarrollo co-localizado (DGS).

Objetivos específicos

- Revisión del estado del arte sobre Línea de Producto Software (LPS), arquitectura de microservicios y desarrollo co-localizado.
- Elaborar la línea de producto de software basada en un análisis de dominio en clínicas veterinarias.
- Implementar los microservicios orientados a la Administración de historias y fichas clínicas.
- Validar los microservicios orientados a la Administración de historias y fichas clínicas.

Hipótesis

La utilización de línea de producto software (LPS) para desarrollar software permite implementar soluciones que se ajustan al requerimiento de las clínicas veterinarias.

Señalamiento de variables

Variable independiente: Línea de Producto Software (LPS).

Variable dependiente: Sistema de Gestión de Clínicas Veterinarias.

Capítulo II:

Fundamentación teórica y referencial

Introducción

En este capítulo se revisará los conceptos, características que sustentan la LPS, y los procesos de métodos Análisis de Dominio Orientado a Características (FODA), que será representada por la arquitectura de microservicios con el desarrollo global de software. Finalmente se realiza una lista de trabajos relacionados con el presente proyecto.

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.

En base a la definición antes mencionada se aprecia que existen dos conceptos fundamentales para la aplicación de LPS:

- Feature: se define como una característica conceptual del sistema, esta se puede relacionar con otras estructuras de los sistemas, o pueden describir la funcionalidad de requerimientos no funcionales, y es usada para describir o distinguir un producto en la línea.
- Core asset**:** es un artefacto de software utilizado en el proceso de desarrollo de uno o más productos de la LPS, un core assets puede ser un componente de

> software, un modelo de procesos, una arquitectura, o cualquier otro resultado de la construcción de un sistema**.**

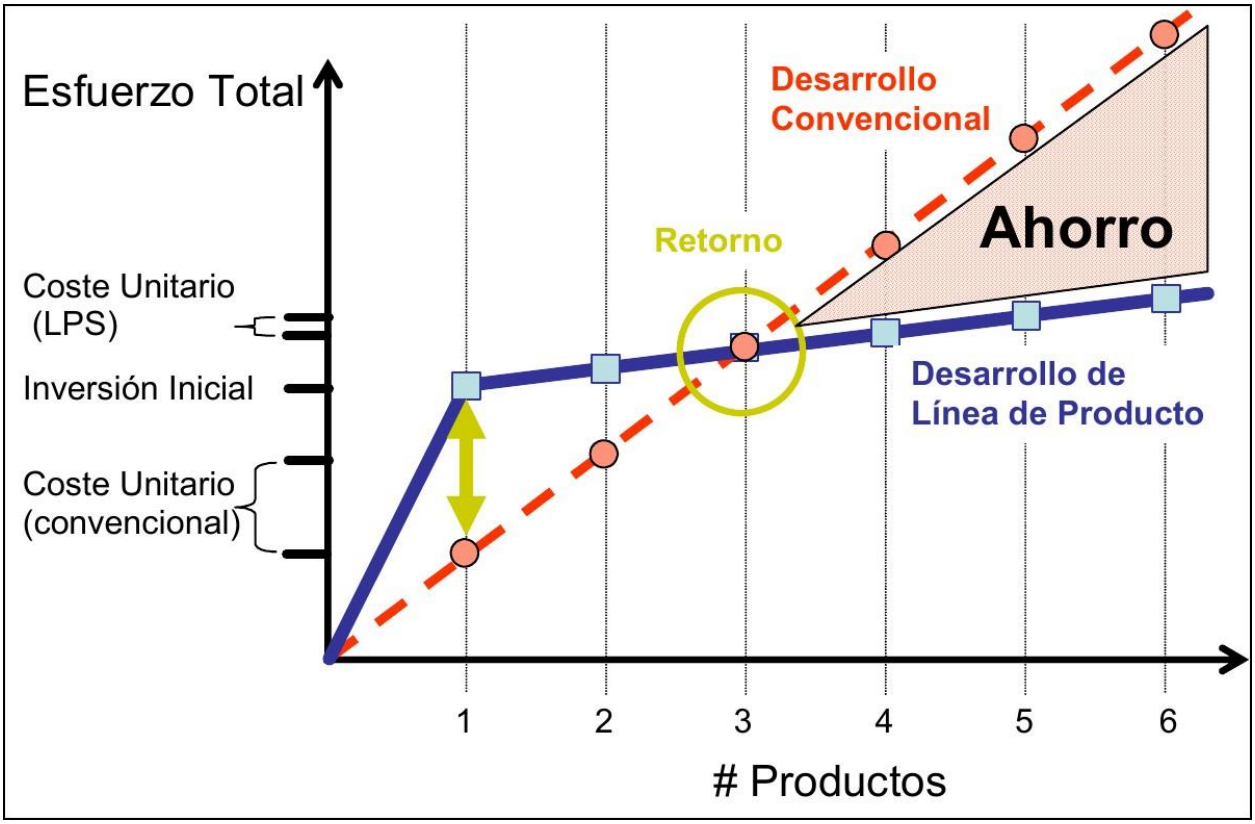
El crecimiento de software aplicando un a LPS se ha ido orientando a distintos segmentos dentro de los mercados, donde su principal objetivo es intentar satisfacer las necesidades de un segmento de mercado y la necesidad de reducción de tiempos en el desarrollo de software gracias a la reutilización de componentes.

Con la aplicación de LPS se puede aumentar la productividad a los desarrolladores disminuyendo el esfuerzo y costo para el desarrollo de software, manteniendo la efectividad y funcionalidad de los productos software. Los estudios de caso han demostrado que las ganancias de productividad han mejorado en comparación con las metodologías convencionales.

En la Figura 1 se representa una gráfica comparativa del esfuerzo realizado al aplicar una metodología convencional frente a la utilización de LPS, observando a largo plazo la reducción del esfuerzo, y por lo tanto costos de producción más bajos.

Figura 1

Actividades esenciales para el desarrollo de 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 manejar un mismo núcleo para varios sistemas similares, resultando así en un ahorro de recursos y tiempo a largo plazo. Los componentes de este núcleo (core assets), permite el desarrollo de múltiples productos a partir de un dominio en común. La LPS sigue siendo un nuevo enfoque, que requiere nuevas arquitecturas y métodos, en donde estos proporcionaran mecanismos para demostrar y representar las características y variabilidades del dominio, donde cada uno de 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.

En la Figura 2 se muestra las actividades que se llevan a cabo dentro del paradigma de la LPS, estas actividades son interactivas, se encuentran fuertemente relacionadas y están en constante desarrollo. 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)

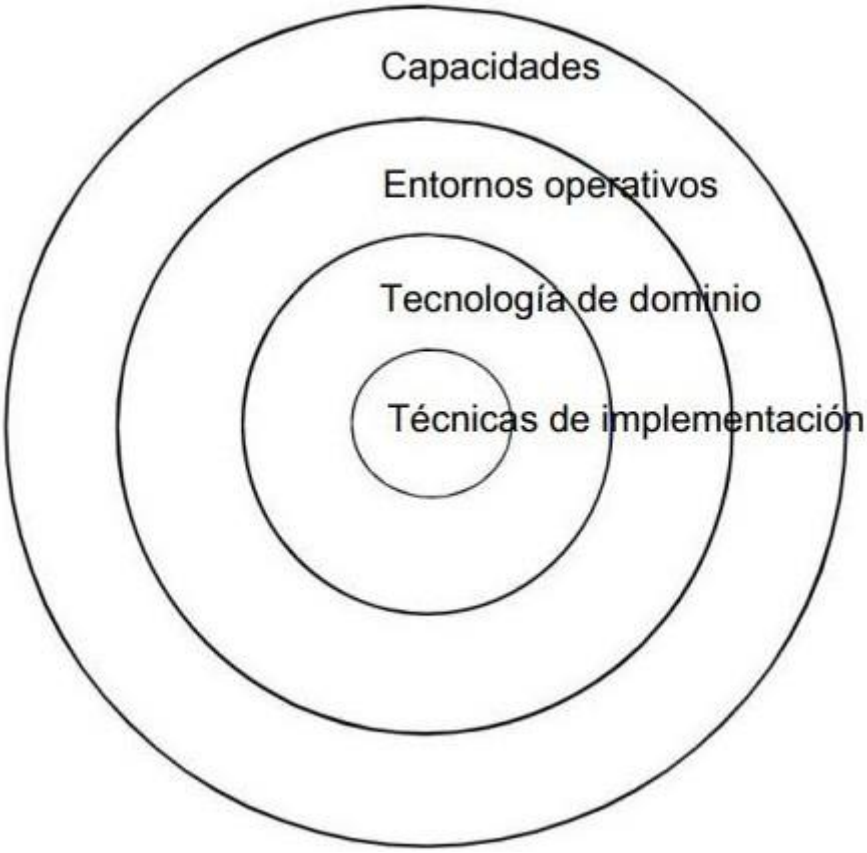
Según (Kang, Cohen, Hess, Novak, & Peterson, 1990) el método FODA permite la reutilización a nivel funcional y arquitectónico. Donde los productos de domino representan la funcionalidad y arquitectura los cuales se pueden adaptar al desarrollo de componentes software. El método FODA permite utilizar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

Parametrización de producto. El objetivo de la parametrización de producto es diseñar componentes genéricos que puedan adecuarse a los valores de parámetros. En donde se implementan de varias formas como subrutinas, genéricos, marcos y procesadores, sin embargo, estas técnicas se aplican principalmente al código. Las aplicaciones en un dominio presentan un conjunto de capacidades comunes, lo que hace que cada aplicación sea diferente a las demás. Estas capacidades desde la visión de un usuario final se modelan como características. Las aplicaciones pueden ejecutarse en diferentes entornos operativos. Pueden ejecutarse en

diferentes hardware o sistemas operativos, o interactuar con diferentes tipos de dispositivos. En la Figura 3 se muestra que al desarrollar una aplicación esta puede tener decisiones que pueden ir variando.

Figura 3

Tipos de decisiones de desarrollo



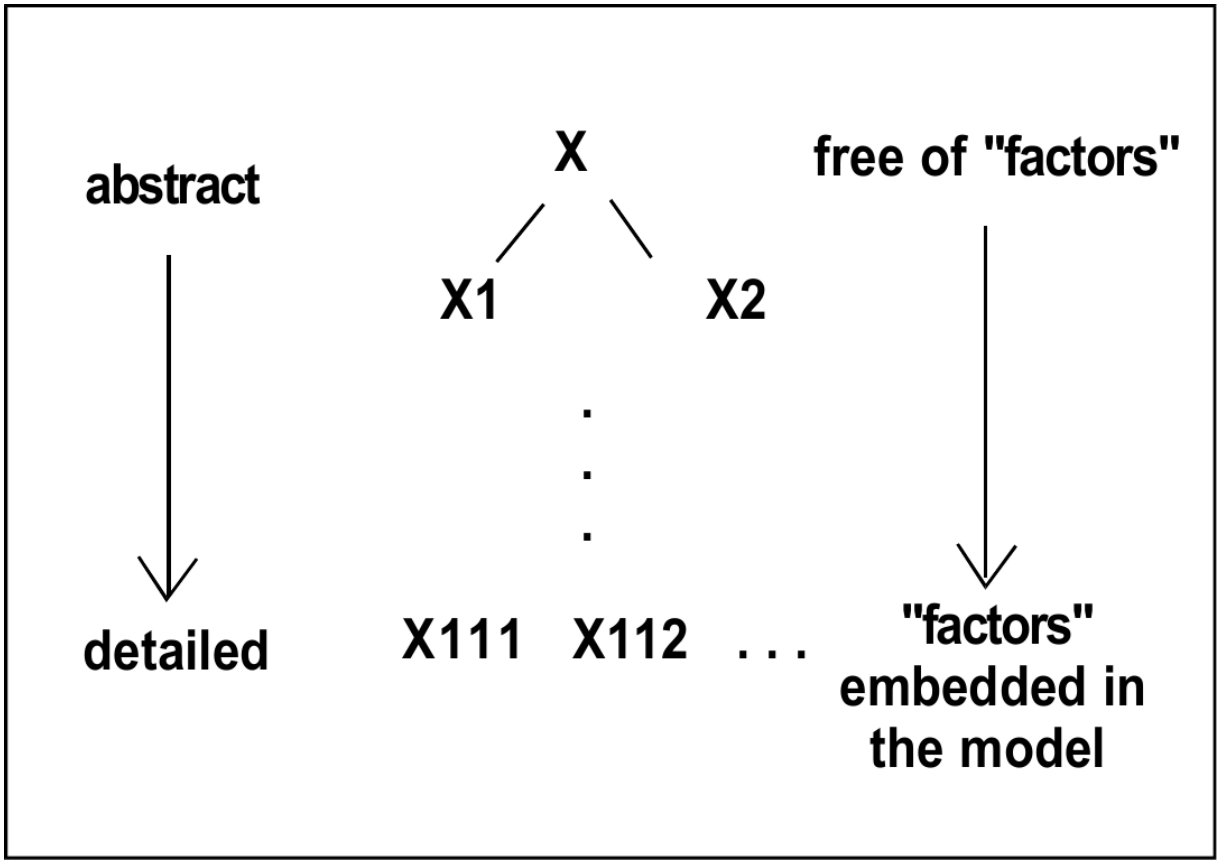
Nota. En la Figura se observa los tipos de decisiones de desarrollo que se presentan al momento de parametrizar los modelos y funciones. Tomado de (Kang, Cohen, Hess, Novak, & Peterson, 1990)

Los problemas y las soluciones, junto con las características, se utilizan para parametrizar los modelos de dominio arquitectónicos y funcionales.

Niveles de abstracción. El método FODA busca que los niveles de abstracción sean lo más básicos posibles, para esto se utiliza varios niveles de abstracción donde una característica de un nivel representa una generalización de los niveles más bajos. Este proceso se realiza hasta obtener características cada vez menos genéricas resultando así en los componentes más esenciales que componen la aplicación. En la Figura 4 se puede observar los distintos niveles de abstracción, hasta llegar a los factores más básicos, los cuales vienen a ser los 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. Tomado de (Kang, Cohen, Hess, Novak, & Peterson, 1990).

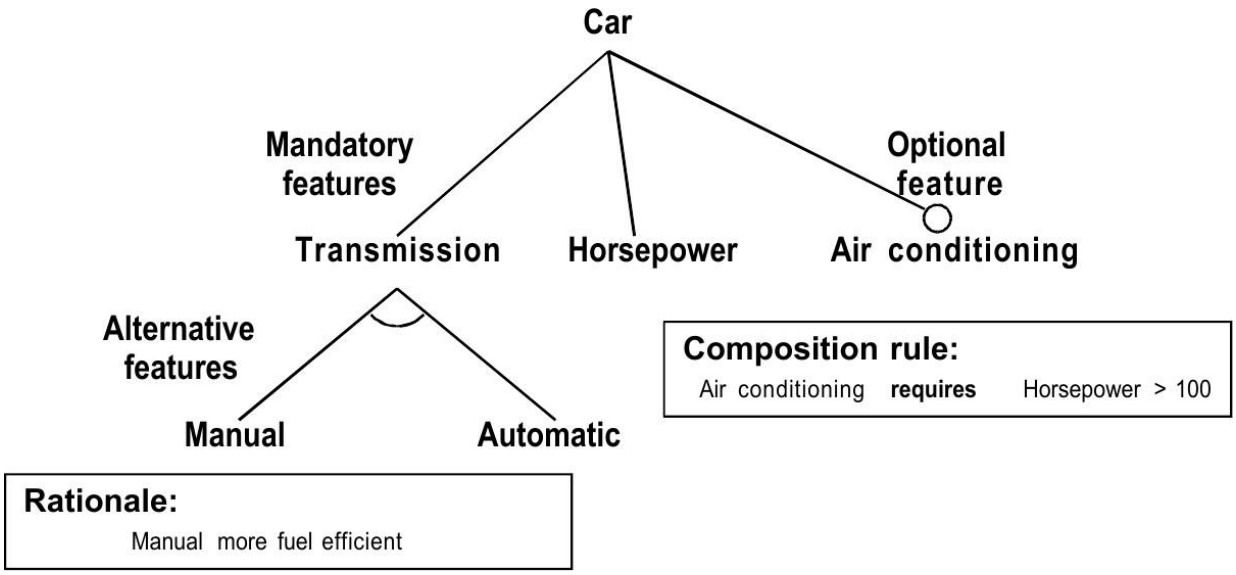
Los productos que genera el dominio son las diferentes abstracciones obtenidas mediante el análisis. Toda la información recopilada se la organiza y se la representa como productos del dominio. Los modelos obtenidos a partir del análisis de dominio son utilizados para realizar los productos de la LPS, en este proceso los analistas de requerimientos determinan si el producto solicitado se encuentra dentro de la LPS; de ser así, se procede a escoger las capacidades del sistema, basándose en las abstracciones obtenidas en el análisis de dominio realizado.

Modelamiento de dominio FODA. Para el desarrollo de software se utiliza diagramas de flujo de datos los cuales son útiles únicamente para el equipo de desarrollo, un cliente no comprendería este diagrama. Los clientes necesitan conocer los componentes y las capacidades esenciales que posee la aplicación, entre ellas: servicios de la aplicación, rendimiento de la aplicación, plataforma de hardware, entre otros. El FODA se enfoca principalmente en la perspectiva de la funcionalidad, donde se muestran las características y los niveles de abstracción, los cuales se pueden utilizar para la generación de productos o parametrizar nuevos modelos.

Las características son los atributos que son de gran importancia para los usuarios finales ya que aquí se muestra la disponibilidad de funciones que puede tener el sistema. En la Figura 5 se muestra las decisiones que debe tomar una persona al comprar un auto; diagramado como 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 muestra los diferentes componentes del dominio y la relación que existe entre ellos. Cada característica es representada por un nombre simple el cual se expande en el diccionario del diagrama. Las características pueden ser de tipo obligatoria u opcional las cuales se determinan en el diagrama como un círculo pintado y uno sin pintar como se muestra en la Figura 5. La selección de características opcionales se realiza en base a los objetivos que tenga el cliente

Arquitectura de microservicios

El objetivo de microservicios para el desarrollo de aplicaciones como un conjunto de pequeños servicios, cada uno de estos son ejecutados en su propio proceso y mecanismo de comunicación, a través de protocolos HTTP mediante APIs. Estos servicios están construidos alrededor de las capacidades o funcionalidades de negocio y con independencia de despliegue

automatizada. La arquitectura de microservicios fomenta el desarrollo y la implementación de aplicaciones que consisten en unidades independientes, modulares, autónomas y autocontenidas de una forma diferente a lo tradicional o monolítica como se muestra 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 está orientada a ser realizada en componentes, por lo tanto, permite el desarrollo de cada uno de ellos de manera independiente. De tal forma que es mucho más fácil realizar grupos de trabajo específicos para cada uno de los microservicios. En combinación con LPS, la arquitectura de microservicios nos permite representar cada uno de los features como un componente del sistema, es decir, un microservicio. Esto nos permite realizar de una forma más organizada y rápida, cada uno de los features y permite que sean mantenibles, reutilizables y escalables. La arquitectura viene a ser un core asset de la línea de

producto software ya que esta se convierte en un activo fundamental a la hora de implementar los diferentes productos de LPS. Los microservicios más esenciales, es decir, los componentes del FODA más básicos, son módulos reutilizables de la 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.

Un proyecto en DGS se puede considerar como un conjunto de varios subproyectos, repartiéndose las diferentes actividades y características que lo conforman entre los diferentes nodos o sitios que colaboran en él, existen tres tipos de distribución de tareas o trabajo: “basado en módulos” consiste en dividir el proyecto en módulos cada uno de ellos puede ser considerado como un artefacto completo y repartirlos entre los sites, “basado en fases” cada fase del desarrollo es asignado a un site o grupo de desarrollo y “follow the sun” cada grupo de desarrollo se encuentra distribuido geográficamente con el objetivo de mantener la producción las 24 horas es decir mientras un equipo duerme el otro trabaja.

Formación de equipos distribuidos

En DGS un punto muy importante es la formación de los equipos distribuidos. Estos equipos deben formarse a partir de los roles que cumple cada uno de los integrantes de los equipos. Obteniendo así equipos basados en características, es decir, equipos que buscan complementar una característica en concreto. 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: Consiste en organizar equipos agrupando varios especialistas dentro de un mismo equipo de trabajo. Garantizando así que el equipo posea conocimientos sobre la mayoría del proceso o del sistema.

Expansión gradual de la responsabilidad: Con este método se realiza pequeñas modificaciones en los integrantes del equipo. Hasta llegar a un equipo de características.

Introducción gradual de equipo de características: En este método se toman las características más importantes del producto backlog y únicamente a ellas se las aplica el cambio a equipo de 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 influye en el ciclo de vida de una aplicación a través de las etapas de planificación, desarrollo, entrega y uso, en donde cada etapa depende de otra y están involucradas dentro de las mismas.

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.).

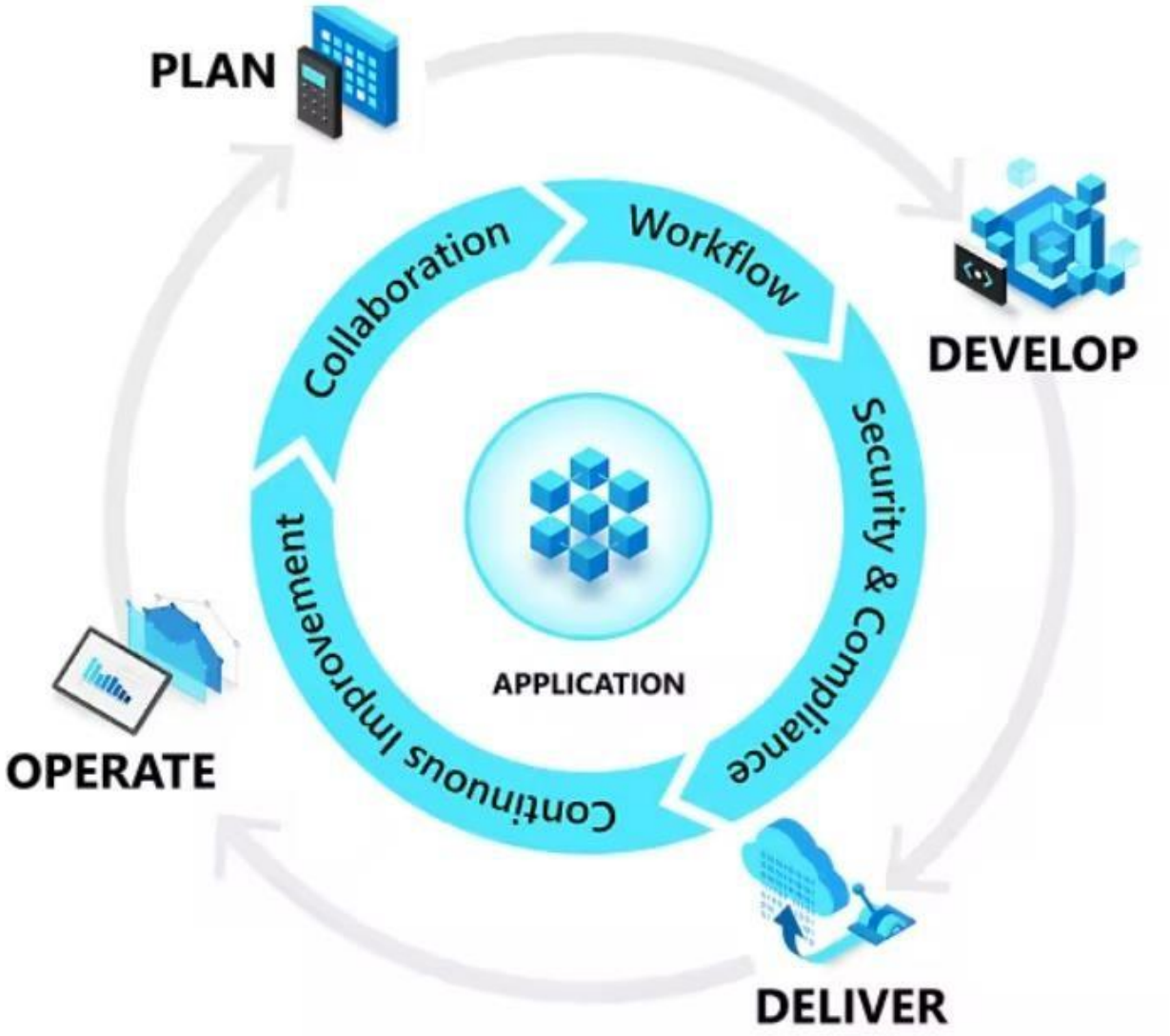
Desarrollo: Incluye todos los aspectos de la programación (escritura, pruebas, revisión e integración), la compilación de ese código que permitan implementar en varios entornos, en donde los equipos buscan innovar la eficiencia sin sacrificar la calidad, la estabilidad o productividad de sistemas (Microsoft, 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 en la nube ha transformado radicalmente la forma en la que los equipos compilan, implementan y usan las aplicaciones, al tener la capacidad de proveer y configurar entornos en la nube de varias regiones los equipos adquieren agilidad para implementar las

aplicaciones creando entornos complejos en la nube y cuando no los necesitan los apagan ofreciendo una innovación más rápida, al tener recursos flexibles, reducir costos operativos, ejecutar infraestructura con más eficacia y escalar a medida que las necesidades de negocio cambien.

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).

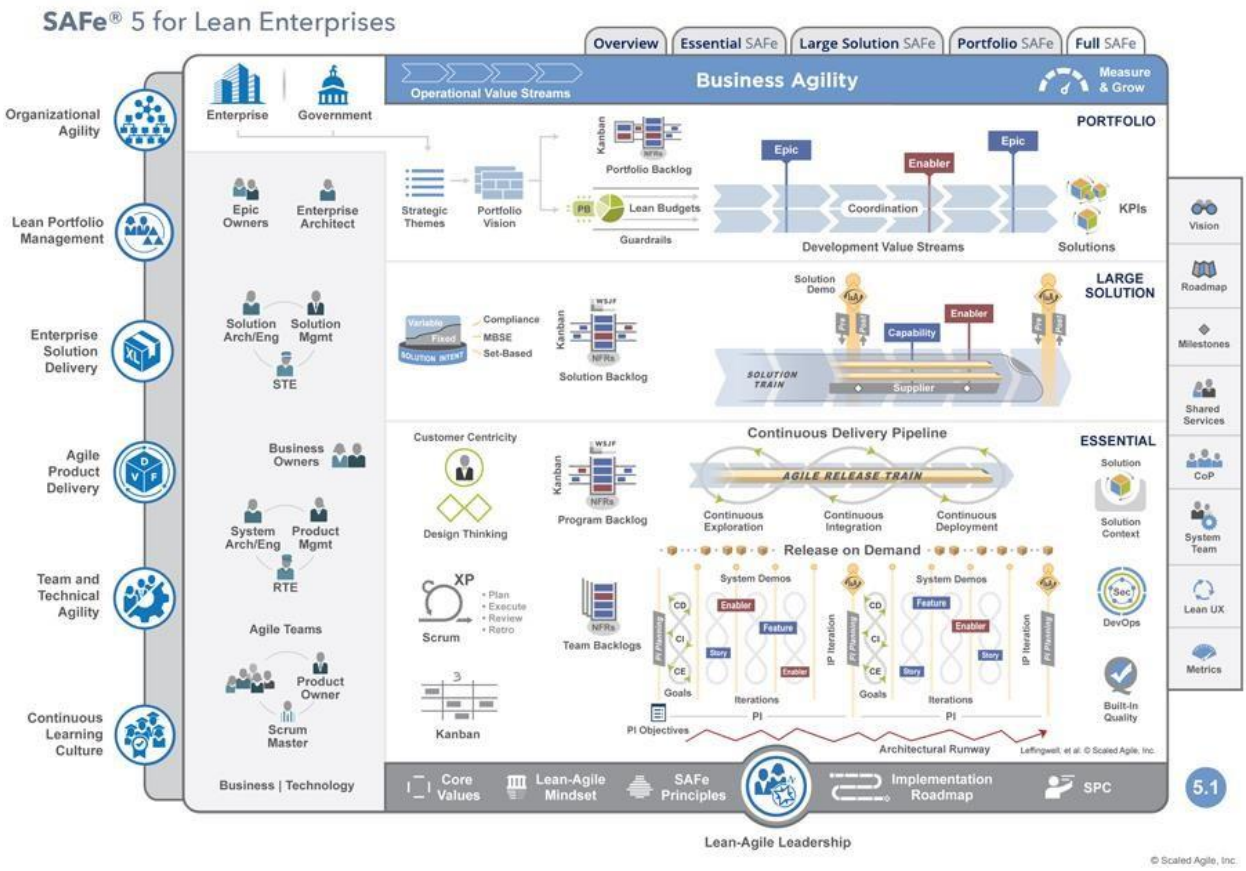
Las organizaciones pueden adaptar el marco SAFe 5 a las necesidades propias del negocio. SAFe 5 está orientado a ser escalable, lo cual permite desde el manejo de pocos equipos de trabajo, hasta los que requieren de cientos o miles de personas.

SAFe 5 es un marco muy adaptable, el cual se va actualizando continuamente, aumentando la agilidad con la que se acopla a las diferentes organizaciones. Un ejemplo de ello fue el COVID-19 el cual obligo a las organizaciones y por lo tanto a SAFe 5 a adaptarse a las circunstancias presentes durante la pandemia.

En la Figura 8 se puede observar un panorama global sobre los componentes principales involucrados dentro del marco de SAFe 5. Como principios fundamentales, valores, mentalidades, roles, artefactos y elementos de implementación que forman parte del marco SAFe 5.

Figura 8

Panorama general SAFe 5



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

Scrum

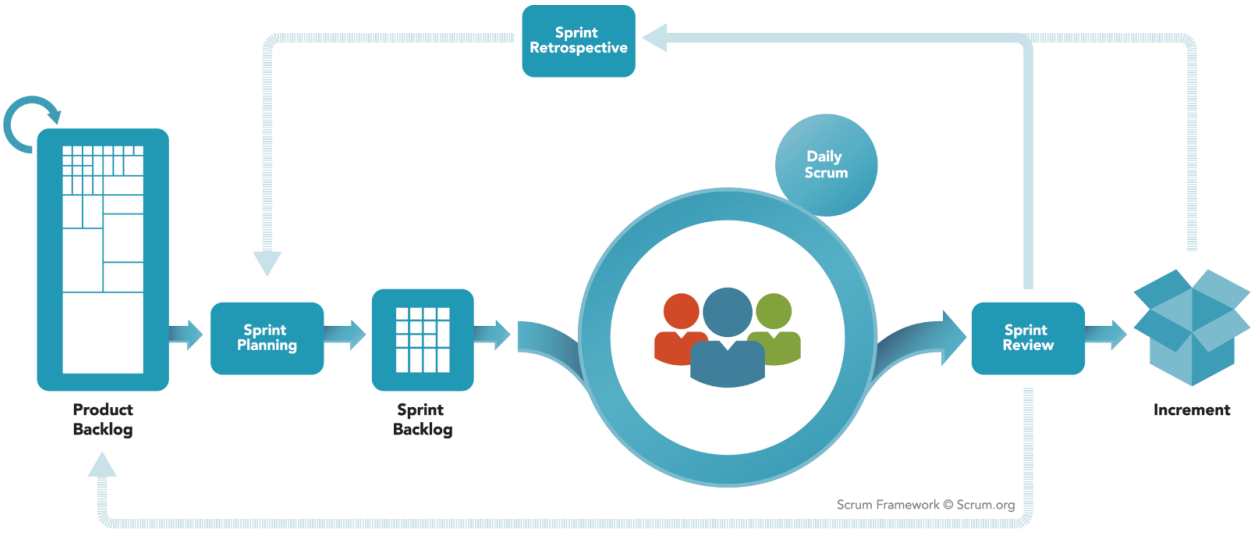
Scrum es una forma de hacer el trabajo en equipo en pequeñas piezas a la vez, con experimentación continua y bucles de retroalimentación en el camino para aprender y mejorar a medida que avanza (Scrum, 2023).

La metodología scrum permite la gestión de proyectos software de una manera rápida y ágil. Scrum se basa principalmente en: la observación, la experiencia y la experimentación. Scrum trabaja de manera iterativa lo que quiere decir que: la metodología va dando

incrementos dentro de los ciclos cortos llamados Sprint, realizando así retroalimentación continua y adaptación a los procesos de entrega

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).

*A*utomatización 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 definió con éxito todos los conceptos que se toman en cuenta dentro del proyecto, también se revisó una lista de varios trabajos que se encuentran relacionados con el presente proyecto.

Capítulo III:

Desarrollo del sistema

Introducción

En el siguiente capítulo se presenta el proceso realizado desde el desarrollo hasta el despliegue en Azure de las características del área de Salud (ficha clínica, receta, carnet, exámenes complementarios y eutanasia) y de Catálogo de Productos las mismas que forman parte del Sistema de Gestión de Clínicas Veterinarias (SGCV). Para desarrollar dichas características se utilizó una arquitectura basada en microservicios.

Metodología

Scrum es un marco de trabajo de procesos que se usa para gestionar y controlar desarrollos complejos de software. Mediante un proceso incremental, Scrum en un proyecto se ejecuta en bloques temporales (Sprints) en un tiempo definido de acuerdo a las necesidades. Cada Sprint proporciona un resultado entregable, un avance susceptible de ser entregado y revisado para que pueda ser aceptado o se soliciten nuevos cambios.

Eventos y artefactos

Los eventos en Scrum son bloques que tienen como finalidad crear una regularidad y consistencia en el proyecto, mantienen una duración máxima de tiempo y ayuda a evitar reuniones innecesarias que entorpecen los procesos del proyecto.

Tabla 1

Eventos de Scrum

Eventos	Definición
Sprint	Ciclo corto de desarrollo con una duración máxima de un mes. Cada Sprint declara un objetivo y entrega una muestra incremental al producto.

Eventos	Definición
Reunión de planificación del Sprint	Reunión en la cual se definen las tareas que formarán parte de cada Sprint. Tiene como propósito definir el objetivo y negociar qué ítems del backlog pasarán a ser ejecutados o desarrollados.
Daily Scrum	Reunión diaria al inicio de cada jornada con una duración máxima de quince minutos. El objetivo es informar sobre el trabajo que se realizó el día anterior y el trabajo que se realizará el día actual, al igual que los obstáculos que han ido surgiendo.
Sprint review	Reunión que se realiza al final de cada Sprint, donde se inspecciona el incremento de esa fase y los resultados obtenidos. El feedback obtenido puede afectar al backlog, con nuevos ítems, modificaciones o incluso eliminaciones.
Sprint retrospective	Evento que finaliza el Sprint. Se trata de una reunión que busca mejorar la forma de desempeño del Scrum Team, donde se inspeccionan a sí mismos en lo relativo a procesos, interacciones y herramientas.

Nota. Se detallan los eventos de Scrum con su definición.

Los artefactos en Scrum son aquellos elementos que ayudan a garantizar el registro y la transferencia de la información de los procesos. Es decir, son los recursos que ajustan las bases de la productividad y calidad del proyecto.

Tabla 2

Artefactos de Scrum

Artefactos	Definición
Product Backlog	Documento central del proyecto. Se trata de un inventario que contiene los requerimientos, casos de uso, tareas y dependencias del proyecto.
Sprint Backlog	

Artefactos	Definición
	Plan de trabajo del Development Team que se realizará durante la etapa de Sprint. Este artefacto ayuda a visualizar los elementos que aún no han empezado a desarrollarse, aquellos que sí y quienes trabajan en los mismos, al igual con aquellos que están esperando a desplegarse o están terminados.
Incremento	Es el resultado de todas las tareas, casos de uso, historias de usuario y todo elemento que se ha desarrollado durante el Sprint, aportando un valor de negocio en el transcurso del producto que se está desarrollando.

Nota. Se detallan los artefactos de Scrum con su definición.

Roles de Scrum

Tabla 3

Integrantes del equipo con su Rol de Scrum

Rol	Integrante	Función
Product Owner	Santiago Jácome Guerrero	Representante de las partes encargadas del sistema.
Scrum Master	José Antonio Núñez Amores	Persona encargada de mantener enfocado los objetivos del proyecto al equipo.
Team Development	Martín Enrique Camacho Moncayo José Antonio Núñez Amores	Personas encargadas del análisis, diseño, desarrollo, despliegue y pruebas del sistema.

Nota. Se detalla los integrantes del equipo con su rol y función durante el proyecto.

Análisis del sistema

Después de definir los roles del equipo Scrum en la metodología, se llevó a cabo una reunión para definir la técnica a utilizar para el levantamiento de requerimientos, con la cual se llegó a un acuerdo de utilizar la técnica de entrevistas que consiste en mantener una

conversación con el entrevistado realizando preguntas específicas para recabar datos que ayudaran al desarrollo del sistema, por ello se realizó una plantilla de preguntas Anexo 1 las cuales se presentaron en cuatro veterinarias (Canopolis, Vital Pet, Royal Hound) ubicadas en la ciudad de Ambato, obteniendo los siguientes resultados en el Anexo 2 respectivamente.

Análisis de dominio

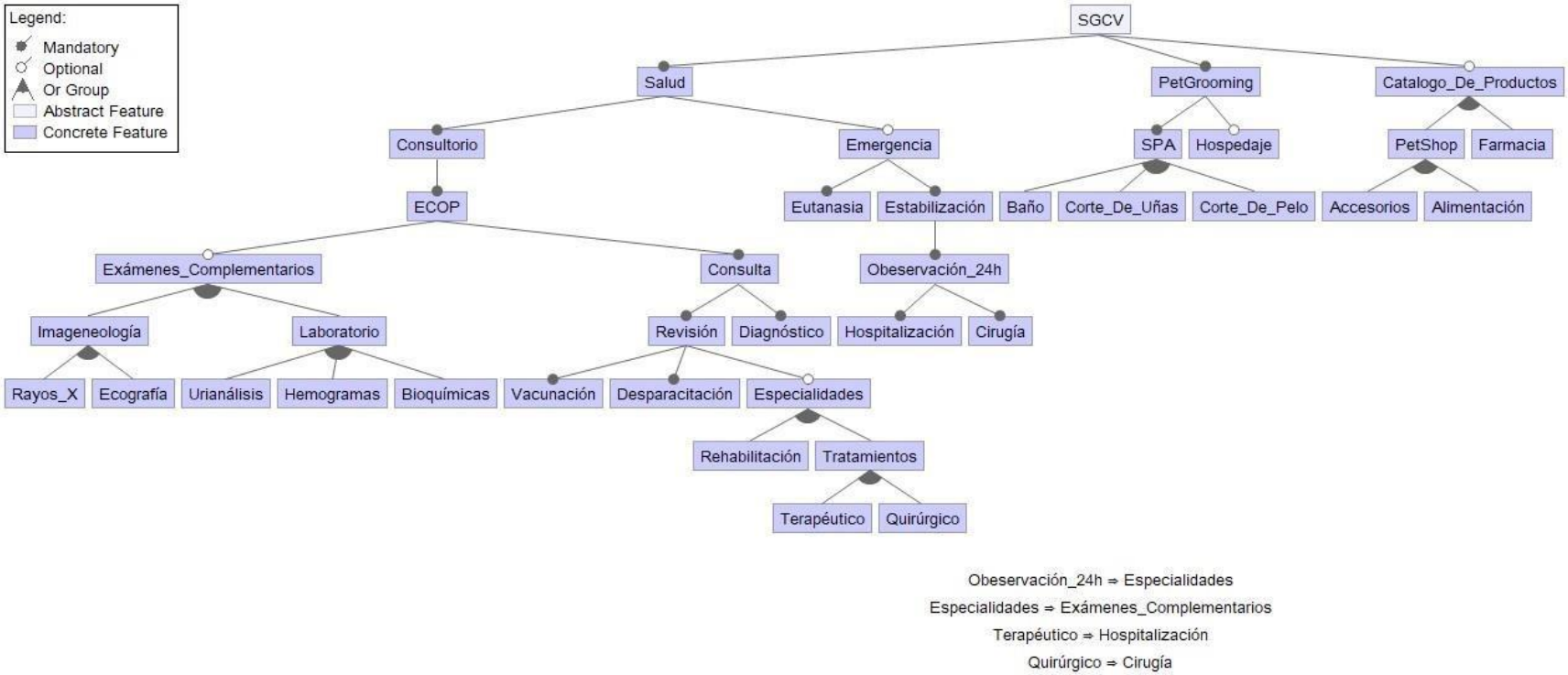
Una vez realizadas las entrevistas se acordó una reunión en la cual se intercambió la información obtenida en las veterinarias y se pudo establecer las características comunes de estas, y posteriormente se realizó el diagrama de Análisis de Dominio Orientado a Objetos (FODA) en su primera versión Anexo 3 diagramado en Lucidchart.

Para la segunda versión del FODA Anexo 3 diagramado en Lucidchart se pactó con el Scrum Team que se usarían términos técnicos acordes a las clínicas veterinarias y relaciones no jerárquicas.

Debido a la falta de información concreta en las entrevistas se realizó una reunión con un experto en el tema para aclarar el flujo de trabajo formal que se lleva en una veterinaria y como consiguiente se obtuvo la versión final del FODA como se muestra en la Figura 10 la cual fue diagramada en featureIDE, un framework para FOSD (Feature-Oriented Software Development) basado en Eclipse.

Figura 10

FODA versión final



Nota. FODA en su versión final diagramado en la herramienta featureIDE.

Aplicando SAFe

Al aplicar el marco de trabajo Scaled Agile Framework (SAFe) se obtuvo las épicas de salud y catálogo de productos como se muestran en la Tabla 4 y 12 respectivamente.

Dentro de la épica de salud se obtuvo las capacidades de Examen Clínico Orientado a Problemas (ECOP) como se muestra en la Tabla 5 con sus características principales que son ficha clínica, carnet, receta y exámenes complementarios como se muestran en las Tablas 6, 7, 8 y 9, al igual que la capacidad de eutanasia como se muestra en la Tabla 10 con su característica principal formulario de eutanasia como se muestra en la Tabla 11.

Dentro de la épica de catálogo de productos que se muestra en la Tabla 12, se obtuvo la característica principal de producto como se muestra en la Tabla 13.

Tabla 4

Épica de salud

Hipótesis Épica	
Fecha de Inicio	19 de septiembre del 2022
Nombre	Servicios de *"**Salud"*** para veterinarias
Coordinador	Julio Castro
Descripción	
Para	Clínicas Veterinarias
Quién	Veterinarias que quieren automatizar sus procesos internos.
La/El	Sistema de gestión de clínicas veterinarias.
¿Qué es?	Aplicación Web Progresiva (PWA) basado en línea de producto software.
Esto	Permite registrarse a los clientes de la clínica veterinaria y a sus mascotas; el veterinario (especialista) puede registrar la ficha clínica del paciente y permite generar el historial clínico, e ingreso a hospitalización y emergencia.
Diferente a	Vetpraxis
Nuestra Solución	Puede ser utilizada tanto por los veterinarios y personas que trabajen en la clínica.
Resultados Comerciales	• Los clientes pueden acceder al sistema para consultar y programar citas.

Hipótesis Épica	
	• Disminuye el papeleo realizado por el veterinario.
	• El veterinario puede registrar y consultar las fichas e historial clínico.
Indicadores Principales	• Aumento de reservas de citas a la clínica veterinaria.
	• Aumento de pacientes en la clínica.
	• Cantidad de clientes registrados en el sistema.
Requerimientos No funcionales	• La seguridad debe estar basada en roles de usuario.
	• El sistema debe estar disponible 24/7.

Nota. Se muestra la Tabla de la épica de salud en el formato proporcionado por SAFe.

Tabla 5

Capacidad ECOP

Capacidad	**Hipótesis de beneficio**
Gestión de ECOP	El usuario autorizado (veterinario) puede crear fichas clínicas y consultar el paciente.
Criterio de aceptación	
• Un usuario autorizado (Veterinario) puede realizar una consulta y determinar su estado de salud.	
• El usuario autorizado (Veterinario) debe llenar el formulario para poder asignar el respectivo procedimiento para la salud del paciente.	

Nota. Se muestra la Tabla de capacidad de ECOP proveniente de la épica de salud en el formato proporcionado por SAFe.

Tabla 6

Característica ficha clínica

Característica	**Hipótesis de beneficio**
Crud de la ficha clínica.	El veterinario puede registrar las fichas clínicas de sus pacientes de manera rápida.
	Los datos por registrar en la ficha son: Motivo de la consulta, tratamiento, temperatura, pulso, frecuencia cardiaca, peso, estado de ánimo y mucosas.
Criterio de aceptación	
• El veterinario autorizado puede registrar las fichas clínicas de cualquier paciente existente o no.	
• El veterinario puede iniciar la ficha clínica de manera automática al realizar una consulta.	

Nota. Se muestra la Tabla de características de ficha clínica proveniente de la capacidad de ECOP en el formato proporcionado por SAFe.

Tabla 7

Característica carnet

Característica	**Hipótesis de beneficio**
Gestión de carnet	Permite al usuario autorizado (Veterinario) gestionar toda la información de vacunas de sus pacientes.
	Los datos por registrar en el carnet son: Ultima fecha de desparasitación y una lista de vacunas la cual contenga la siguiente información: nombre de la vacuna, laboratorio, fecha de revacunación, descripción de la vacuna.
Criterio de aceptación	
• El usuario autorizado (Veterinario) puede administrar de manera correcta, permitiéndole crear, modificar, borrar y buscar información respecto a los carnets.	

Nota. Se muestra la Tabla de características de carnet proveniente de la capacidad de ECOP en el formato proporcionado por SAFe.

Tabla 8

Característica receta

Característica	**Hipótesis de beneficio**
Gestión de receta	Permite al veterinario administrar las recetas de todos sus pacientes.
	Los datos por registrar en la receta son: indicaciones generales, fecha de emisión y los medicamentos con sus respectivas indicaciones.
Criterio de aceptación	
• El veterinario puede administrar de manera correcta y ordenada las recetas de sus pacientes.	

Nota. Se muestra la Tabla de características de gestión de receta proveniente de la capacidad de ECOP en el formato proporcionado por SAFe.

Tabla 9

Característica exámenes complementarios

Característica	**Hipótesis de beneficio**
Administración de los exámenes complementarios	El veterinario puede crear un formulario con los resultados del paciente que va enlazado al historial clínico.
	Los datos por registrar al realizar un examen complementario son: Tipo de examen, nombre del examen, resultados, descripción, y archivos adjuntos.
Criterio de aceptación	
• Un usuario autorizado puede crear un formulario para los resultados de los exámenes complementarios de un paciente.	
• El usuario autorizado debe llenar el formulario con los resultados obtenidos del paciente.	
• El usuario autorizado puede modificar el formulario de resultados cuando sea pertinente.	
• El usuario autorizado puede ver el formulario de resultados cada vez que lo requiera.	

Nota. Se muestra la Tabla de características de exámenes complementarios proveniente de la capacidad de ECOP en el formato proporcionado por SAFe.

Tabla 10

Capacidad gestión de eutanasia

Capacidad	**Hipótesis de beneficio**
Gestión de eutanasia	El veterinario podrá registrar el proceso de eutanasia de los pacientes. Los datos por registrar en la eutanasia son: Motivo, fecha de aprobación y fecha de emisión.
Criterio de aceptación	
El veterinario registrara la eutanasia de los pacientes dentro del sistema.	

Nota. Se muestra la Tabla de capacidad de gestión eutanasia proveniente de la épica de salud en el formato proporcionado por SAFe.

Tabla 11

Características formulario de eutanasia

Característica	**Hipótesis de beneficio**
Crud de formulario de eutanasia	El veterinario llenara un formulario preparando así la impresión de un documento para la aprobación de la eutanasia por parte del tutor del paciente.
Criterio de aceptación	

El veterinario llenara el formulario y generara el documento listo para imprimir.

Nota. Se muestra la Tabla de características de formulario de eutanasia proveniente de la capacidad de eutanasia en el formato proporcionado por SAFe.

Tabla 12

Épica de productos

Hipótesis Épica	
Fecha de Inicio	19 de septiembre del 2022
Nombre	Servicios de venta de productos para clínicas veterinarias.
Coordinador	Martin Camacho
Descripción	
Para	Clínicas Veterinarias
Quien	Clínicas veterinarias que quieren automatizar sus procesos internos.
La/El	Sistema de gestión de clínicas veterinarias.
Que es	Sistema web basado en línea de producto software.
Esto	Permite realizar el control de inventario de los productos que se venden en la clínica veterinaria como lo son medicamentos, accesorios y comida de las mascotas.
Diferente a	VetPraxis
Nuestra Solución	Puede ser utilizada tanto por los veterinarios y los clientes de la clínica.
Resultados Comerciales	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.
Indicadores Principales	Más eficiencia en la venta de productos.
	Economizar en la clínica veterinaria y no tener productos en exceso ni sin disponibilidad.
	Saber que productos son los más vendidos y cuáles no.
	La seguridad debe estar basada en roles de usuario.

Requerimientos funcionales	**No**	El sistema debe estar disponible 24/7.

Nota. Se muestra la Tabla de la épica de productos en el formato proporcionado por SAFe.

Tabla 13

Característica productos

Característica	**Hipótesis de beneficio**
Crud de productos	Realizar un registro de los productos tanto de entrada como salida.
	La información por registrar de cada producto es: Nombre, tipo de producto (farmacéutico, alimento, accesorio), cantidad existente, precio de venta, precio de compra, código único del producto.
Criterio de aceptación	
• Un usuario autorizado puede verificar la cantidad de productos disponibles.	
• Un usuario autorizado puede registrar, actualizar o eliminar un determinado producto según sea el caso.	
• Los datos de los medicamentos como nombre o código deben validarse que no se repitan antes de ser almacenados.	

Nota. Se muestra la Tabla de características de productos proveniente de la épica de productos en el formato proporcionado por SAFe.

Tabla 14

Historias de Usuario

Código	**Rol**	**Descripción**	**Responsables**	**Resultado**
HU003	Como usuario	Quiero crear, actualizar, listar, eliminar una ficha clínica, para llevar los registros de la consulta.	Martin Camacho y José Núñez	El usuario puede realizar una consulta de la se crea una ficha clínica.

Código	Rol	Descripción	Responsables	Resultado
HU004	Como usuario	Quiero crear, actualizar, listar y eliminar recetas, para tener un registro de los medicamentos aplicados al paciente.	Martin Camacho y José Núñez	El usuario podrá crear, actualizar, buscar y eliminar recetas.
HU005	Como usuario	Quiero crear, actualizar, listar y eliminar exámenes complementarios, para poder tener un diagnóstico más claro acerca de la salud del paciente.	Martin Camacho y José Núñez	El usuario podrá crear, actualizar, buscar y eliminar exámenes complementarios.
HU006	Como usuario	Quiero crear, actualizar, listar y eliminar carnets, para poder tener un registro de las vacunas de los pacientes.	Martin Camacho y Jose Núñez	El usuario puede crear, actualizar, buscar y eliminar los carnets de los pacientes.
HU007	Como usuario	Quiero crear, actualizar, listar y eliminar medicamentos, para poder tener un registro de los medicamentos asignados en una receta para un paciente.	Martin Camacho y Jose Nuñez	El usuario puede crear, actualizar, buscar y eliminar un medicamento.

Código	Rol	Descripción	Responsables	Resultado
HU008	Como usuario	Quiero crear, actualizar, listar y eliminar vacunas, para poder tener un registro de las vacunas aplicadas en un paciente.	Martin Camacho y Jose Nuñez	El usuario puede crear, actualizar, buscar y eliminar vacunas.
HU014	Como usuario	Quiero crear, actualizar, listar y eliminar productos, para poder tener un registro con la información de los productos que entran y salen de la veterinaria.	Martin Camacho y Jose Núñez	El usuario puede crear, actualizar, buscar y eliminar productos.

Nota. Se detallan las historias de usuario de las características de cada capacidad.

Product Backlog

Tabla 15

Product backlog

Historia de Usuario	Estimación	Fecha Inicio	Fecha Fin	Sprint	Actividades realizadas
HU003	20	7/10/2022	26/10/2022	1	Creación de Tablas en base de datos. Creación de Repositorio y ramas. Conexión de base de datos. Creación del proyecto e instalación de
HU004	15	27/10/2022	10/11/2022	1	
HU005	15	11/11/2022	25/11/2022	2	
HU006	13	26/11/2022	8/12/2022	2	

Historia de Usuario	Estimación	Fecha Inicio	Fecha Fin	Sprint	Actividades realizadas
HU007	18	9/12/2022	26/12/2022	1	dependencias. Desarrollo de rutas, controladores. Creación de validaciones. Pruebas en Postman. Despliegue de microservicios. Pruebas de rendimiento.
HU008	13	27/12/2022	8/1/2023	2	
HU014	12	9/12/2022	20/1/2023	3	

Nota. Se detalla el Product Backlog de las historias de usuario descritas en la Tabla 14.

Diseño del sistema

En este apartado se detalla el proceso de construcción que se siguió para la planificación de la solución del sistema.

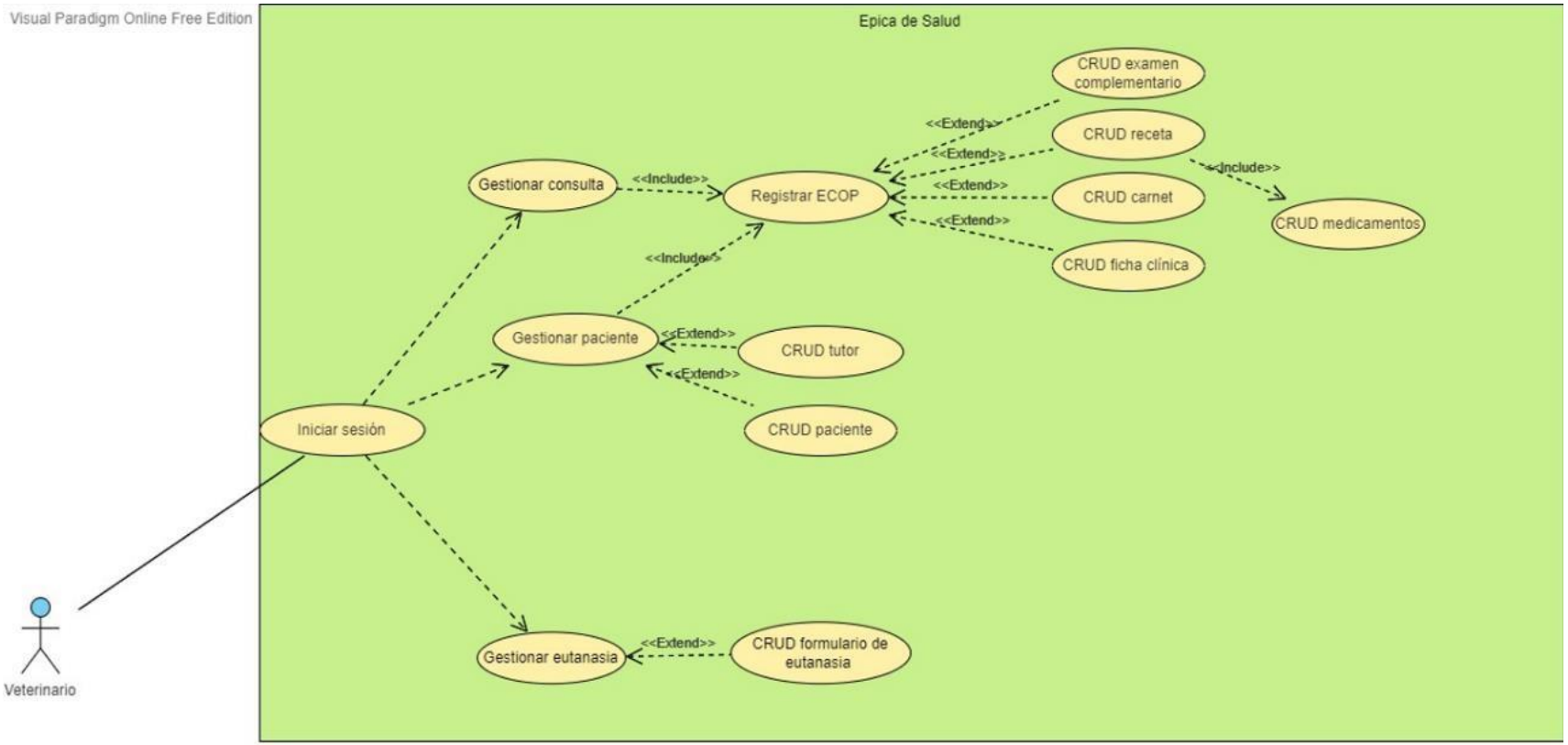
Modelado UML

Realizar el modelado UML permite demostrar visualmente la estructura y el comportamiento que se llevó a cabo para el desarrollo del sistema.

Casos de uso. En la Figura 11 se muestra la relación desde el momento en que se inicia la sesión y se accede a las funcionalidades para el área de salud.

Figura 11

Diagrama de casos de uso del Área de Salud

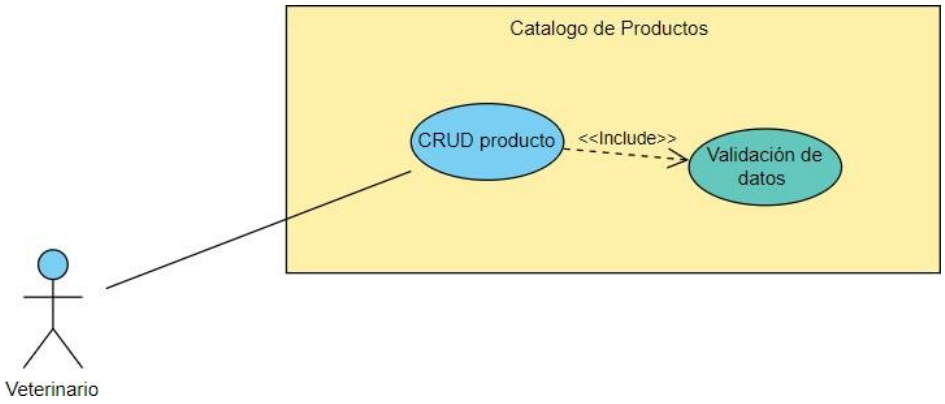


Nota. Se muestra de manera detallada las funcionalidades a las que el veterinario tendrá acceso en el área de Salud.

En la Figura 12 se muestra la interacción del usuario con el CRUD producto.

Figura 12

Diagrama de casos de uso de Catálogo de Productos



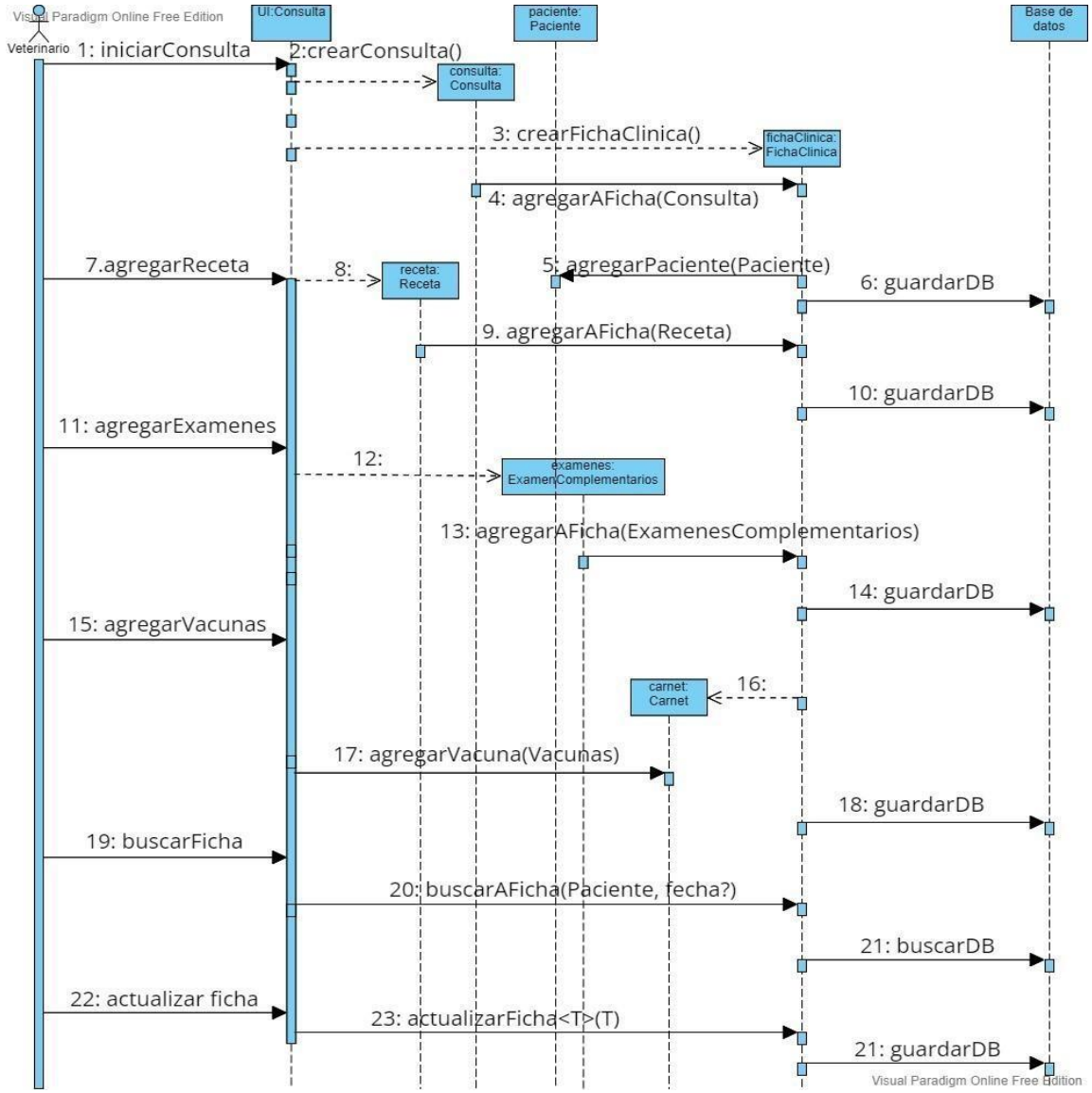
Nota. Se muestra de manera detallada las funcionalidades a las que el veterinario tendrá acceso en el área de Salud.

Casos de uso extendido. Los casos de uso extendido se encuentran alojados en el Anexo 4.

Diagrama de secuencia. En la Figura 13, se muestra la comunicación entre los distintos componentes que interactúan en el área de salud.

Figura 13

Diagrama de Secuencia de la capacidad de ECOP perteneciente al área de Salud

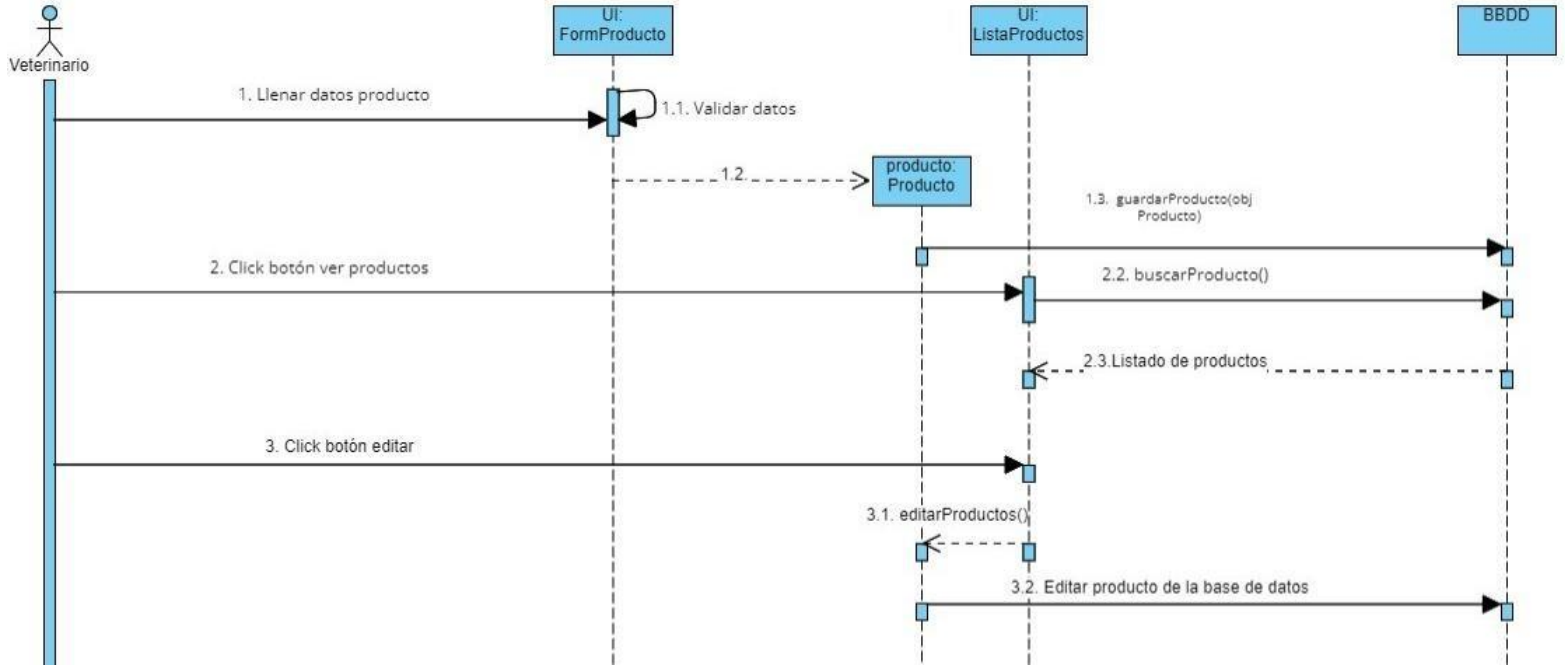


Nota. Se muestra las interacciones entre las historias de usuario obtenidas en la Tabla 14.

En la Figura 14, se muestra la comunicación entre los distintos componentes que interactúan en el área de Catálogo de Productos.

Figura 14

Diagrama de secuencia de la épica catálogo de producto

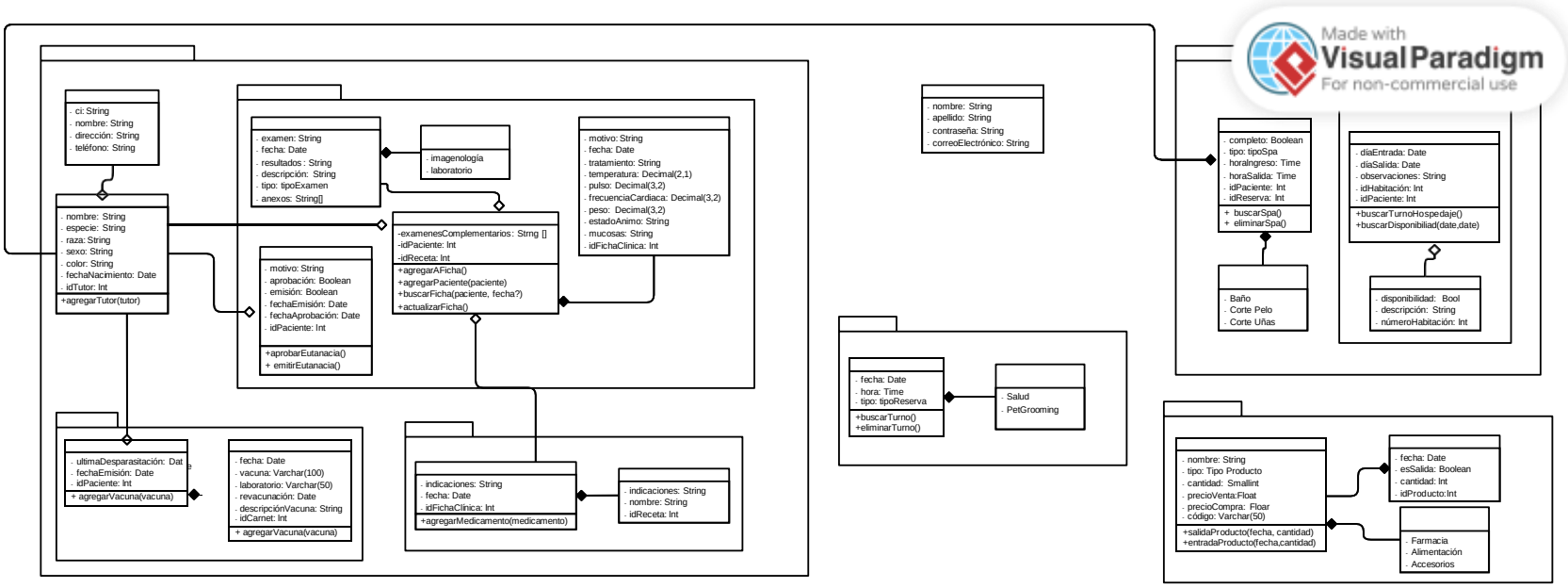


Nota. Se muestra las interacciones entre las historias de usuario obtenidas en la Tabla 14.

Diagrama clases. El diagrama de clase se desarrolló basándonos en las historias de usuario, las que vamos a tratar en este proyecto se encuentran en la Tabla 14. Este diagrama es clave para la construcción de la base de datos el cual explicaremos posteriormente.

Figura 15

Diagrama de clases del SGCV

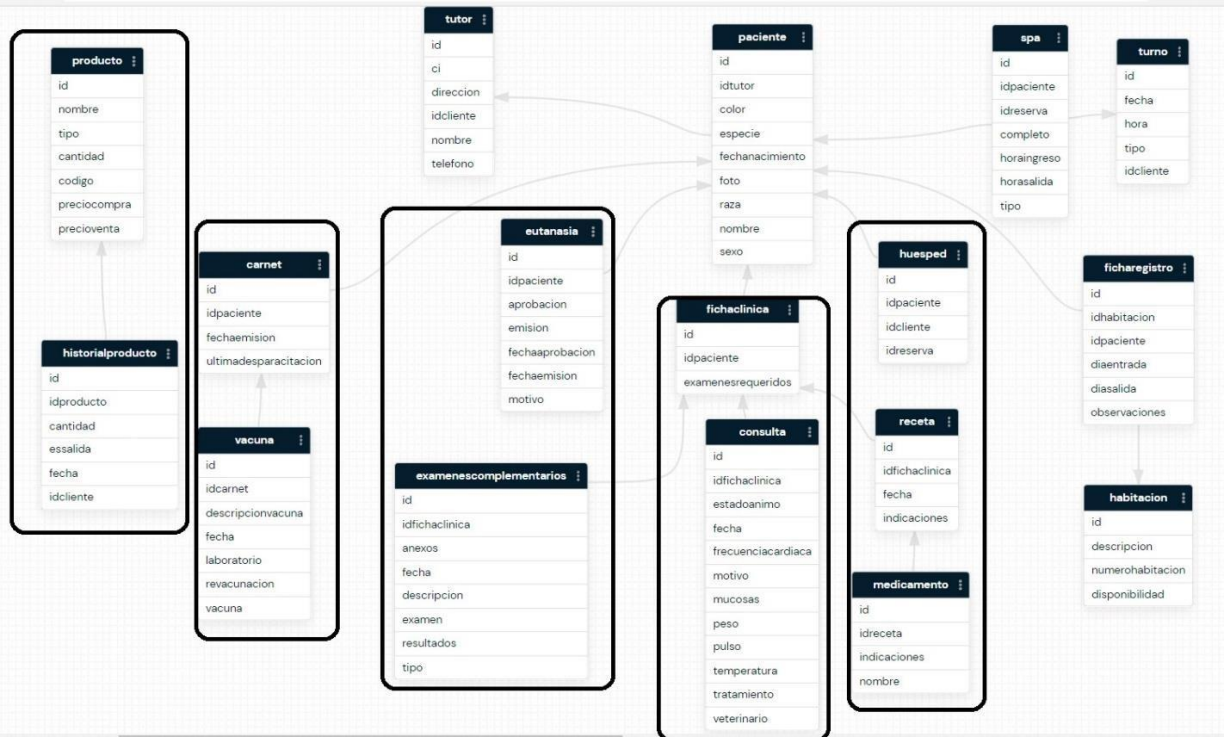


Nota. Se muestra el diagrama de clases.

Diagrama Entidad-Relación. Se desarrolló el diagrama Entidad-Relación que fue generado a partir del diagrama de clases de la Figura 15 ya que cada clase representa una entidad. En la Figura 16, se puede observar las entidades señaladas que han sido asignadas al equipo de trabajo.

Figura 16

Diagrama Entidad-Relación

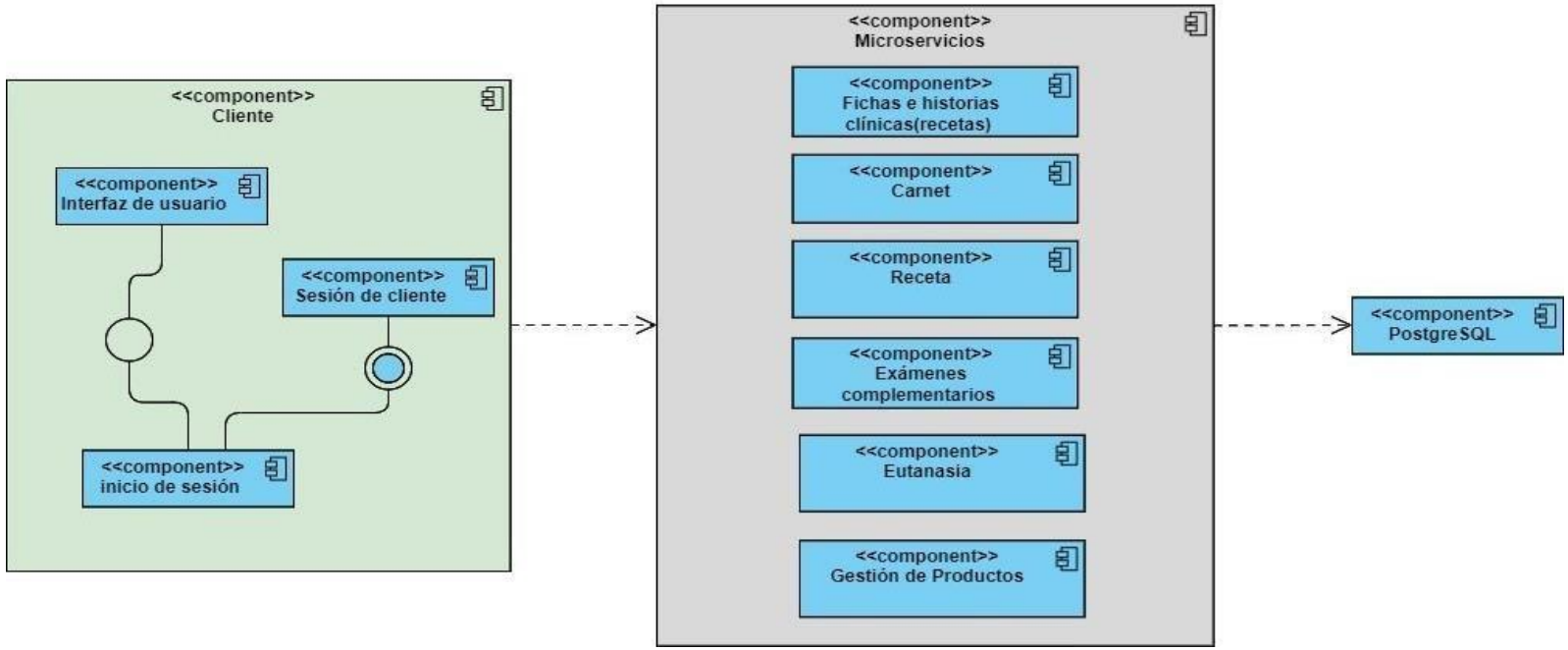


Nota. Se muestra el diagrama de entidad relación basado en el diagrama de clases.

Diagrama de Componentes. En la Figura 17, se muestra una representación visual de las dependencias entre los componentes que forman parte del área de Salud y Catálogo de Productos, su relación entre ellos y como interactúan.

Figura 17

Diagrama de componentes para los microservicios de Salud y Catálogo de Productos

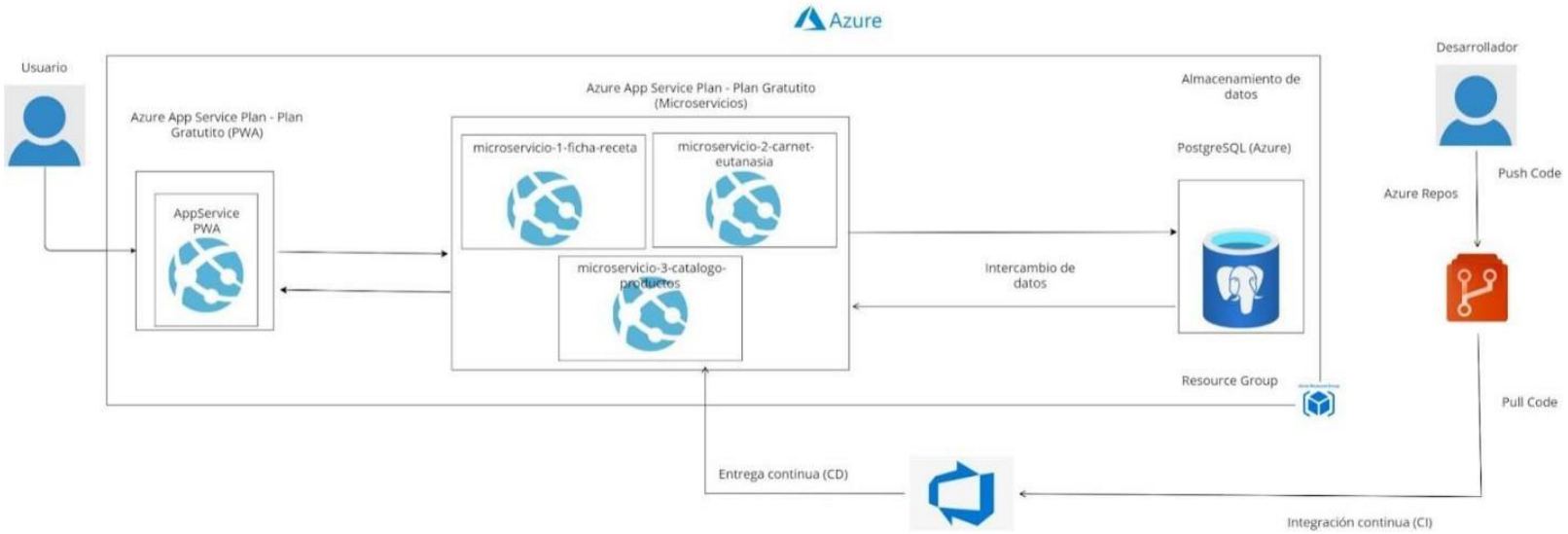


Nota. Se muestra el diagrama de componentes para los microservicios de salud y catálogo de productos.

Despliegue. En la Figura 18, se muestra el diagrama de despliegue de los microservicios de Salud y Catálogo de Productos, mediante una representación gráfica para visualizar como el sistema está configurado en producción.

Figura 18

Diagrama de despliegue para los microservicios de Salud y Catálogo de Productos

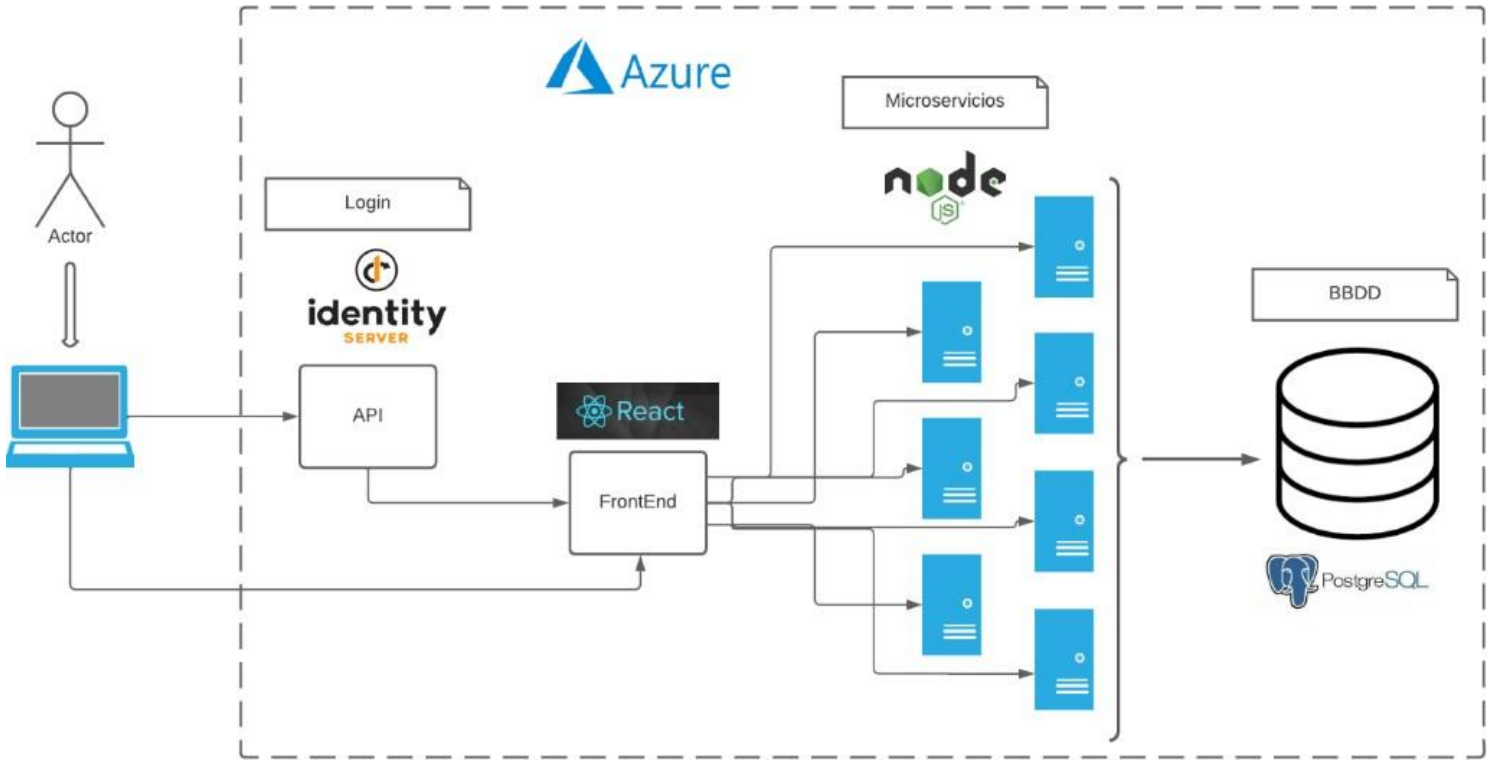


Nota. Se muestra el diagrama de despliegue basado en la arquitectura, herramientas y desarrollo continuo.

Arquitectura. En la Figura 19, se muestra la representación visual de la estructura del sistema, está basado en los componentes claves y como están enlazados entre sí.

Figura 19

Diagrama de la arquitectura orientada en microservicios del SGCV



Nota. Se muestra el diagramade la arquitectura del SGCV.

Desarrollo del sistema

Herramientas

Para desarrollar los microservicios orientados al área de Salud y Catálogo de Productos se planificó realizar tres Sprint y usar las siguientes herramientas para su desarrollo.

Tabla 16

Herramientas de desarrollo

N°	Nombre	Descripción
1	Node.js	Es un entorno de servidor multiplataforma de tiempo real de ejecución JavaScript de back-end puede ejecutarse en Windows, Linux, Unix, macOs se utilizó en su versión 18.12.1 para los microservicios orientados al área de Salud y Catálogo de Producto.
2	PostgreSQL	Para el almacenamiento de base de datos se utilizó este sistema de gestión de base de datos relacionales orientado a objetos y con código abierto usando la versión 14.
3	pgAdmin	Herramienta que permite gestionar de manera gráfica la base de datos para este caso se usó la versión 4.
4	Liquibase	Es una librería de código abierto que permite gestionar los cambios en la base de datos especialmente en un entorno de desarrollo ágil.

N°	Nombre	Descripción
5	Azure DevOps	El Server de Microsoft permite llevar el control de versiones, generar un repositorio por cada microservicio desarrollado, gestionar los requerimientos en base a las actividades de cada sprint permitiendo así tener el código mejor organizado y más seguro.
6	Visual Studio Code	Es un editor de código fuente que se ejecuta en el escritorio y permite depuración de código.
7	pg	Es un módulo de la colección node-Postgres que permite interactuar con una base de datos PostgreSQL.
8	Express.js	Es un framework de desarrollo minimalista para Node.js y permite estructurar una aplicación web de manera ágil.
9	express-validator	Es un middleware de express.js para validar información.
10	nvm	Software para gestión de múltiples versiones de Node.js en un mismo ordenador.
11	nodemon	Es una herramienta que permite monitorear cualquier cambio realizado en nuestra aplicación de Node.js
12	git	Es un sistema de control de versiones distribuido que realiza un seguimiento de los cambios en cualquier conjunto de archivos de una computadora.
13	gitflow	Esta herramienta basada en Git, ofrece un modelo de ramificación estricto acorde a gitflow y se puede instalar como plugin adicional en Visual Studio Code.
14	Postman	Herramienta que permite testear APIs.

Nota. Se muestra de manera detallada las herramientas utilizadas en el aplicativo con su respectiva definición.

Sprints

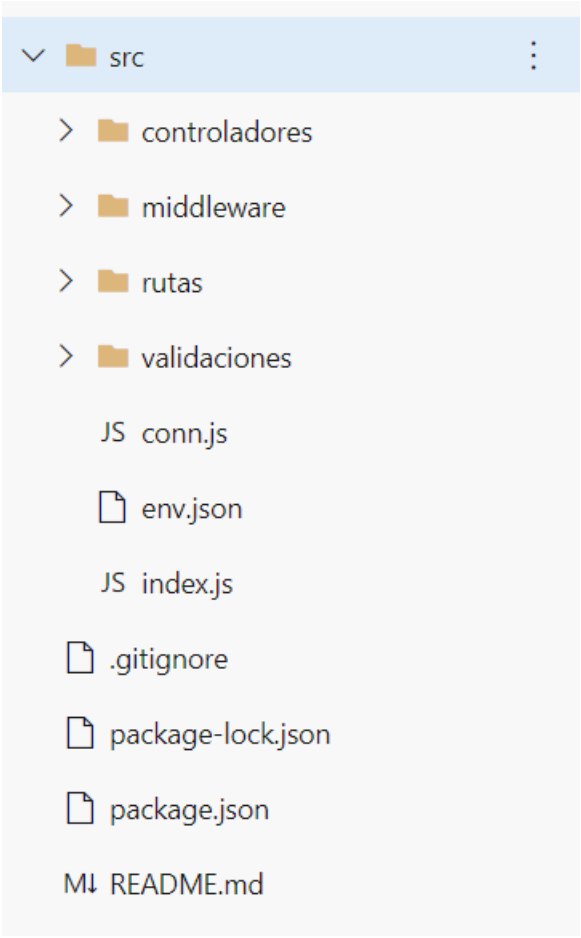
En el desarrollo de cada Sprint se tuvo en cuenta el orden jerárquico del diagrama FODA, lo cual permite dar prioridad a las historias de usuario primordiales para el desarrollo del sistema.

En cada Sprint se desarrolló un microservicio, los cuales se definieron por las características mencionadas en la Tabla de Épicas, capacidades y características.

En el desarrollo de cada microservicio se mantuvo una estructura de directorios para facilitar el desarrollo y organizar los archivos como se muestra en la Figura 21. Se manejo una carpeta fuente (src) la cual contiene directorios para controladores, rutas, middleware y validaciones, a su vez almacena archivos que permiten la conexión a la base de datos mediante el módulo pg junto con el archivo index.js, responsable de inicializar la aplicación. Para la validación de los campos correspondientes a cada microservicio se utilizó express-validator, un middleware que permite validar una solicitud, y si una regla de validación configurada falla devuelve una respuesta de error.

Figura 20

Estructura de directorios



Nota. Se muestra la estructura de directorios y archivos de los microservicios.

Base de datos. El desarrollo de la base de datos se llevó a cabo en PostgreSQL, un gestor de base de datos relacional orientado a objetos. Se desarrolló en el editor de código fuente Visual Studio Code usando la herramienta Liquibase que permitió sincronizar la base de datos con el repositorio en Azure DevOps y tener un registro de los cambios realizados.

Para el desarrollo de la base de datos se tomó en cuenta el diagrama entidad-relación, el cual facilito la creación de las Tablas con sus respectivos campos y relaciones, teniendo en cuenta que al usar la herramienta Liquibase se debe registrar quien realizo el cambio con su contexto y comentario correspondiente, como se muestra en la Figura 21.

Una vez que se realizó algún cambio en la base de datos se utiliza el comando Liquibase update el cual permite sincronizar la base de datos. Para que los cambios sean aceptados se realiza un pull request a la rama principal lo cual ayuda a desplegar la base de datos con sus cambios, y esto se mostrara en el gestor pgAdmin como se muestra en la Figura 22.

Figura 21

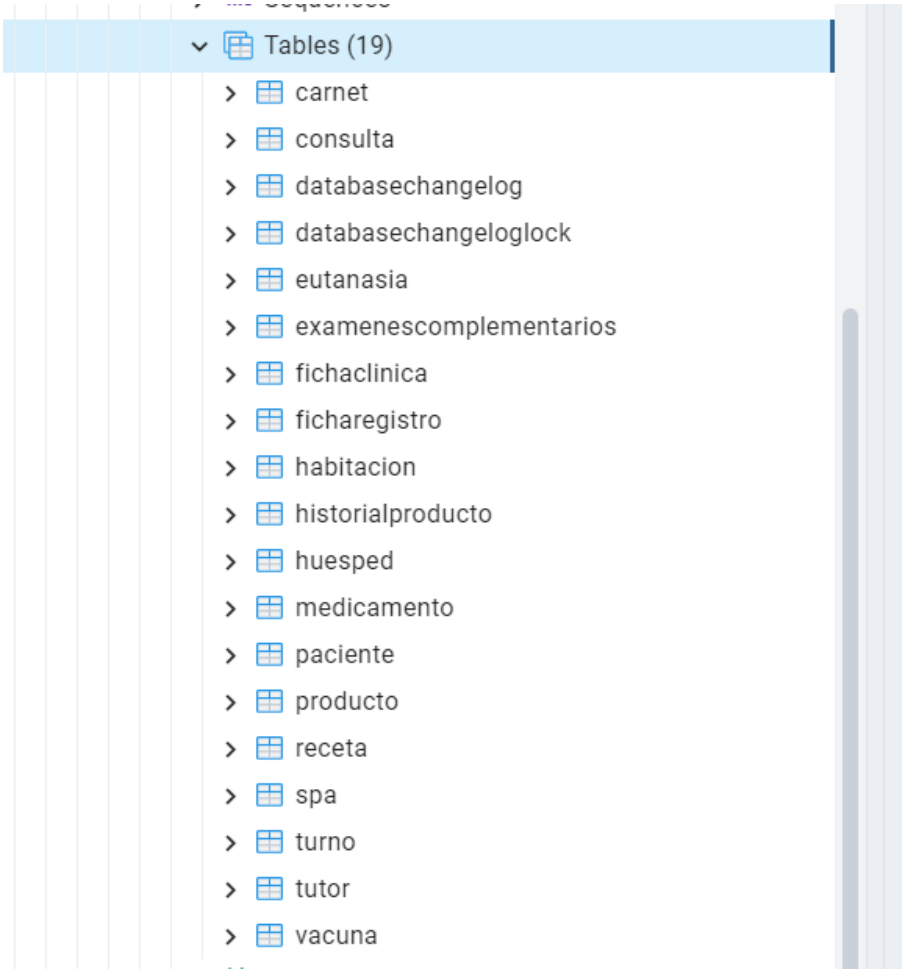
Estructura del código de desarrollo de la base de datos

```
--liquibase formatted sql

--changeset martin.camacho:1 labels:implementacion-receta context:16-h17
--comment: crear tabla receta y medicamento

create table receta (
    id INT GENERATED ALWAYS AS IDENTITY,
    indicaciones text null,
    fecha date,
    primary key (id)
);
create table medicamento (
    id INT GENERATED ALWAYS AS IDENTITY,
    indicaciones text null,
    idReceta int not null,
    primary key (id)
);

alter table medicamento
    add constraint fkRecetaMedicamento
    foreign key (idReceta)
    references receta (id);

--rollback DROP TABLE medicamento,receta;
```

Nota. Se muestra la estructura que se utilizó para el desarrollo de la base de datos.

Figura 22

Despliegue de base de datos



Nota. Se muestra la estructura de la base de datos desplegada en pgAdmin.

Sprint 1. En la Tabla 17, 18 y 19 se muestran las historias de usuario de las características: Ficha clínica, Receta y Medicamento respectivamente, que pertenecen al Sprint 1, esto ayuda al desarrollo de los microservicios. Cada historia de usuario describe las funcionalidades del sistema.

Tabla 17

Historia de usuario detallada de Ficha Clínica

Ficha Clínica		
Código: HU003		Estimación: 20 días
Como Veterinario, quiero crear, actualizar, listar, eliminar una ficha clínica, para llevar los registros de la consulta.		
Criterios de aceptación		
1.	Crear Ficha Clínica	• El veterinario autorizado puede crear una nueva ficha clínica para un paciente cuando asista a consulta.
2.	Actualizar Ficha Clínica	• El veterinario autorizado puede modificar la información dentro de la ficha clínica del paciente.
3.	Listar Ficha Clínica	• Al momento de realizar una búsqueda de la ficha clínica se deberá ingresa la información o nombre del paciente que se está buscando la ficha clínica.
4.	Eliminar Ficha Clínica	• Debe existir una ficha clínica registrada y eliminar todos los registros asociados.

Nota. Se muestra la historia de usuario de ficha clínica que se desarrolló en el primer Sprint.

Tabla 18

Historia de usuario detallada de Receta

Receta		
Código: HU004		Estimación: 15 días
Como Veterinario, quiero crear, actualizar, listar y eliminar recetas para tener un registro de los medicamentos aplicados al paciente.		
Criterios de aceptación		
1.	Crear Receta	• El veterinario recibe un mensaje de confirmación si la receta se guardó correctamente.
2.	Actualizar Receta	• El veterinario puede actualizar la información de una receta las veces que sea

Receta		
		necesario y recibirá un mensaje cuando se guarden los datos.
3.	Listar Receta	• El veterinario puede buscar recetas de cualquier paciente.
4.	Eliminar Receta	• El veterinario puede borrar tantas recetas como sea necesario.

Nota. Se muestra la historia de usuario de receta que se desarrolló en el primer Sprint.

Tabla 19

Historia de usuario detallada de Medicamento

Medicamento		
Código: HU007		Estimación: 18 días
Como Veterinario, quiero crear, actualizar, listar y eliminar medicamentos para poder tener un registro de los medicamentos asignados en una receta para un paciente.		
Criterios de aceptación		
1.	Crear Medicamento	• El veterinario puede crear uno o más medicamentos en una receta siempre que sea necesario.
2.	Actualizar Medicamento	• El veterinario puede actualizar la información de los medicamentos las veces que sea necesario.
3.	Listar Medicamento	• El veterinario puede buscar medicamentos de las recetas de un paciente.
4.	Eliminar Medicamento	• El veterinario puede borrar tantos medicamentos como sea necesario.

Nota. Se muestra la historia de usuario de medicamento que se desarrolló en el primer Sprint.

Sprint 2. En la Tabla 20, 21 y 22 se muestran las historias de usuario de las características: Exámenes Complementarios, Carnets y Vacunas respectivamente, que pertenecen al Sprint 2, esto ayuda al desarrollo de los microservicios. Cada historia de usuario describe las funcionalidades del sistema.

Tabla 20

Historia de usuario detallada de Exámenes Complementarios

Exámenes Complementarios		
Código: HU005		Estimación: 15 días
Como Veterinario, quiero crear, actualizar, listar y eliminar exámenes complementarios, para poder tener un diagnóstico más claro acerca de la salud del paciente.		
Criterios de aceptación		
1.	Crear Examen Complementario	• El veterinario puede crear uno o más exámenes complementarios para un paciente siempre que sea necesario.
2.	Actualizar Examen Complementario	• El veterinario puede actualizar la información de los exámenes complementarios las veces que sea necesario.
3.	Listar Examen Complementario	• El veterinario puede buscar exámenes complementarios de un paciente.
4.	Eliminar Examen Complementario	• El veterinario puede borrar los exámenes complementarios que desee.

Nota. Se muestra la historia de usuario de exámenes complementarios que se desarrolló en el segundo Sprint.

Tabla 21

Historia de usuario detallada de Carnets

Carnets	
Código: HU006	Estimación: 13 días
Como Veterinario, quiero crear, actualizar, listar y eliminar carnets, para poder tener un registro de las vacunas de los pacientes.	
Criterios de aceptación	

Carnets		
1.	Crear Carnet	El veterinario puede crear un carnet de vacunas preventivas para nuevos pacientes.
2.	Actualizar Carnet	El veterinario puede aumentar nuevas vacunas a un carnet creado anteriormente.
3.	Listar Carnet	La información del carnet se mostrará al buscar mediante campos específicos.
4.	Eliminar Carnet	El veterinario puede eliminar el carnet de un paciente.

Nota. Se muestra la historia de usuario de carnets que se desarrolló en el segundo Sprint.

Tabla 22

Historia de usuario detallada de Vacunas

Vacunas		
Código: HU008		Estimación: 13 días
Como Veterinario, quiero crear, actualizar, listar y eliminar vacunas, para poder tener un registro de las vacunas aplicadas en un paciente.		
Criterios de aceptación		
1.	Crear Vacuna	El veterinario puede crear vacunas en un carnet las veces que sea necesario.
2.	Actualizar Vacuna	El veterinario puede modificar la información de la vacuna siempre que se lo requiera.
3.	Listar Vacuna	La información de las vacunas se mostrará cada que se consulte un carnet.

Vacunas		
4.	Eliminar Vacuna	El veterinario puede eliminar la vacuna de un carnet si es necesario.

Nota. Se muestra la historia de usuario de vacunas que se desarrolló en el segundo Sprint.

Sprint 3. En la Tabla 23 se muestra la historia de usuario de la característica Catálogo de Productos que pertenece al Sprint 3, esto ayuda al desarrollo de los microservicios. Cada historia de usuario describe las funcionalidades del sistema.

Tabla 23

Historias de usuario detallada de Catálogo de Productos

Catálogo de Productos		
Código: HU012		Estimación: 12 días
Como Veterinario, quiero crear, actualizar, listar y eliminar productos, para poder tener un registro con la información de los productos que entran y salen de la veterinaria.		
Criterios de aceptación		
1.	Crear Producto	Se muestra una notificación cuando la información se haya almacenado correctamente.
2.	Actualizar Producto	El veterinario puede modificar información de un producto cuando requiera.

Catálogo de Productos		
3.	Listar Producto	Si el producto buscado existe se debe mostrar toda la información existente del mismo.
4.	Eliminar Producto	Se muestra una notificación cuando el registro del producto se haya eliminado correctamente.

Nota: Se muestra la historia de usuario de catálogo de productos que se desarrolló en el tercer Sprint.

Resultados de los Sprint

Se realizó una lista de chequeo por cada Sprint, el cual permite indicar si los objetivos (historias de usuario), se cumplen o no, en el caso de los Sprint planteados se obtuvo un resultado positivo, cumpliendo todos los objetivos de cada Sprint.

Resultados Sprint 1. En las Tablas 24, 25 y 26 se muestran las listas de chequeo de los resultados del Sprint 1, basándonos en las historias de usuario desarrolladas en dicho Sprint.

Tabla 24

Lista de chequeo Ficha Clínica

Ficha Clínica		
Código: HU003	Estimación: 20 días	
Como Veterinario, quiero crear, actualizar, listar, eliminar una ficha clínica, para llevar los registros de la consulta.		
Prueba	Detalles	Estado
El veterinario autorizado puede crear una nueva ficha clínica para	El sistema valida cada dato ingresado en el formulario y en caso de tener información errónea no la guarda.	✓

Ficha Clínica		
un paciente cuando asista a consulta.	El veterinario puede crear fichas clínicas por cada vez que un paciente asista a consulta.	✓
El veterinario autorizado puede modificar la información dentro de la ficha clínica del paciente.	El veterinario puede modificar una ficha clínica existente cada que así lo requiera.	✓
Al momento de realizar una búsqueda de la ficha clínica se deberá ingresa la información o nombre del paciente que se está buscando la ficha clínica.	El sistema devuelve la información acorde al criterio de búsqueda.	✓
Debe existir una ficha clínica registrada y eliminar todos los registros asociados.	Se elimina toda la información asociada a un registro de ficha clínica existente.	✓

Nota. Se muestra los objetivos a cumplir de la historia de usuario de ficha clínica.

Tabla 25

Lista de chequeo Receta

Receta		
Código: HU004	Estimación: 15 días	
Como Veterinario, quiero crear, actualizar, listar y eliminar recetas para tener un registro de los medicamentos aplicados al paciente.		
Prueba	Detalles	Estado
El veterinario recibe un mensaje de confirmación si la receta se guardó correctamente.	Al momento de guardar un veterinario.	✓
El veterinario puede actualizar la información de una receta las veces que sea necesario y recibirá un mensaje cuando se guarden los datos.	El veterinario puede modificar una receta existente cada que así lo requiera.	✓
El veterinario puede buscar recetas de cualquier paciente.	El veterinario puede listar todas las recetas pertenecientes a un paciente cada que lo requiera.	✓
El veterinario puede borrar tantas recetas como sea necesario.	El veterinario puede borrar cualquier receta cuando así lo requiera.	✓

Nota. Se muestra los objetivos a cumplir de la historia de usuario de receta.

Tabla 26

Lista de chequeo de la historia de usuario de Medicamentos

Medicamento		
Código: HU007	Estimación: 18 días	
Como Veterinario, quiero crear, actualizar, listar y eliminar recetas para tener un registro de los medicamentos aplicados al paciente.		
Prueba	Detalles	Estado
El veterinario puede crear uno o más medicamentos en una receta siempre que sea necesario.	El veterinario puede agregar los medicamentos que desee a una receta cuando lo requiera.	✓
El veterinario puede actualizar la información de los medicamentos las veces que sea necesario.	El veterinario puede modificar uno o más medicamentos en caso de cometer errores al llenar el formulario.	✓
El veterinario puede buscar medicamentos de las recetas de un paciente.	El veterinario puede ver todos los medicamentos disponibles en una receta.	✓
El veterinario puede borrar tantos medicamentos como sea necesario.	El veterinario puede borrar los medicamentos asociados a una receta las veces que sea necesario.	✓

Nota. Se muestra los objetivos a cumplir de la historia de usuario de medicamento.

Resultados Sprint 2. En las Tablas 27 y 28 se muestran las listas de chequeo de los resultados del Sprint 2, basándonos en las historias de usuario desarrolladas en dicho Sprint.

Tabla 27

Lista de chequeo de la historia de usuario de Exámenes Complementarios y Carnets

Exámenes complementarios		
Código: HU005	Estimación: 15 días	
Como Veterinario, quiero crear, actualizar, listar y eliminar exámenes complementarios, para poder tener un diagnóstico más claro acerca de la salud del paciente.		
Prueba	Detalles	Estado

Exámenes complementarios		
El veterinario puede crear uno o más exámenes complementarios para un paciente siempre que sea necesario.	El veterinario puede crear exámenes complementarios de un paciente las veces que lo requiera y recibe un mensaje de confirmación.	✓
El veterinario puede actualizar la información de los exámenes complementarios las veces que sea necesario.	El veterinario puede modificar la información de uno o más campos de un examen complementario y recibe. una notificación cuando se guardan los cambios.	✓
El veterinario puede buscar exámenes complementarios de un paciente.	El veterinario puede ver los exámenes complementarios realizados a un paciente las veces que sea necesario.	✓
El veterinario puede borrar los exámenes complementarios que desee.	El veterinario puede borrar uno o más exámenes complementarios asociados a un paciente y recibe una notificación cuando se elimina la información.	✓

Carnets		
Código: HU006	Estimación: 13 días	
Como Veterinario, quiero crear, actualizar, listar y eliminar carnets, para poder tener un registro de las vacunas de los pacientes.		
Prueba	Detalles	Estado
El veterinario puede crear un carnet de vacunas preventivas para nuevos pacientes.	El veterinario puede crear un carnet de vacunación para cada nuevo paciente que ingresa y recibe una notificación al momento que se crea.	✓
El veterinario puede aumentar nuevas vacunas a un carnet creado anteriormente.	El veterinario puede agregar nuevas vacunas a un carnet existente y recibe una notificación cuando se guarda.	✓
La información del carnet se mostrará al buscar mediante campos específicos.	El veterinario puede buscar el carnet de un paciente cada que lo requiera.	✓
El veterinario puede eliminar el carnet de un paciente.	El veterinario puede borrar el carnet de vacunación existente de un paciente y recibe una notificación al momento que se elimina.	✓

Nota. Se muestra los objetivos a cumplir de la historia de usuario de exámenes complementarios y carnets

Tabla 28

Lista de chequeo de la historia de usuario de Vacunas

Vacunas		
Código: HU008	Estimación: 13 días	
Como Veterinario, quiero crear, actualizar, listar y eliminar vacunas, para poder tener un registro de las vacunas aplicadas en un paciente.		
Prueba	Detalles	Estado
El veterinario puede crear vacunas en un carnet las veces que sea necesario.	El veterinario puede agregar las vacunas que sea necesario en un carnet y recibirá una notificación cada que se cree una vacuna.	✓
El veterinario puede modificar la información de la vacuna siempre que se lo requiera.	El veterinario puede modificar la información de cualquier vacuna asociada a un carnet y recibirá una notificación cuando se guarden los cambios.	✓
La información de las vacunas se mostrará cada que se consulte un carnet.	El veterinario puede ver la información de las vacunas asociadas a un carnet cada vez que sea necesario.	✓
El veterinario puede eliminar la vacuna de un carnet si es necesario.	El veterinario puede eliminar la vacuna de un carnet si es necesario y recibirá una notificación cuando se elimine.	✓

Nota. Se muestra los objetivos a cumplir de la historia de usuario de vacunas.

Resultados Sprint 3. En la Tabla 29 se muestra la lista de chequeo de los resultados del Sprint 3, basándonos en las historias de usuario desarrolladas en dicho Sprint.

Tabla 29

Lista de chequeo de la historia de usuario de catálogo de productos

Catálogo de Productos	
Código: HU012	Estimación: 12 días

Catálogo de Productos		
Como Veterinario, quiero crear, actualizar, listar y eliminar productos, para poder tener un registro con la información de los productos que entran y salen de la veterinaria.		
Prueba	Detalles	Estado
Se muestra una notificación cuando la información se haya almacenado correctamente.	El veterinario puede agregar un producto cuando lo requiera y recibe una notificación al crearse el producto.	✓
El veterinario puede modificar información de un producto cuando requiera.	El veterinario puede modificar la información de un producto las veces que sea necesario y recibirá una notificación cuando se guarden los datos.	✓
Si el producto buscado existe se debe mostrar toda la información existente del mismo.	El veterinario puede observar toda la información de un producto existente si así lo requiere.	✓
Se muestra una notificación cuando el registro del producto se haya eliminado correctamente.	El veterinario puede eliminar el registro de un producto cuando lo requiera y recibirá una notificación.	✓

Nota. Se muestra los objetivos a cumplir de la historia de usuario de catálogo de productos.

Despliegue del sistema

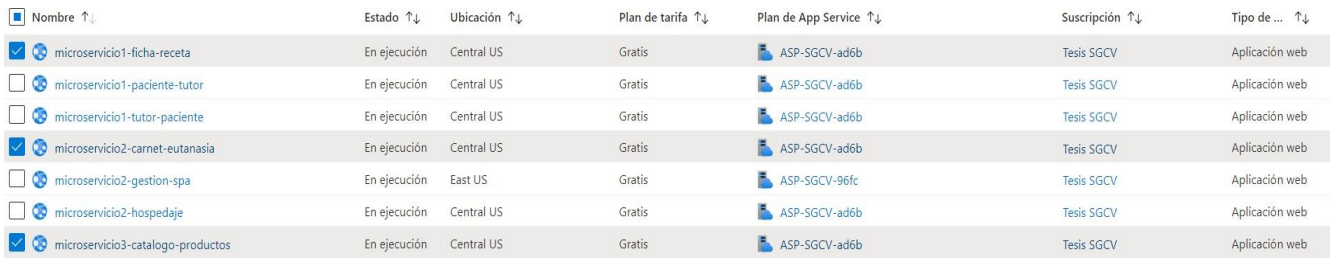
Una vez terminados y validados los microservicios se los pone en producción mediante un AppService de Azure que nos permite alojar aplicaciones web en la nube y nos devuelve una URL mediante la cual se tiene acceso al microservicio.

AppService

Para el despliegue de cada microservicio se creó un AppService en Azure como se muestra en la Figura 23.

Figura 23

AppServices para los microservicios del área de Salud y Catálogo de Productos



Nombre	Estado	Ubicación	Plan de tarifa	Plan de App Service	Suscripción	Tipo de ...
microservicio1-ficha-receta	En ejecución	Central US	Gratis	ASP-SGCV-ad6b	Tesis SGCV	Aplicación web
microservicio1-paciente-tutor	En ejecución	Central US	Gratis	ASP-SGCV-ad6b	Tesis SGCV	Aplicación web
microservicio1-tutor-paciente	En ejecución	Central US	Gratis	ASP-SGCV-ad6b	Tesis SGCV	Aplicación web
microservicio2-carnet-eutanasia	En ejecución	Central US	Gratis	ASP-SGCV-ad6b	Tesis SGCV	Aplicación web
microservicio2-gestion-spa	En ejecución	East US	Gratis	ASP-SGCV-96fc	Tesis SGCV	Aplicación web
microservicio2-hospedaje	En ejecución	Central US	Gratis	ASP-SGCV-ad6b	Tesis SGCV	Aplicación web
microservicio3-catalogo-productos	En ejecución	Central US	Gratis	ASP-SGCV-ad6b	Tesis SGCV	Aplicación web

Nota. Los AppServices desplegados para los microservicios de ficha-receta, carnet-eutanasia y catálogo-productos.

Proceso para el despliegue

Configuración de la aplicación para el despliegue. Para el despliegue en la nube de Azure se tuvo que configurar la aplicación de node en los archivos index.js y package.json como se muestra en la Figura 24 y 25 respectivamente.

Figura 24

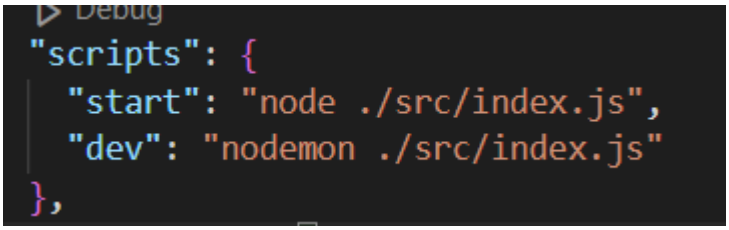
Configuración index.js

```
app.get("/", (req, res)=>{
    res.send ("Microservicio Catalogo de productos");
});

app.listen (process.env.PORT || 8181);
```

Nota. Se muestra la última línea el comando process.env.PORT que toma por defecto el puerto del entorno donde va a correr la aplicación.

Figura 25

Configuración package.json

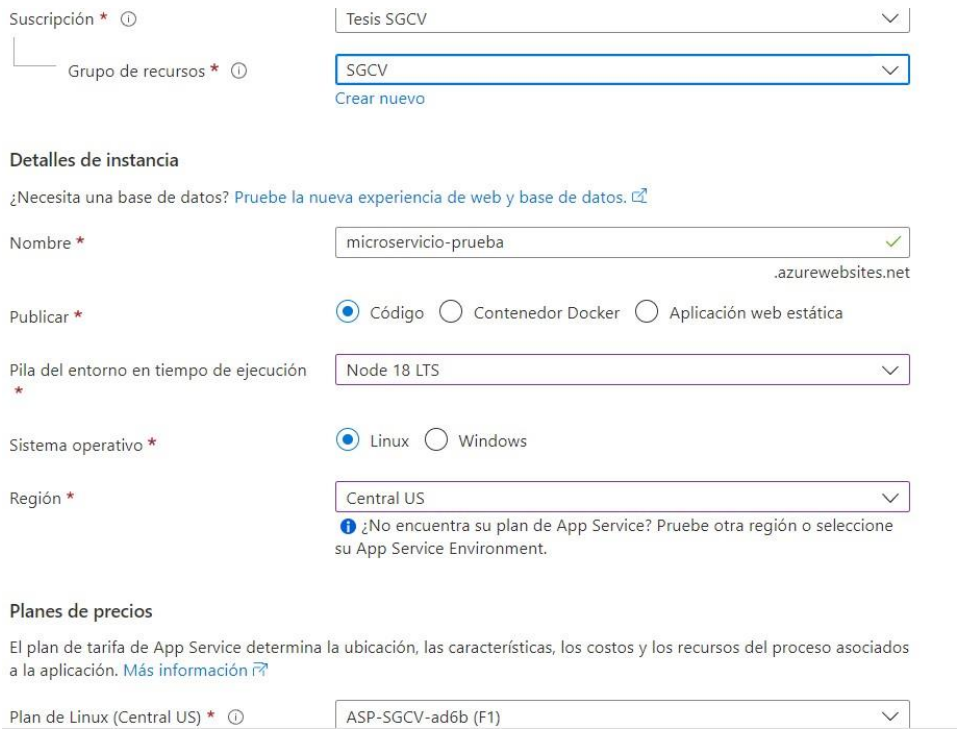


Nota. En el apartado de scripts se agregó la línea de start que es el comando por defecto con el que el entorno de despliegue de Azure va a correr la aplicación.

Configurar el entorno Azure para despliegue. Al momento de crear un AppService se tomó en cuenta las características mostradas en la Figura 26.

Figura 26

Configuración para crear un AppService

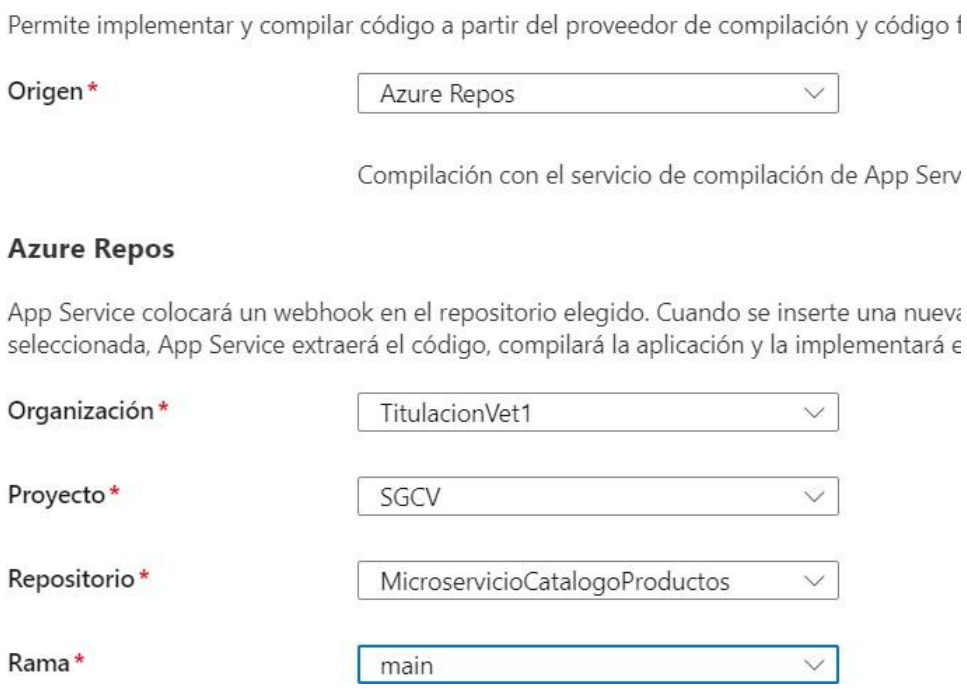


Nota. Configuración para los AppService detallada.

Implementación. Para la implementación de los microservicios se debe realizar un pull request desde la rama develop a la rama main ya que esta es la rama de producción, luego de eso en el centro de implementación del AppService se realizó la configuración mostrada en la Figura 27.

Figura 27

Configuración del AppService para implementación

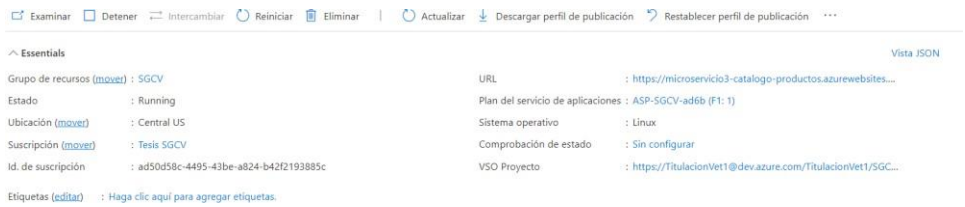


Nota. Se observa detalladamente el repositorio y rama a partir de la cual se realizó la implementación.

Azure devuelve una URL a partir de la cual se puede consumir el microservicio con las diferentes rutas y métodos HTTP como se muestra en la Figura 28.

Figura 28

Microservicio desplegado



Nota. Se observa como Azure devuelve la URL.

Se puede visualizar el despliegue del microservicio ingresando la URL en un navegador como se muestra en la Figura 29.

Figura 29

Microservicio en ejecución

Nota. Se observa como el microservicio ya está ejecutándose correctamente.

Validación del sistema

En este apartado se realizó las pruebas de rendimiento y funcionalidad del back end para los microservicios de ficha-clínica, receta, exámenes-complementarios y carnet-eutanasia que pertenecen al área de Salud y el microservicio para el área de Catálogo de Productos.

Para probar los microservicios de manera correcta deben estar publicados con anterioridad en un AppService en Azure.

En la Tabla 30 se muestran las URLs base de cada microservicio desplegado.

Tabla 30

Microservicios con su URL base

Microservicios	URL base
Microservicio ficha-receta	HTTPs://microservicio1-ficha-receta.azurewebsites.net
Microservicio carnet-eutanasia-exámenes complementarios	HTTPs://microservicio2-carnet-eutanasia.azurewebsites.net
Microservicio catálogo-productos	HTTPs://microservicio3-catalogo-productos.azurewebsites.net/

Nota: Enlaces donde se encuentran alojados los microservicios.

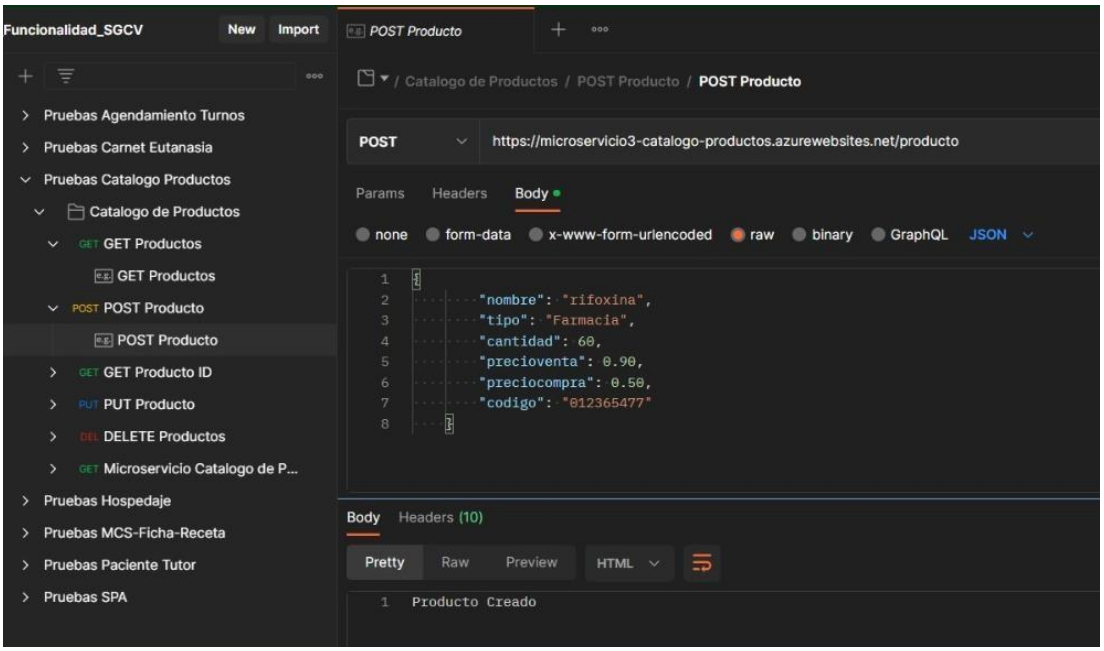
Definición de herramienta de evaluación Postman

Para evaluar las URLs detalladas en la Tabla 30 se creó colecciones en Postman para lo cual se llevó a cabo distintos casos de prueba con los distintos métodos HTTP según las rutas de los microservicios.

Para el caso de prueba del método HTTP POST se envió un objeto en formato JSON con información en los campos respectivos que se muestran en el diagrama entidad relación, teniendo en cuenta el tipo de dato como se muestra en la Figura 30.

Figura 30

Método HTTP POST

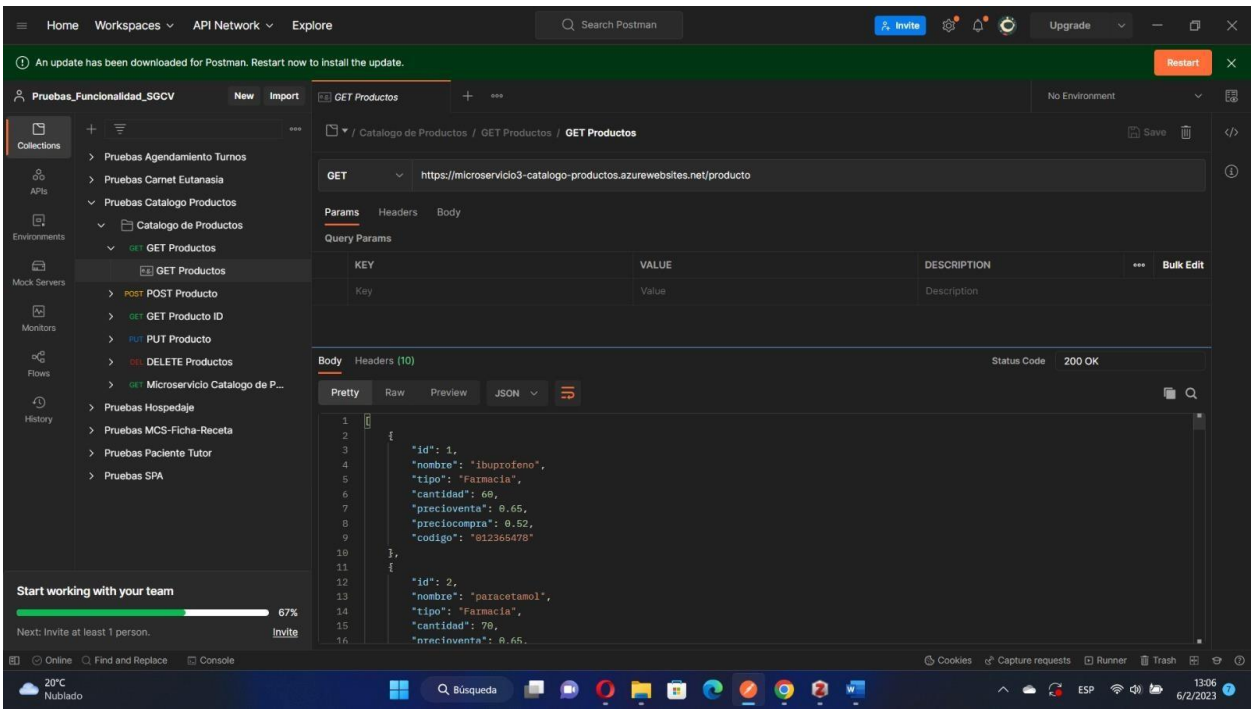


Nota. Petición HTTP con el método POST.

Para el caso de prueba del método HTTP GET se envió una petición a la URL del microservicio correspondiente, teniendo en cuenta que algunos microservicios contienen un criterio de búsqueda diferente, el ejemplo se muestra en la Figura 31.

Figura 31

Método HTTP GET

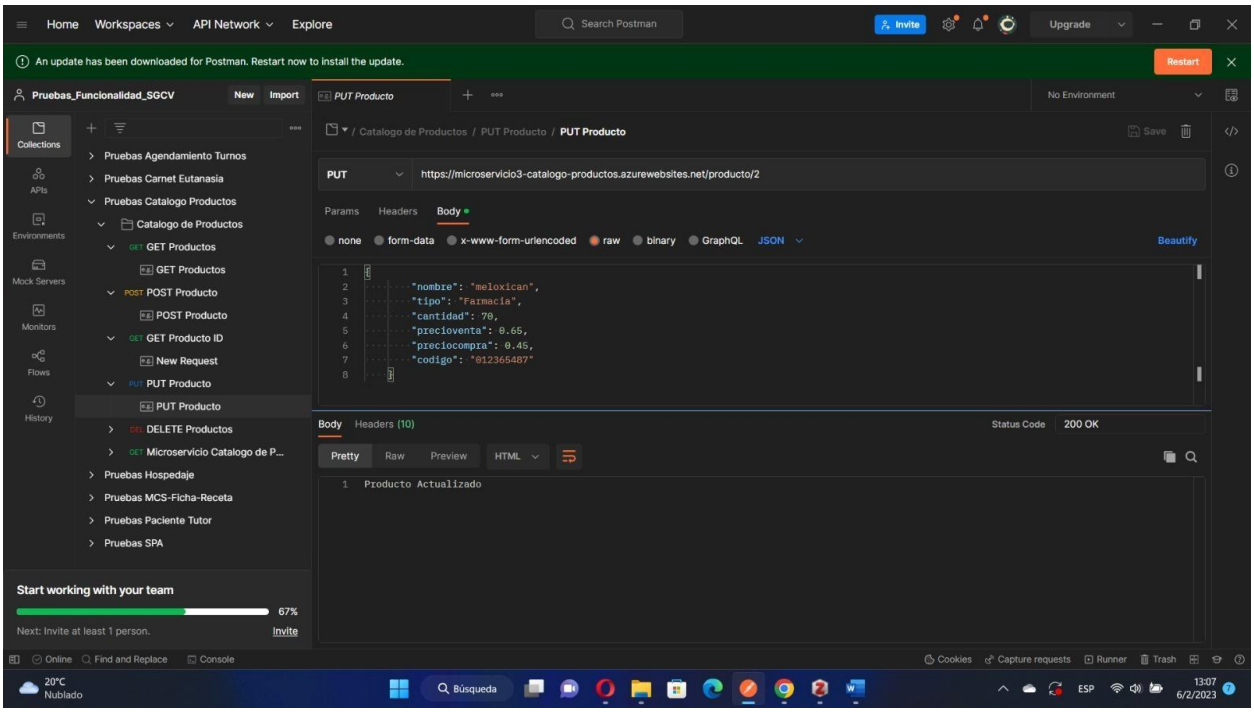


Nota. Petición HTTP con el método GET.

Para el caso de prueba del método HTTP PUT se envió un objeto en formato JSON con información modificada en los campos respectivos, teniendo en cuenta que la petición se envía al id del elemento de la Tabla a editar Figura 32.

Figura 32

Método HTTP PUT

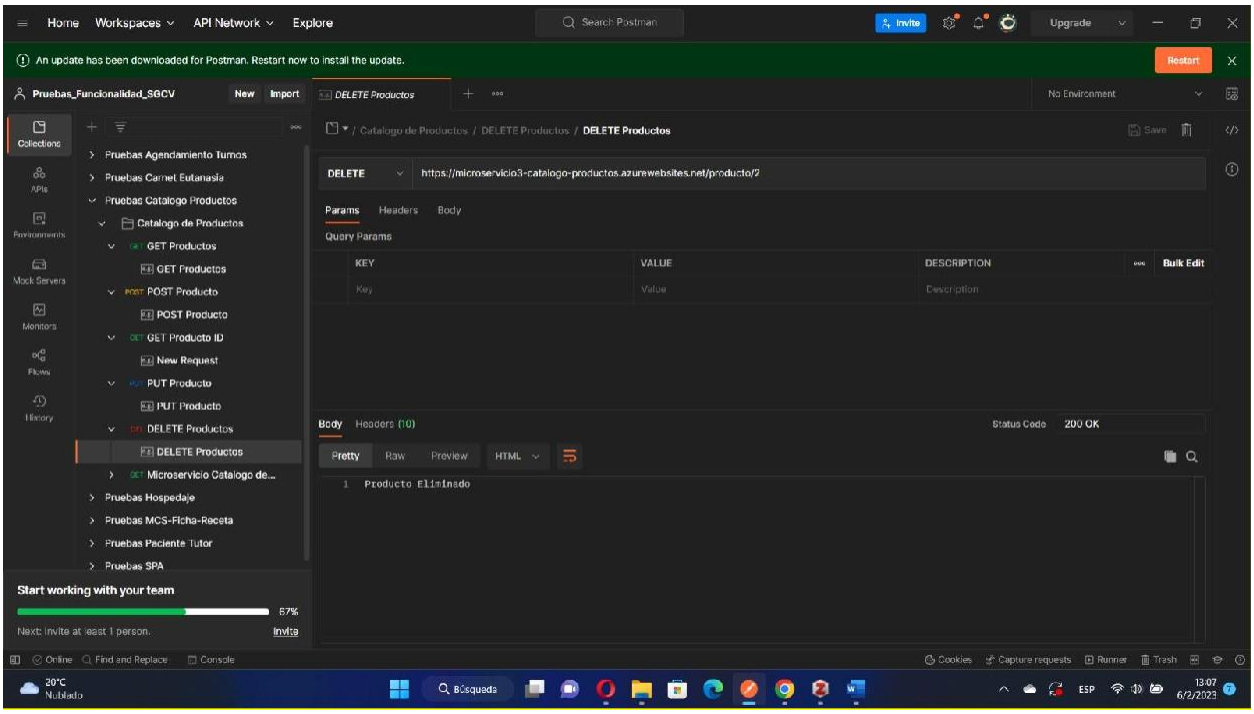


Nota. Petición HTTP con el método PUT.

Para el caso de prueba del método HTTP DELETE se envió una petición a la URL del microservicio correspondiente, teniendo en cuenta el id del elemento de la Tabla a eliminar como se muestra en la Figura 33.

Figura 33

Método HTTP DELETE



Nota. Petición HTTP con el método DELETE.

Pruebas de funcionalidad Sprint 1. Se realizó las pruebas de los microservicios de ficha clínica y receta como se muestra en el apartado 3.7.1, se adjunta los resultados en el Anexo 5.

Pruebas de funcionalidad Sprint 2. Se realizó las pruebas de los microservicios de carnet, exámenes complementarios y eutanasia como se muestra en el apartado 3.7.1, se adjunta los resultados en el Anexo 6.

Pruebas de funcionalidad Sprint 3. Se realizó las pruebas de los microservicios de catálogo de productos como se muestra en el apartado 3.7.1, se adjunta los resultados en el Anexo 7.

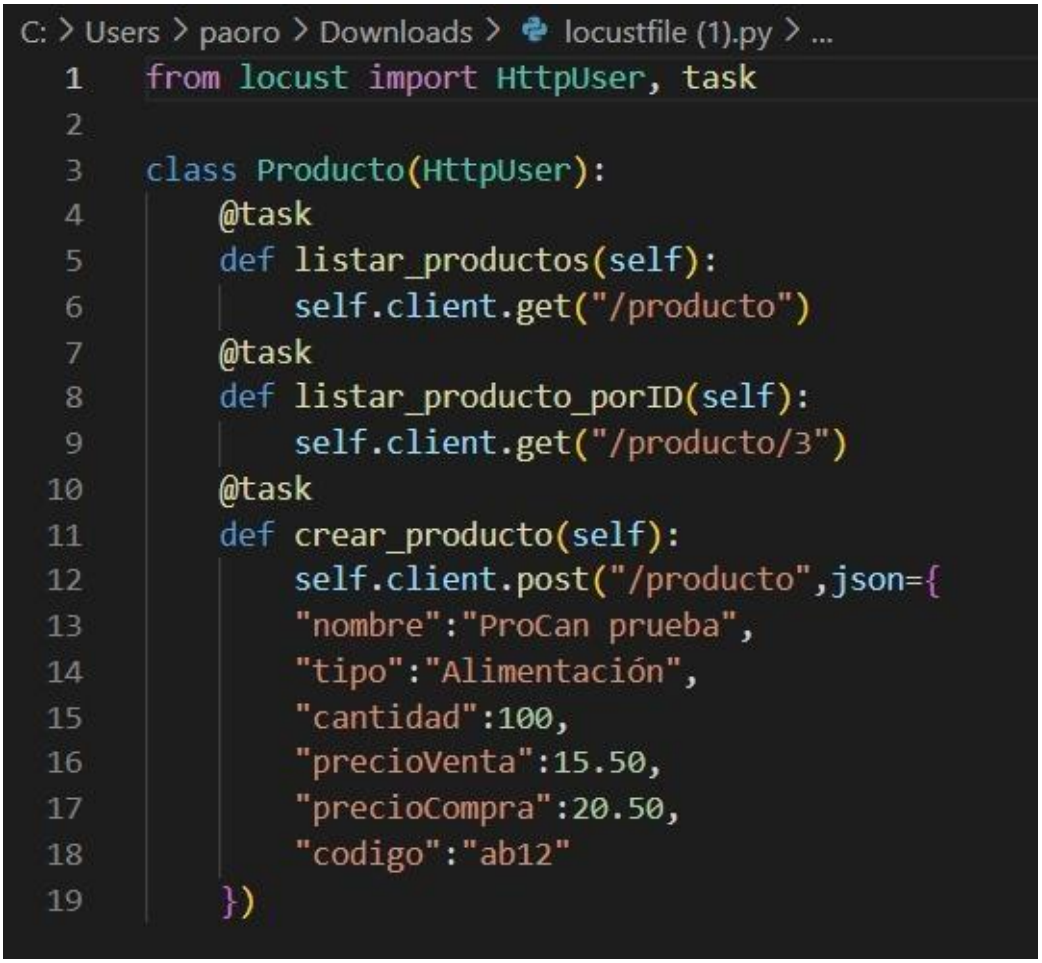
Definición de herramienta de evaluación Locust

Para evaluar las URLs detalladas en la Tabla 30 se creó un archivo de prueba con nombre locustfile.py el cual permite simular tráfico real sobre los microservicios para observar su comportamiento.

Se crea una clase con el nombre del microservicio a probar y se generan task que corresponden a los métodos HTTP GET y POST como se muestra en la Figura 34.

Figura 34

Archivo de pruebas locustfile.py



Nota. Se muestra la estructura del archivo locustfile.py

Pruebas de rendimiento Sprint 1. Se realizó las pruebas de los microservicios de ficha clínica y receta, se adjunta el reporte con los resultados en el Anexo 8.

Pruebas de rendimiento Sprint 2. Se realizó las pruebas de los microservicios de carnet, exámenes complementarios y eutanasia, se adjunta el reporte con los resultados en el Anexo 8.

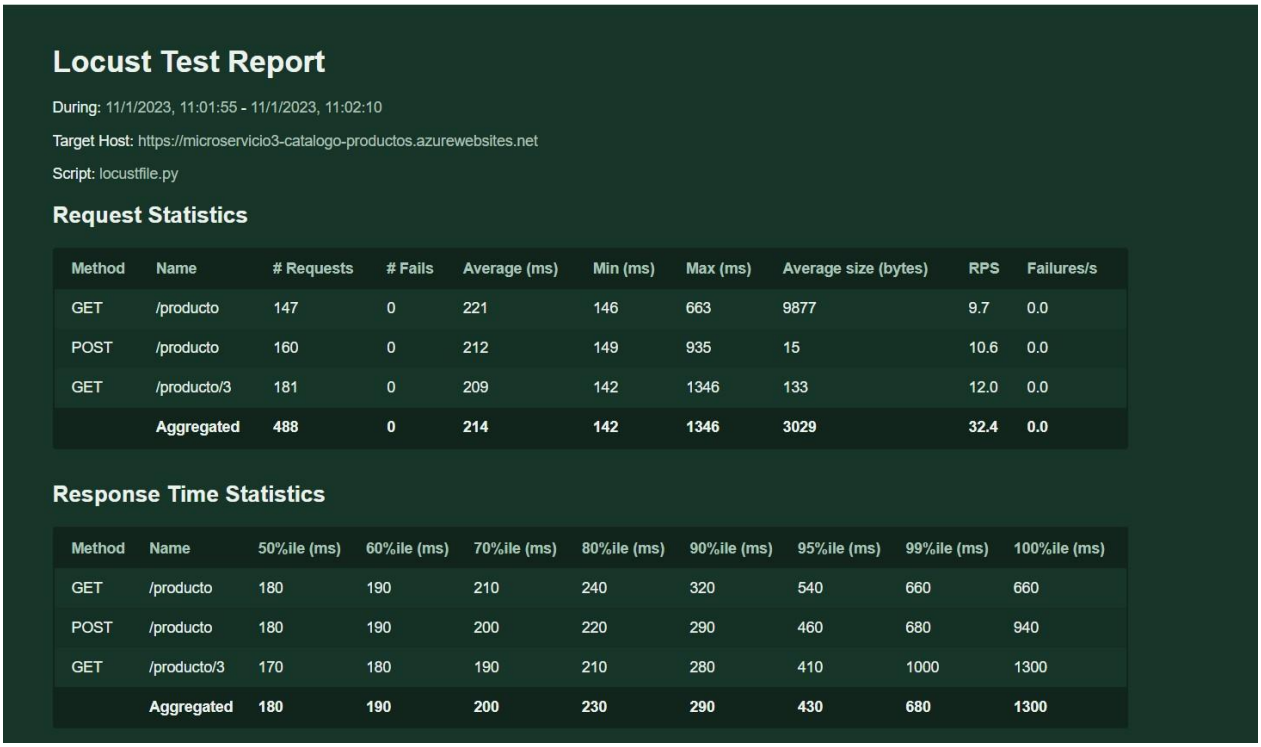
Pruebas de rendimiento Sprint 3. Se realizó las pruebas de los microservicios de catálogo de productos, se adjunta el reporte con los resultados en el Anexo 8.

Reportes. Locust devuelve reportes con los resultados de las pruebas de rendimiento como se muestra en la Figura 35.

Los datos que muestra Locust en la Figura 35 son los Request Statics (Estadísticas de las peticiones) y se pueden interpretar como está en la Tabla 31 y Response Time Statics (Estadísticas de los tiempos de respuesta) nos devuelve los tiempos de respuesta aproximados basados en percentiles.

Figura 35

Reporte de pruebas de Locust



Locust Test Report

During: 11/1/2023, 11:01:55 - 11/1/2023, 11:02:10

Target Host: https://microservicio3-catalogo-productos.azurewebsites.net

Script: locustfile.py

Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/producto	147	0	221	146	663	9877	9.7	0.0
POST	/producto	160	0	212	149	935	15	10.6	0.0
GET	/producto/3	181	0	209	142	1346	133	12.0	0.0
	Aggregated	**488**	**0**	**214**	**142**	**1346**	**3029**	**32.4**	**0.0**

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/producto	180	190	210	240	320	540	660	660
POST	/producto	180	190	200	220	290	460	680	940
GET	/producto/3	170	180	190	210	280	410	1000	1300
	Aggregated	**180**	**190**	**200**	**230**	**290**	**430**	**680**	**1300**

Nota. Reporte de pruebas para el microservicio de catálogo de productos

Tabla 31

Interpretación de los resultados de las pruebas en Locust

Campo	**Interpretación**
Method	Método HTTP al que se hizo la prueba.
Name	Nombre de la ruta de prueba.
# Requests	Número de peticiones realizadas.
# Fails	Número de fallos.
Average (ms)	Promedio del tiempo en el que se demora en realizar la petición.
Min (ms)	Tiempo mínimo en el que se demora realizar la petición.
Max (ms)	Tiempo máximo en el que se demora realizar la petición.
Average size (bytes)	Tamaño promedio que ocupa en memoria realizar las peticiones.

Campo	Interpretación
RPS	Número de peticiones realizadas por segundo.
Failure/s	Fallos al momento de realizar peticiones.

Nota. Se muestra la Tabla de interpretación de los reportes en Locust.

Como se puede observar en la Figura 36 son los gráficos de las pruebas realizadas que permiten una mejor comprensión de los datos y se puede observar variaciones en tiempos de respuesta, número de usuarios y número de peticiones por segundo.

Figura 36

Gráficos de pruebas Locust

Nota. Se puede observar las gráficas de los reportes de pruebas.

Capítulo IV:

Conclusiones y recomendaciones

Conclusiones

- Se ha culminado el desarrollo del Sistema de Gestión para Clínicas Veterinarias usando el paradigma de LPS.
- Al implementar la LPS nos permite desarrollar funcionalidades que se puedan implementar en clínicas veterinarias de cualquier tamaño gracias a que se realizó un análisis de dominio en varias clínicas veterinarias con diferentes características y se desarrollaron los microservicios para cumplir toda necesidad de dichas clínicas.
- El uso de una arquitectura orientada a microservicios junto con LPS permite que el sistema desarrollado escale al mismo tiempo que el crecimiento de una clínica veterinaria.
- Los microservicios implementados se pueden usar en varias veterinarias ya se aplica el paradigma de LPS.
- Los microservicios implementados fueron validados para evitar inconsistencias en el ingreso de información para evitar caídas en el sistema y poder tener disponibilidad total.

Recomendaciones

- Debido al desarrollo co-localizado es recomendable realizar reuniones diarias y a una hora fija para poder hacer un feedback acerca de lo realizado en el día, para cumplir con los tiempos estimados del sprint.
- Al realizar la encuesta para la toma de requisitos se recomienda desarrollar preguntas cerradas que especifiquen cada proceso de gestión de la veterinaria.
- Se recomienda realizar varias versiones del diagrama FODA hasta entender de manera correcta el análisis de dominio.
- Se debe tener en cuenta el flujo de trabajo de git para poder realizar un despliegue correcto y sin interrupciones en la funcionalidad.

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.

Anexos