DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SOFTWARE

TRABAJO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO DE SOFTWARE

TEMA:

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)

AUTORES:

CAMACHO MONCAYO, MARTIN ENRIQUE

NÚÑEZ AMORES, JOSÉ ANTONIO

DIRECTOR:

Dr. JACOME GUERRERO, PATRICIO SANTIAGO

LATACUNGA FEBRERO, 2023

Contenido

- **Planteamiento del problema y objetivos**
- Fundamentación teórica
- Metodología
- Análisis y diseño del sistema
- Desarrollo y validación del sistema
- Conclusiones

Formulación del problema

- ➔ Recolección de información.
- ➔ Falta de organización de los expedientes físicos.
- ➔ Registros no legibles por la calidad de la caligrafía.
- ➔ Pérdida de carnets
- ➔ Pérdida de clientes por procesos lentos.

Objetivos

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).

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.

Contenido

- Planteamiento del problema y objetivos
- **Fundamentación teórica**
- Metodología
- Análisis y diseño del sistema
- Desarrollo y validación del sistema
- Conclusiones

Línea de Producto Software (LPS)

Definición:
Es un paradigma que propone la reutilización planificada, para aprovechar elementos comunes y variables en un mismo dominio.

Características :

- Permite crear diversas variantes.
- Cada variante puede ser diseñada y adaptada para cumplir con necesidades específicas.
- El proceso de construcción consta de tres fases: ingeniería de dominio, ingeniería de aplicación, gestión de la configuración como se muestra en la Figura 1.

Figura 1: Actividades esenciales para el desarrollo de LPS

Análisis de Dominio Orientado a Características (FODA)

Definición:
Es una técnica usada en LPS para identificar y analizar las características más importantes o dominantes.

Características:

- Se enfoca en la perspectiva de la funcionalidad a través del diagrama de características.
- Permite la reutilización a nivel funcional y arquitectónico.
- Las características pueden ser de tipo opcional u obligatorio esta selección se realiza en base a los objetivos del cliente como se muestra en la Figura 2.

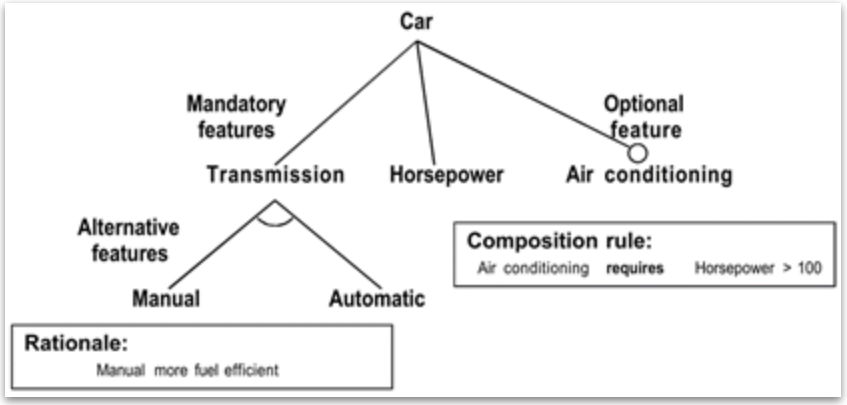


Figura 2: Ejemplo de diagrama de características mostrando la composición de un auto

Fundamentación Teórica

Microservicios

Enfoque de diseño de software en el que un sistema se divide en múltiples servicios independientes

Desarrollo Global de Software (DGS)

Es el proceso de desarrollar aplicaciones de software utilizando un equipo distribuido que se encuentra en diferentes regiones geográficas o países.

DevOps

Es un enfoque que combina el desarrollo de software (**Dev**) y las operaciones de tecnología de la información (**Ops**) para permitir un desarrollo y una entrega de software más rápidos y eficientes.

Scrum

Es un marco ágil para administrar y completar proyectos complejos, especialmente en el desarrollo de software.

Marco Ágil Escalado - SAFe

Definición:

Scaled Agile Framework (SAFe) es un marco para implementar metodologías ágiles en empresas de gran escala. SAFe proporciona un conjunto de prácticas y herramientas para mejorar la colaboración entre equipos, aumentar la eficiencia y la calidad, y acelerar el tiempo de comercialización.

Características:

- Escalabilidad: SAFe es un marco diseñado para ser escalable a cualquier tamaño de organización, desde pequeñas empresas hasta grandes empresas con miles de empleados.
- Metodologías ágiles: SAFe está basado en metodologías ágiles, lo que significa que las empresas pueden adaptarse y responder rápidamente a los cambios en el mercado.

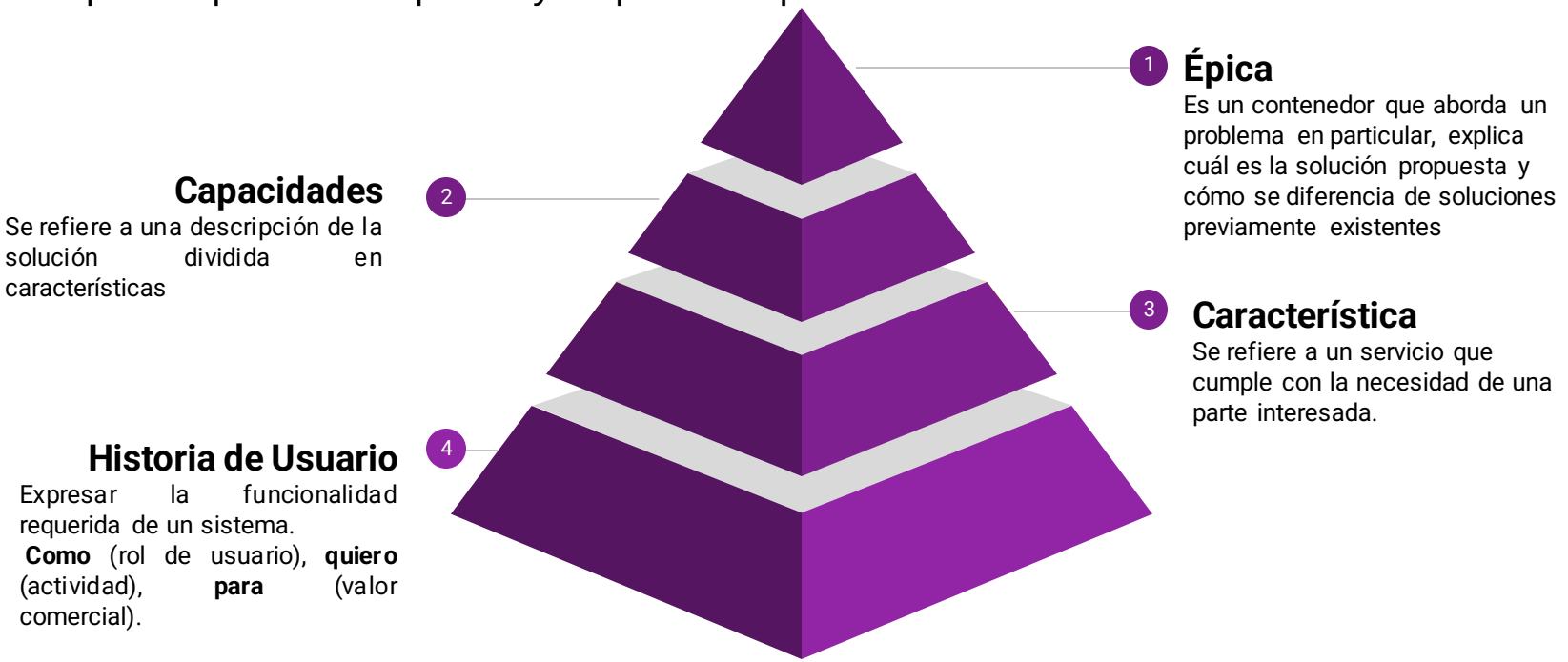


Figura 6: Niveles jerárquicos de SAFe

Contenido

- Planteamiento del problema y objetivos
- Fundamentación teórica
- **Metodología**
- Análisis y diseño del sistema
- Desarrollo y validación del sistema
- Conclusiones

Metodología - Asignación de equipos y roles

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

En la tabla 1, se observa las distintas funciones asignadas a cada miembro del equipo y las respectivas responsabilidades.

En la tabla 2, se observa los distintos roles asignados al equipo, y también se define a la persona encargada de evaluar el sistema.

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

Metodología - Azure DevOps y Git Flow

2	Servicios de "Petgrooming" para clínicas veterinarias	jfpoveda1
1	Servicios de "Salud" para veterinarias	Julio Castro
3	Servicios de venta de productos para clínicas veterinarias	Julio Castro

Branch			Co...	Author	Authored...
feature					
16-h17			942c6e8	Martin	8 dic 2022
6-h7			ab4192:	Martin	29 dic 2...
21-h22-revert-from-develop			e26c3e6	mecamacho10	9 ene
main	Default	Compare	def344:	jose1n8a98	26 ene

Contenido

- Planteamiento del problema y objetivos
- Fundamentación teórica
- Metodología
- **Análisis y diseño del sistema**
- Desarrollo y validación del sistema
- Conclusiones

Análisis del sistema - Levantamiento de requisitos

1

Se aplicaron entrevista a las personas encargadas de las clínicas veterinarias Royal Hound, Vital Pet, Canopolis y Pet's- Cli- Lab de la ciudad de Ambato. con el objetivo de determinar los servicios comunes y opcionales que tenía cada veterinaria. Al concluir las encuestas se obtuvo lo siguiente:

Servicios generales (Royal-Hound)
- Consultas
- Vacunas
- Desparacitaciones
- Cirugía, traumatología, y neurocirugía
- Hospitalización
- Ambulancias
- PetGrooming
- Venta de accesorios y alimento

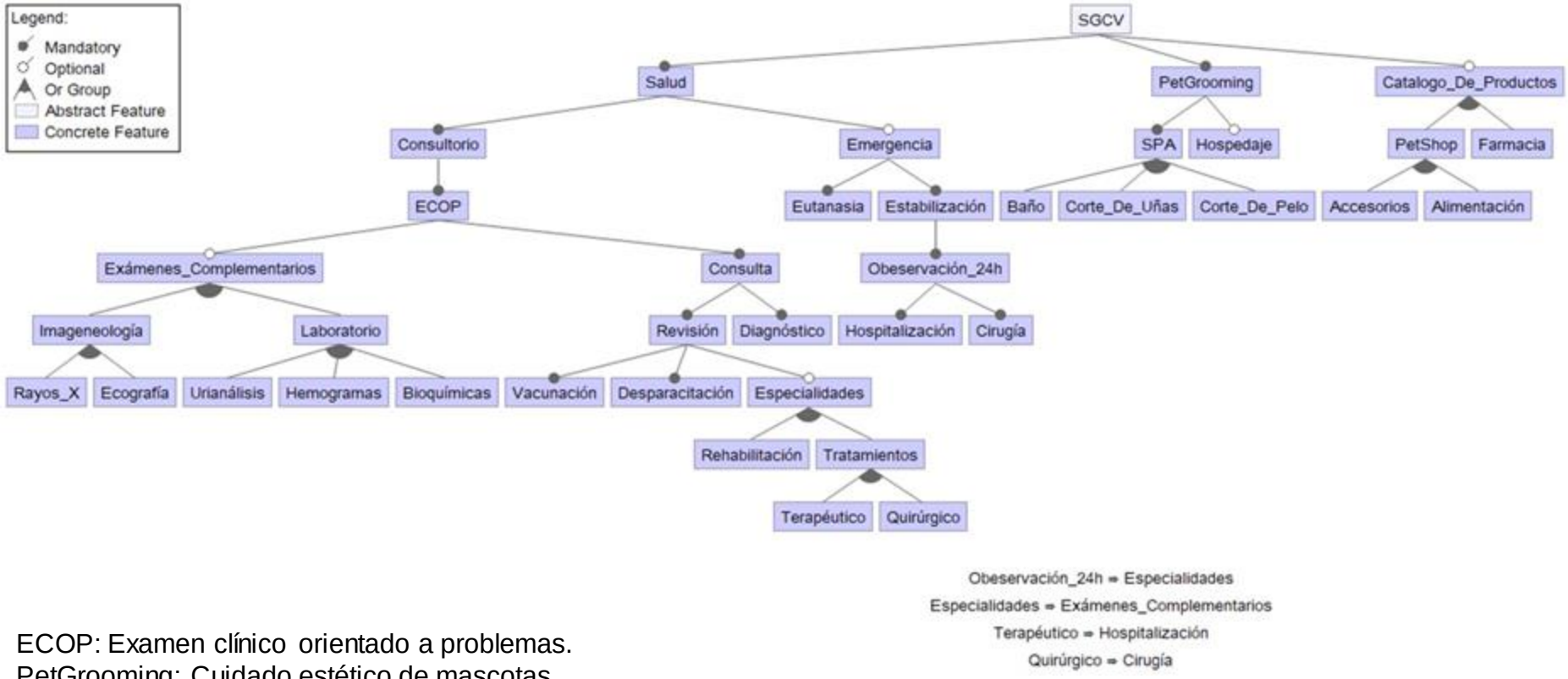
Servicios generales (Vital-Pet)
- Consultas
- Hospitalización
- Laboratorio Clínico
- Radiografías
- Cirugías
- Venta de accesorios
- Farmacia
- PetGrooming
- Vacunación

Servicios generales (Cannopolis)
- Consulta
- Cirugía general
- Petshop
- Farmacia
- Laboratorio
- Ecografías
- Planes de vacunación
- Hospitalización

Servicios generales (PETS-CLI-LAB)
- Tienda
- Consultorio
-Hospitalización
-PetGrooming
-Eutanasia
-Administracion de seguimiento de medicamentos
-Hospedaje

Análisis del sistema - Modelado del dominio

2 Se realiza el modelado del sistema a través del diagrama de características FODA. Para el mismo se pasó por tres versiones siendo la siguiente la versión final, con sus características opcionales y obligatorias.



ECOP: Examen clínico orientado a problemas.
PetGrooming: Cuidado estético de mascotas
PetShop: Tienda de mascotas

Análisis del sistema - Implementación de SAFe

3 Una vez realizado el FODA, se aplica SAFe y se obtienen las épicas, capacidades, características e historias de usuario que tendrá el sistema.

Épicas	Capacidades	Características	Historias de usuario
Salud	ECOP	Ficha clínica	Gestión Ficha clínica
		Carnet de vacunación	Gestión Carnet de vacunación
		Receta medica	Gestión Receta medica
		Exámenes complementarios	Gestión Exámenes complementarios
Gestión de Productos		Producto	Gestión producto

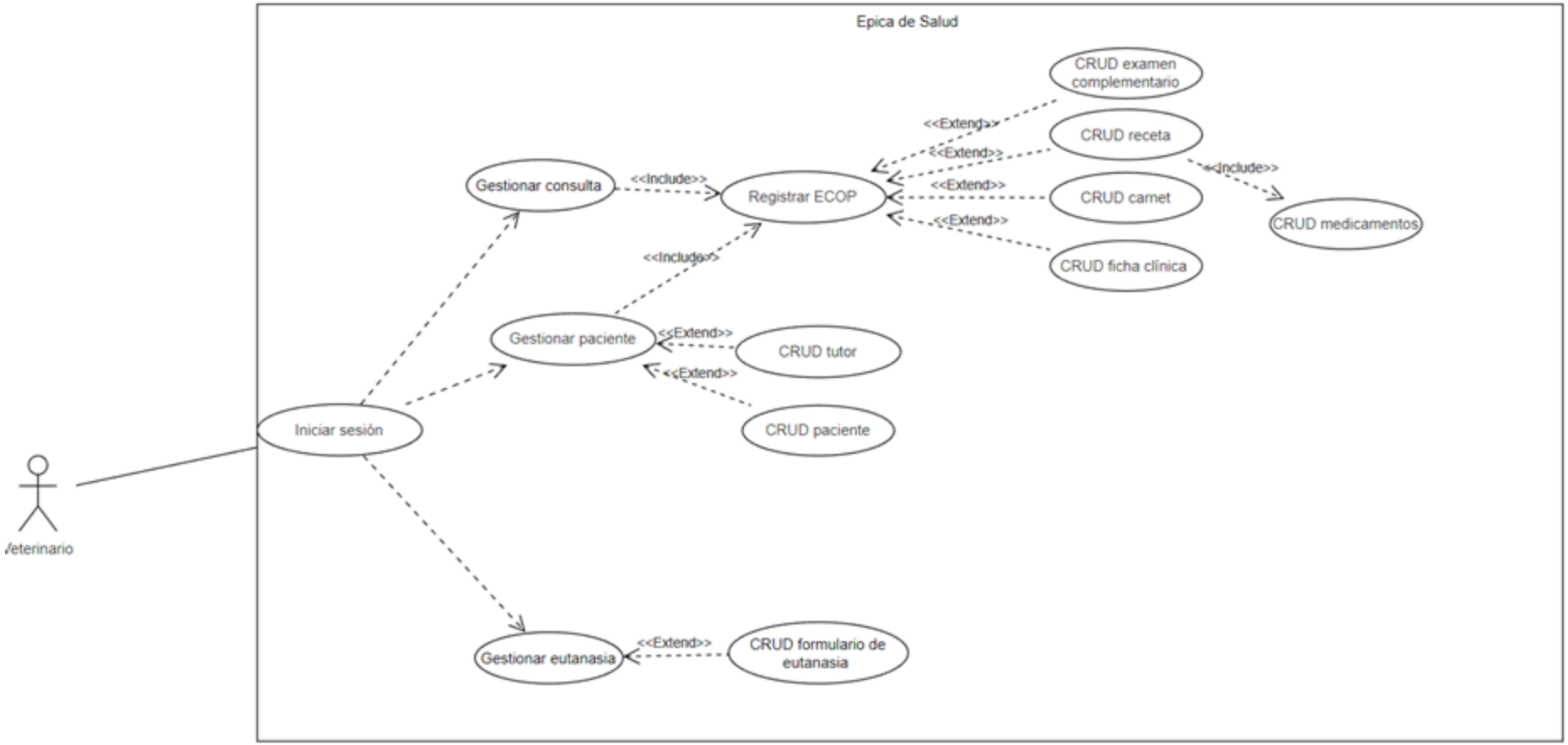
Análisis del sistema – Historias de Usuarios

Código	Descripción	Responsables
HU003	Quiero crear, actualizar, listar, eliminar una ficha clínica, para llevar los registros de la consulta	Martín Camacho y José Núñez
HU004	Quiero crear, actualizar, listar y eliminar recetas, para tener un registro de los medicamentos aplicados al paciente.	Martín Camacho y José Núñez
HU005	Quiero crear, actualizar, listar y eliminar exámenes complementarios, para poder tener un diagnóstico más claro acerca de la salud del paciente.	Martín Camacho y José Núñez
HU006	Quiero crear, actualizar, listar y eliminar carnets, para poder tener un registro de las vacunas de los pacientes.	Martín Camacho y José Núñez
HU007	Quiero crear, actualizar, listar y eliminar medicamentos, para poder tener un registro de los medicamentos asignados en una receta para un paciente.	Martín Camacho y José Núñez
HU008	Quiero crear, actualizar, listar y eliminar vacunas, para poder tener un registro de las vacunas aplicadas en un paciente	Martín Camacho y José Núñez
HU014	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.	Martín Camacho y José Núñez

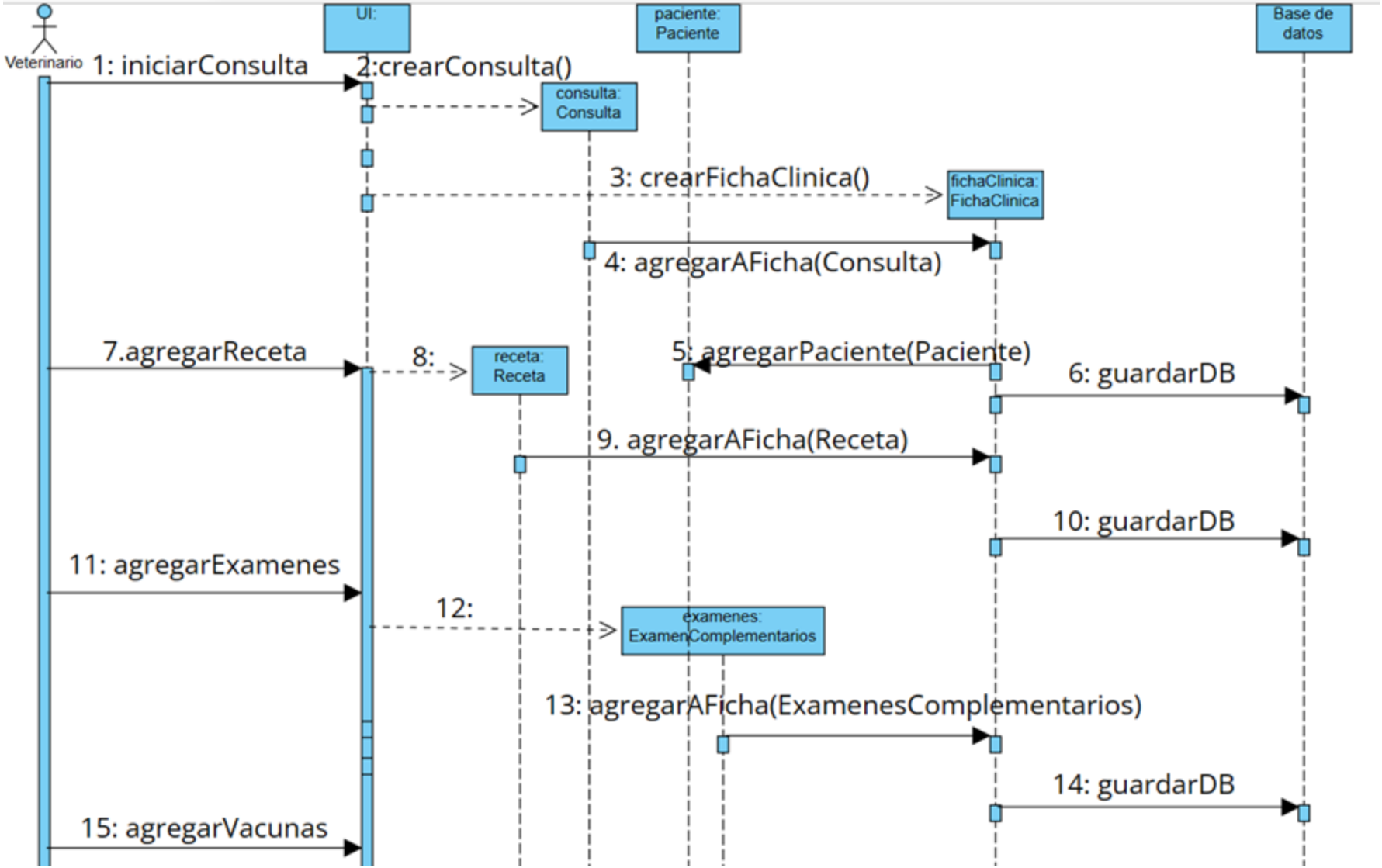
Análisis del sistema – Product Backlog

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.
HU004	15	27/10/2022	10/11/2022	1	Creación del proyecto e instalación de dependencias.
HU005	15	11/11/2022	25/11/2022	2	Desarrollo de rutas, controladores.
HU006	13	26/11/2022	8/12/2022	2	Creación de validaciones.
HU007	18	9/12/2022	26/12/2022	1	Pruebas en Postman.
HU008	13	27/12/2022	8/1/2023	2	Despliegue de microservicios.
HU014	12	9/12/2022	20/1/2023	3	Pruebas de rendimiento.

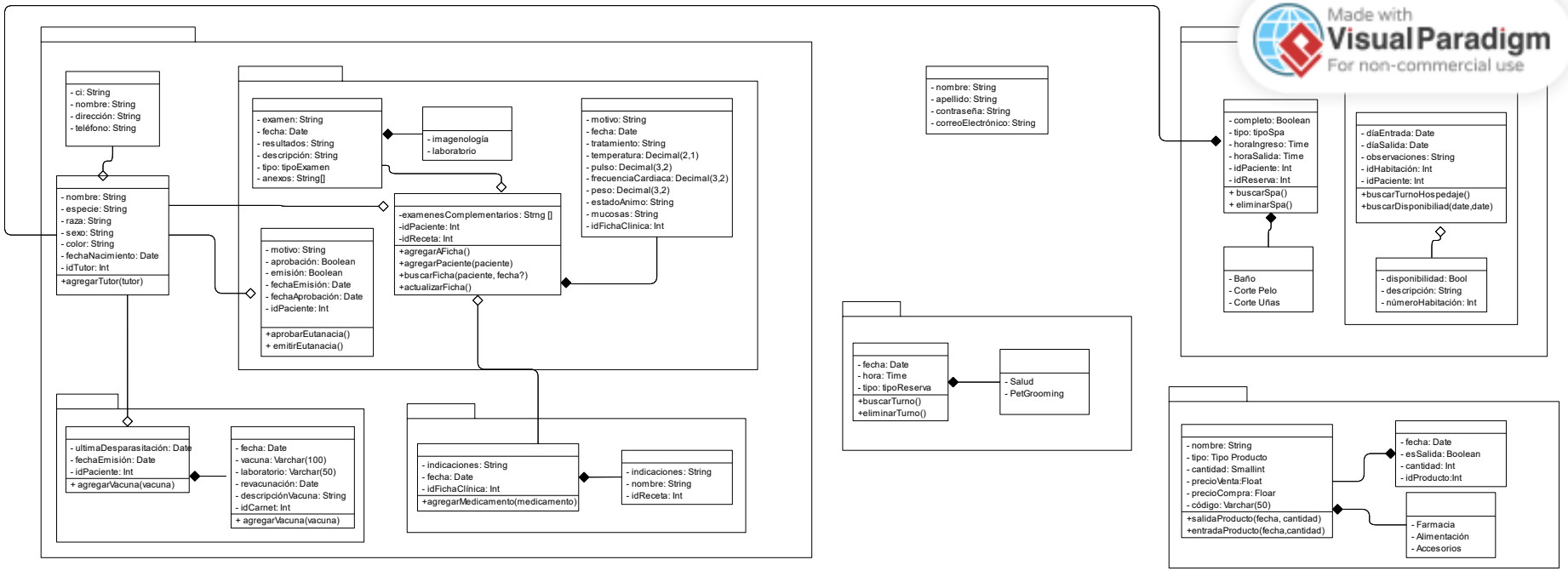
Diseño del sistema - Diagrama de casos de uso



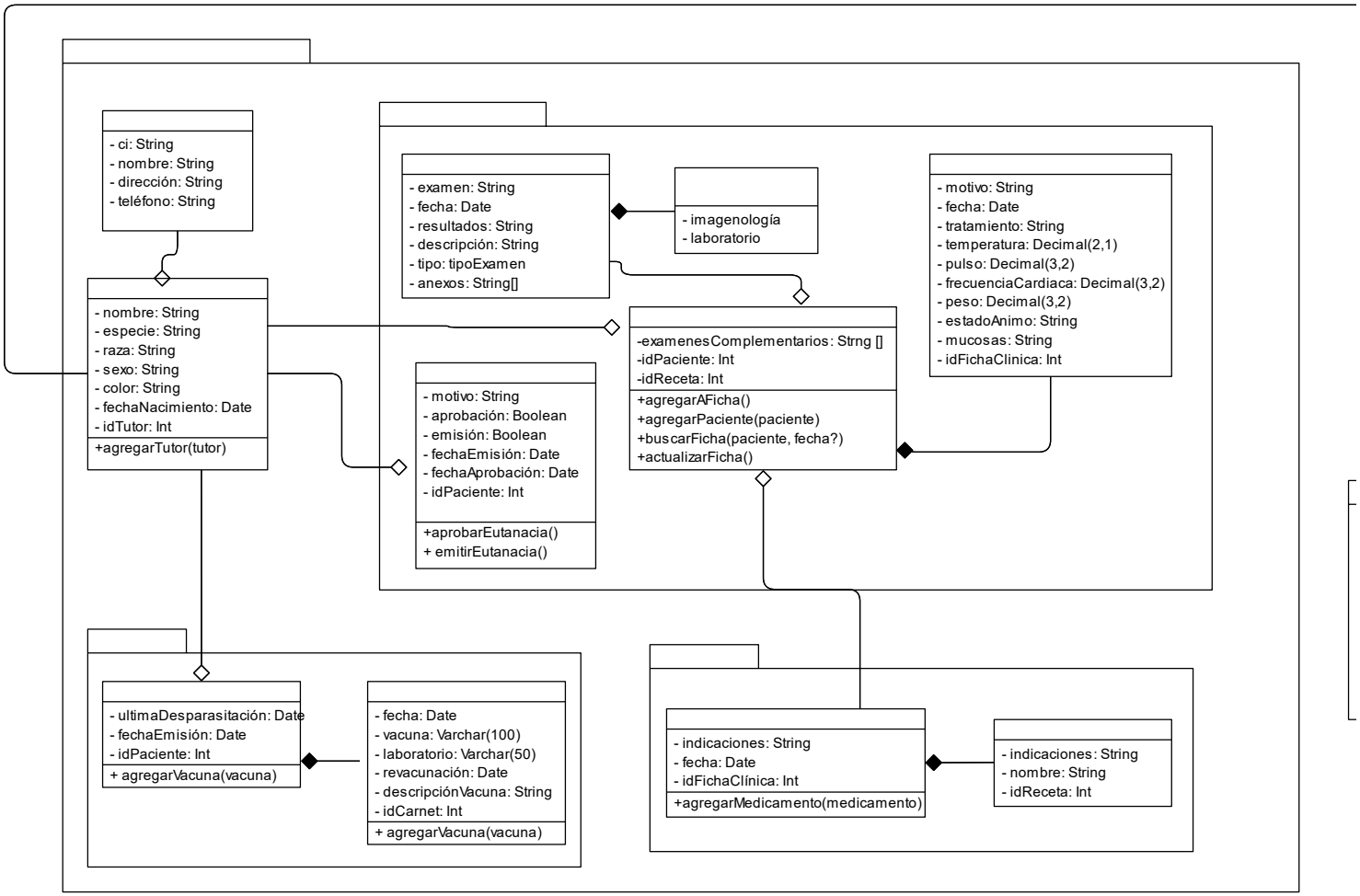
Diseño del sistema - Diagrama de secuencia salud



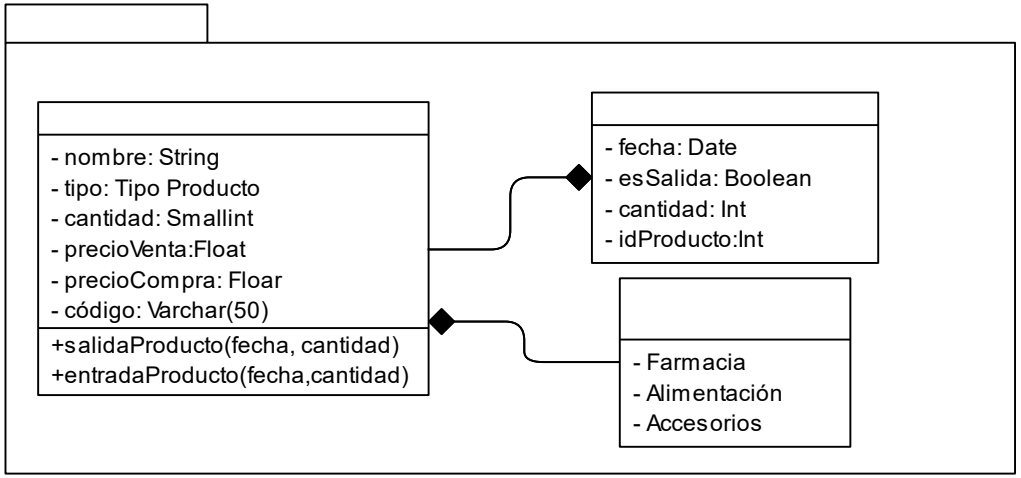
Diseño del sistema - Diagrama de clases



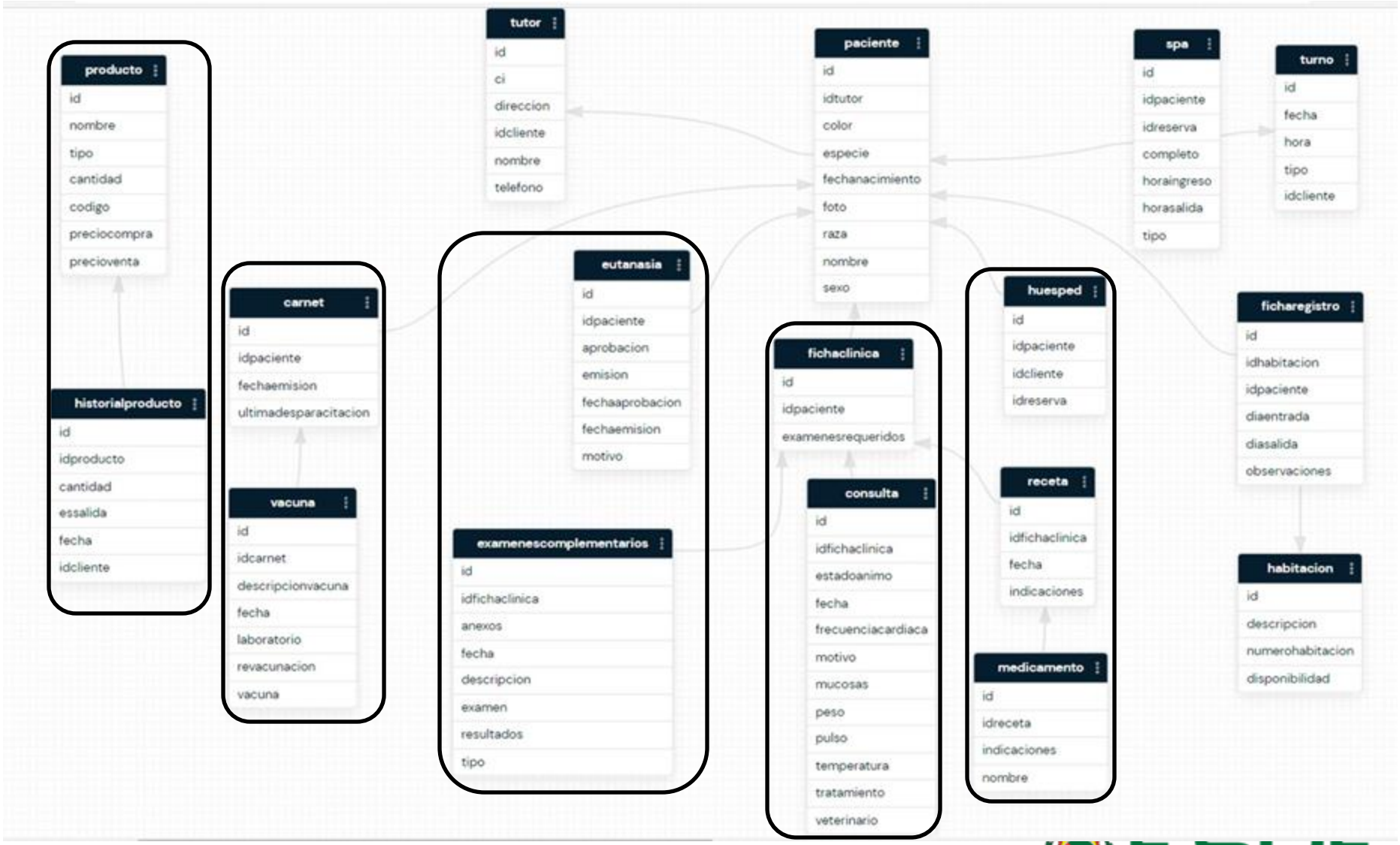
Diseño del sistema - Salud



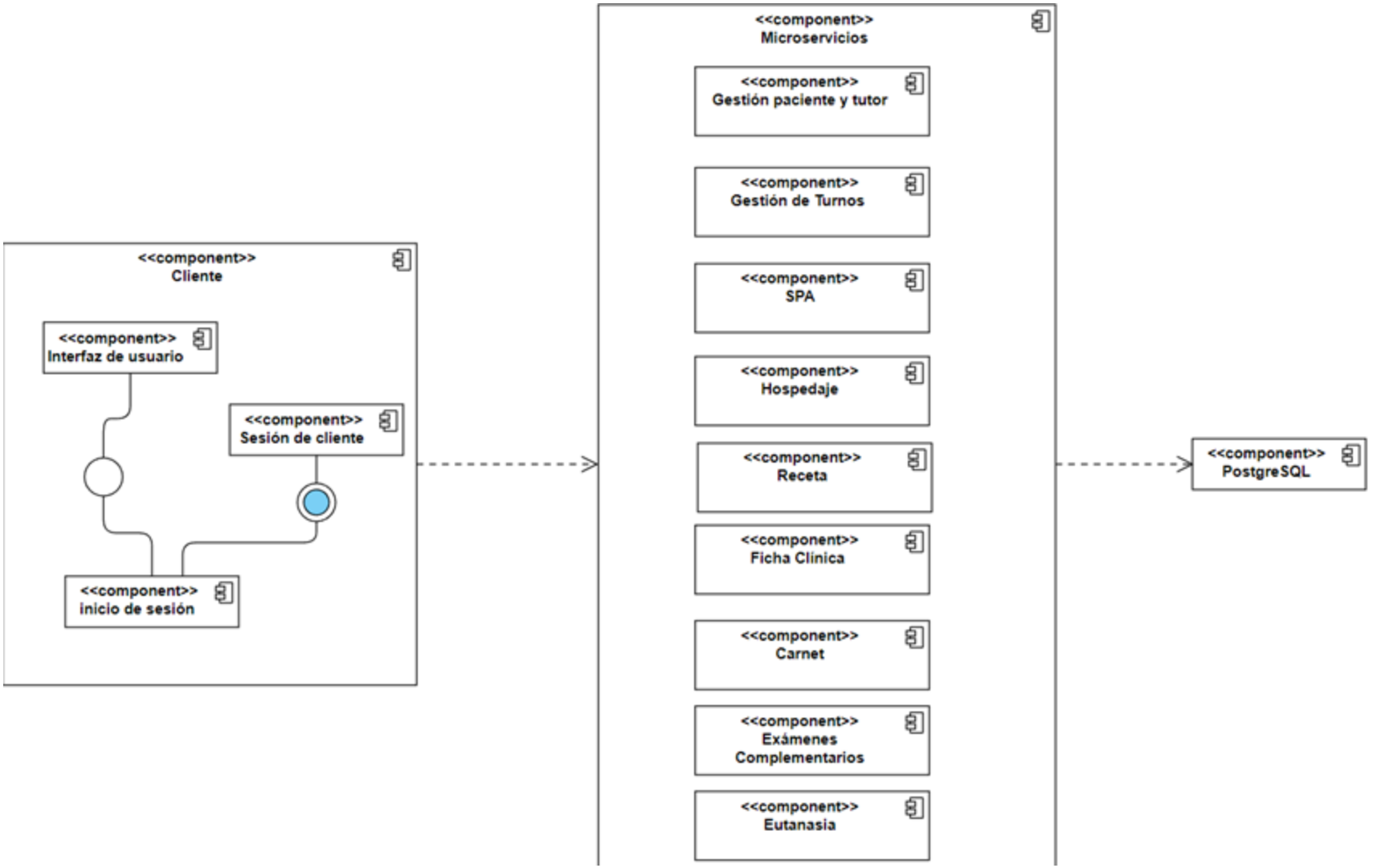
Diseño del sistema – Catálogo de productos



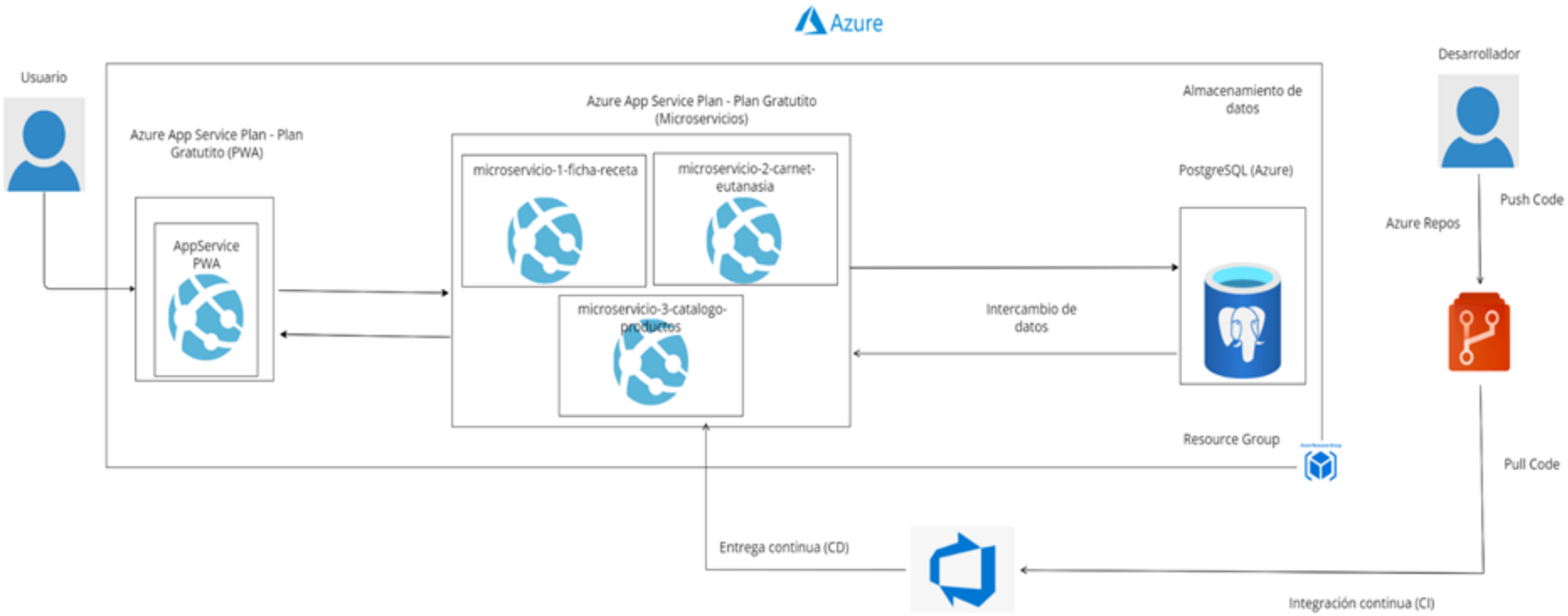
Diseño del sistema - Modelo entidad relación



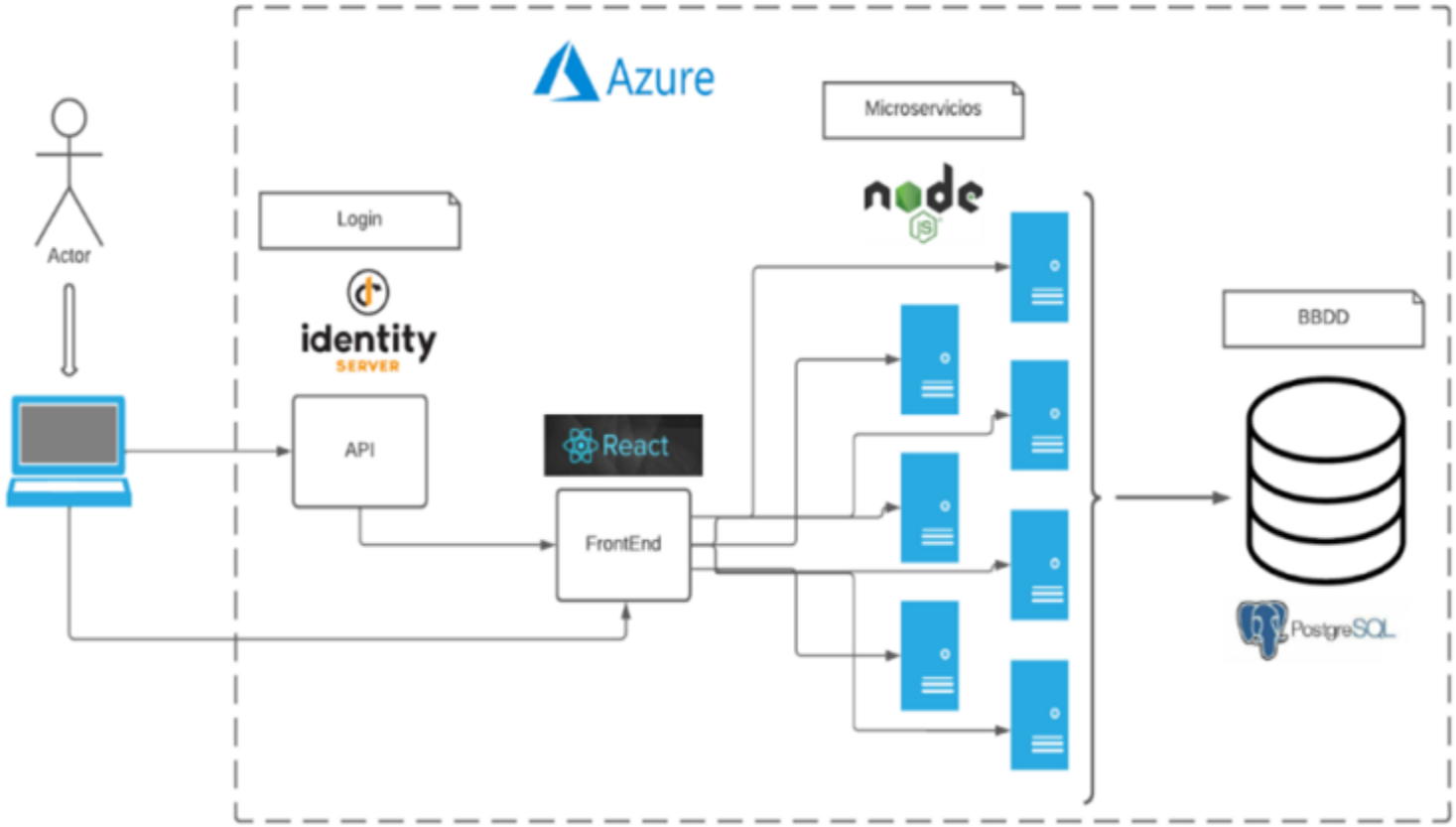
Diseño del sistema - Diagrama de componentes



Diseño del sistema - Diagrama de despliegue



Arquitectura



Contenido

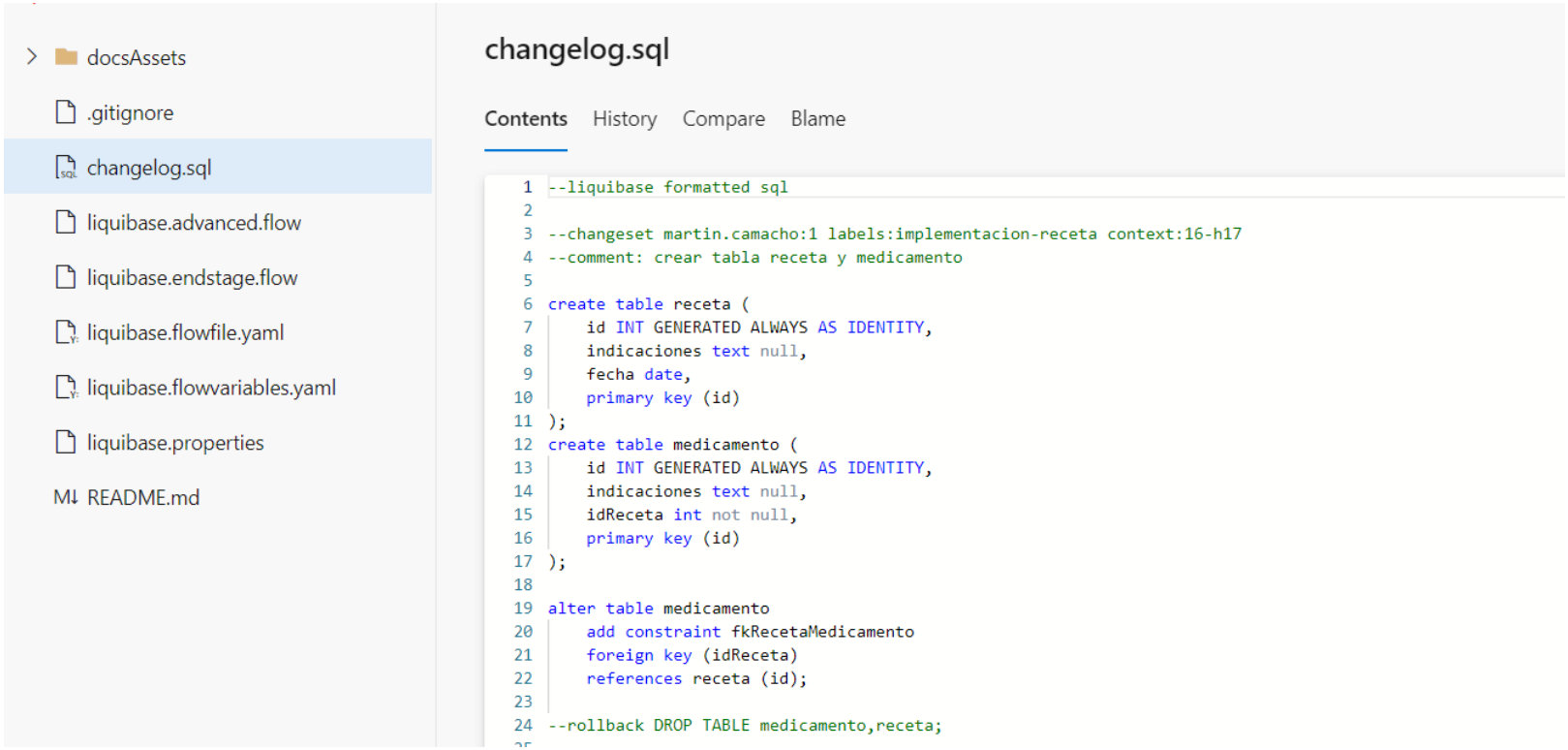
- Planteamiento del problema y objetivos
- Fundamentación teórica
- Metodología
- Análisis y diseño del sistema
- **Desarrollo y validación del sistema**
- Conclusiones

Herramientas de desarrollo

Actividad	Herramienta Descripción
Codificación de microservicios	Se utiliza el lenguaje de programación NodeJS.
Control de flujo de trabajo	Se utiliza la herramienta GitFlow, la cual tendrá una rama develop de prepublicación y la rama main que es la principal y se sincroniza con las ramas personales
Base de datos	Se utilizará PostgresSQL. Se utiliza la herramienta Liquibase, para la sincronización de los cambios en la base de datos.
Pruebas de funcionalidad	Se utiliza la herramienta Postman
Pruebas de rendimiento	Se utiliza la herramienta Locust
Gestión del ciclo de vida del proyecto y herramientas DevOps	Se utiliza la aplicación Azure DevOps, que permite gestionar repositorios, gestión de backlogs y herramientas de código.
Despliegue de Microservicios	Se utiliza la nube Azure

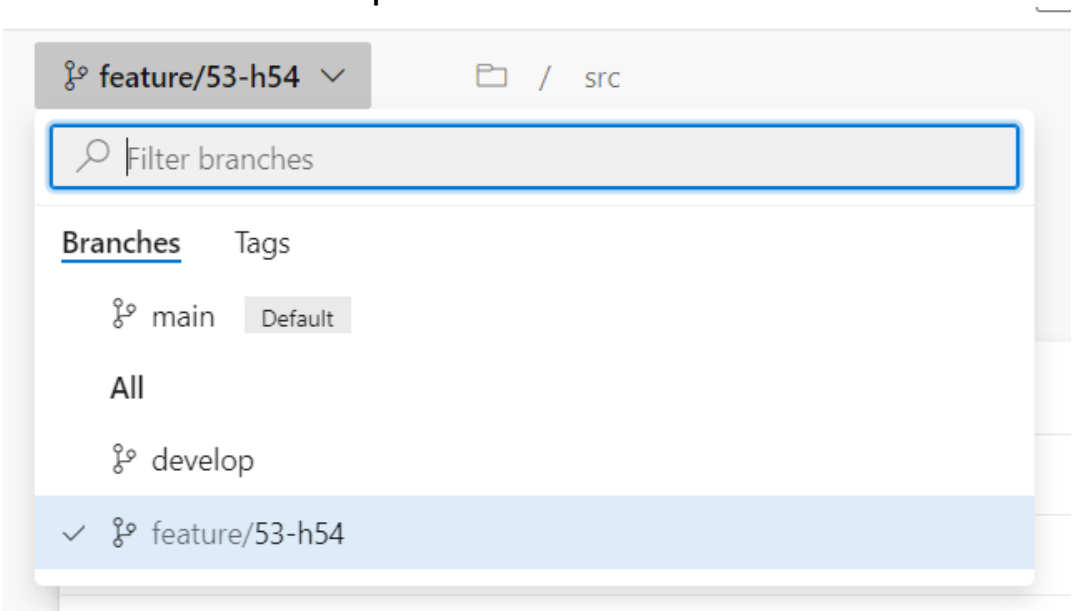
Gestión de la base de datos

1. Clonar el repositorio de la base de datos
2. Se crean las ramas correspondientes a cada microservicio.
3. Sincronizar los cambios con la rama develop.
4. Subir los cambios a la base de datos mediante el comando “**Liquibase Update**”.



Desarrollo de microservicios

1. Crear un repositorio en Azure DevOps por cada microservicio.
2. Crear las ramas de acuerdo a las historias de usuario, por ejemplo para el repositorio Catálogo de Productos se crea la rama 53-h54 perteneciente a catálogo e historial de productos.
3. Crear rutas, controladores, middlewares, validaciones e index.
4. Sincronizar las ramas con la rama develop. En la siguiente imagen, se podrá ver un ejemplo de repositorio con sus respectivas ramas.



Listado de microservicios

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

Despliegue de microservicios

Se prepara el entorno, en el archivo index se configura el puerto **app. listen (process.env. PORT || 8181)**. En el archivo package json se pone "**start": "node ./src/index.js"**

1. Configuración

2. Sincronización de Ramas

Se sincronizan las ramas develop hacia la rama main.

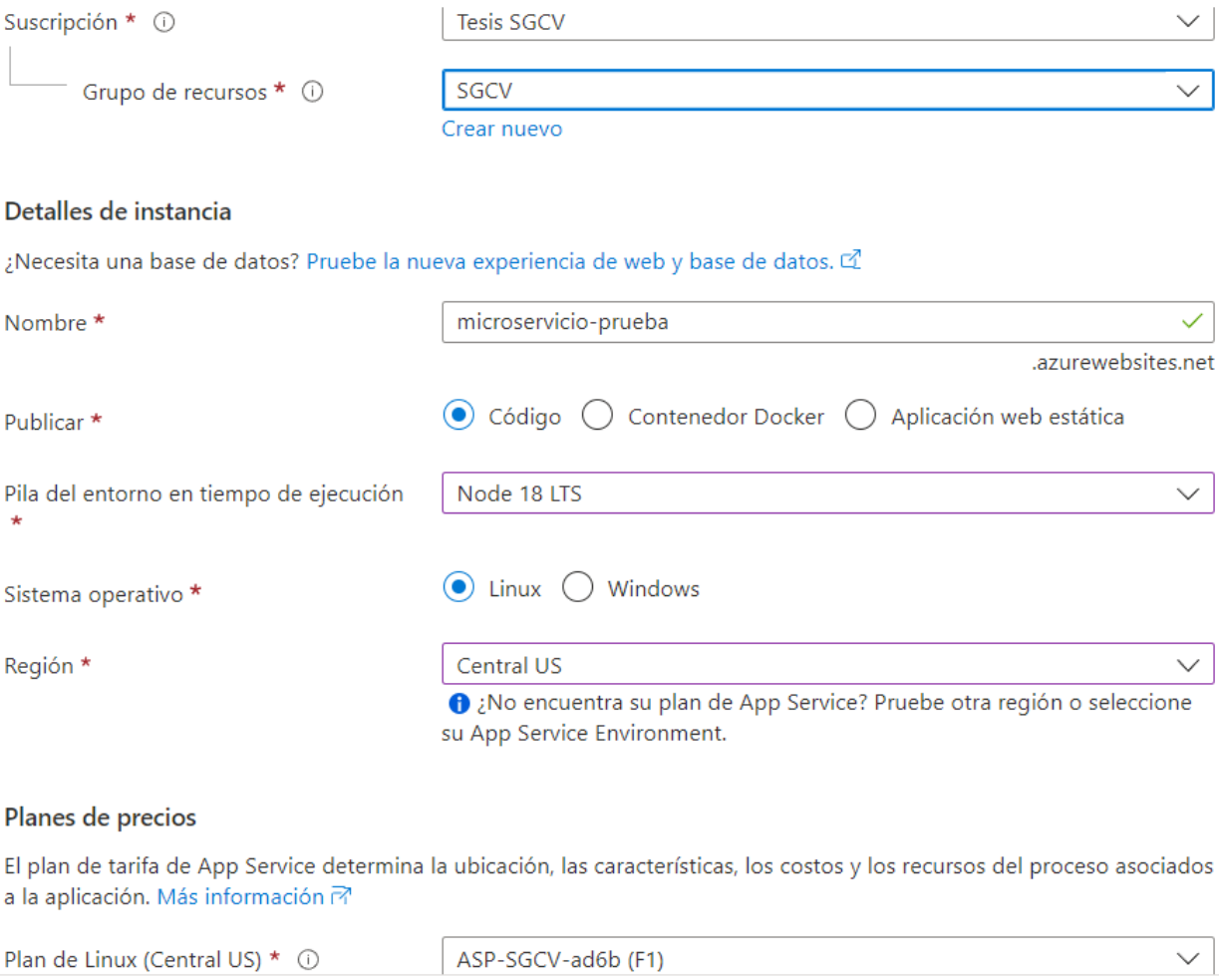
En Azure se selecciona el grupo de recursos SGCV, y en buscar se selecciona aplicación web.

3. Creación de App Service

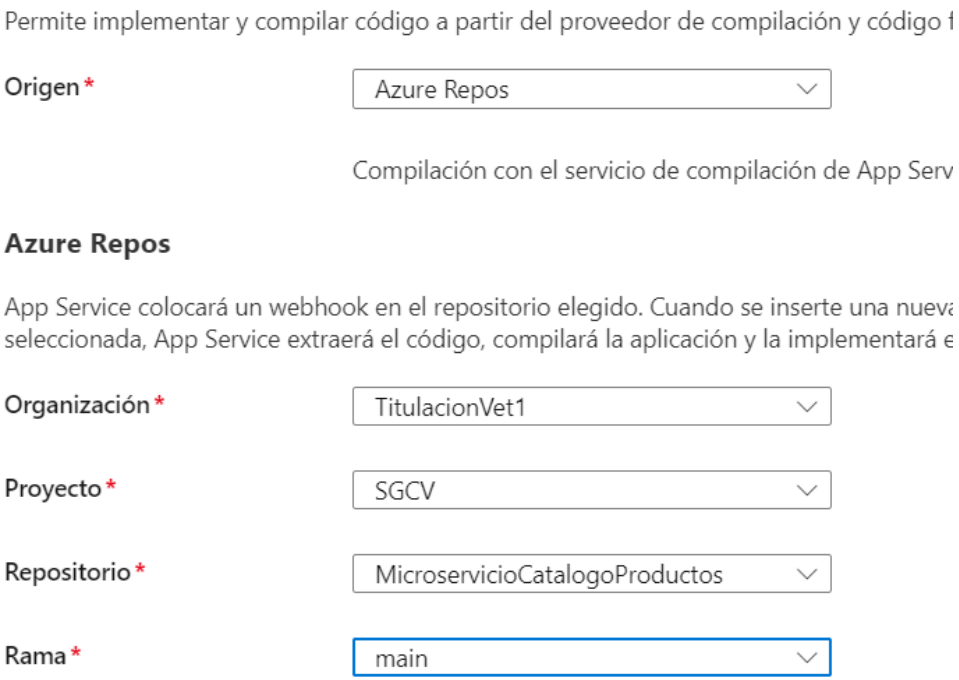
4. Configuración de App Service

Se llenan los datos requeridos que son asignarle un nombre al microservicio, escoger el lenguaje de programación y el sistema operativo, el plan de pago y la zona horaria.

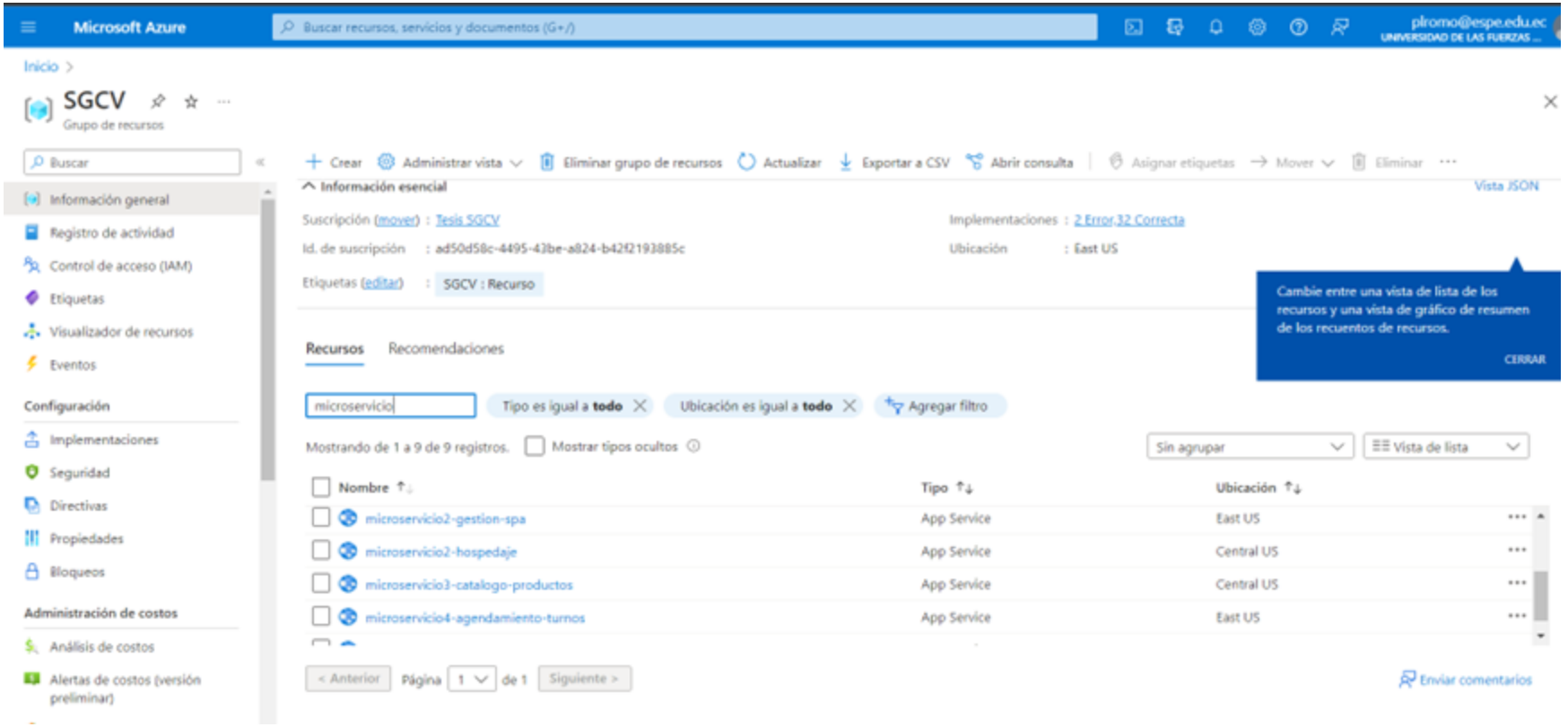
Despliegue de microservicios - Configuración para crear un AppService



Despliegue de microservicios - Configuración del AppService para implementación

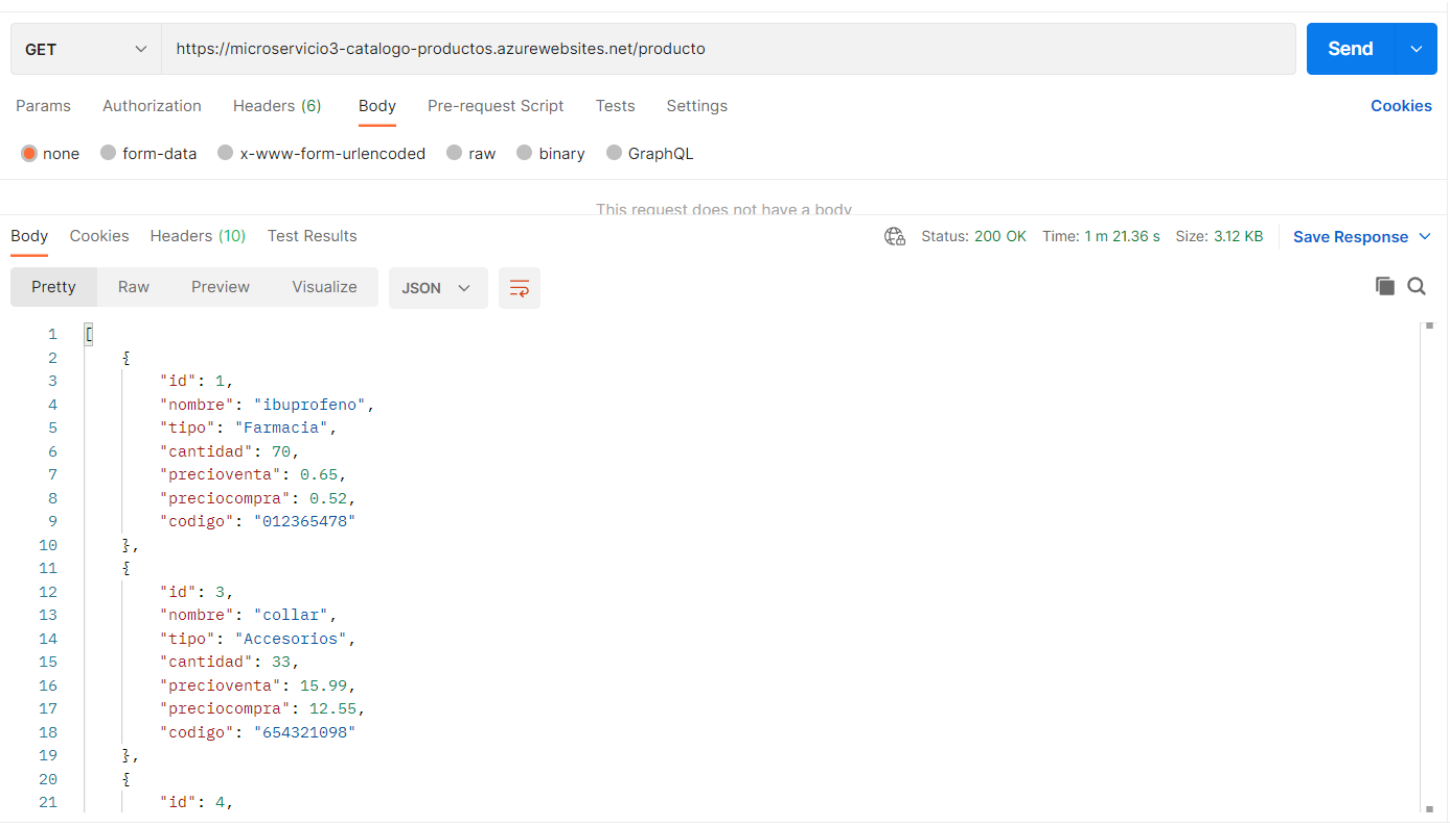


Despliegue de microservicios

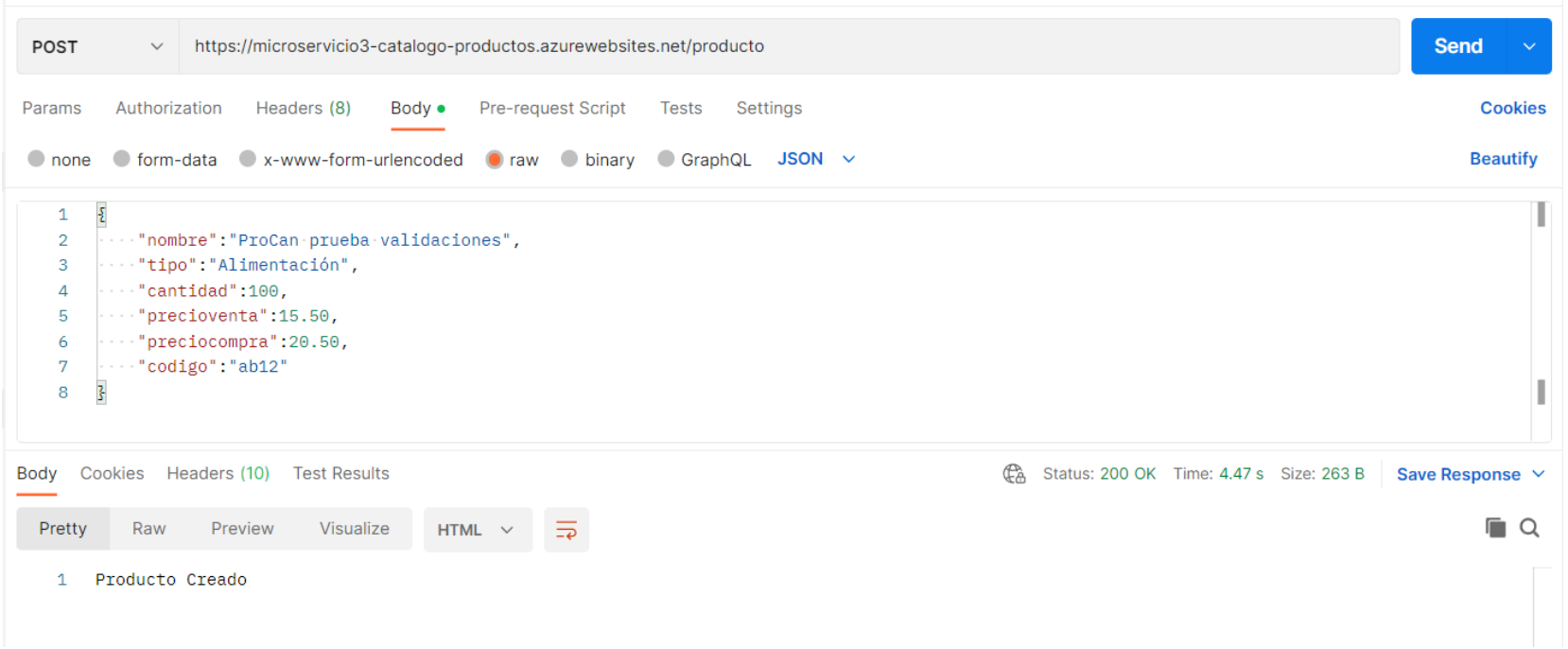


Validación de los microservicios - Postman

En las pruebas de funcionalidad se busca comprobar que los microservicios funcionan de manera correcta de acuerdo con los requisitos, las mismas se harán de acuerdo con las rutas. Las peticiones HTTP a evaluar son GET,POST, PUT,DELETE



Pruebas de funcionalidad - Método POST



Validación de los microservicios - Locust

La herramienta se encargará de las pruebas de rendimiento. Para las mismas primero se crea un archivo con las peticiones GET y POST, este archivo llamado locustfile.py, se lo corre en la terminal con el comando locust. Se pedirá que se haga las pruebas de ingreso de 10 usuarios cada 1 segundo. Locust empezará con las pruebas y enviará una cantidad masiva de datos. Se genera un reporte.

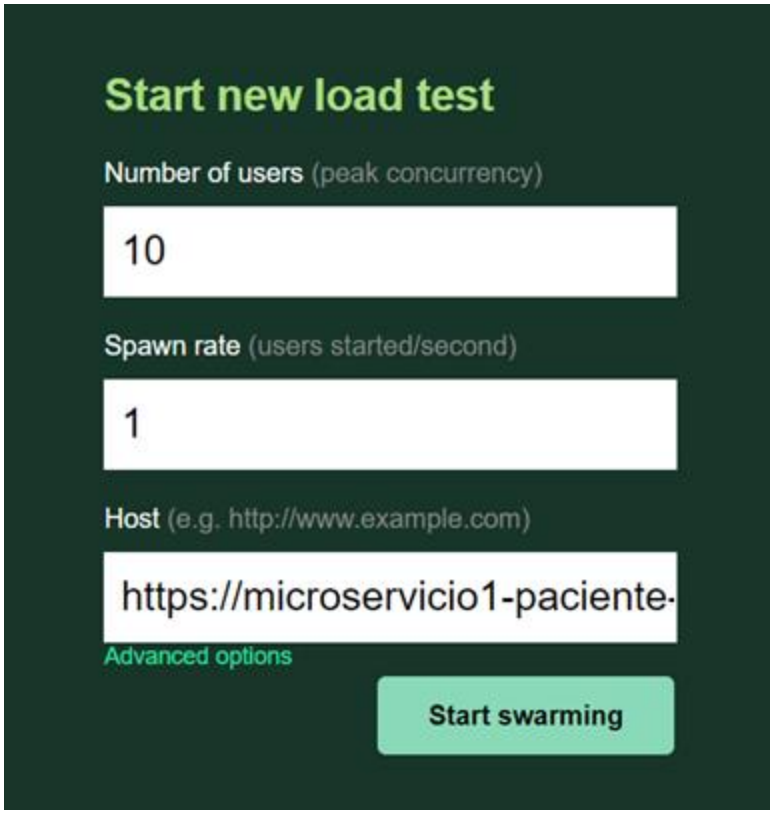
Get Started | locustfile.py

locustfile.py > Tutor > crear_tutores

```python
from locust import HttpUser, task

class Tutor(HttpUser):
    @task
    def listar_tutores(self):
        self.client.get("/tutores")
    @task
    def listar_tutores_porID(self):
        self.client.get("/tutores/5")
    @task
    def crear_tutores(self):
        self.client.post("/tutores",json={

        "nombre": "Andrea Jimenez",
        "ci": "0501724306",
        "direccion": "Ambato, Huchi Chico",
        "telefono": "0991452428",
    })
```



Pruebas de rendimiento - Locust

Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
POST	/tutores	475	0	170	142	1314	26	15.9	0.0
GET	/tutores/5	513	0	159	135	539	112	17.2	0.0
GET	/tutores	518	0	172	140	740	32071	17.4	0.0
	Aggregated	**1506**	**0**	**167**	**135**	**1314**	**11077**	**50.5**	**0.0**

Pruebas de rendimiento - Locust

Listas de chequeo

- Mediante la técnica de listas de chequeo, se verifica el cumplimiento de los requerimientos planteados ya que cada criterio de aceptación de las historias de usuario planteadas se convierte en una prueba.

- Las pruebas realizadas evidencian el cumplimiento de los requisitos planteados previamente como historias de usuario.

Casos de prueba

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.	✓

Validación de la hipótesis

Se ha cumplido con la hipótesis establecida, ya que al seguir el proceso de la Línea de Producto de Software se ha creado un sistema que se ajusta a los requerimientos pedidos por las clínicas veterinarias.

Contenido

- Planteamiento del problema y objetivos
- Fundamentación teórica
- Metodología
- Análisis y diseño del sistema
- Desarrollo y validación del sistema
- **Conclusiones**

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.

GRACIAS