UNIVERSIDAD NACIONAL
"PEDRO RUIZ GALLO"
ESCUELA DE POSGRADO

MAESTRÍA EN INGENIERÍA DE SISTEMAS CON MENCIÓN EN GERENCIA DE TECNOLOGÍAS DE LA INFORMACIÓN Y GESTIÓN DEL SOFTWARE

"Arquitectura de software basada en microservicios para mejorar la disponibilidad de historias clínicas electrónicas odontológicas, Chiclayo – Lambayeque, 2020"

TESIS

Presentada para optar el Grado Académico de Maestro en Ingeniería de Sistemas con Mención en Gerencia de Tecnologías de la Información y Gestión del Software

Autor:
Ing. Juan Carlos Arcila Díaz

Asesor:
Mg. Carlos Alberto Valdivia Salazar

Lambayeque – Perú

2021

“**Arquitectura de software basada en microservicios para mejorar la disponibilidad de historias clínicas electrónicas odontológicas, Chiclayo – Lambayeque, 2020”.**

Ing. Juan Carlos Arcila Díaz
Autor

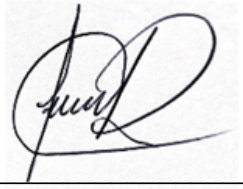
Mg. Carlos Alberto Valdivia Salazar
Asesor

Tesis presentada a la Escuela de Posgrado de la Universidad Nacional Pedro Ruiz Gallo para optar el Grado de: MAESTRO EN INGENIERÍA DE SISTEMAS CON MENCIÓN EN GERENCIA DE TECNOLOGÍAS DE LA INFORMACIÓN Y GESTIÓN DEL SOFTWARE.

Aprobado por:

Dr. Jessie Leila Bravo Jaico
Presidente del jurado

Dr. Gilberto Carrión Barco
Secretario del jurado



Mg. Nilton Cesar German Reyes
Vocal del jurado

Lambayeque, 2021

Acta de sustentación

UNPRG	ESCUELA DE POSGRADO *M.Sc. Francis Villena Rodriguez*	Versión:	01
		Fecha de Aprobación	29-8-2020
UNIDAD DE INVESTIGACION	FORMATO DE ACTA DE SUSTENTACIÓN VIRTUAL DE TESIS	Pág. 1	

ACTA DE SUSTENTACIÓN VIRTUAL DE TESIS

Siendo las 04:05 p.m. del día miércoles 11 de agosto de 2021, se dio inicio a la Sustentación Virtual de Tesis soportado por el sistema Google Meet, preparado y controlado por la Unidad de Tele Educación de la Escuela de Posgrado de la Universidad Nacional Pedro Ruiz Gallo de Lambayeque, con la participación en la Video Conferencia de los miembros del Jurado, nombrados con Resolución Nº105-2020-EPG, de fecha 29 de enero de 2021, conformado por:

Dra. JESSIE LEILA BRAVO JAICO	Presidente
Dr. GILBERTO CARRION BARCO	Secretario
Mg. NILTON CESAR GERMAN REYES	Vocal
Mg. CARLOS ALBERTO VALDIVIA SALAZAR	Asesor

Para evaluar el informe de tesis del tesista JUAN CARLOS ARCILA DIAZ, candidato a optar el grado de MAESTRO EN INGENIERIA DE SISTEMAS CON MENCION EN GERENCIA DE TECNOLOGIAS DE LA INFORMACION Y GESTION DEL SOFTWARE con la tesis titulada "ARQUITECTURA DE SOFTWARE BASADA EN MICROSERVICIOS PARA MEJORAR LA DISPONIBILIDAD DE HISTORIAS CLINICAS ELECTRONICAS ODONTOLOGICAS, CHICLAYO – LAMBAYEQUE, 2020".

El Sr. Presidente, después de transmitir el saludo a todos los participantes en la Video Conferencia de la Sustentación Virtual ordenó la lectura de la Resolución N°581-2021-EPG de fecha 03 de agosto de 2021 que autoriza la Sustentación Virtual del Informe de Tesis correspondiente, luego de lo cual autorizó al candidato a efectuar la Sustentación Virtual, otorgándole 30 minutos de tiempo y autorizando también compartir su pantalla. Culminada la exposición del candidato, se procedió a la intervención de los miembros del jurado, exponiendo sus opiniones y observaciones correspondientes, posteriormente se realizaron las preguntas al candidato.

Culminadas las preguntas y respuestas, el Sr. Presidente, autorizó el pase de los miembros del Jurado a la sala de video conferencia reservada para el

debate sobre la Sustentación Virtual del Informe de Tesis realizada por el candidato, evaluando en base a la rúbrica de sustentación y determinando el resultado total de la tesis con puntos 18, equivalente a_MUY BUENO, quedando el candidato apto para optar el Grado de MAESTRO EN INGENIERIA DE SISTEMAS CON MENCION EN GERENCIA DE TECNOLOGIAS DE LA INFORMACION Y GESTION DEL SOFTWARE.

Se retornó a la Video Conferencia de Sustentación Virtual, se dio a conocer el resultado, dando lectura del acta y se culminó con los actos finales en la Video Conferencia de Sustentación Virtual.

Siendo las _05:20_p.m. se dio por concluido el acto de Sustentación Virtual.

PRESIDENTE **SECRETARIO**

VOCAL **ASESOR**

Dr. Luis Jaime Collantes Santisteban
Director Académico

NOTA: La existencia del acta en los archivos de la Escuela de Posgrado de la Universidad Nacional Pedro Ruiz Gallo; ha sido verificada por Sra. Gloria Luisa Carranza Velásquez, quien con su firma da fe de lo mencionado.

Lic. Gloria Luisa Carranza Velásquez
Personal Administrativo

Lambayeque, 27 de agosto de 2021

Declaración jurada de originalidad

Yo Juan Carlos Arcila Díaz investigador principal, y Carlos Alberto Valdivia Salazar, asesor del trabajo de investigación "Arquitectura de software basada en microservicios para mejorar la disponibilidad de historias clínicas electrónicas odontológicas, Chiclayo – Lambayeque, 2020", declaramos bajo juramento que este trabajo no ha sido plagiado, ni contiene datos falsos. En caso se demostrará lo contrario, asumo responsablemente la anulación de este informe y por ende el proceso administrativo a que hubiere lugar. Que puede conducir a la anulación del título o grado emitido como consecuencia de este informe.

Lambayeque, 2 de junio del 2021.

Ing. Juan Carlos Arcila Díaz
Autor

Mg. Carlos Alberto Valdivia Salazar
Asesor

Dedicatoria

A mi esposa Fiorella, por su compañía y apoyo para cumplir metas personales y profesionales.

A mis hermanos Luis, Liliana, Cesy, mis sobrinos Julio y Dayiro, quienes son el futuro de mi familia para seguir cumpliendo nuestras metas.

A mi sobrina Nikol, partiste muy joven al cielo, tu recuerdo estará presente en mi corazón.

Agradecimiento

Con el sentimiento más sincero a Dios nuestro padre creador por permitirme alcanzar este logro en mi vida.

A mis padres Ana y Juan por apoyarme en

todo momento, todos mis logros son suyos.

A mi asesor Mg. Carlos Valdivia por su orientación en el desarrollo de esta

investigación.

A todos mis docentes y compañeros de esta maestría.

¡Gracias!

Índice General

Índice de Figuras

Índice de Tablas

Índice de Anexos

Resumen

Una persona puede atenderse en diferentes Proveedores de Servicios odontológicos, iniciando su atención con el registro de sus datos de paciente, Historia Clínica Odontológica y odontograma, si se desea mantener disponible esta información para la consulta posterior del Proveedor de Servicio Odontológico es necesario el diseño de un sistema de software.

En este trabajo se diseña una arquitectura de software basada en el enfoque de Microservicios para permitir la disponibilidad de las Historias Clínicas Odontológicas. Se identificaron los atributos de calidad y requerimientos funcionales para diseñar la arquitectura, determinando que debe estar compuesta por 4 Microservicios Paciente, HCO, Odontograma y Proveedor de Servicio Odontológico, cada microservicio implementa su base de datos independiente, se realiza la comunicación segura entre los microservicios y los clientes mediante un API Gateway de recursos HTTP y un Token de autenticación.

Se evaluaron los Microservicios que componen la arquitectura diseñada utilizando los atributos de calidad de los Microservicios: Cohesión, acoplamiento y reusabilidad.

Para evaluar la arquitectura de software se elaboró un prototipo que implementa cada uno de sus componentes, se desplego la arquitectura diseñada utilizando contenedores para cada microservicio y el API Gateway, sobre este prototipo se realizaron pruebas de caja negra para evaluar el atributo de Funcionalidad y Seguridad, se utilizó pruebas de escalabilidad para evaluar el atributo de calidad Escalabilidad, logrando determinar que se cumple con los requisitos identificados. Se evaluó la Disponibilidad y Desempeño utilizando pruebas de carga determinando que se puede disponer hasta 21 HCO por segundo con una disponibilidad del 100%, y si la demanda de peticiones aumenta la arquitectura escala horizontalmente de manera automática.

Palabras clave: Historia Clínica Odontológica, Microservicios, disponibilidad, arquitectura de Software.

Abstract

A person can be treated at different Dental Service Providers, starting their care with the registration of their patient data, Dental Clinical History and odontogram, if you want to keep this information available for subsequent consultation of the Dental Service Provider, it is necessary to design a software system.

In this work, a software architecture based on the Microservices approach is designed to allow the availability of Dental Medical Records. The quality attributes and functional requirements were identified to design the architecture, determining that it must be composed of 4 Microservices Patient, HCO, Odontogram and Dental Service Provider, each microservice implements its independent database, secure communication between the microservices and clients using an API Gateway of HTTP resources and Authentication by Token.

The Microservices that make up the designed architecture were evaluated using the quality attributes of Microservices: Cohesion, coupling and reusability.

To evaluate the software architecture, a prototype was developed that implements each component, the designed architecture was deployed using containers for each microservice and the API Gateway, on this prototype, black box tests were applied to evaluate the Functionality and Security attribute, tests of scalability for evaluate the quality attribute Scalability, determining that the identified requirements are met. Availability and Performance were evaluated using load tests determining that up to 21 HCOs per second can be available with 100% availability, and if the demand for requests increases the architecture scales horizontally automatically.

Keywords: Dental Clinical History, Microservices, availability, Software architecture.

Introducción

Una persona puede atender sus problemas odontológicos en diferentes proveedores ya sea en clínicas odontológicas privadas, establecimientos de salud públicos o con diferentes cirujanos dentistas de manera particular. En Perú algunas clínicas odontológicas y establecimientos de salud utilizan software de gestión de información para el registro de los servicios clínicos brindados y registro de historias clínicas; mientras que otras clínicas o establecimientos odontológicos gestionan la información de sus historias clínicas en papel (HCP). Si el paciente acude por primera vez a la atención con un odontólogo en cualquier proveedor de servicios odontológicos se procede a registrar los datos iniciales del paciente, este proceso implica llenar formularios en donde se ingresan los datos del paciente en su Historia Clínica Odontológica (Colegio Odontológico del Perú, 2019) y su odontograma siguiendo la normativa de salud correspondiente (Ministerio de Salud Perú, 2019), está información se almacena ya sea en una HCP o Historia Clínica Electrónica (HCE) utilizando su sistema de información, esta historia clínica solo es gestionada por el proveedor de servicio odontológico que hizo el registro. Muchas veces el paciente debe de cambiar de proveedor de servicios odontológicos ya sea porque este no cuenta con los servicios requeridos, por el cambio de ubicación física del paciente o motivado por razones económicas (en ciertos proveedores odontológicos los tratamientos son más baratos), como la información registrada en el proveedor de servicio odontológico anterior no se comparte con este nuevo proveedor el proceso de registro de la historia clínica odontológica se repite ocasionando uso adicional de tiempo y recursos, además de la pérdida de información de los procesos odontológicos realizados anteriormente.

Este problema se presenta también en las historias clínicas registradas en papel de los pacientes en establecimientos de salud en Perú (Rojas Mezarina, Cedamentos Medina, y Vargas Herrera, 2015), frente a estos inconvenientes, surge la HCE, que ofrece muchas ventajas frente a la HCP. El problema de la multiplicidad de historias clínicas no se resuelve solamente con las HCE, sino que es necesario también una herramienta tecnológica que permita interoperar y gestionar las HCE.

En Perú mediante Ley 30024 (Congreso de La República del Perú, 2013), se creó el Registro Nacional de Historias Clínicas Electrónicas (RENHICE).

El RENHICE es un directorio electrónico en el que los profesionales de la salud de los establecimientos de salud o servicios médicos de apoyo, a través de sus sistemas de información de HCE debidamente acreditados, podrán acceder y consultar en qué otros establecimientos, un paciente tiene una HCE y permitirle acceder a esta o estas, sin que se pueda almacenar en su sistema de información (base de datos del consultante) ni en la del RENHICE, los datos de la HCE del establecimiento consultado (Rojas Mezarina et al., 2015).

Siendo el RENHICE una base de datos de filiación de cada persona con la relación de los establecimientos de salud que le han brindado atención y generado una HCE es necesario que estos establecimientos registren las HCE estandarizadas y aseguren su disponibilidad.

Las historias clínicas odontológicas (HCO) se deben de gestionar de la misma manera, permitiendo la disponibilidad desde cualquier proveedor de servicios odontológicos, para asegurar la disponibilidad se deben almacenar sobre una plataforma que permita la seguridad e integridad de la información almacenada en las HCO siguiendo un formato estándar; considerando que la mayoría de establecimientos odontológicos no cuenta con un software para gestionar sus HCE

(LOLIMSA, 2014) y los pocos establecimientos que si gestionan sus HCE lo almacenan en su propia base de datos. Siendo necesario entonces que el software que se desarrolle tenga una arquitectura de software que permita la disponibilidad de historias clínicas manteniendo la seguridad e integridad de la información y un formato estándar de HCO.

Actualmente es común en las empresas encontrar software desarrollados utilizando una arquitectura monolítica teniendo inconvenientes en la escalabilidad, flexibilidad y sobre todo no permiten la comunicación adecuada con otros proyectos de software. Una alternativa de arquitectura de software que si va a permitir la disponibilidad de HCO es la Arquitectura Orientada al Servicio (SOA), SOA debe soportar las necesidades de interoperabilidad, diferentes grados de integración, varios patrones de mensajes y diferentes fases del ciclo de vida del sistema (Sánchez Reyna, 2015).

Existe otro estilo de arquitectura de software llamado Microservicios que permite que las aplicaciones se dividen en sus componentes más pequeños y sean independientes entre sí, permitiendo además la disponibilidad de información. Este enfoque de microservicios viene siendo analizado en diferentes proyectos de gestión e intercambio de datos relacionados con la asistencia sanitaria basada en la nube (Esposito, Castiglione, Tudorica, y Pop, 2017).

En este trabajo de investigación se ha diseñado una arquitectura de software basada en el enfoque de Microservicios para permitir la disponibilidad de las Historias Clínicas Odontológicas. Se inició con el proceso de diseño de la arquitectura identificando los requerimientos funcionales revisando antecedentes en la literatura referentes a sistemas de Historias clínicas, revisión de la normatividad peruana y los requerimientos funcionales según la documentación revisada, se consolidaron los

siguientes atributos de calidad relevantes: Funcionalidad, disponibilidad, seguridad, mantenibilidad, escalabilidad y desempeño.

Utilizando un diagrama de casos de uso y la descomposición en sustantivos se determinó que la arquitectura de software a diseñar debe estar compuesta por 4 Microservicios Paciente, HCO, Odontograma y Proveedor de Servicio Odontológico, cada microservicio implementa su base de datos independiente, se realiza la comunicación segura entre los microservicios y los clientes mediante un API Gateway de recursos HTTP y un Token de autenticación.

Los microservicios que componen la arquitectura de software diseñada fueron evaluados utilizando los atributos de calidad de los Microservicios (Kharbuja, 2016): Cohesión, acoplamiento y reusabilidad.

Para evaluar la arquitectura de Software diseñada se desarrolló un prototipo que implementa cada uno de los componentes, se desplego la arquitectura diseñada utilizando contenedores con características predefinidas para cada microservicio y el API Gateway, sobre este prototipo se realizaron pruebas de caja negra para evaluar el atributo de Funcionalidad y Seguridad, y pruebas de escalabilidad para evaluar el atributo de calidad Escalabilidad, logrando determinar que la arquitectura de Software cumple con los requisitos identificados.

Se ha evaluado la Disponibilidad y Desempeño utilizando pruebas de carga determinando que con una sola instancia del Microservicio HCO se puede disponer hasta 21 HCO por segundo con una disponibilidad del 100%, y si la demanda de peticiones aumenta la arquitectura escala horizontalmente de manera automática.

Capítulo I. Diseño Teórico

1.1 Antecedentes de la Investigación

Se presenta los antecedentes de investigación en tesis y artículos científicos recientemente publicados.

1.1.1. A nivel internacional

Se vienen realizando investigaciones para mejorar la gestión de HCE, Rivera López, Santander Acosta y Sixto Fuentes (2021) han desarrollado una Arquitectura de información para la gestión de la historia clínica digital en Oftalmopediatría, su estudio es cuali-cuantitativo con métodos teóricos y empíricos que analizan la evolución y desarrollo de la historia clínica de Oftalmopediatría, crearon un prototipo de software con un gran impacto social para las entidades donde se implemente, beneficiando tanto al paciente como a los profesionales.

Por su parte Soto Vidal y Cuello (2020) en su tesis denominada "Plataforma para la atención médica a pacientes en época de Covid-19" consideraron como objetivo principal desarrollar una plataforma como mecanismo de comunicación entre pacientes y médicos en la pandemia de Covid, esta plataforma está constituida por 3 aplicaciones, un aplicativo web para la gestión de las solicitudes de citas médicas, un aplicativo móvil para pacientes y otra para médicos que permita realizar el proceso de tele consulta; para la obtención de requisitos funcionales y no funcionales analizaron la literatura existente y utilizaron una encuesta. Sus aplicaciones se interconectan utilizando un API REST.

Para el monitoreo de pacientes con diagnóstico de Alzheimer Serna Flórez y Giraldo Plaza (2020) plantean un diseño arquitectónico basado en microservicios para soportar el monitoreo propuesto, su diseño compuesto por 5 microservicios, usuarios, verificación médica, ubicación médica, mensajes y contacto, realizaron un prototipo de su diseño arquitectónico en el cual se muestra las diferentes funcionalidades del sistema de monitoreo.

Para la validación de los atributos de calidad se diseñaron escenarios de validación, demostrando que su diseño de arquitectura sirve de ayuda para el monitoreo adecuado de pacientes.

Calderón-Gómez et al. (2019) en su proyecto "Desarrollo de aplicaciones eHealth basadas en microservicios en una arquitectura de Cloud", presentan una propuesta para fortalecer el Registro Médico Electrónico (RME) de Panamá y los sistemas de información disponibles ya que presentan una escasa disponibilidad en el acceso y en el procesamiento de sus datos limitando de esta manera la toma de decisiones clínico-médicas. Desarrollaron dos sistemas de información con arquitecturas de software usando el paradigma de la computación en la nube (SaaS) y orientado a la implementación de microservicios, ambas arquitecturas presentan capas estables y modificables, adaptadas a las necesidades de cualquier organización, escalables y replicadas en varios contenedores.

Para evaluar la eficiencia de las arquitecturas de software (3 y 5 capas) de los sistemas de información desarrollados con este enfoque se utilizó pruebas de rendimiento y pruebas de estrés para obtener valores cuantitativos y medir el rendimiento general de la aplicación y su interacción con los usuarios finales. Se ha utilizado un entorno de pruebas controladas con el propósito de generar una carga de trabajo que aumenta con el tiempo emulando un número de específico de sesiones de 30 usuarios virtuales concurrentes.

Para la gestión de historias clínicas odontológicas de pacientes Granda Vinueza (2018), en su trabajo de investigación desarrolló un sistema informático conformado por dos capas, Back-End y Front-End, basado en un enfoque de microservicios. Para validar su sistema informático realizó pruebas de caja negra y pruebas de estrés.

Esposito et al. (2017) en su trabajo "Security and Privacy for Cloud-Based Data Management in the Health Network Service Chain: A Microservice Approach", investigan los problemas de seguridad y privacidad causados por el uso de la computación en la nube

para la gestión e intercambio de datos relacionados con la atención médica entre los diferentes proveedores, se proponen soluciones a estos problemas dentro del contexto de los microservicios. Presentan en detalle los medios de seguridad y privacidad mediante la descripción de los requisitos, un modelo novedoso de control de acceso basado en la semántica, la preservación de la privacidad del almacenamiento de datos y la notificación de violación de datos. Los resultados experimentales solo muestran la viabilidad del nuevo sistema propuesto en un escenario específico.

Por su parte Villa Benavides (2017) en su trabajo "Diseño de una arquitectura que soporte la interoperabilidad de la historia clínica electrónica de pacientes en situaciones de emergencia", aborda el diseño de una arquitectura de sistema para la interoperabilidad de la historia clínica electrónica de pacientes en situaciones de emergencia, en el contexto del sistema de salud en Colombia. Se diseño la arquitectura para la interoperabilidad de la Electronic Health Record (EHR) para pacientes en situaciones de emergencia, en esta se establece los lineamientos que permiten el intercambio de información clínica dentro de un ambiente colaborativo entre los actores del sistema de salud de Colombia y brinda la posibilidad de integrar la información clínica de los pacientes en un repositorio único de historias clínicas electrónicas gestionadas por los pacientes. Para validar la arquitectura se consideraron los drivers de la arquitectura, propuestas arquitectónicas y escenarios.

La arquitectura propuesta es analizada y discutida de manera cualitativa, considerando como factores de éxito: la aceptación del sistema por parte de los usuarios, la participación del personal médico, la sostenibilidad financiera de la infraestructura, el aseguramiento de la confidencialidad de la información sensible y los lineamientos del gobierno para el establecimiento mecanismos que promuevan la interoperabilidad de la información del sector salud.

1.1.2. A nivel nacional

Por otro lado, Cayo Alcos (2018) propone el desarrollo de un software como servicio usando una arquitectura de microservicios para mejorar la gestión de procesos de servicio médico para la Clínica Universitaria de la Universidad Nacional Mayor de San Marcos, permitiendo mejorar la interoperabilidad entre las áreas médicas y la consulta de historias clínicas electrónicas. Utilizando una base de datos relacional para la gestión del examen médico y una base de datos no relacional para la gestión de la historia clínica.

Para la evaluación del software propuesto se ha utilizado pruebas de bajo nivel (unitarias e integración) y de alto nivel (pruebas de sistemas y aceptación) obteniendo resultados exitosos.

Como menciona Ruelas Acero (2017), la composición de servicios permite integrar varios servicios para brindar una solución de software, su investigación doctoral tiene como objetivo proponer un modelo de composición de microservicios para la implementación de una aplicación web de comercio electrónico. En su proyecto para la identificación de los microservicios a implementar propone utilizar la técnica de casos de uso, la arquitectura lógica de su modelo propuesto está conformado por 4 microservicios y el API Gateway que permite la comunicación entre los microservicios.

Para la evaluación de sus microservicios implementados se utilizó los atributos de calidad: Granularidad, acoplamiento, cohesión, complejidad y reusabilidad. Se utilizo además pruebas de carga para evaluar tiempo de respuesta, disponibilidad y rendimiento de su modelo propuesto frente a una aplicación monolítica, determinando una mejora considerable.

Sánchez Reyna (2015) en su investigación, “Recuperación de historias clínicas electrónicas a partir de un repositorio digital usando una arquitectura orientada a servicios”, implementa un servicio Web que permita el registro y la recuperación de las historias clínicas electrónicas

(HCE) a partir de un repositorio centralizado, sólo se considera los formatos médicos: Formato de Atención Integral (del niño, del adolescente, del adulto y del adulto mayor), Formato de Emergencia, Formato de Consulta Externa, Formato de Hospitalización y Ficha Familiar.

Para el diseño e implementación del Web Service se define la arquitectura de software basándose en el modelo "4+1" que considera 5 vistas: lógica, implementación, procesos, física y de casos de uso; como mecanismo de seguridad del Web Service se implementa la pseudonimización que permite garantizar la privacidad de los datos de identificación del paciente, y la firma digital con cifrado simétrico y asimétrico que aseguran la integridad y veracidad de la información almacenada en el repositorio, además de la seguridad del servicio web mediante certificados, restricción de accesos por IP y tokens.

El acceso al Web Service implementado permite integrar y mantener actualizadas todas las HCE de los pacientes en un repositorio digital solucionando los problemas de ilegibilidad, deterioro, pérdida de los registros y sobre todo la no disponibilidad de las historias clínicas de los pacientes.

1.2 Base Teórica

1.2.1. Arquitectura de Software

1.2.1.1. Definición de Arquitectura de Software

Existen diferentes definiciones del concepto de Arquitectura de software, siendo la definición siguiente propuesta por Software Engineering Institute (SEI) aceptada ampliamente. La arquitectura de software es la representación de un programa o sistema informático que permite comprender su comportamiento.

La arquitectura de software sirve como modelo tanto para el sistema como para el proyecto que lo desarrolla, definiendo las tareas de trabajo que deben realizar los equipos de diseño e implementación. La arquitectura es el portador principal de las

cualidades del sistema, como el rendimiento, la modificabilidad (Bass, Clements, y Kazman, 2013).

1.2.1.2. Proceso de la Arquitectura de Software

En la figura 1 se muestra un proceso de arquitectura iterativo simple de 3 pasos que puede ser utilizado para guiar las actividades durante el proceso de la Arquitectura de Software (Gorton, 2011), estos 3 pasos son:

1. Definir los requisitos de la arquitectura.

2. Diseño de la arquitectura: Implica definir la estructura y responsabilidades de los componentes que conformarán la arquitectura.

3. Validación: esto implica "probar" la arquitectura, que por lo general se realiza revisando si el diseño cumple con los requisitos existentes.

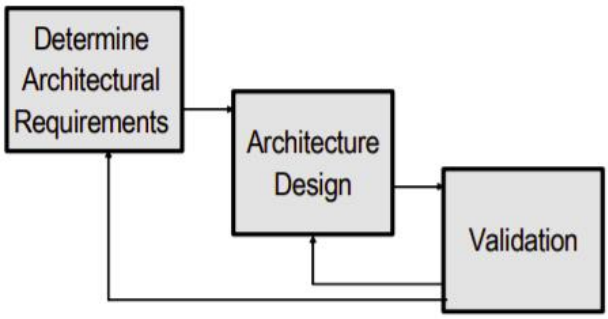


Figura 1. Un proceso de diseño de arquitectura de tres pasos.

Fuente: Gorton (2011)

Este proceso de arquitectura es esencialmente iterativo. Una vez propuesto el diseño, validarlo puede demostrar que el diseño necesita modificaciones o que ciertos requisitos deben mejorarse para su mejor comprensión. Toda arquitectura de software debe ser documentada con suficiente detalle, sin ambigüedades, y organizarla de manera que permita que los interesados la puedan implementar correctamente.

Determinar los requerimientos de la arquitectura

Antes de empezar el diseño de una solución de Arquitectura de Software es necesario tener una idea bastante clara de los requisitos de arquitectura de la aplicación (Gorton, 2011).

Según Cervantes Maceda, Velasco-Elizondo, y Castro Careaga (2017), "Requerimiento es una especificación que describe alguna funcionalidad, atributo o factor de calidad de un sistema de software. Puede describir también algún aspecto que restringe la forma en que se construye ese sistema" (p. 10).

No todos los requerimientos tienen la misma relevancia es por ello que la etapa de requerimientos se centra en la identificación de los requerimientos con mayor influencia a este subconjunto de requerimientos se le conoce como drivers arquitectónicos, sin importar la fuente todos los requisitos abarcan las siguientes categorías (Bass et al., 2013):

a) Drivers funcionales, son un subconjunto de los requerimientos funcionales que se eligen considerando su importancia para lograr los objetivos del negocio, la complejidad técnica, representan un escenario relevante para la arquitectura.
b) Drivers de atributos de calidad, requerimientos que establecen criterios sobre la calidad del sistema, para la identificación y especificación de drivers arquitectónicos de calidad se puede usar estándares como el ISO/IEC 9126.
c) Drivers de restricciones, aspectos que limitan el proceso de desarrollo del sistema, tenemos dos tipos de restricciones: restricciones técnicas que son solicitudes expresas sobre el uso, durante el desarrollo del sistema, hardware, lenguaje de programación, licencias. Restricciones administrativas, por lo general están relacionadas con el costo, tiempo de desarrollo, equipo de desarrollo.

Según Cervantes Maceda et al. (2017) para realizar la identificación, especificación y priorización de drivers arquitectónicos existen algunos métodos y modelos, algunos de ellos son:

- Taller de atributos de calidad (QWA)
- Método de diseño centrado en la arquitectura (ACDM)
- Funcionalidad, usabilidad, confiabilidad, desempeño y soporte (FURPS+)

Diseño de la arquitectura

Según Cervantes Maceda et al. (2017), "La etapa del diseño de la arquitectura puede verse como una transformación, que realiza el arquitecto, de los drivers hacia las distintas estructuras que componen la arquitectura" (p. 32).

Una característica principal de la arquitectura de software es que facilita la realización de cambios para lograrlo se aplican principios de modularidad al descomponer en partes del sistema y hacer paralelo el desarrollo; otro principio es la cohesión alta para simplificar la realización de cambios en el código y el principio de acoplamiento bajo es decir la dependencia mínima entre los módulos que componen el sistema.

Las entradas al paso de diseño son los requisitos de la arquitectura. La etapa de diseño en sí tiene dos pasos, que son de naturaleza iterativa. El primero implica elegir una estrategia general para la arquitectura, es conveniente buscar soluciones probadas a los problemas recurrentes en esta etapa (Gorton, 2011). A continuación, se describen categorías de estas soluciones:

a) Patrones, son soluciones conceptuales a problemas recurrentes en la etapa de diseño de la arquitectura (Alexander, Ishikawa y Silverstein, 1977). Al diseñar desde cero un sistema la clase de patrón con la que se inicia son los estilos

arquitectónicos, algunos ejemplos de ellos: Filtros y tuberías, capas, cliente/servidor, N-Tercios, par a par, publicador-suscriptor.

b) Tácticas, son técnicas probadas de las ciencias de la computación para resolver problemas relacionados con los atributos de calidad (Bass et al., 2013).

c) Frameworks, tanto patrones como tácticas son conceptos que deben ser implementados posteriormente en el código. Los frameworks son elementos reutilizables de software que permiten la solución de un problema específico como por ejemplo el diseño de interfaces, persistencia de objetos en una base de datos relacional. En la actualidad existen muchos frameworks. Los frameworks implementan una variedad de patrones y tácticas.

Para llevar a cabo el proceso de diseño de la arquitectura existen diversos métodos, entre estos tenemos:

- Diseño guiado por atributos (ADD), en este método la arquitectura se diseña de manera iterativa y en cada iteración se toma un elemento o parte y se descompone en subelementos. Este método produce una arquitectura puramente conceptual empleando patrones y tácticas resultando complicado mapear hacia las tecnologías necesarias para su implementación.
- ACDM, define un proceso de apoyo en la realización de tareas en el ciclo de la arquitectura de software. Este método proporciona diversos criterios y guías para diseñar (Lattanze, 2008).

El segundo paso en la etapa de diseño de la arquitectura implica especificar los componentes individuales que componen la aplicación, mostrar cómo encajan en el marco general y asignarles responsabilidades. Según Gorton (2011) Para definir los componentes principales del diseño debe realizarse lo siguiente:

- Identificar los componentes principales de la aplicación y cómo se conectan.

- Identificar la interfaz o los servicios que soporta cada componente.
- Identificar las responsabilidades del componente.
- Identificación de dependencias entre componentes.
- Identificar particiones en la arquitectura que son candidatas para distribución a través de servidores en una red.

Validación de la arquitectura

Esta etapa consiste en inspeccionar la arquitectura de software buscando inconsistencias o errores evitando de esta manera riesgos que involucren costos e impactos en el proyecto. Las evaluaciones tienen como principio la detección temprana de los defectos en la arquitectura para que no afecte las fases posteriores o el uso del producto, deben siempre enfocarse en la satisfacción de los drivers arquitectónicos pues son la base de la calidad del producto (Cervantes Maceda et al., 2017).

Según Gorton (2011) hay dos técnicas principales que resultan útiles en la validación de una arquitectura.

a) La primera técnica implica esencialmente la prueba manual de la arquitectura utilizando escenarios de prueba.
b) La segunda técnica implica la construcción de un prototipo que crea un arquetipo simple de la aplicación deseada, de modo que se pueda evaluar su capacidad para satisfacer los requisitos. Los prototipos son una solución muy reducida de rápido desarrollo que permite confirmar o negar una hipótesis en la cual se basa la arquitectura.

Documentación de la arquitectura

Crear una arquitectura no es suficiente, es necesario documentarla con suficiente detalle, sin ambigüedades, y organizarla de manera que permita que los interesados la puedan utilizar correctamente para hacer su trabajo. La documentación de la arquitectura debe ser lo suficientemente transparente y accesible para el rápido entendimiento, básicamente tiene 3 usos: sirve como medio de educación, es el vehículo principal para la comunicación entre las partes interesadas y sirve como base para el análisis y la construcción del sistema (Bass et al., 2013).

Un concepto ampliamente utilizado al elaborar la documentación es la vista, una vista describe una o más estructuras de arquitectura de software representando sus elementos. Una vista que conforma de un diagrama que representa los elementos de la estructura e información textual que describe las propiedades y relaciones entre los elementos (Cervantes Maceda et al., 2017).

En la etapa de documentación se reconocen varios tipos de vista, siendo recomendable que el arquitecto documente al menos una de las siguientes:

a) Vistas lógicas, describe las estructuras arquitectónicas y su organización en términos de unidades de implementación considerando sus propiedades y relaciones. Las unidades de implementación utilizadas incluyen clases paquetes, módulos o subsistemas. Algunas relaciones usadas con frecuencia incluyen es-parte-de, depende-de, usa, es-un. En la figura 2 se muestra un ejemplo de vista lógica.

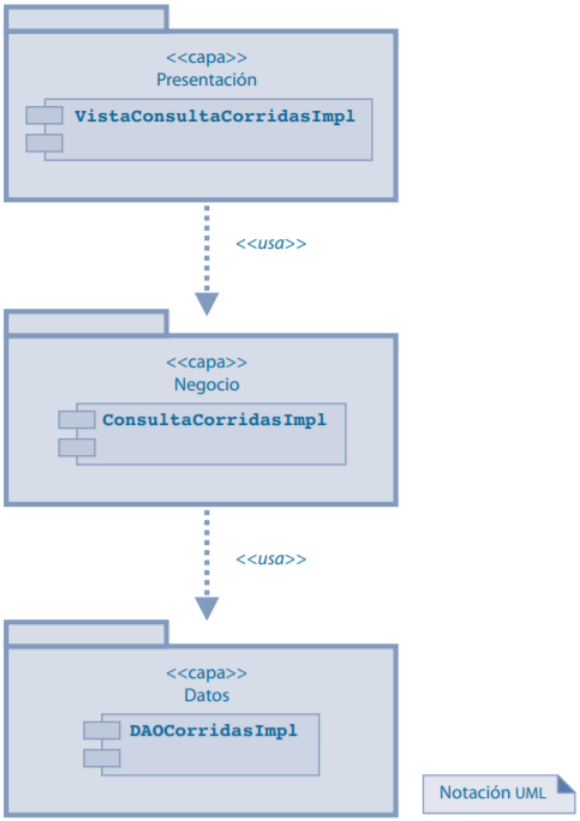


Figura 2. Ejemplo de vista lógica.

Fuente: Cervantes Maceda et al. (2017)

b) Vistas de comportamiento, describen estructuras conformadas por elementos que representan entidades visibles en tiempo de ejecución, por ejemplo, instancias, procesos, objetos, clientes, servidores o almacenes de datos. Un ejemplo de relaciones que pueden emplearse en esta vista son por ejemplo flujo-de-datos, llamada-a-procedimiento-remoto, acceso-compartido, broadcasting e incluso SOAP. La figura 3 muestra un ejemplo de vista de comportamiento que describe aspectos relacionados con la funcionalidad, representado con un diagrama de secuencia UML, este diagrama muestra una secuencia de interacciones entre instancias de elementos extraídos de la documentación estructural (Bass et al., 2013).

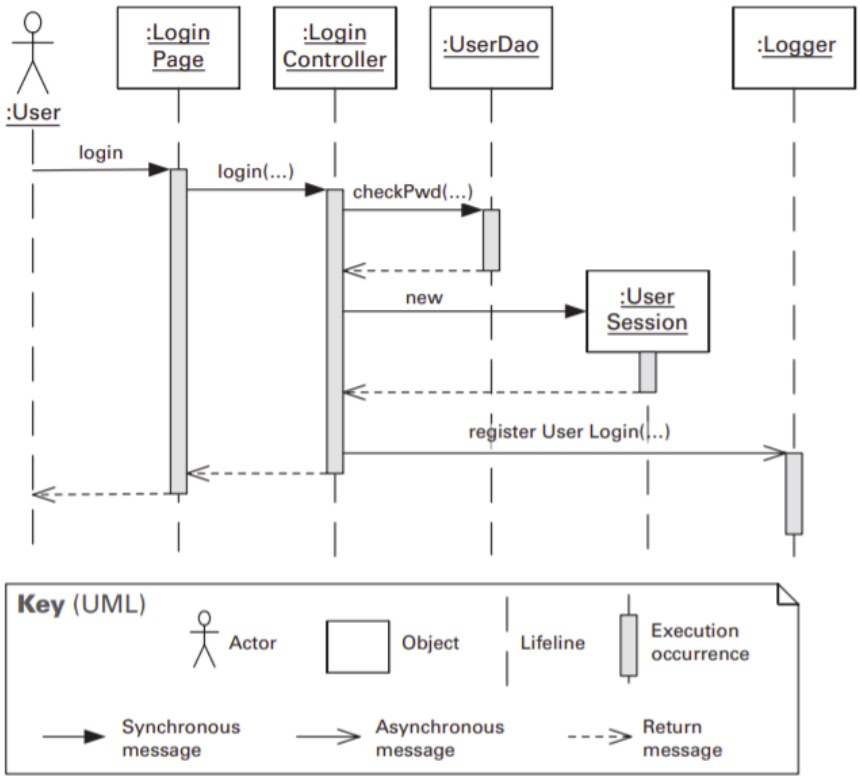


Figura 3. Ejemplo de vista de comportamiento. Diagrama de secuencia UML.

Fuente: Bass et al. (2013)

c) Vistas físicas, en esta vista se puede utilizar elementos de dos categorías: hardware, como equipos de cómputo, dispositivos incluyendo muchas veces valores específicos y software que incluyen unidades de implementación como clases, paquetes, módulos, procesos, objetos, almacenes de datos, estructuras de directorios. En las vistas físicas las relaciones como reside-en o se ejecuta-en son utilizados con frecuencia. En la figura 4 se ejemplifica una vista física.

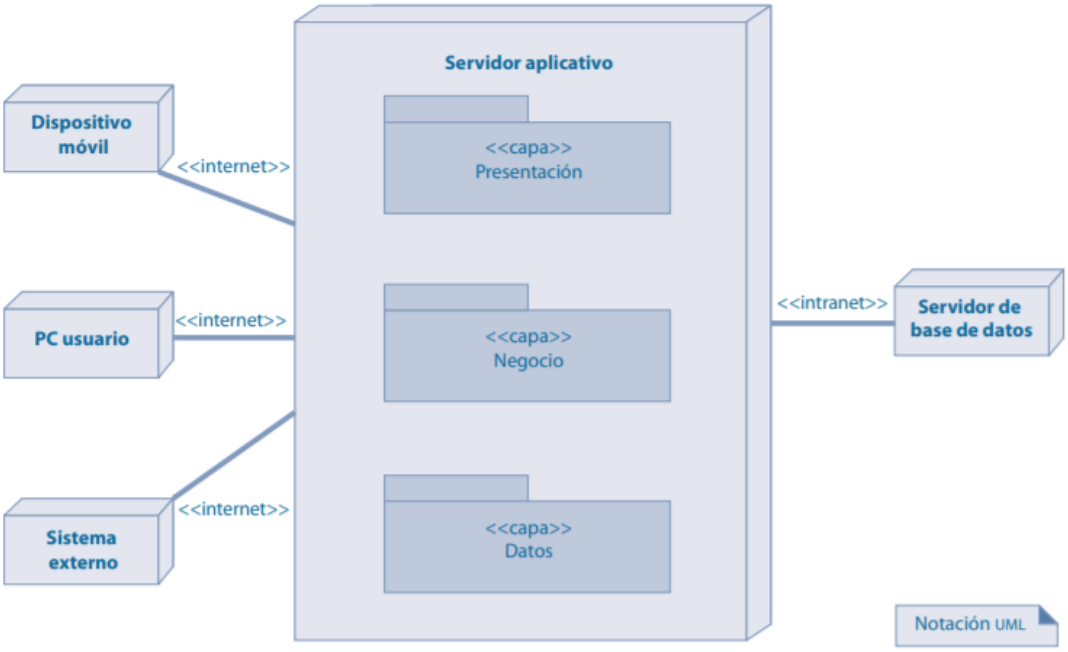


Figura 4. Ejemplo de vista física.

Fuente: Cervantes Maceda et al. (2017)

Para comunicar información sobre las estructuras de la arquitectura mediante la documentación es necesario el uso de notaciones con una sintaxis y semántica que permita su interpretación de manera correcta. Las notaciones pueden clasificarse en notaciones informales, notaciones semiformales y notaciones formales.

Implementación de la arquitectura

La arquitectura está destinada a servir como modelo para la implementación, una tarea clave para los implementadores es ejecutar fielmente las prescripciones de la arquitectura (Bass et al., 2013).

La implementación es la acción de transformar especificaciones abstractas y no concretas en un sistema que comprende elementos de software y hardware que interactúan entre sí de tal manera que cumplan las funcionalidades requeridas por los usuarios (Cervantes Maceda et al., 2017).

La implementación comprende:

- Diseñar de manera detallada los módulos del sistema de software basándose en la arquitectura permitiendo soportar los requerimientos funcionales.
- Integrar los módulos desarrollados y/o adquiridos de tal manera que funcionen correctamente.
- Probar la integración de los módulos y corregir cualquier error.

1.2.2. Arquitectura monolítica

Es una arquitectura de software tradicional. Un sistema de software desarrollada en la arquitectura monolítica es una aplicación centralizada que consta de una sola pieza, tiene como característica el uso de una base de código, no tiene sus módulos independientes y estos están estrechamente acoplados. Esta arquitectura es adecuada para desarrollar sistemas de software poco complejos que por lo general serán utilizados en un corto periodo de tiempo.

Esta arquitectura no es muy recomendable para aplicaciones grandes ya que son difíciles de mantener (Chumpolsathien, 2019).

En la figura 5 se puede ver un ejemplo de aplicaciones empresariales que a menudo se compilan en tres partes principales: una interfaz de usuario del lado del cliente (HTML y Javascript ejecutándose en el navegador web del cliente), una base de datos y una aplicación del lado del servidor que maneja las solicitudes HTTP, ejecutará la lógica del dominio, gestionará los datos de la base de datos y generará las vistas HTML para enviarlas al navegador. Esta aplicación es un monolito, un único ejecutable lógico, cualquier cambio en el sistema va a implicar un nuevo despliegue de toda la aplicación (Lewis y Fowler, 2014).

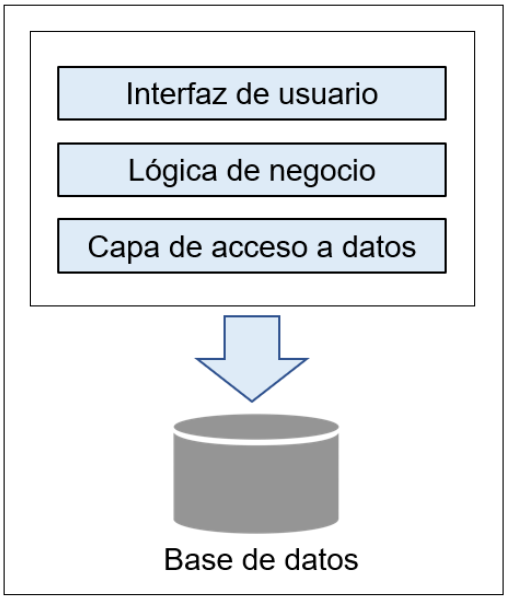


Figura 5. Arquitectura monolítica de una aplicación.

Fuente: Lewis y Fowler (2014)

1.2.3. Arquitectura SOA

La arquitectura orientada a servicios (SOA) es un enfoque de diseño en el que múltiples servicios o procesos del sistema completamente separados colaboran para proporcionar un conjunto final de capacidades. La comunicación entre estos servicios se produce a través de llamadas a través de una red, no como en el caso de aplicaciones monolíticas que se hacen llamados a métodos dentro de un límite de proceso.

SOA surgió como un enfoque para combatir los desafíos de las grandes aplicaciones monolíticas, este enfoque tiene como objetivo promover la reutilización del software, por ejemplo, dos o más aplicaciones de usuario final podrías usar los mismos servicios.

Muchos de los problemas planteados en la puerta de SOA son en realidad problemas con cosas como protocolos de comunicación (por ejemplo, SOAP), middleware del proveedor, falta de orientación sobre la granularidad del servicio o la orientación incorrecta sobre cómo elegir lugares para dividir su sistema (Newman, 2015).

En la figura 6 se ilustra a un nivel muy alto un sistema basado en SOA, este consta de servicios, consumidores de servicios que descubren y utilizan servicios y una infraestructura SOA que conecta aplicaciones con servicios proporcionando un mecanismo de comunicación estándar entre consumidores y servicios. Cada servicio representa idealmente una funcionalidad comercial y proporciona una interfaz que se invoca a través de un formato de datos y un protocolo que todos los consumidores entienden (Kajko-Mattsson y Lewis, 2014).

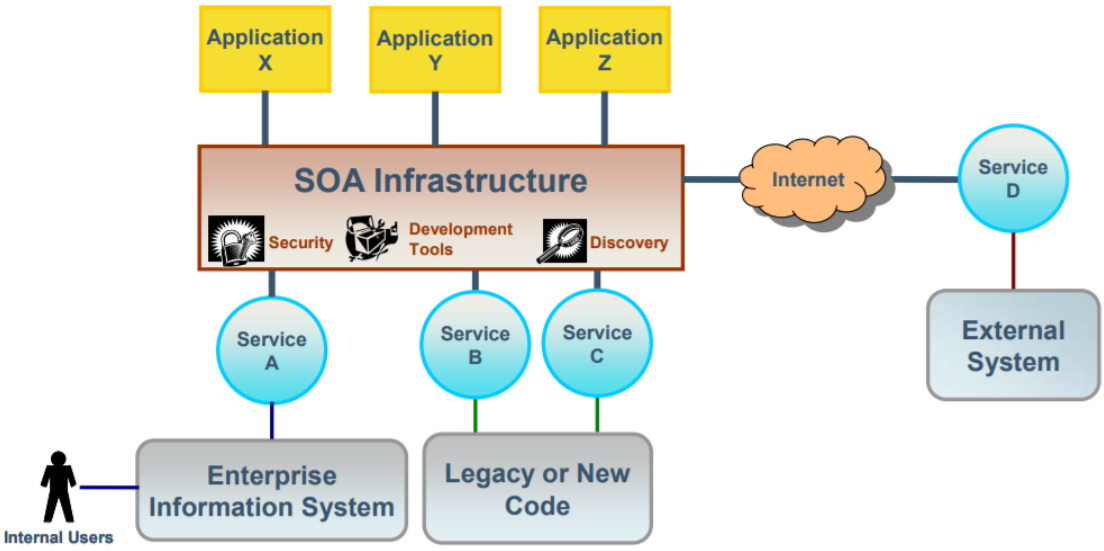


Figura 6. Componentes de un sistema basado en SOA.

Fuente: Kajko-Mattsson y Lewis (2014)

1.2.4. Arquitectura de Microservicios

El enfoque de microservicio ha surgido del uso en el mundo real de la arquitectura SOA, por lo tanto se debería pensar en los microservicios como un enfoque específico para SOA de la misma manera que XP o Scrum son enfoques específicos para el desarrollo de software ágil (Newman, 2015).

La arquitectura de microservicio es un enfoque para desarrollar una sola aplicación de software como un conjunto de pequeños servicios desplegados independientemente, cada uno de los cuales se ejecuta en su propio proceso y se

comunica con mecanismos ligeros, a menudo una API de recursos HTTP. Existe un mínimo de gestión centralizada de estos servicios, que pueden escribirse en diferentes lenguajes de programación y utilizar diferentes tecnologías de almacenamiento de datos (Lewis y Fowler, 2014), en la figura 7 se muestra un ejemplo de esta arquitectura.

1.2.4.1. Características y beneficios de los microservicios

En Amazon Web Services (2019) se indicó las siguientes características de los microservicios:

- Autónomos, entre los servicios no es necesario compartir su código e implementaciones permitiendo que se puedan desarrollar, implementar, operar y escalar de manera independiente.
- Especializados, los servicios son diseñados para resolver un problema específico.

Y los siguientes beneficios:

- Agilidad, para el desarrollo de cada microservicio es necesario la organización de equipos pequeños e independientes.
- Escalado flexible, los microservicios permiten que cada servicio se escale de forma independiente para satisfacer la demanda de la característica de la aplicación que respalda.
- Implementación sencilla
- Los microservicios permiten la integración y la entrega continua, facilitando probar nuevas ideas hasta lograr un correcto funcionamiento.
- Libertad tecnológica, esta arquitectura no sigue un enfoque de "diseño único", el equipo de desarrollo de cada microservicio puede elegir la mejor herramienta para resolver sus problemas.

- Resistencia, debido a que los microservicios son independientes si hay un error en un servicio la aplicación continúa funcionando dando de baja a solo el servicio que presenta errores.

En la figura 7, vemos una arquitectura basada en microservicios, esta se fundamenta en la división de módulos independientes denominados microservicios que se construyen para lograr funcionalidades de negocio muy concretas (autenticación, directorio, correo, CRUD de items, etc) y que se comunican entre si mediante mecanismos de interacción relativamente sencillos como RPC (Remote Procedure Communications) o RESTFul a un API, cada microservicio se despliega de manera independiente, pueden ser desarrollados en tecnologías diferentes, pero deben exponer sus funcionalidades a través de una API bien documentada y conocida (Roldán Martínez, Valderas Aranda, & Torres Bosh, 2018).

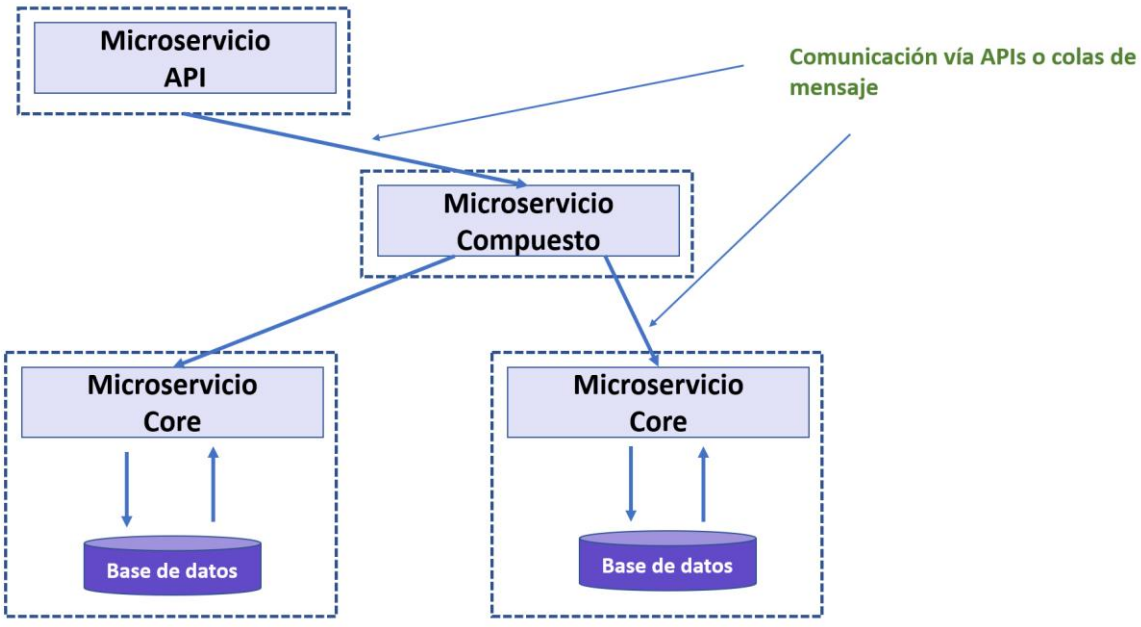


Figura 7. Arquitectura de microservicios de una aplicación.

Fuente: (Roldán Martínez et al., 2018)

1.2.4.2. Arquitectura monolítica en comparación con la arquitectura de microservicios

Con las arquitecturas monolíticas las aplicaciones al tener sus procesos estrechamente asociados y ejecutándose en un solo servicio limitan la modificación de los procesos o integración de nuevas características en la aplicación, además si un proceso de la aplicación experimenta un pico de demanda, se debe escalar toda la arquitectura sin tener la opción de escalar solo el proceso con mayor demanda.

Con una arquitectura de microservicios, una aplicación se crea con componentes independientes que ejecutan cada proceso de la aplicación como un servicio permitiendo de esta manera que cada servicio se puede actualizar, implementar y escalar para satisfacer la demanda de funciones específicas de una aplicación (Amazon Web Services, 2019).

1.2.4.3. Modelamiento de microservicios

Kharbuja (2016) presenta dos enfoques para modelar microservicios.

Se puede modelar microservicios utilizando casos de uso, este es una forma eficiente y elegante para capturar requisitos enfocándose en las preocupaciones que son valiosas para las partes interesadas.

Otro enfoque con el que se puede modelar microservicios es Domain-driven design (DDD), este respalda el modelado basado en la realidad del negocio como relevante para sus casos de uso. Describe áreas de problemas independientes como Contextos delimitados (cada contexto delimitado se correlaciona con un microservicio) y enfatiza un lenguaje común para hablar sobre estos problemas (Microsoft, 2019).

Existen tres partes básicas para implementar DDD:

1. Lengua ubicua
2. Diseño estratégico

3. Diseño táctico

Para el modelado de microservicios solo son relevantes las dos primeras partes (Kharbuja, 2016).

1.2.5. Composición de servicios

Componer servicios es la acción de establecer mecanismos que permitan la comunicación eficiente entre dos o más servicios que se ejecutan independientemente en ambientes distintos. La orquestación y coreografía están relacionados directamente con la composición de servicios.

1.2.5.1. Orquestación y coreografía

El término orquestación de servicios se refiere a la coordinación de múltiples servicios a través de un mediador centralizado, como un consumidor de servicios o un centro de integración, se llama orquestación debido a que al igual que una orquesta un número de músicos están tocando diferentes instrumentos y todos ellos son coordinados por el conductor central (Richards, 2016). Este concepto se ilustra en la figura 8.

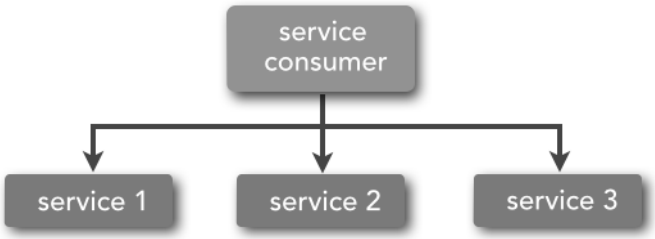


Figura 8. Orquestación de servicios.

Fuente: Richards (2016)

La coreografía de servicios se refiere a la coordinación de múltiples llamadas de servicio sin un mediador central. Con la coreografía de servicio, un servicio llama a otro servicio, que puede llamar a otro servicio y así sucesivamente, realizando lo que

también se conoce como encadenamiento de servicios (Richards, 2016). El diagrama de la figura 9 ilustra este concepto.

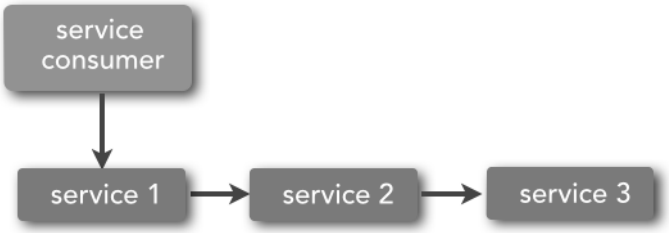


Figura 9. Coreografía de servicios.

Fuente: Richards (2016)

1.2.6. Contenedores

La virtualización nos permite dividir un servidor físico en hosts separados, cada uno de los cuales puede ejecutar cosas diferentes, esta virtualización tiene muchas desventajas, una de ellas es que requieren espacio y procesamiento adicional ya que las máquinas virtuales se ejecutan directamente en el hardware con su propio sistema operativo y su propio núcleo.

En linux existen los contenedores, cada contenedor es efectivamente un subárbol del conjunto árbol de proceso del sistema. Estos contenedores pueden tener recursos físicos asignados a ellos, algo que el núcleo maneja por nosotros. Esto significa que nuestro sistema operativo host podría ejecutar Ubuntu y nuestros contenedores CentOS, siempre que ambos puedan compartir el mismo núcleo. No solo nos beneficiamos de los recursos ahorrados al no necesitar un hipervisor, también ganamos en términos de retroalimentación. Los contenedores de Linux son mucho más rápidos de aprovisionar que las máquinas virtuales completas (Newman, 2015). En la figura 10 se muestra un diagrama que compara la virtualización estándar y la virtualización con contenedores.

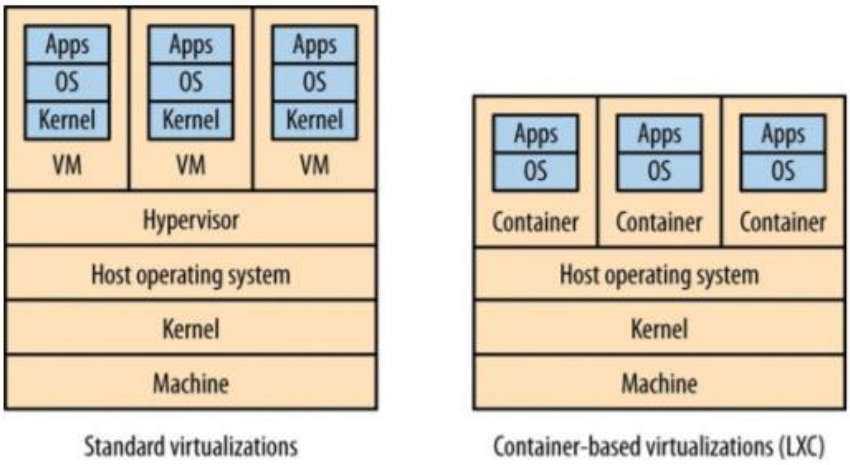


Figura 10. Comparación entre una virtualización estándar y una virtualización basada en contenedores.

Fuente: Newman (2015)

1.2.6. Historias clínicas

La historia clínica es una valoración del estado médico que incluye un relato escrito o verbal de la patología que se va a tratar del paciente junto con un episodio biográfico médico del paciente. Normalmente está ordenada cronológicamente, y debe ser clara, legible, precisa, tener máxima fidelidad con lo explicado por el paciente (Colegio Odontológico del Perú, 2019).

1.2.6.1. Estándar de historias clínicas electrónicas

Debido a la necesidad de almacenar y tener acceso a las historias clínicas mediante sistemas de software surge la Historia Clínica electrónica (HCE).

La información almacenada en las historias clínicas debe seguir un estándar para poder ser interpretada por cualquier sistema de software independientemente de su plataforma tecnológica o de su lenguaje de desarrollo.

Health Level Seven (HL7) es una organización que desarrolla estándares para minimizar las incompatibilidades entre sistemas de información en salud, permitiendo la interacción y el intercambio productivo de datos.

Clinical Document Architecture (CDA) – HL7

CDA (Clinical Document Architecture) fue reconocida como estándar ANSI en noviembre del 2000.

Este estándar está basado en XML para el marcaje de documentos que especifica la estructura y semántica de documentos clínicos para el propósito de facilitar su intercambio en un entorno de interoperabilidad de diferentes sistemas de software (Subcomité Técnico V3-CDA HL7 Spain, 2007).

Fast Healthcare Interoperability Resources (FHIR) – HL7

FHIR es un estándar que describe elementos y formatos de datos y una interfaz de programación de aplicaciones diseñado para permitir el intercambio de información para apoyar la provisión de atención médica en una amplia variedad de entornos. La especificación se basa y adapta prácticas RESTful modernas y ampliamente utilizadas para permitir la provisión de atención médica integrada en una amplia gama de equipos y organizaciones (HL7, 2019).

1.3 Definiciones Conceptuales

Acoplamiento:

Unir elementos de manera que ajusten perfectamente.

API:

Application Programing Interface – Interfaz de Programación de Aplicaciones. Conjunto de procedimientos y funciones para permitir el acceso de terceros a servicios creados.

Aplicación:

Una aplicación de software es un programa de computador diseñado para facilitar la realización de funciones específicas en un computador.

Contenedor:

Paquete de elementos de software que permiten ejecutar una aplicación en cualquier sistema operativo.

Disponibilidad:

Capacidad de sistema de software para ofrecer un servicio.

Escalabilidad:

Propiedad deseable de un sistema, proceso o una red, que indica su habilidad para reaccionar y adaptarse sin perder calidad para satisfacer una demanda más grande.

Interfaz de usuario:

Medio con que el usuario puede comunicarse con una máquina, equipo, computadora o dispositivo.

JSON:

JavaScript Object Notation - formato de texto sencillo para el intercambio de datos.

Máquina virtual:

Software que simula un sistema de computación y puede ejecutar programas como si fuese una computadora real.

Módulo:

Un módulo es una porción de un programa de ordenador.

Patrón de diseño:

Técnicas para resolver problemas comunes en el desarrollo de software.

Requerimiento:

Características y funcionalidades que debe tener el sistema de software.

Rest:

REpresentational State Transfer. arquitectura que se ejecuta sobre HTTP.

RestFul:

servicio web que implementa la arquitectura REST.

XML:

Lenguaje de marcado extensible, es una especificación de W3C.

1.4 Operacionalización de Variables

Variables	Definición de la Variable	Dimensión	Indicadores	Instrumento
Variable Independiente	Arquitectura de software basada en microservicios	Atributos de calidad de los Microservicios	Acoplamiento	Test de calidad
			Cohesión	
			Reusabilidad	
		Atributos de calidad de la Arquitectura	Funcionalidad	Test de caja negra
			Seguridad	
			Escalabilidad	Test de escalabilidad
Variable Dependiente	Disponibilidad de HCO	Desempeño	Tiempo de respuesta	Test de carga
			Rendimiento	
		Disponibilidad	Porcentaje de respuestas correctas	

1.5 Hipótesis

Una arquitectura de software basada en microservicios ayudará a mejorar la disponibilidad de las historias clínicas electrónicas odontológicas.

Capítulo II. Métodos y Materiales

2.1 Tipo de Investigación

La investigación es de tipo experimental tecnológico porque manipula directamente la variable independiente, la arquitectura de software basada en microservicios, para medir los efectos en la variable dependiente, la disponibilidad de historias clínicas electrónicas odontológicas.

2.2 Método de Investigación

El método de investigación es de campo exploratoria, ya que fue necesario la participación del investigador en el lugar donde se encuentra el objeto de estudio permitiendo la búsqueda y recolección de datos que permitan resolver la problemática.

2.3 Diseño de Contrastación

En la presente investigación se ha contrastado la hipótesis de la siguiente manera:

O1 ⟶ X ⟶ O2

Donde:

O1 = Disponibilidad de las historias clínicas electrónicas odontológicas antes de implementar la arquitectura de software basada en microservicios.

X = Arquitectura de software basada en microservicios

O2 = Disponibilidad de las historias clínicas electrónicas odontológicas después de implementar la arquitectura de software basada en microservicios.

2.4 Población, Muestra y Muestreo

Población

La población está conformada por las peticiones de consumo realizadas a la arquitectura de software basada en microservicios en un tiempo determinado.

Teniendo en cuenta para el 2020 existirán 971 cirujanos dentistas colegiados en el departamento de Lambayeque (IDEA FRI, 2016), esta sería la población total de peticiones en un determinado momento.

Muestra

La selección de la muestra ha sido de tipo no probabilística utilizando el muestreo por conveniencia, en la investigación de Ruelas Acero (2017) se utilizó 20 peticiones por segundo para evaluar una aplicación web implementada en un modelo de composición de microservicios, en esta investigación se utilizó también 20 peticiones por segundo.

2.5 Técnicas, Instrumentos, Equipos y Materiales de Recolección de Datos

2.5.1. Técnicas

Técnica de encuesta

Se utilizó esta técnica para analizar el estado actual de la gestión de Historias clínicas odontológicas en la provincia de Chiclayo.

Técnica de Observación

Se utilizó esta técnica para obtener información de seguimiento de la arquitectura de software basada en microservicios.

2.5.2. Instrumentos

Ficha de encuesta

Se empleo una ficha de encuesta para analizar el trabajo actual de los proveedores de servicios odontológicos respecto al manejo de Historias Clínicas Odontológicas en la provincia de Chiclayo, departamento de Lambayeque.

Pruebas de caja negra

Son pruebas funcionales en las que no se toma en cuenta el código sino se centran principalmente en las funcionalidades, para poder realizar estas pruebas se necesitan entradas con el fin de obtener una salida específica.

Las pruebas de caja negra tienen varias técnicas, en esta investigación se utilizará la técnica de partición de equivalencia, siendo una de las técnicas más eficientes permitiendo inspeccionar valores válidos e inválidos de entrada en el sistema.

Se ha usado esta prueba para evaluar la funcionalidad y seguridad de la arquitectura desarrollada en esta investigación, utilizando el instrumento:

Ficha de prueba de caja negra

Pruebas de escalabilidad

Las pruebas de escalabilidad permiten verificar la capacidad de una aplicación de escalar cualquiera de sus características no funcionales, como por ejemplo número de transacciones, carga que soporta.

Para determinar la capacidad de escalabilidad de la arquitectura desarrollada en esta investigación ante un incremento de demanda de peticiones de HCO se ha utilizado el instrumento:

Ficha de prueba de escalabilidad

Pruebas de carga

Esta prueba se realiza generalmente para observar el comportamiento de un sistema de software bajo una cantidad de peticiones esperada.

Para evaluar el desempeño y disponibilidad de la arquitectura de software diseñada para responder a las peticiones de HCO se utilizó el instrumento:

Ficha de prueba de carga

2.5.3. Equipos

Se ha utilizado el servicio en la nube de Microsoft Azure para desplegar la arquitectura propuesta y probar nuestra implementación, se ha configurado cuatro contenedores con las siguientes características:

Nombre del servicio: APP Services – Microsoft Azure

Velocidad Procesador: 1 Core 3,7 GHz

Memoria RAM: 1.75GB

Almacenamiento: 10GB

Región: South Central USS

Sistema Operativo: Windows

2.5.4. Materiales de recolección de datos

JMeter

JMeter es un proyecto de Apache que puede ser utilizado como una herramienta de prueba de carga para medir el rendimiento de aplicaciones que brindan servicios web. Se ha utilizado esta herramienta para realizar las pruebas de carga y pruebas de escalabilidad.

IBM Spss Statistics

SPSS es un programa estadístico informático muy usado en las ciencias sociales y aplicadas. Se ha utilizado esta herramienta para analizar los datos obtenidos de la encuesta aplicada para de analizar el estado actual de la gestión de HC odontológicas en la provincia de Chiclayo.

2.6 Procesamiento y Análisis de Datos

Atributos de calidad de servicios

Para la evaluación de la arquitectura basada en microservicios se utilizó los atributos de calidad empleados en los trabajos de Kharbuja (2016) y Ruelas Acero (2017): el acoplamiento, la cohesión y la reutilización.

El acoplamiento de un servicio, fuerza de la dependencia entre servicios en el sistema.

$$acoplamiento = \frac{n_c + n_p}{n_s}$$

Donde:

n_c: Número de servicios consumidores.

n_p: Número de servicios dependientes.

n_s: Número total de servicios.

La cohesión, fuerza de la relación entre las operaciones en un servicio.

$$cohesión = \frac{n_s}{M(s)}$$

Donde:

M(s): Conjunto de mensajes de las operaciones de un servicio.

n_s: Número total de servicios.

La reusabilidad se define como la capacidad de reutilización de un servicio.

$$reusabilidad = S_{consumidores}$$

Donde:

$S_{consumidores}$: Número de consumidores del servicio.

Disponibilidad de las HCO

Desempeño

Para la evaluación del desempeño de la arquitectura diseñada en esta investigación se utilizó los indicadores: Tiempo de respuesta y rendimiento.

Tiempo de respuesta, tiempo de respuesta promedio de la arquitectura de software, este indicador será calculado en ms.

$$tiempo\ de\ respuesta = tiempo\ final\ -\ tiempo\ inicial$$

Donde:

Tiempo final: Tiempo exacto después de realizada la petición.

Tiempo inicial: Tiempo inicial antes de realizar la petición.

Rendimiento, será el total de peticiones que se pueden realizar en un determinado espacio de tiempo, para esta investigación se determinara el total de peticiones que se pueden realizar en 60 000 ms, es decir 1 minuto.

$$rendimiento = \frac{peticiones\ realizadas}{tiempo}$$

Donde:

Peticiones realizadas: Peticiones realizadas en un tiempo determinado.

Tiempo: Tiempo establecido para determinar la cantidad de peticiones que se puede realizar.

Disponibilidad

Para la evaluación de la disponibilidad de historias clínicas se utilizó el indicador: porcentaje de respuestas correctas.

Este porcentaje será establecido teniendo en cuenta el número de peticiones realizadas a la arquitectura de software basada en microservicios y los errores que presente durante estas peticiones.

$$disponibilidad = \frac{peticiones\ correctas\ x\ 100}{total\ de\ peticiones}$$

Donde:

Peticiones correctas: Número de peticiones correctas.

Total de peticiones: Número total de peticiones realizadas.

Capítulo III. Resultados

3.1. Diagnóstico del estado actual de la gestión de HCE odontológicas en el departamento de Lambayeque

Para analizar el estado actual de la gestión de Historias Clínicas Electrónicas Odontológicas de la provincia de Chiclayo se realizó una encuesta a los especialistas cirujanos dentistas que trabajan como proveedores de servicios odontológicos (anexo 1).

Según IDEA FRI (2016) en el departamento de Lambayeque para el año 2020 existirán 971 cirujanos dentistas Colegiados, para el cálculo de la muestra a la que será aplicada la encuesta utilizamos la siguiente ecuación para poblaciones finitas.

$$n = \frac{z^2(pqN)}{Ne^2 + z^2(pq)}$$

Donde:

Z=nivel de Confianza

p=probabilidad a favor

q=probabilidad en contra

N=universo

e=error de estimación

n=tamaño de la muestra

Considerando los valores:

Z=95%

p=50%

q=50%

N=971

e=5%

n=?

Obteniendo como muestra un total de 83 médicos cirujanos a los que fue aplicada la encuesta. Se puede ver el resultado detallado de la encuesta en el anexo 12, se ha realizado el análisis de fiabilidad de los datos recolectados utilizando IBM SPSS Statics versión 25, obteniendo un resultado aceptable con valor 0,755 (George y Mallery, 2003).

Reliability Statistics

Cronbach's Alpha	N of Items
.755	9

Los resultados más relevantes de esta encuesta corresponden a:

1. Almacenamiento de HCO

   En la tabla 1 podemos ver que el 34.9% de los encuestados almacena sus HCO siempre, el 41% lo almacena a veces y el 24.1% nunca almacena sus HCO, ya sea en papel o en formato digital.

   Tabla 1
   Resultados de la encuesta sobre almacenamiento de HCO

Respuesta	Frecuencia absoluta	Frecuencia relativa
Nunca	20	24.1
A veces	34	41.0
Siempre	29	34.9
Total	83	100.0

   Nota. Fuente: Resultados de encuesta del anexo 12.

   En la tabla 2 podemos revisar que de los cirujanos dentistas encuestados solo el 14.5% almacena sus HCO digitalmente, es decir utiliza un software informático, procesador de texto u hoja de cálculo para almacenar digitalmente sus HCO.

   Tabla 2
   Resultados de la encuesta sobre almacenamiento de HCO en formato digital

Respuesta	Frecuencia absoluta	Frecuencia relativa
Nunca	63	75.9
A veces	8	9.6
Siempre	12	14.5
Total	83	100.0

   Nota. Fuente: Resultados de encuesta del anexo 12.

2. Necesidad de consultar antecedente de atención odontológica

   El 88.0% de los encuestados indico que a veces se presentan algunos inconvenientes por no contar con la información de HCO de sus pacientes almacenada para poder realizar consultas, en la tabla 3 podemos revisar el detalle.

Tabla 3
Resultados de la encuesta sobre inconvenientes por no contar con registro de HCO

Respuesta	Frecuencia absoluta	Frecuencia relativa
Nunca	5	6.0
A veces	73	88.0
Siempre	5	6.0
Total	83	100.0

Nota. Fuente: Resultados de encuesta del anexo 12.

3. Compartir información de HCO

   En la tabla 4 podemos ver que el 55.4% de los encuestados indico que estaría dispuesto a compartir la información de las HCO de sus pacientes siempre con otros proveedores de servicios odontológicos, el 30.1% indico que a veces podría compartir la información de sus pacientes y el 14.5% indico que nunca estaría dispuesto a compartir la información de HCO.

Tabla 4
Resultados de la encuesta sobre compartir HCO

Respuesta	Frecuencia absoluta	Frecuencia relativa
Nunca	12	14.5
A veces	25	30.1
Siempre	46	55.4
Total	83	100.0

Nota. Fuente: Resultados de encuesta del anexo 12.

4. Utilidad de un repositorio digital para almacenar HCO

   En la tabla 5 podemos ver que el 78.3% de los cirujanos dentistas encuestados considera de utilidad que las historias clínicas odontológicas se encuentren almacenadas en un

repositorio digital seguro y confiable para ser consultadas por el especialista autorizado en el momento oportuno.

Tabla 5
Resultados de la encuesta sobre almacenamiento de HCO para consultas

Respuesta	Frecuencia absoluta	Frecuencia relativa
Nunca	7	8.4
A veces	11	13.3
Siempre	65	78.3
Total	83	100.0

Nota. Fuente: Resultados de encuesta del anexo 12.

Podemos revisar el detalle de respuesta de cada una de las preguntas de la encuesta en el anexo 12.

3.2. Contexto del Proyecto

En el presente proyecto se abordó el problema de la no disponibilidad de las Historias Clínicas Odontológicas (HCO) por parte de los proveedores de servicios odontológicos, esta no disponibilidad en la mayoría de casos se debe a que no todos cuentan con un sistema de información de gestión de HCO. Como solución a este problema se propuso una arquitectura de software para estructurar un sistema que permita la disponibilidad de las HCO. Este sistema soporta una HCO como un servicio en la nube que podrá ser compartida con los diferentes proveedores se servicios odontológicos, se utilizó el modelo conceptual de HCO utilizado por el Colegio Odontológico del Perú como componente principal para el manejo de la información odontológica de los pacientes. En la figura 11 se presenta el contexto del proyecto.

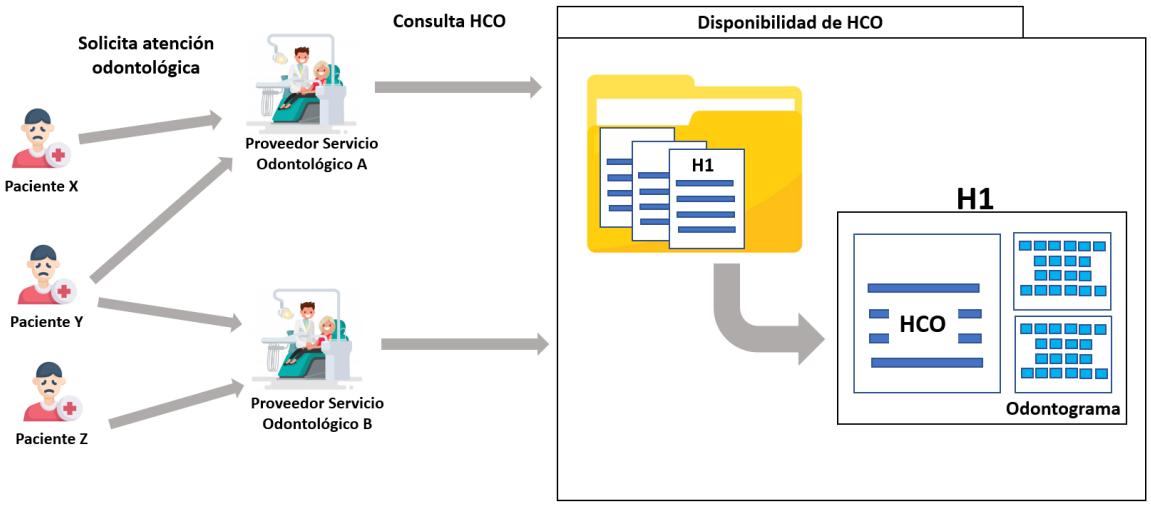


Figura 11. Contexto del proyecto.

Fuente: Elaboración propia

Con la implementación de este sistema cada vez que un paciente solicita una atención odontológica el proveedor de este servicio que tiene autorización para el uso del sistema puede consultar el historial odontológico del paciente.

Para la elaboración de este sistema de software se ha aplicado métodos y técnicas en cada una de las etapas del ciclo de desarrollo de la arquitectura de software, estas etapas se muestran en la figura 12.

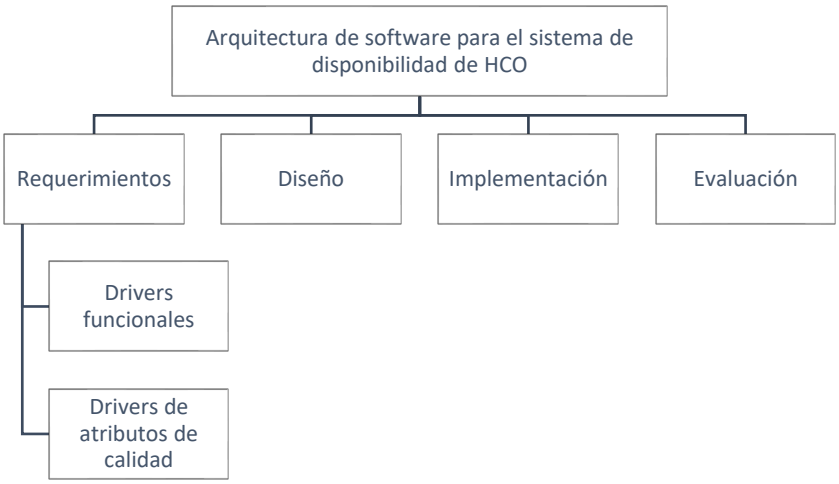


Figura 12. Etapas de desarrollo de la arquitectura.

Fuente: Elaboración propia

3.3. Requerimientos arquitectónicos del sistema de software para la disponibilidad de HCO

La arquitectura propuesta satisface los requerimientos fundamentales para permitir la disponibilidad de las HCO, en este apartado mostramos una visión general del software a través de los requerimientos funcionales y de requerimientos de atributos de calidad.

3.3.1. Drivers funcionales

Los requisitos de la arquitectura se identificaron mediante las siguientes actividades:

Revisión de la literatura referentes a sistema de historias clínicas

Revisión de la normatividad peruana referente a la historia clínica.

Encuesta a especialistas cirujanos dentistas y revisión de documentos del colegio odontológico del Perú referentes a la historia clínica odontológica.

Requisitos identificados en la literatura

Existen diferentes trabajos de interoperabilidad entre sistemas de software de la Electronic Health Record (EHR), Historia Clínica Electrónica, estos trabajos identifican requisitos comunes para el diseño de un sistema de gestión de EHR, en la tabla 6 se muestran esos requisitos comunes para soluciones EHR identificados y consolidados en el trabajo de Villa Benavides (2017).

Tabla 6

Requisitos de la EHR

Requisito	Atributo De Calidad
Uso de estándares de salud. Intercambio de información clínica administrativa y de afiliación del paciente.	Interoperabilidad
Información disponible 7 x 24 x 365. Cumplimiento de los acuerdos de niveles de servicios. Acceso a registro médicos, previa validación de privilegios.	Disponibilidad
Ajuste de funcionalidades a los cambios normativos. Incluir nuevas funcionalidades sin afectar el sistema.	Mantenibilidad

Confidencialidad de información sensible. Autenticación de usuarios. Proteger la integridad de la historia clínica.	Seguridad
Niveles de respuesta óptimos. Consulta oportuna de los registros médicos.	Desempeño
Soportar picos de transacciones. Adaptarse a diferentes entornos.	Adaptabilidad
Facilidad de uso.	Usabilidad
Sistema autosostenible. Reducir costos del equipo de TI. Mejorar relación costo/beneficio.	Sostenibilidad
Cumplimiento de las leyes nacionales e internacionales. Cumplimiento de los acuerdos interinstitucionales.	Funcionalidad

Nota. Fuente: Villa Benavides (2017)

Requisitos identificados en la normatividad peruana

Para el diseño de la arquitectura propuesta se realizó una revisión de la normatividad peruana, específicamente de la NTS 150-MINSA-2019/DGIESP (Ministerio de Salud Perú, 2019), Norma Técnica de salud para el uso del Odontograma y la Ley 30024 Ley que crea el Registro Nacional de Historias Clínicas Electrónicas.

En la tabla 7 se presenta los requisitos identificados en la normatividad asociados con los atributos de calidad que la arquitectura soporta.

Tabla 7

Requisitos asociados a la normatividad peruana

Norma	Requisito	Atributo De Calidad
Ley 30024 Artículo 4	Estandarizar los datos, la información clínica de las Historias Clínicas Electrónicas (HCE) y las características y funcionalidades de los sistemas de Información de historias clínicas electrónicas.	Funcionalidad

Norma NTS 150-MINSA-2019/DGIESP	Estandarización del uso del odontograma.	Funcionalidad
Ley 30024 Artículo 4	Asegurar la disponibilidad de la información clínica contenida en las HCE para el paciente, representante legal y profesional de salud autorizado.	Disponibilidad
Ley 30024 Artículo 4	Intercambio de información clínica para asegurar la continuidad de la atención de salud del paciente en los establecimientos de salud y servicios médicos de apoyo.	Compatibilidad
Ley 30024 Artículo 7	Confidencialidad de las historias clínicas.	Seguridad
Ley 30024 Artículo 8	Autenticación de identidad de las personas para acceder al Registro nacional de HCE	Seguridad

Nota. Fuente: Ley 30024 (Ministerio de Salud Perú, 2019)

Requisitos de Negocio

Para identificar los requisitos del negocio se realizó una revisión de los documentos públicos del Colegio Odontológico del Perú y de la investigación de Villa Benavides (2017), además se tuvo en cuenta el análisis de la encuesta realizada a los especialistas cirujanos dentistas de la ciudad de Chiclayo – Perú. Se tienen en cuenta como requisitos prioritarios aquellos que están relacionados con la funcionalidad del sistema de software, la disponibilidad de las HCO, la facilidad de manejo del sistema de software, seguridad e integridad de los datos.

En la tabla 8 se presentan los requisitos consolidados de negocio identificados asociados con los atributos de calidad que la arquitectura soporta.

Tabla 8

Requisitos de negocio

Id	Requisito	Atributo De Calidad
R01	El sistema debe permitir la gestión de pacientes, HCO, odontogramas, proveedores de servicios odontológicos.	Funcionalidad
R02	El sistema debe permitir la disponibilidad de la HCO entre los proveedores de servicios odontológicos.	Disponibilidad
R03	El sistema debe de utilizar el formato de Historia Clínica y odontograma estándar del Colegio Odontológico del Perú.	Disponibilidad
R04	El acceso a los datos de los pacientes, HCO y odontograma debe utilizar mecanismos de autenticación y autorización.	Seguridad
R05	El sistema debe permitir incorporar nuevas funcionalidades.	Modificabilidad
R06	La disponibilidad del sistema debe ser 24 x 7	Disponibilidad
R07	El sistema debe permitir al menos 20 usuarios concurrentes en ciertos periodos de tiempo.	Confiabilidad
R08	El sistema debe responder a las solicitudes de consulta de HCO en un tiempo promedio de 2 segundos y nunca exceder los 5 segundos.	Desempeño
R09	El sistema deberá funcionar en diferentes plataformas y desde diferentes dispositivos.	Portabilidad

R10	El sistema podrá aumentar su capacidad de procesamiento cuando la demanda de la disponibilidad de HCO aumente.	Escalabilidad

Nota. Fuente: Elaboración propia

Después del consolidado de los requerimientos funcionales que especifican las interacciones entre los usuarios y el sistema serán especificados usando la técnica de casos de uso (Cockburn, 2001). En la figura 13 se presenta el modelo de casos de uso del sistema usando un diagrama UML.

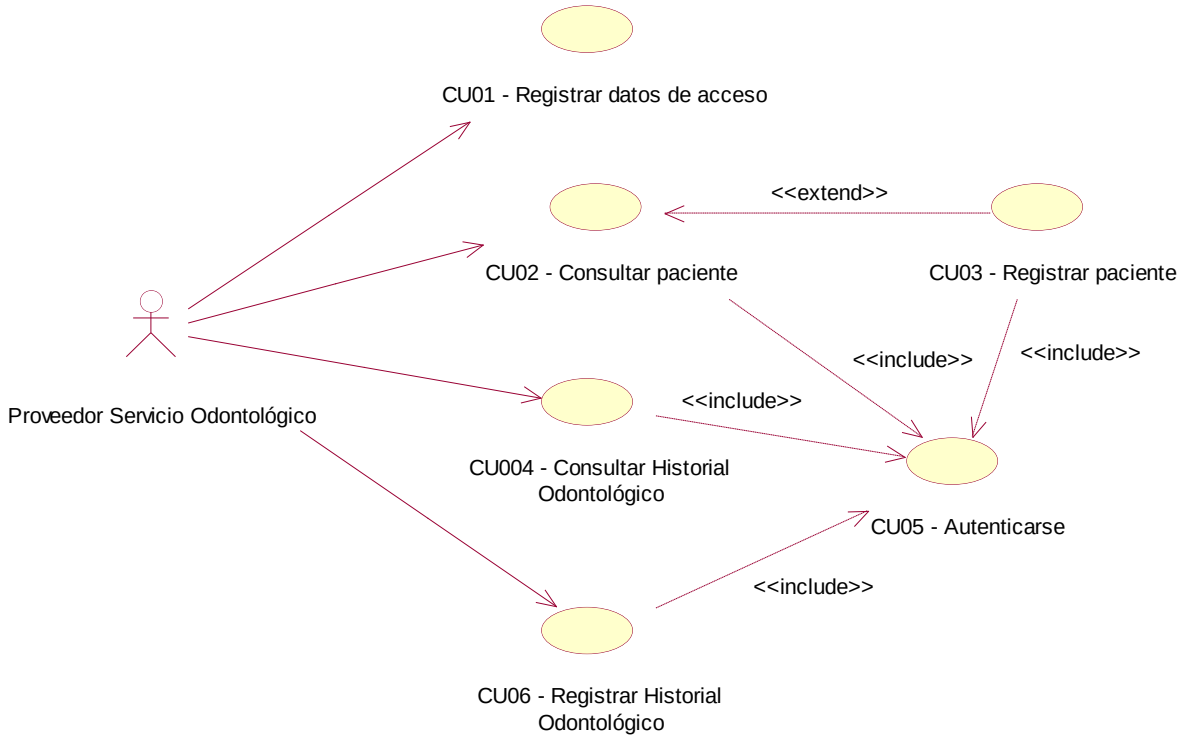


Figura 13. Diagrama de casos de uso.

Fuente: Elaboración propia

3.3.2. Drivers de atributos de calidad

Se identifican los atributos de calidad relevantes para el sistema.

En la tabla 9 se muestra los atributos de calidad relevantes que se han tenido en cuenta para diseñar la arquitectura de software que permita la disponibilidad de HCO basado en la terminología de la norma ISO/IEC25010.

Tabla 9

Drivers de arquitectura para la disponibilidad de las HCO

Driver	Justificación
FUNCIONALIDAD	La arquitectura debe permitir la gestión de HCO, odontogramas, paciente y proveedores de servicios odontológicos según formatos establecidos por el COP.
DISPONIBILIDAD	La arquitectura debe permitir la disponibilidad de las HCO y odontogramas de cada paciente.
SEGURIDAD	El acceso a los datos de los pacientes, HCO y odontograma debe utilizar mecanismos de autenticación y autorización.
MANTENIBILIDAD	La arquitectura debe soportar modificaciones y permitir agregar nuevas funcionalidades con un bajo impacto en el funcionamiento del sistema.
ESCALABILIDAD	La arquitectura debe permitir aumentar su capacidad de procesamiento cuando la demanda de HCO aumente.
DESEMPEÑO	Se debe asegurar la disponibilidad de la HCO de un paciente con un tiempo óptimo en un determinado momento.

Nota. Fuente: Elaboración propia

3.4. Diseño de la arquitectura de software basado en microservicios para mejorar la disponibilidad de las HCO

Para permitir la disponibilidad de las HCO teniendo en cuenta los drivers identificados se puede usar el patrón de arquitectura SOA, pero se ha descartado este patrón por su complejidad, esfuerzo, costo requerido y porque el enfoque de Microservicios presenta ventajas sobre este patrón de arquitectura SOA.

Existen métodos y guías para la definición de la arquitectura de software, muchos de los cuales se centran en los requisitos funcionales, en la investigación de Ruelas Acero (2017) se consideró los requisitos funcionales para el diseño de su arquitectura, de igual manera en esta investigación se ha considerado los requisitos funcionales para del diseño de la arquitectura de software basado en el enfoque de Microservicios.

Descomposición en Microservicios

Utilizando el diagrama de casos de uso de la figura 13 se identificó los microservicios usando la descomposición basada en sustantivos, respetando el principio de Responsabilidad individual (SRP, Single Responsability Principle) de Martin, Robert C. (2003). En la tabla 10 se presenta los microservicios identificados agrupando funcionalidades relacionadas con una entidad (sustantivo) en particular.

Se ha separado la Historia Clínica Odontológica del Odontograma en dos servicios diferentes, esto para dar la posibilidad de utilizar estos recursos de manera independiente o de manera compuesta a la vez.

Tabla 10

Descomposición en Microservicios basada en sustantivos

Microservicio	Tareas (Responsabilidad Individual)	Entidad De Datos
	Registrar datos de acceso	

Proveedor de Servicio Odontológico	Actualizar datos de perfil Actualizar password de acceso Autenticarse y obtener token Recuperar datos de acceso	Proveedor de Servicio Odontológico
Paciente	Registrar datos de paciente Actualizar datos de paciente Obtener datos de paciente por su documento	Paciente
Historia Clínica Odontológica	Registrar historial Odontológico Actualizar historial Odontológico Obtener HCO por documento de paciente Obtener HCO por código de identificación	Historia Clínica Odontológico
Odontograma	Registrar Odontograma Actualizar Odontograma Consultar Odontograma por identificador de HCO Obtener tratamientos Odontológicos	Odontograma

Nota. Fuente: Elaboración propia

Comunicación entre Microservicios

Para poder realizar peticiones a cada uno de los microservicios identificados estos tienen un punto final público (endpoint) o rutas finales del recurso al que se puede acceder realizando una petición utilizando el protocolo HTTP REST (Representational State Transfer), utilizando los endpoint y el protocolo HTTP REST se ha construido una API para cada

microservicio, en la tabla 11 se muestran los microservicios con las rutas finales del recurso y el tipo de petición.

Tabla 11

Microservicios y tareas del modelo propuesto

Microservicio	Tareas	Tipo de Petición	Rutas Finales
Proveedor de Servicio Odontológico	Registrar datos de acceso	POST	/ProveedorServicioOdontologico
	Actualizar datos de perfil	PUT	/ProveedorServicioOdontologico/{id}
	Actualizar password de acceso	PUT	/ProveedorServicioOdontologico/CambiarPassword/{id}
	Autenticarse y obtener token	POST	/Security/Login
	Recuperar datos de acceso	POST	/Security/RecuperarPassword
Paciente	Registrar datos de paciente	POST	/Paciente
	Actualizar datos de paciente	PUT	/Paciente/{id}
	Obtener datos de paciente por su documento	GET	/Paciente/getByDocumento/{documento}
Historia Clínica Odontológico	Registrar historial Odontológico	POST	/Historia
	Actualizar historial Odontológico	PUT	/Historia/{id}
	Obtener HCO por documento de paciente	GET	/Historia/getByPacienteDocumento/{documento}
	Obtener HCO por código de identificación	GET	/Historia/ id}
Odontograma	Registrar Odontograma	POST	/Odontograma

Actualizar Odontograma	PUT	/Odontograma/{id}
Consultar Odontograma por identificador de HCO	GET	/Odontograma/getByHistoriaId/{id}
Obtener tratamientos Odontológicos	GET	/DefinicionOperativa/getByNombre/{nombre}

Nota. Fuente: Elaboración propia

Los códigos de estado HTTP devueltos por las API cada vez que se realice una petición serán los ofrecidos por el estándar RFC 2616 que se detalle en la tabla 12.

Tabla 12

Estándar RFC 2616

Código	Descripción
1XXX	Respuestas informativas
2XXX	Peticiones correctas
3XXX	Redirecciones
4XXX	Errores del cliente
5XXX	Errores del Servidor

Nota. Fuente: Estándar RFC 2616

La comunicación entre los clientes y los microservicios se realizarán mediante una API Gateway, se ha optado por el uso de un API Gateway ya que permite encapsular la estructura interna la arquitectura de software, se desacopla los clientes de los microservicios, se mantiene segura la arquitectura ya que no se exponen todos los microservicios. Se puede aplicar además el patrón “back-end para front-end” (BFF) y agregar otros API Gateway proporcionando una puerta de enlace distinta a cada tipo de aplicación cliente.

En la figura 14 se muestra la arquitectura lógica de modelo propuesto, está conformado de 5 principales componentes lógicos: Microservicio Proveedor Servicio Odontológico, Paciente, Historia Clínica Odontológica, Odontograma y Api Gateway.

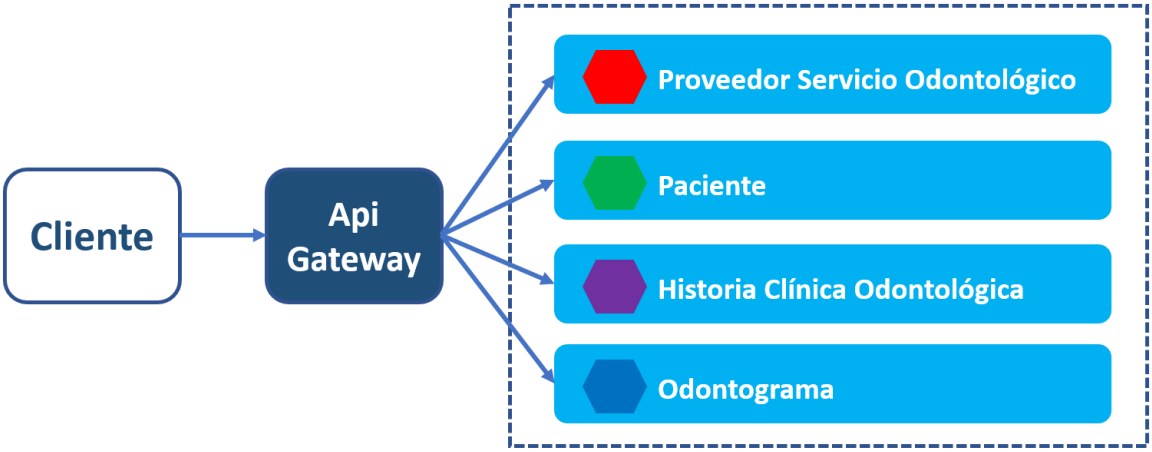


Figura 14. Arquitectura lógica del modelo de composición de microservicios.

Fuente: Elaboración propia

Se realiza la comunicación segura entre los microservicios el API Gateway y el cliente utilizando Token de autenticación. Los tokens son almacenados en el cliente, no hay información de estado permitiendo que la arquitectura sea totalmente escalable y evitando ataques CSRF (Cross-Site Request Forgery). Se trabajará la autenticación en el Microservicio Proveedor de Servicio Odontológico.

El cliente de la arquitectura de software enviará y recibirá las peticiones response y request en formato JSON.

Cada microservicio tendrá responsabilidad sobre su base de datos, el microservicio será el único componente de la arquitectura de software que realiza consultas de inserción, actualización y selección en la base de datos para exponerlos a través de su API REST. Cada microservicio asegura la integridad de sus datos almacenados.

Cada Componente lógico que conforma la arquitectura de software estará dentro de un contenedor, permitiendo de esta manera la escalabilidad automática.

En la figura 15 se muestra la arquitectura física de modelo propuesto y en la tabla 13 se describe cada uno de los elementos que la componen.

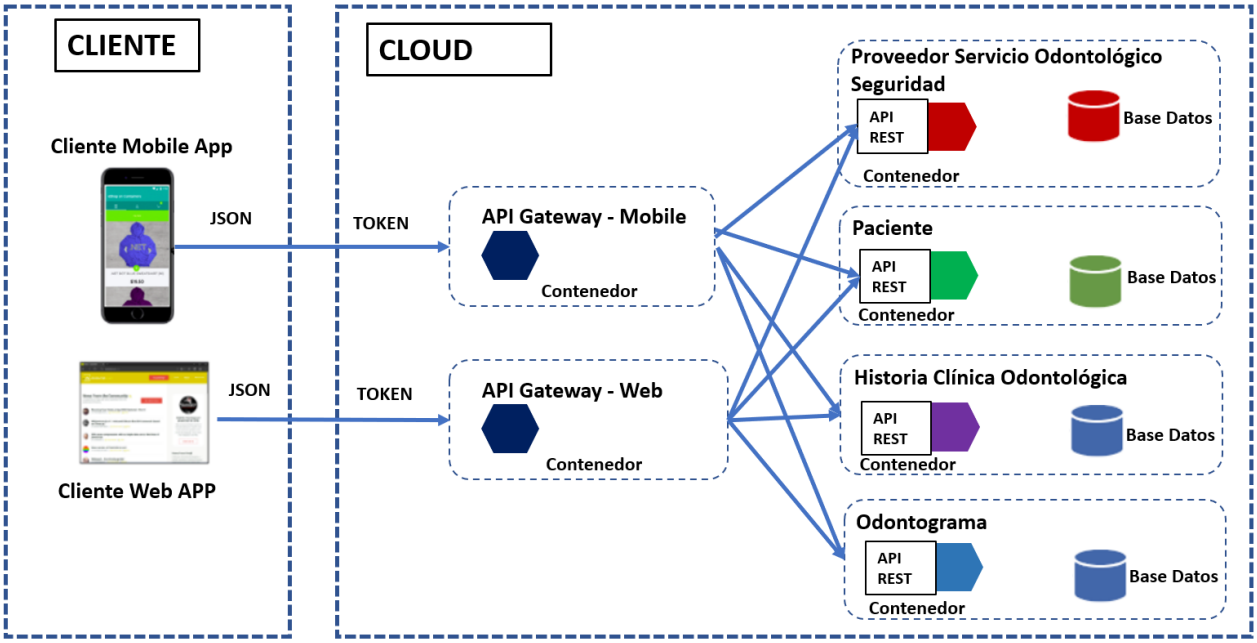


Figura 15. Arquitectura física del modelo de composición de microservicios.

Fuente: Elaboración propia

Tabla 13

Elementos de la arquitectura de Software

Elemento	Responsabilidad	Propiedades
API Gateway	La puerta de enlace proporciona un único punto de conexión o dirección URL para las aplicaciones cliente, a continuación, asigna internamente las solicitudes a los microservicios. En este API Gateway tiene la funcionalidad de balanceador de carga.	Solicitudes HTTP

Cliente	Los clientes son consumidores de los recursos brindados por parte de un microservicio, estos deben autenticarse para generar un Token de comunicación, este token debe enviarse en cada una de sus peticiones Response y Request a la API Gateway.	- Peticiones Response en formato JSON - Peticiones Request en formato JSON
Token	El cliente se autentica en la arquitectura a través de la cuenta de acceso del Proveedor de Servicios Odontológico si los datos son correctos la arquitectura le devuelve un token único. Cada petición HTTP que haga el cliente va acompañada de un Token	Componente de Software
Microservicio	Servicio independiente que se comunica a través de una API REST y que desempeña funcionalidades específicas para satisfacer los requerimientos funcionales del negocio.	Protocolo REST para las llamadas a las API
Base de datos	El microservicio será el único componente de la arquitectura de software que realiza consultas de inserción, actualización y selección en la base de datos para exponerlos a través de su API REST.	Sistema de Gestión de base de datos (SGBD)
Contenedor	Contenedor que contiene cada uno de los componentes de la arquitectura.	Bibliotecas, dependencias y archivos necesarios para ejecutar los

componentes de la arquitectura

Nota. Fuente: Elaboración propia

3.5. Implementación de la arquitectura de Software para la disponibilidad de HCO

Se ha desarrollado un prototipo de implementación de cada uno de los componentes que conforman la arquitectura de Software propuesta, en la figura 16 se muestra la arquitectura de Software física teniendo en cuenta los lenguajes de programación, librerías, frameworks, sistema de gestión de base de datos utilizados para la implementación del prototipo.

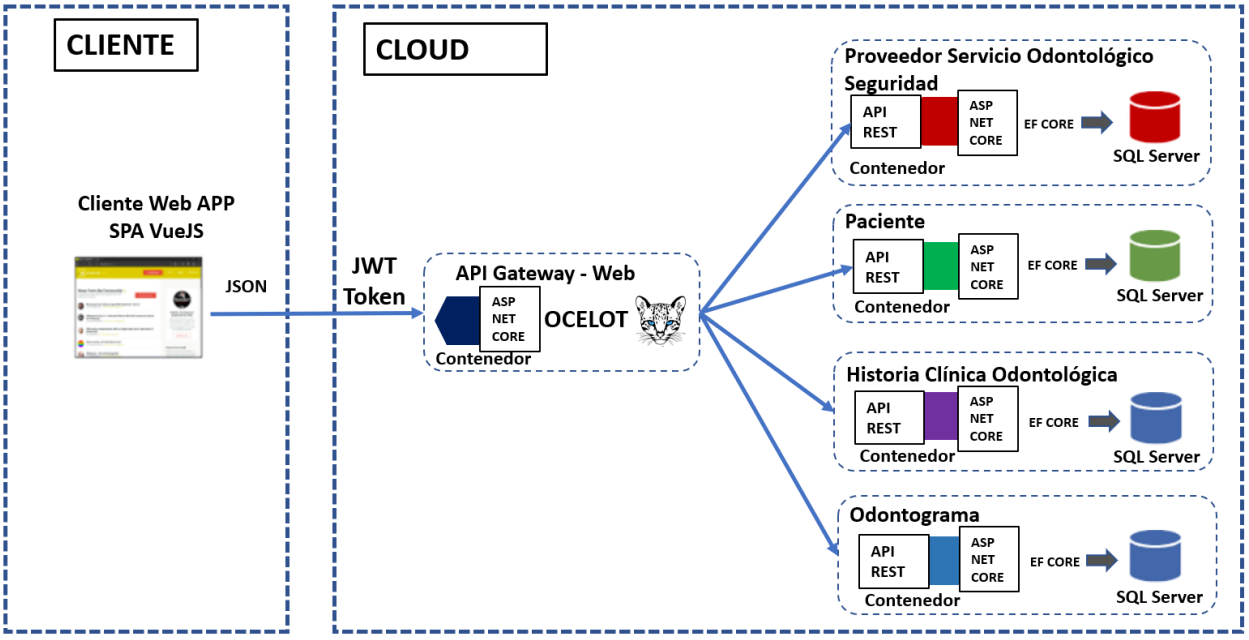


Figura 16. Arquitectura física del prototipo de implementación de la arquitectura de Software.

Fuente: Elaboración propia

Se ha implementado el prototipo para evaluar la arquitectura de software considerando solo necesario crear un cliente web y un solo API Gateway de puerta de enlace para la comunicación con los microservicios y el cliente web.

En la tabla 14 se describe cada uno de los elementos que componen el prototipo, los frameworks, librerías y otras tecnologías seleccionadas para implementar la arquitectura se han seleccionado teniendo en cuenta la experiencia de desarrollo de software del autor de esta investigación.

Se ha seleccionado el Sistema manejador de base de datos relacional SQL Server teniendo en cuenta la investigación de Diaz Montejo, Bellon Cely y González Sanabria (2015) donde determinan que un Sistema de Gestión de Base de Datos Relacional (SGBDR) muestra mejores beneficios para almacenar Historias clínicas.

Tabla 14

Elementos del prototipo de la arquitectura de Software

Elemento	Responsabilidad	Propiedades
API Gateway Web	La puerta de enlace proporciona un único punto de conexión o dirección URL para que las aplicaciones clientes puedan realizar peticiones a los servicios. Esta puerta de enlace realiza también la tarea de balanceador de carga.	- Framework ASP Net Core 3.1 - Paquete Ocelot
Cliente Web App **SPA VueJS**	Para poder realizar pruebas a la arquitectura de Software propuesta mediante peticiones request se ha desarrollado un cliente web Single Page Application (SPA).	- Framework VueJS - Peticiones request JSON y response JSON utilizando la librería Axios - Librería Vuetify
Token	El cliente se autentica en la arquitectura a través de la cuenta de acceso del Proveedor de Servicios Odontológico si los datos son correctos la	JSON Web Token (JWT)

	arquitectura le devuelve un token único utilizando la librería JWT, después de autenticarse cada petición HTTP que realice debe contener en su cabecera el token.	
Microservicio	Cada microservicio maneja información según los requerimientos funcionales del negocio, expone estas funcionalidades a través de una Api de recursos HTTP.	- Framework ASP Net Core 3.1 - Framework Entity Framework Core
Base de datos	Cada microservicio gestiona su propia información, lo almacena y gestiona en su propia base de datos.	Sistema de gestión de base de datos relacional SQL Server
Contenedor	Cada microservicio y el API Gateway se implementarán en contenedores diferentes, esto permite la escalabilidad independiente de cada Microservicio aplicando políticas de escalabilidad en periodos de alta demanda.	Procesador APP Service Microsoft Azure - 1 Core 3,7 GHz - 1.75Gb RAM - 10GB Almacenamiento - Región South Central USS

Nota. Fuente: Elaboración propia

3.5.1. Implementación de los Microservicios

Cada Microservicio administra su propio repositorio de base de datos y desempeña funcionalidades específicas para satisfacer los requerimientos funcionales del Negocio, el

prototipo desarrollado tiene 4 microservicios internos que se publican a través de la API Gateway, en la figura 17 se muestra la estructura de la solución.

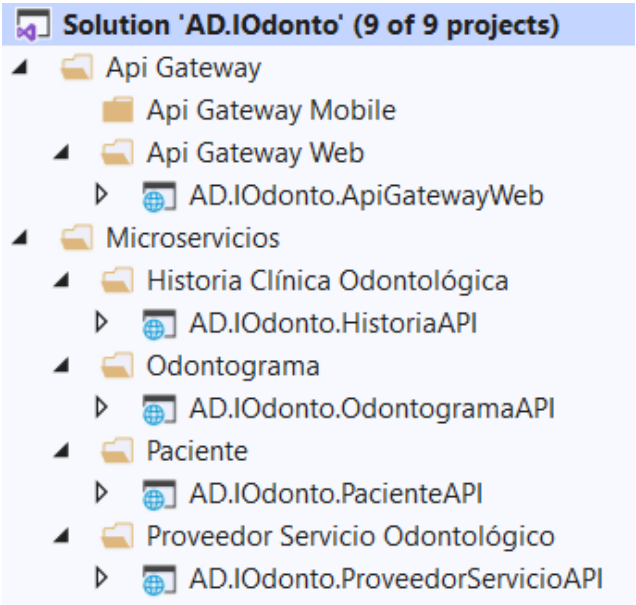


Figura 17. Estructura de la solución: Microservicios y el API Gateway.

Fuente: Elaboración propia

Cada microservicio expone su API Rest, para crear la API de cada microservicio se ha utilizado el framework ASP Net Core y para el acceso a datos almacenados en la base de datos SQL Server se ha utilizado el Framework Entity Framework Core. A continuación, se detalle la estructura de cada microservicio.

Microservicio Proveedor Servicio Odontológico

En este Microservicio se han expuesto dos API Rest, una de ellas es para gestionar la seguridad de la arquitectura y la otra API Rest para gestionar los recursos correspondientes a los proveedores de servicios odontológicos. En la figura 18 podemos ver el detalle de la estructura de la implementación del microservicio Proveedor de Servicio Odontológico en ASP Net Core.

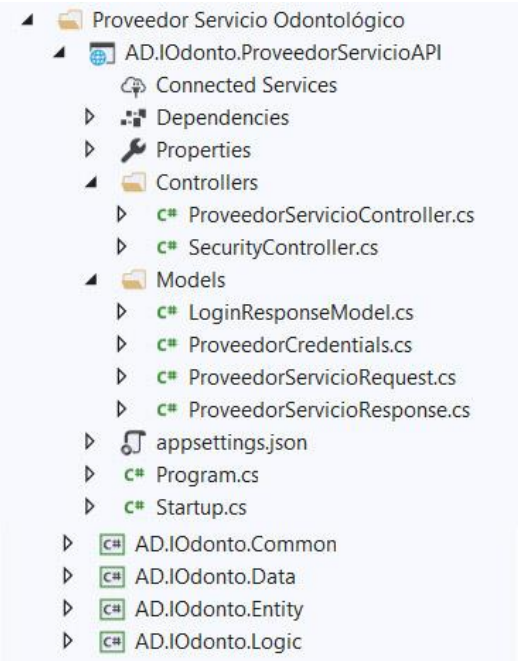


Figura 18. Estructura de la implementación del microservicio Proveedor Servicio Odontológico.

Fuente: Elaboración propia

Después de realizar el análisis de los requerimientos funcionales se ha obtenido el esquema de datos que debe de administrar este microservicio y se ha diseñado una base de datos independiente para almacenar la información específica de este microservicio, en la figura 19 se muestra la base de datos diseñada en SQL Server.

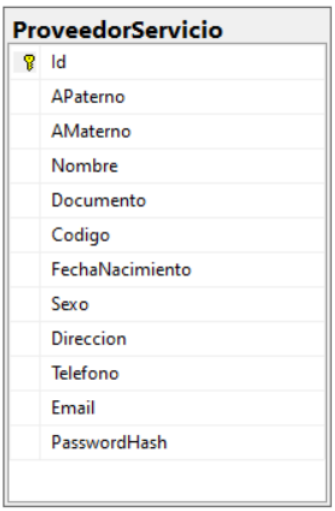


Figura 19. Estructura de la base de datos del microservicio Proveedor Servicio Odontológico.

Fuente: Elaboración propia

Esta base de datos es la que se ha considerado para el desarrollo del prototipo básico que implementa cada componente de la arquitectura de software diseñada en esta investigación Este microservicio cumple con las funcionalidades específicas que se le han asignado en la etapa de desarrollo de la arquitectura, en la figura 20 se muestra las rutas finales del recurso y el tipo de petición de cada API Rest. En el anexo 3 se tiene un detalle de la información de entrada (request), salida (response) y parámetros de cada una de las rutas finales de este microservicio.

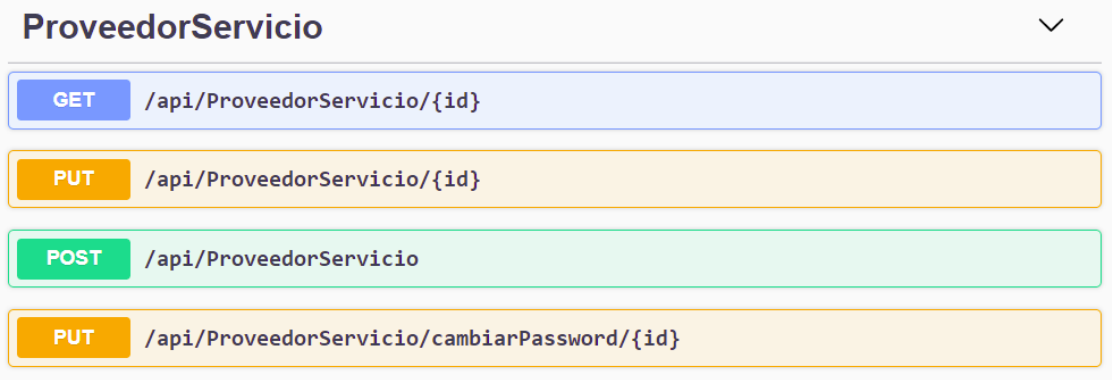


Figura 20. Rutas finales del microservicio Proveedor Servicio Odontológico.

Fuente: Elaboración propia

Las únicas rutas finales a las que se puede acceder sin autenticarse son las rutas "/ProveedorServicio" en la petición POST ya que mediante esta ruta se puede registrar los datos de acceso de un nuevo proveedor de servicio odontológico y la ruta "/Security/Login" ruta que espera las credenciales de acceso para autenticar al usuario y generarle un Token; todas las demás rutas finales requieren que se envíe el token de autenticación en la cabecera de la petición HTTP.

Microservicio Paciente

En este Microservicio se han expuesto una API Rest, esta permite gestionar los recursos correspondientes de los pacientes. En la figura 21 podemos ver el detalle de la estructura de la implementación del microservicio Paciente en ASP Net Core.

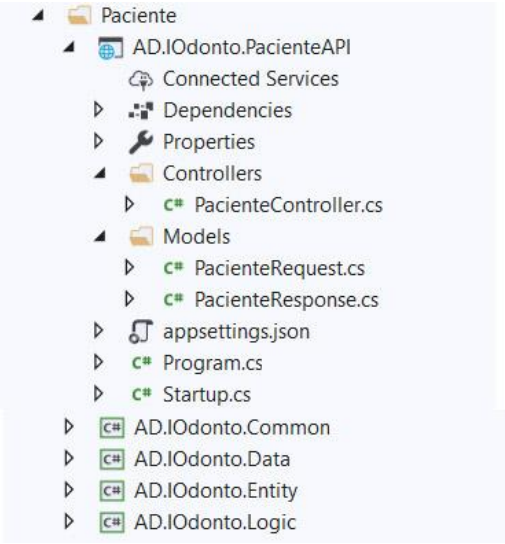


Figura 21. Estructura de la implementación del microservicio Paciente.

Fuente: Elaboración propia

Después de realizar el análisis de los requerimientos funcionales y los datos necesarios para la historia clínica odontológica del anexo 1 se ha obtenido el esquema de datos que debe de administrar este microservicio y se ha diseñado una base de datos independiente para almacenar la información específica de este microservicio, en la figura 22 se muestra la base de datos diseñada en SQL Server.

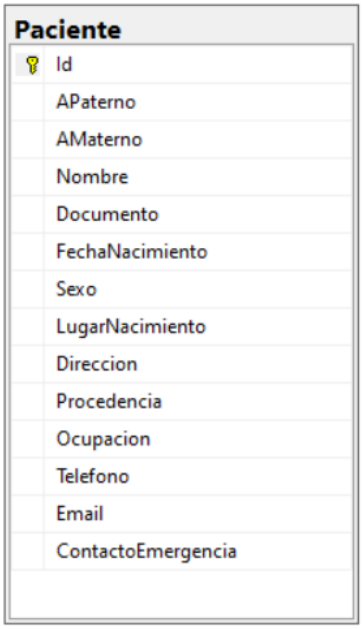


Figura 22. Estructura de la base de datos del microservicio Paciente.

Fuente: Elaboración propia

Este microservicio cumple con las funcionalidades específicas que se le han asignado en la etapa de desarrollo de la arquitectura, en la figura 23 se muestra las rutas finales del recurso y el tipo de petición de cada API Rest. En el anexo 3 se tiene un detalle de la información de entrada (request), salida (response) y parámetros de cada una de las rutas finales de este microservicio.

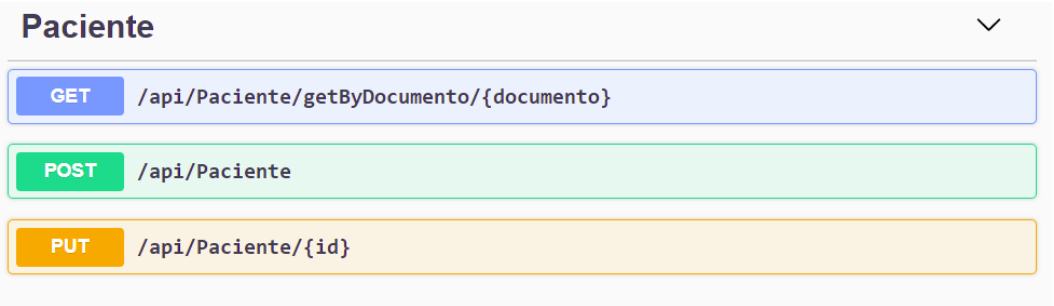


Figura 23. Rutas finales del microservicio Paciente.

Fuente: Elaboración propia

Para acceder a cada una de las rutas finales de este microservicio se debe de enviar en la cabecera de la petición HTTP un Token válido.

Microservicio Historia Clínica Odontológica

En este Microservicio se han expuesto una API Rest, esta permite gestionar los recursos correspondientes de las Historias clínicas odontológicas. En la figura 24 podemos ver el detalle de la estructura de la implementación del microservicio Historia Clínica Odontológica en ASP Net Core.

Figura 24. Estructura de la implementación del microservicio HCO.

Fuente: Elaboración propia

Después de revisar la documentación correspondiente de la historia clínica odontológica del COP (anexo 1) se ha obtenido el esquema de datos que debe de administrar este microservicio, se ha optado por crear una base de datos relacional teniendo en cuenta los resultados obtenidos en la investigación "Estudio comparativo entre bases de datos temporales y bases de datos relacionales aplicado a historias clínicas electrónicas" de Diaz Montejo et al. (2015) cuyos autores concluyen que los sistemas de base de datos relacionales en cuanto a interoperabilidad tienen un mayor desempeño que los sistemas de gestión de base de datos transaccional. En la figura 25 se muestra la base de datos diseñada en SQL Server, se ha considerado que la HCO está conformada por una serie de registros individuales donde se registra cada atención médica.

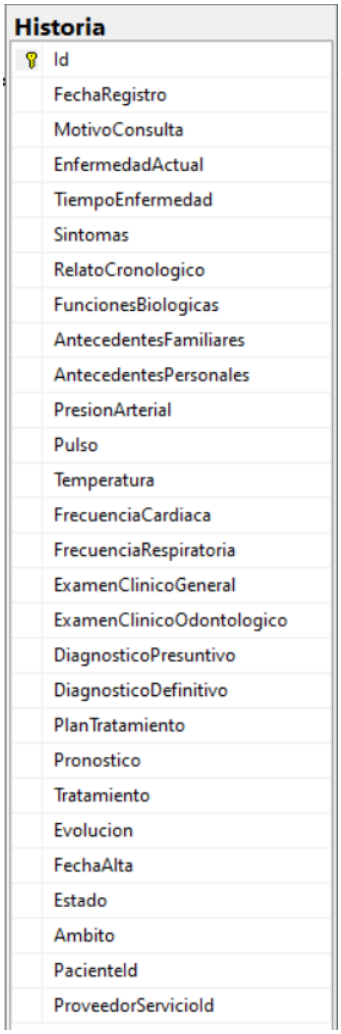


Figura 25. Estructura de la base de datos del microservicio HCO.

Fuente: Elaboración propia

Este microservicio cumple con las funcionalidades específicas que se le han asignado en la etapa de desarrollo de la arquitectura, en la figura 26 se muestra las rutas finales del recurso y el tipo de petición de cada API Rest. En el anexo 3 se tiene un detalle de la información de entrada (request), salida (response) y parámetros de cada una de las rutas finales de este microservicio.

Figura 26. Rutas finales del microservicio HCO.

Fuente: Elaboración propia

Para acceder a cada una de las rutas finales de este microservicio se debe de enviar en la cabecera de la petición HTTP un Token válido.

Microservicio Odontograma

En este Microservicio se han expuesto una API Rest, esta permite gestionar los recursos correspondientes de los Odontogramas y sus definiciones operativas. En la figura 27 podemos ver el detalle de la estructura de la implementación del microservicio Odontograma en Asp Net Core.

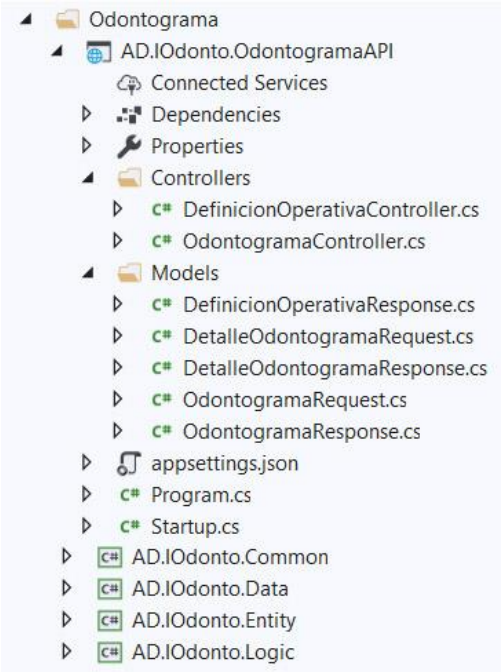


Figura 27. Estructura de la implementación del microservicio Odontograma.

Fuente: Elaboración propia

Después de revisar el formato de Odontograma propuesto por el COP (anexo 2) y la documentación correspondiente para la elaboración del Odontograma según la "Norma Técnica de Salud Para el uso del Odontograma" (NTS 150-MINSA-2019/FGIESP) se ha obtenido el esquema de datos que debe de administrar este microservicio. En la figura 28 se muestra la base de datos diseñada en SQL Server, cada Odontograma tiene detalles por cada pieza dentaria, esos detalles son almacenados en la tabla DetalleOdontograma y para cada detalle de la pieza dentaria se hace referencia a la definición operativa correspondiente, si bien no hay una relación directa entre el detalle del odontograma y la definición operativa ya que solo se almacena el nombre de la Definición operativa o el nombre de la actividad realizada a la pieza dentaria, esto para permitir la interoperabilidad de odontogramas en un futuro.

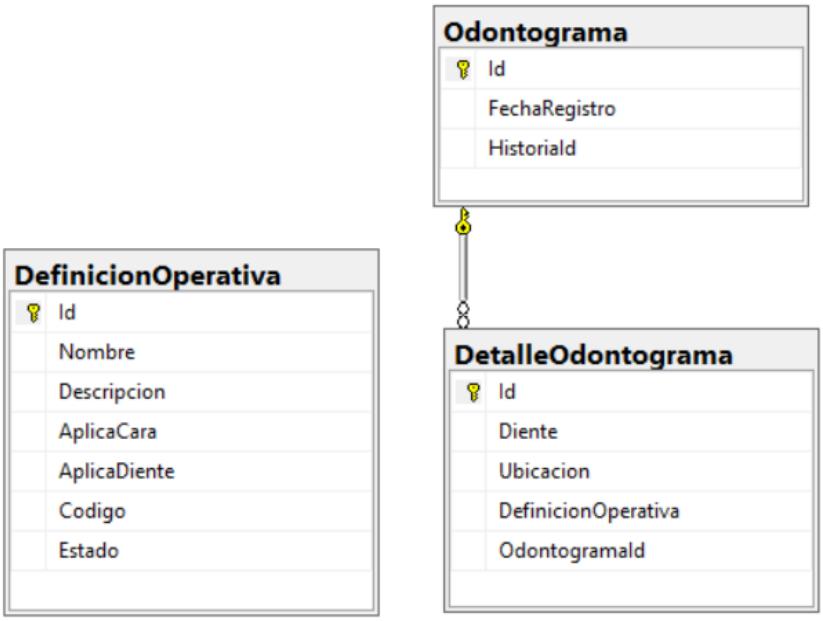


Figura 28. Estructura de la base de datos del microservicio Odontograma.

Fuente: Elaboración propia

Este microservicio cumple con las funcionalidades específicas que se le han asignado en la etapa de desarrollo de la arquitectura, en la figura 29 se muestra las rutas finales del recurso y el tipo de petición de cada API Rest. En el anexo 3 se tiene un detalle de la información de

entrada (request), salida (response) y parámetros de cada una de las rutas finales de este microservicio.

Figura 29. Rutas finales del microservicio Odontograma.

Fuente: Elaboración propia

Para acceder a cada una de las rutas finales de este microservicio se debe de enviar en la cabecera de la petición HTTP un Token válido.

API Gateway

Para implementar el API Gateway se ha creado un proyecto en ASP Net Core y se ha utilizado el paquete OCELOT, de esta manera tendremos una sola puerta de acceso para consumir los microservicios mejorando la seguridad de la arquitectura, administrar el balanceo de carga, QoS, rastreo distribuido. En la figura 30 se muestra la estructura de configuración de nuestro API Gateway con Ocelot.

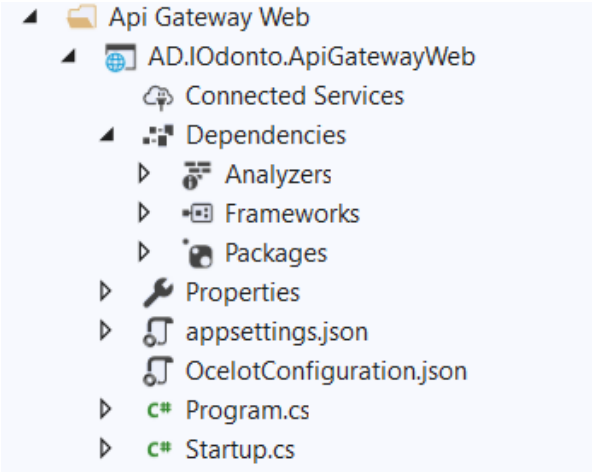


Figura 30. Estructura de la implementación del API Gateway.

Fuente: Elaboración propia

En el anexo 5 se detalla la configuración del API Gateway, en la figura 31 se muestra el funcionamiento lógico del API Gateway para comunicar al cliente con un microservicio específico.

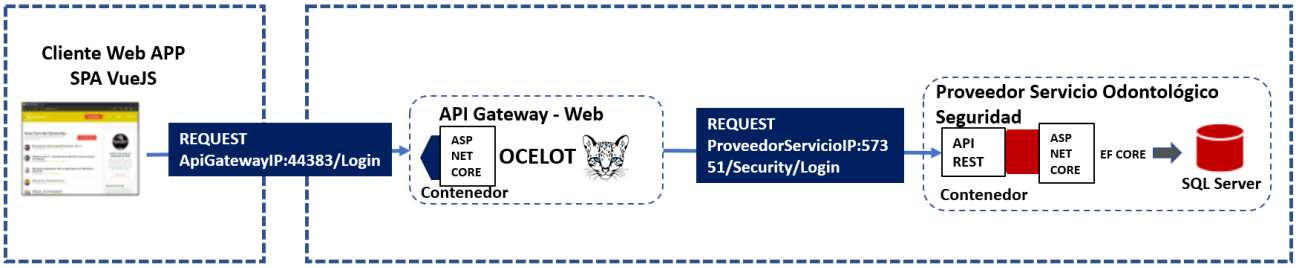


Figura 31. Funcionamiento lógico del API Gateway.

Fuente: Elaboración propia

Token

El cliente se debe autenticar en la arquitectura a través de la cuenta de acceso del Proveedor de Servicios Odontológicos si los datos de acceso son correctos la arquitectura le devuelve un TOKEN único y en formato común (definido por RFC 7519) para poder ser usado por todos los Microservicios, en nuestra implementación ese TOKEN se crea utilizando la librería JSON Web Token (JWT), después de que el cliente se autentica en la cabecera de cada petición HTTP que realice debe contener el TOKEN. En la figura 32 se muestra la implementación del método para crear el TOKEN de autenticación.

```
private string CreateToken(LoginResponseModel userInfo)
{
    var tokenHandler = new JwtSecurityTokenHandler();
    var key = Encoding.ASCII.GetBytes("xecretKeywqejane");
    var tokenDescriptor = new SecurityTokenDescriptor
    {
        Subject = new ClaimsIdentity(new List<Claim>
        {
            new Claim("userId", userInfo.Id.ToString()),
            new Claim("userEmail", userInfo.Email),
            new Claim("userNombre", (userInfo.Nombre+' '+userInfo.APaterno+' '+userInfo.AMaterno+' '))
        }),
        Expires = DateTime.UtcNow.AddHours(720),
        SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
    };
    var token = tokenHandler.CreateToken(tokenDescriptor);

    string tokenStr = tokenHandler.WriteToken(token);

    return tokenStr;
}
```

Figura 32. Uso de JSON Web Token en la implementación.

Fuente: Elaboración propia

Contenedor

Según la arquitectura propuesta cada Microservicio y el API Gateway deben de estar en contenedores distintos, esto permite la escalabilidad independiente de cada Microservicio aplicando políticas se puede establecer que, en períodos de mayor actividad para un Microservicio específico, aumente de manera incremental los recursos disponibles y, después, cuando disminuya la actividad, se reduzca para optimizar costos.

Se ha utilizado el servicio en la nube de Microsoft Azure para desplegar nuestros Microservicios y probar nuestra implementación, para cada microservicio se ha creado un contenedor cuyas características se muestran en la tabla 15. En la figura 33 se muestra la lista de los microservicios implementados en contenedores distintos utilizando el App Services de Microsoft Azure.

Tabla 15

Características del Contenedor

Característica	Valor
Nombre del Servicio	APP Services – Microsoft Azure
Velocidad Procesador	1 Core 3,7 GHz

Memoria RAM	1.75GB
Almacenamiento	50GB
Región	South Central USS
Sistema Operativo	Windows

Nota. Fuente: Microsoft Azure (2021)

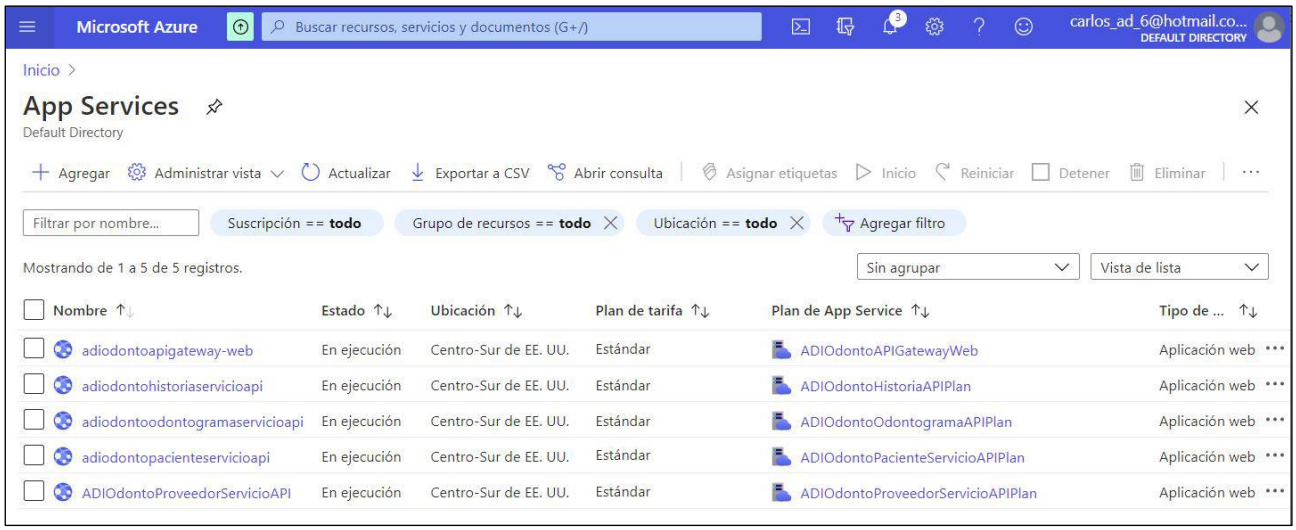


Figura 33. Despliegue de los Microservicios en Microsoft Azure.

Fuente: Elaboración propia

Existen varios planes de servicios en la nube y cada uno de ellos tiene un costo asignado por cada espacio de tiempo de uso, se ha optado por desplegar nuestra arquitectura en Microsoft Azure por la facilidad de publicar cada Microservicio y porque este ya tiene el SDK .Net Core necesario para ejecutar nuestras API Rest. Todas las pruebas realizadas se han hecho teniendo en cuenta estas características del contenedor.

Cliente

Se ha implementado un Cliente Web Single Page Application (SPA) utilizando el Framework VueJS, se usó la librería Vuetify para el diseño visual de la interfaz, se ha utilizado la librería Axios para realizar las solicitudes HTTP y consumir los recursos de los Microservicios a través de su API Rest. La implementación del cliente web ha permitido revisar la correcta

implementación del prototipo de la arquitectura de software propuesta, determinando que existe una correcta comunicación con cada una de las API Rest de los microservicios que conforman la arquitectura de software propuesta y se puede crear un flujo real de trabajo.

En la figura 34 se muestra la interfaz de consulta de HCO en el Cliente Web SPA, no se detalla mucho el desarrollo de este Cliente Web porque no es interés directo de la investigación, pero para más detalle en el anexo 6 se muestra la guía de uso de este Cliente Web SPA.

Figura 34. Interfaz de consulta de HCO en el Cliente Web SPA.

Fuente: Elaboración propia

3.6. Evaluación de la arquitectura de software basada en microservicios para la disponibilidad de HCO

3.6.1. Evaluación de la calidad de los microservicios implementados

Se ha evaluado la arquitectura propuesta utilizando los atributos de calidad de los microservicios propuestos por Kharbuja (2016) al igual que en la investigación denominada "Modelo de composición de Microservicios para la implementación de una aplicación web de comercio electrónico utilizando kubernetes" de Ruelas Acero (2017). Para evaluar cada uno de los atributos de calidad se han utilizado las métricas de calidad y su correspondiente valor según la arquitectura de software propuesta en esta investigación, en la tabla 11 se muestran las métricas de calidad y su valor correspondiente para la arquitectura propuesta.

Tabla 16

Métricas de calidad utilizadas para calcular los atributos de calidad

Métrica	Símbolo	Valor Arquitectura Microservicios
Número de mensajes proporcionados por un servicio	M(S)	3
Número de servicios	n_s	4
Número de servicios conectados	n_{c+} n_p	2
Número de servicios consumidores	n_c	1
Número de servicios dependientes	n_p	1
Número de consumidores del servicio	$S_{consumidores}$	4

Nota. Fuente: Adaptado de Kharbuja (2016)

Después de implementar la arquitectura de software basada en microservicios se ha calculado los atributos de calidad teniendo en cuenta las métricas de la tabla 16, en la tabla 17 se muestra los resultados de los atributos de calidad.

Tabla 17

Resultados de evaluar los atributos de calidad

Atributos De Calidad	Resultado
Acoplamiento	0.5
Cohesión	1.33
Reusabilidad	4

Nota. Fuente: Elaboración propia

Los cálculos de los resultados obtenidos se describen a continuación:

Acoplamiento

El acoplamiento de un servicio es la fuerza de la dependencia entre servicios en el sistema. Utilizando la ecuación 1 (Kharbuja, 2016) se obtiene un acoplamiento de 0.5, indicando que los microservicios son altamente independientes.

$$acoplamiento = \frac{n_c + n_p}{n_s} \quad \text{(Ecuación 1)}$$

El bajo nivel de acoplamiento permite a las aplicaciones ser más independientes y que, por tanto, puedan ser actualizadas afectando en menor medida al resto del sistema, posibilitando ciclos de entrega más cortos.

Cohesión

La cohesión es la relación entre las funciones implementadas y sus propósitos.

Utilizando la ecuación 2 (Kharbuja, 2016) se obtiene una cohesión de 1.33, indicando que los microservicios desempeñan las funcionalidades necesarias.

$$cohesión = \frac{n_s}{M(s)} \quad \text{(Ecuación 2)}$$

El principio de los microservicios es que tengan una alta cohesión, ya que cada microservicio realiza una sola funcionalidad.

Reusabilidad

La reusabilidad se define como la capacidad de reutilización de un servicio.

Utilizando la ecuación 3 (Kharbuja, 2016) se obtiene una reutilización de 4,

$$reusabilidad = S_{consumidores} \quad \text{(Ecuación 3)}$$

Los microservicios al realizar funcionalidades especializadas se pueden reutilizar independientemente para la implementación de otras aplicaciones.

3.6.2. Evaluación de los atributos de calidad de la arquitectura

Funcionalidad

Para validar la arquitectura de software de acuerdo al atributo de calidad de Funcionalidad se han realizado pruebas de caja negra, los resultados del informe de estas pruebas están detalladas en el anexo 11. Para evaluar la Funcionalidad y determinar que la arquitectura de software cumple con los requisitos identificados se han realizado 3 pruebas, para evaluar la gestión de Proveedores de Servicios odontológicos, Pacientes, HCO y Odontogramas teniendo en cuenta los formatos establecidos.

Después de realizar las pruebas de funcionalidad sobre el prototipo diseñado e implementado en esta investigación se ha determinado que la arquitectura de Software cumple con los requisitos identificados en el atributo de calidad de Funcionalidad.

Seguridad

Para validar la arquitectura de software de acuerdo al atributo de calidad de Seguridad se han realizado pruebas de caja negra, los resultados del informe de estas pruebas están detalladas en el anexo 11. Para evaluar la Seguridad y determinar que la arquitectura de software cumple con los requisitos de autenticación y autorización identificados se han realizado 2 pruebas, una de ellas permite validar el acceso y generación de Token y la otra la autorización para consultar una HCO y odontograma.

Después de realizar las pruebas de caja negra sobre el prototipo diseñado e implementado en esta investigación se ha determinado que la arquitectura de Software cumple con los requisitos identificados en el atributo de calidad de Seguridad.

Escalabilidad

Para validar la arquitectura de software de acuerdo al atributo de calidad de Escalabilidad se han realizado pruebas de escalabilidad, los resultados del informe de estas pruebas están detalladas en el anexo 10.

El requisito principal de este atributo es que la arquitectura debe permitir aumentar su capacidad de procesamiento cuando la demanda de HCO aumente, para esto se ha configurado el contenedor de la arquitectura para poder realizar el escalado horizontal ante un aumento de demanda, es decir cuando una instancia del servicio de HCO no soporte la demanda se cree otra instancia para poder apoyar en responder la demanda de peticiones.

Si se requiere que el escalado tenga un mayor impacto se debe primero implementar el servicio sobre un contenedor con mejores características, en esta investigación se ha trabajado con el contenedor cuyas características se encuentran en el apartado de equipos (tabla 15), con este contenedor se ha realizado esta prueba de escalabilidad.

Tabla 18

Resultado de disponibilidad de HCO con 3 instancias de HCO

Peticiones/Segundo	Tiempo promedio (ms)	Disponibilidad (%)
30	1056	100.00%
35	1048	100.00%
40	1054	100.00%
45	1099	100.00%
50	1021	100.00%
55	1074	100.00%
60	1042	100.00%
62	1166	100.00%
63	1199	100.00%
65	2632	**98.33%**
70	**4638**	**92.05**

Nota. Fuente: Elaboración propia

En la tabla 18 se muestra que con 3 instancias del servicio de HCO se puede soportar hasta 63 peticiones por segundo con una disponibilidad del 100% y cumpliendo el indicador de tiempo establecido para responder una HCO, si comparamos estos resultados con los de la tabla 20 donde se utiliza solo una instancia del microservicio podemos determinar que si se ve un aumento en la respuesta de peticiones cuando el microservicio tiene 3 instancias, en conclusión, la arquitectura de software propuesta puede escalar de manera correcta cuando se tiene un aumento en la demanda de peticiones.

3.6.3. Evaluación del desempeño de la arquitectura para la disponibilidad de las HCO

Para evaluar el desempeño de la arquitectura de software propuesta se ha utilizado pruebas de carga (anexo 7) para medir los indicadores de tiempo de respuesta y rendimiento, el informe de resultados de esta prueba de carga se muestra en el anexo 8.

En la tabla 15 se muestra las características de los contenedores (5 App Services) donde se ha desplegado el prototipo que implementa cada uno de los componentes de la arquitectura propuesta, las pruebas de carga se han realizado sobre esta infraestructura.

Los indicadores de aceptación en la prueba de carga fueron considerados teniendo en cuenta la investigación (Villa Benavides, 2017), donde se establece que un sistema que gestione historias clínicas electrónicas debe procesar en promedio 20 peticiones por segundo con un tiempo esperado de 2 segundos y nunca debe exceder los 5 segundos; deben tener una disponibilidad del 99%.

Tiempo de respuesta

En la tabla 19 se muestra el tiempo de respuesta promedio que tarda la arquitectura de software propuesta para resolver 2, 5, 10, 15, 20, 21, 22 y 23 peticiones por segundo, de esta tabla se puede demostrar que la arquitectura puede responder hasta 20 peticiones en un segundo determinado cumpliendo de esta manera con el indicador de aceptación que indica que la arquitectura debe permitir la disponibilidad de hasta 20 HCO en 1 segundo; vemos

además que incluso puede resolver hasta 21 peticiones dentro del tiempo de 1 segundo, cuando la arquitectura recibe más de 21 peticiones ya tarda más de 1 segundo en resolver estas peticiones.

Tabla 19

Tiempo de respuesta para la obtención de una HCO

Peticiones/Segundo	Tiempo Respuesta (Ms)
2	849
5	748
10	696
15	686
20	680
21	662
22	4393
23	13765

Nota. Fuente: Elaboración propia

Rendimiento

El rendimiento es la capacidad total de HCO que se puede disponer en un minuto utilizando la arquitectura de software propuesta en esta investigación.

Teniendo en cuenta que la arquitectura de software propuesta permite disponer de hasta 21 HCO con una disponibilidad del 100% se puede determinar que el rendimiento de la arquitectura propuesta es de 1260 HCO por minuto, considerando una instancia del contenedor y las características específicas de la tabla 15.

3.6.4. Evaluación de la arquitectura para la disponibilidad de las HCO

Para evaluar la disponibilidad de HCO de la arquitectura de software propuesta se ha utilizado pruebas de carga (anexo 7) para medir el porcentaje de respuestas correctas a las peticiones, el informe de resultados de esta prueba de carga se muestra en el anexo 8.

Disponibilidad

En la tabla 20 se muestra el resultado del test de carga correspondiente para evaluar la disponibilidad, este indicador representa el porcentaje de peticiones correctas del total de peticiones realizadas. Se logra determinar un 100% de disponibilidad de una HCO con la arquitectura de software propuesta para disponer 20 HCO cumpliendo de esta manera el indicador de aceptación de disponibilidad, además se puede verificar que la arquitectura puede disponer de hasta 21 HCO sin ningún margen de error y a partir de 22 HCO ya se tendría peticiones resueltas de manera incorrecta, pero si desea seguir manteniendo una disponibilidad del 100% se aplicarían reglas de escalabilidad para aumentar de manera automática las instancias del microservicio.

Tabla 20

Porcentaje de disponibilidad de una HCO

Peticiones/Segundo	Disponibilidad (%)
2	100.00%
5	100.00%
10	100.00%
15	100.00%
20	100.00%
21	100.00%
22	99.84%

Nota. Fuente: Elaboración propia

Capítulo IV. Discusión

Para el diseño de la arquitectura de software propuesta en esta investigación se han tomado diferentes decisiones con el objetivo de satisfacer los drivers de atributos de calidad identificados.

Funcionalidad

La arquitectura de software basada en microservicios diseñada en esta investigación permite implementar 4 servicios identificados para gestionar de manera independiente la información de Pacientes, Historias Clínicas Odontológicas, Odontogramas y Proveedores de servicios Odontológicos, permitiendo que cada microservicio desempeñe funcionalidades específicas para satisfacer los requerimientos funcionales del negocio, cada microservicio se implementa en su propio contenedor con una capa de Backend y Base de datos, estos microservicios se comunican con los clientes mediante una capa API Gateway mediante recursos HTTP en formato JSON. Al igual que Soto Vidal y Cuello (2020), en esta investigación se han identificado los requerimientos funcionalidades analizando la literatura existente y realizando una encuesta.

El número de microservicios que componen la arquitectura de software está relacionado al proyecto, Serna Flórez y Giraldo Plaza (2020) identificaron 5 microservicios para el monitoreo de paciente con Alzheimer.

En esta investigación para lograr la satisfacción de los drivers arquitectónicos se ha utilizado el prototipo como técnica para validar la arquitectura al igual que en la investigación de Santander Acosta y Sixto Fuentes (2021) que implementan un prototipo para validar su arquitectura que permite la gestión de la historia clínica digital en Oftalmopediatría y se ha utilizado la prueba funcional de caja negra al igual que Granda Vinueza (2018) en su desarrollo de un sistema informático de gestión de historias clínicas odontológicas.

Disponibilidad

La arquitectura de software diseñada en esta investigación permite disponer de hasta 21 HCO con un 100% de disponibilidad en un segundo determinado, desplegando el microservicio HCO en una sola instancia, con esto se logra cumplir con lo que se establece en la investigación de Villa Benavides (2017) que un sistema que gestione historias clínicas electrónicas debe procesar en promedio 20 peticiones por segundo con un tiempo esperado de 2 segundos y nunca debe exceder los 5 segundos; la arquitectura diseñada además puede escalar horizontalmente si se tiene una mayor demanda de peticiones, se ha probado la arquitectura hasta con 3 instancias del microservicio HCO y se ha logrado responder hasta 63 peticiones por segundo con un 100% de disponibilidad. En la investigación de Ruelas Acero (2017) se evalúa su arquitectura propuesta basada en microservicios con pruebas de carga, realizando hasta 20 peticiones por segundo se obtiene una disponibilidad del 81%. Calderón-Gómez et al. (2019) desarrolla una aplicación eHealth basada en microservicios y lo prueba con 30 usuarios virtuales concurrentes obteniendo una disponibilidad del 100%.

Seguridad

Al igual que la investigación de Sánchez Reyna (2015) donde se implementa la seguridad mediante certificados y tokens, en esta presente investigación se ha implementado el Token como mecanismo de autenticación y autorización, el Proveedor de Servicio Odontológico que desee consultar una HCO debe autenticarse previamente para que se genere el Token en el cliente, este Token permitirá realizar consultas de HCO del paciente. La evaluación que se ha hecho sobre la arquitectura diseñada en esta investigación ha determinado que esta cumple con los requisitos de autenticación y autorización identificados.

El mecanismo de seguridad mediante Token evita ataques CSRF y es utilizado también en la investigación de Ruelas Acero (2017) en su modelo de composición de microservicios

para comercio electrónico, Villa Benavides (2017) en su arquitectura que soporte la interoperabilidad de la historia clínica electrónica de pacientes en situaciones de emergencia, Soto Vidal y Cuello (2020) en su plataforma para atención médica a pacientes en época de covid-19.

Además del Token, se tiene como mecanismo de seguridad el API Gateway, este permite mantener segura la arquitectura debido a que no se exponen todos los microservicios para comunicarse con el cliente; es considerado también un mecanismo de seguridad en la investigación de Esposito et al. (2017), Cayo Alcos (2018) usa este mecanismo para la comunicación de sus microservicios.

Mantenibilidad

La arquitectura propuesta se ha diseñado bajo el enfoque de Microservicios permitiendo de esta manera un bajo acoplamiento, cada microservicio se despliega independientemente resultando más fácil las frecuentes actualizaciones afectando en menor medida al resto del sistema y posibilitando ciclos de entrega más cortos. En la investigación de Villa Benavides (2017) se diseñó la arquitectura para la interoperabilidad de la Electronic Health Record (EHR) de igual manera bajo el enfoque de microservicios, ya que brinda la posibilidad de integrar la información clínica de los pacientes en un repositorio único de historias clínicas electrónicas. Un indicador de mantenibilidad es la reusabilidad, la arquitectura diseñada en esta investigación tiene una reusabilidad de 4, es decir los microservicios al realizar funcionalidades especializadas se pueden reutilizar independientemente para la implementación de otras aplicaciones.

Escalabilidad

Se puede configurar la escalabilidad horizontal de la arquitectura de manera automática ya que ha sido diseñada bajo el enfoque de microservicios, cada microservicio implementado

en un contenedor distinto puede escalar y aumentar su capacidad de manera autónoma si experimenta un aumento en su demanda, y ajustar su capacidad según la demanda actual utilizando eficientemente los recursos, este enfoque de microservicios es seleccionado en la investigación de Ruelas Acero (2017), Cayo Alcos (2018), Villa Benavides (2017), Calderón-Gómez et al. (2019), (Esposito et al., 2017), Serna Flórez y Giraldo Plaza (2020) porque permite la escalabilidad horizontal de cada microservicio de manera independiente de la aplicación de manera sencilla. Se ha evaluado la escalabilidad de la arquitectura diseñada en esta investigación utilizando pruebas de escalabilidad, logrando determinar que la arquitectura es escalable y se genera una nueva instancia del microservicio cuando este tiene un aumento de demanda, se ha logrado revisar que con 3 instancias del servicio de HCO se puede soportar hasta 63 peticiones por segundo con una disponibilidad del 100% y cumpliendo el indicador de tiempo establecido para responder una HCO.

Desempeño

Considerando la investigación de Villa Benavides (2017) donde se establece que un sistema que gestione historias clínicas electrónicas deben tener una disponibilidad del 99%, se ha aplicado un test de carga para evaluar la eficiencia de la arquitectura de software diseñada en esta investigación se ha determinado que permite disponer de hasta 1260 HCO con una disponibilidad de 100% en un minuto, esto cuando solo se trabaja con una instancia del microservicio HCO, cuando aumenta la demanda y se tiene hasta 3 instancias del microservicio de HCO se puede disponer de hasta 3780 HCO por minuto con una disponibilidad del 100%. Para evaluar el desempeño se han utilizado pruebas de carga y pruebas de escalabilidad, Calderón-Gómez et al. (2019) ha realizado pruebas de carga y pruebas de estrés para evaluar su arquitectura eHealth basada en microservicios logrando hasta 30 peticiones por segundo, es decir se tiene un rendimiento de hasta 1800 respuestas

por minuto con una disponibilidad del 100%, en la investigación de Ruelas Acero (2017) se tiene un rendimiento de 740 respuestas por minuto con una disponibilidad del 100%.

Conclusiones

- El análisis del estado actual de las HCO en la provincia de Chiclayo mediante la ficha de encuesta ha logrado determinar que solo el 14.5% de los Proveedores de Servicios Odontológicos gestiona sus HCO ya sea mediante un sistema informático, hoja de cálculo o documento de texto digital y que el 78.3% de los cirujanos dentistas encuestados considera de utilidad que las historias clínicas odontológicas se encuentren almacenadas en un repositorio digital seguro y confiable para ser consultadas por el especialista autorizado en el momento oportuno.
- Los atributos de calidad para el diseño de la arquitectura de software han sido identificados teniendo en cuenta la normatividad peruana, antecedentes en la literatura referentes a sistemas de Historias clínicas, formatos establecidos por el Colegio Odontológico del Perú, encuesta a los especialistas cirujanos dentistas de Chiclayo, se consolidaron los siguientes atributos de calidad relevantes: Funcionalidad, disponibilidad, seguridad, mantenibilidad, escalabilidad y desempeño.
- Se ha utilizado el enfoque de Microservicios para el diseño de la arquitectura propuesta permitiendo de esta manera la funcionalidad, disponibilidad, mantenibilidad y escalabilidad independiente de cada microservicio identificado.
- Utilizando un diagrama de casos de uso y la descomposición en sustantivos se determinó que la arquitectura de software a diseñar debe estar compuesta por 4 Microservicios, Paciente, HCO, Odontograma y Proveedor de Servicio Odontológico, cada microservicio implementa su base de datos independiente, se realiza la comunicación segura entre los microservicios y los clientes mediante un API Gateway de recursos HTTP y un Token de autenticación.
- La implementación de la arquitectura de Software diseñada en esta investigación se puede realizar de manera independiente para cada microservicio y considerando los

conocimientos tecnológicos del equipo de desarrollo para poder implementar cada componente.

- Se ha evaluado la arquitectura diseñada utilizando los atributos de calidad de los Microservicios propuesto por Kharbuja (2016), obteniendo como resultado un acoplamiento de 0.5 indicando que los microservicios son altamente independientes, una cohesión de 1.33 indicando que los microservicios desempeñan las funcionalidades necesarias y una reusabilidad de 4 es decir se puede reutilizar cada uno de los 4 servicios de manera independiente en otros sistemas de software.
- Para validar la arquitectura de software de acuerdo al atributo de calidad Funcionalidad y Seguridad se han realizado pruebas de caja negra sobre el prototipo desarrollado que implementa la arquitectura de Software diseñada, logrando determinar que la arquitectura de Software cumple con los requisitos identificados.
- Para validar la arquitectura de software de acuerdo al atributo de calidad Escalabilidad se han realizado pruebas de escalabilidad sobre el prototipo desarrollado que implementa la arquitectura de Software diseñada, logrando determinar que la arquitectura de software diseñada puede escalar de manera correcta cuando se tiene un aumento en la demanda de peticiones.
- Se evaluaron los atributos de Desempeño y Disponibilidad utilizando pruebas de carga sobre el prototipo desarrollado que implementa la arquitectura de Software diseñada, obteniendo como resultado que con sólo una instancia de microservicio HCO se puede disponer de hasta 21 HCO en un segundo determinado con una disponibilidad del 100%, teniendo en cuenta la prueba de escalabilidad se determina que con 3 instancias del microservicio HCO se puede disponer hasta de 63 HCO con una disponibilidad del 100%.

Recomendaciones

- Se recomienda utilizar el enfoque de Microservicios para el diseño de Sistema de software que tendrán un alto volumen de transacciones, ya que con este enfoque se puede escalar horizontalmente cada servicio según sea necesario, se puede usar diferentes tecnológicas para el desarrollo de cada servicio y se pueden realizar modificaciones en cada servicio sin afectar el funcionamiento total del sistema de software.
- Para la identificación de los Microservicios se recomienda utilizar la descomposición en sustantivos mediante un diagrama de caso de usos.
- Mejorar la arquitectura de Software diseñada en investigación para poder obtener los datos de los pacientes desde un servicio público externo que incluya el uso de la firma digital de cada paciente para la autorización que permita disponer de su HCO.
- Utilizar prácticas de Integración/Distribución continuas (CI/CD) para el desarrollo con el enfoque de Microservicios.

Referencias Bibliográficas

Amazon Web Services. (2019). Microservicios. Retrieved December 11, 2019, from https://aws.amazon.com/es/microservices/

Bass, L., Clements, P., & Kazman, R. (2013). *Software Architecture in Practice third Edition*. Addison-Wesley.

Calderón-Gómez, H., Navarro-Marín, F., Gómez-Pulido, J. M., Castillo-Sequera, José Luis, Garcés-Jiménez, A., Polo-Luque, M.-L., Sanz-Moreno, J., … Vargas-Lombardo, M. (2019). Desarrollo de aplicaciones eHealth basadas en microservicios en una arquitectura de Cloud. *Iberian Journal of Information Systems and Technologies*, (July). Retrieved from https://www.researchgate.net/publication/337673427_Desarrollo_de_aplicaciones_eHealth_basadas_en_microservicios_en_una_arquitectura_de_Cloud

Cayo Alcos, D. B. (2018). *EH-UNMSM : E-Health Cloud para la mejora de procesos de la clínica universitaria UNMSM mediante una arquitectura de microservices*. Universidad Nacional Mayor de San Marcos. Retrieved from http://cybertesis.unmsm.edu.pe/handle/cybertesis/7913

Cervantes Maceda, H., Velasco-Elizondo, P., & Castro Careaga, L. (2017). *Arquitectura de Software*. CENGAGE Learning.

Chumpolsathien, N. (2019). Microservices: the Future of Distributed System. *School of Computer Science and Technology - Beijing Institute of Technology*.

Colegio Odontológico del Perú. (2019a). Historia clínica odontológica. Retrieved from http://www.cop.org.pe/wp-content/uploads/2019/08/HISTORIA-CLÍNICA-ODONTOLÓGICA-2019.pdf

Colegio Odontológico del Perú. (2019b). Odontograma. Retrieved from http://www.cop.org.pe/wp-content/uploads/2019/08/ODONTOGRAMA-2019.pdf

Congreso de La República del Perú. Ley 30024 (2013). Perú.

Diaz Montejo, J. E., Bellon Cely, D. F., & González Sanabria, J. S. (2015). Estudio comparativo entre bases de datos temporales y bases de datos relacionales aplicado a historias clínicas electrónicas. *Ing. USBMed*, *6*(1), 46–53.

Esposito, C., Castiglione, A., Tudorica, C., & Pop, F. (2017). Security and Privacy for Cloud-Based Data Management in the Health Network Service Chain: A Microservice Approach. *IEEE Communications Magazine*, (September), 102–108. https://doi.org/10.1109/MCOM.2017.1700089

Gorton, I. (2011). *Essential Software Architecture (2 . ed .)* (Springer). Australia.

Granda Vinueza, P. E. (2018). *Desarrollo de un sistema informático para controlar la información de consultorios odontológicos que gestionen las historias clínicas de pacientes. Caso Centro Médico AXXIS*. UNIVERSIDAD CENTRAL DEL ECUADOR.

HL7. (2019). Fast Healthcare Interoperability Resources (FHIR). Retrieved from https://www.hl7.org/fhir/overview-dev.html

IDEA FRI. (2016). *Estudio para determinar la brecha de oferta y demanda de los servicios profesionales de Odontología en Perú 2016*. Lima.

Kajko-Mattsson, M., & Lewis, G. A. (2014). A Framework for Roles for Development , Evolution and Maintenance of SOA- Based Systems. *ResearchGate*, (February). https://doi.org/10.1109/SDSOA.2007.1

Kharbuja, R. (2016). *Designing a Business Platform using Microservices*. TECHNISCHE UNIVERSITÄT MÜNCHEN Master's.

Lattanze, A. (2008). *Architecting Software Intensive Systems: A Practitioners Guide*. Auerbach Publications.

Lewis, J., & Fowler, M. (2014). Microservices. Retrieved from

https://martinfowler.com/articles/microservices.html

LOLIMSA. (2014). Lolimsa: Solo el 11% de las historias clínicas son virtuales. Retrieved from https://elcomercio.pe/economia/negocios/lolimsa-11-historias-clinicas-son-virtuales-324020-noticia/?ref=ecr

Microsoft. (2019). Design a DDD-oriented microservice. Retrieved December 11, 2019, from https://docs.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/ddd-oriented-microservice

Ministerio de Salud Perú. Norma Técnica de Salud para el Uso del Odontograma N° 150-MINSA/2019/DGIESP (Aprobada por la RM 272-2019/MINSA) (2019). Perú. Retrieved from http://www.cop.org.pe/wp-content/uploads/2019/04/RM-272-2019-MINSA-y-NTS-150-MINSA-2019-DGIESP-1_8690.pdf

Newman, S. (2015). *Building Microservices*. O'Reilly Media.

Richards, M. (2016). *Microservices vs. service-oriented architecture*. O'Reilly. Retrieved from https://www.oreilly.com/radar/microservices-vs-service-oriented-architecture/

Rivera López, M., Santander Acosta, R., & Sixto Fuentes, S. (2021). Arquitectura de información para la gestión de la historia clínica digital en oftalmopediatría. *Revista de Ciencias Médicas de Pinar Del Río*, *25*(2). Retrieved from http://revcmpinar.sld.cu/index.php/publicaciones/article/view/4853

Rojas Mezarina, L., Cedamentos Medina, A., & Vargas Herrera, J. (2015). Registro nacional de historias clínicas electrónicas en perú. *Revista Peruana de Medicina Experimental y Salud Pública*, *32*(2), 395–396.

Roldán Martínez, D., Valderas Aranda, P. J., & Torres Bosh, V. (2018). *Microservicios un enfoque integral*. (RA-MA, Ed.). Madrid.

Ruelas Acero, D. A. (2017). *Modelo de composición de Microservicios para la*

implementación de una aplicación web de comercio electrónico utilizando kubernetes. Universidad Nacional del Altiplano.

Sánchez Reyna, K. S. H. (2015). *Recuperación de Historias Clínicas Electrónicas a partir de un Repositorio Digital usando una Arquitectura Orientada a servicios*. Pontificia Universidad Católica del Perú.

Serna Flórez, M., & Giraldo Plaza, J. E. (2020). Sistema distribuido y sensible al contexto para el monitoreo de pacientes con mal de Alzheimer. *Revista Cintex*, *25*(1), 21–31.

Soto Vidal, D. F., & Cuello, C. G. C. (2020). *PLATAFORMA PARA LA ATENCION MEDICA A PACIENTES EN EPOCA DE COVID-19*. Universidad de Córdoba.

Subcomité Técnico V3-CDA HL7 Spain. (2007). *Guía para el desarrollo de documentos CDA*.

Villa Benavides, L. E. (2017). *Diseño de una arquitectura que soporte la interoperabilidad de la historia clínica electrónica de pacientes en situaciones de emergencia*. Universidad de San Buenaventura. Retrieved from http://hdl.handle.net/10819/5469

Anexos

Anexo 1: Encuesta realizada a los cirujanos dentistas de la provincia de Chiclayo

ENCUESTA REALIZADA A LOS CIRUJANOS DENTISTAS DE LA PROVINCIA DE CHICLAYO

Objetivo: Conocer la forma de trabajo de los proveedores de servicios odontológicos respecto al manejo de Historias Clínicas Odontológicas.

Instrucciones: Esta información solo se utilizará con fines académicos.

A. Almacenamiento de las Historias Clínicas Odontológicos

1. En la clínica o establecimiento de salud donde usted labora se registra información de las historias clínicas Odontológicas de sus pacientes.
   a) Siempre b) A veces c) Nunca
2. Almacena manualmente la información de las Historias Clínicas de sus pacientes.
   a) Siempre b) A veces c) Nunca
3. Almacena digitalmente la información de las Historias Clínicas Odontológicas de sus pacientes, considere digitalmente a un sistema informático, documento digital de texto, hoja de cálculo o base de datos.
   a) Siempre b) A veces c) Nunca
4. De la información de la Historia Clínica Odontológica que usted registra se entrega una copia al paciente.
   a) Siempre b) A veces c) Nunca

B. Consulta de las Historias Clínicas Odontológicas

5. ¿Cuándo asiste un paciente a atenderse a su clínica o establecimiento de salud ¿Es necesario consultar su historial médico?
   a) Siempre b) A veces c) Nunca
6. El paciente le brinda la información adecuada para que usted pueda hacer su diagnóstico sin perder mucho tiempo.
   a) Siempre b) A veces c) Nunca
7. Se ha presentado inconvenientes porque no se tiene la información de Historia Clínica Odontológica completa de su paciente.
   a) Siempre b) A veces c) Nunca
8. ¿Estaría dispuesto a compartir la información de las Historias Clínicas Odontológicas de sus pacientes con otros proveedores de servicios odontológicos?
   a) Siempre b) A veces c) Nunca
9. Considera de utilidad que las Historias Clínicas Odontológicas se encuentren almacenadas en un repositorio digital seguro y confiable para que puedan ser consultadas por el especialista propietario y autorizado en el momento oportuno.
   a) Siempre b) A veces c) Nunca

Fuente: Elaboración propia

Anexo 2: Formato de Historia Clínica Odontológica

HISTORIA CLÍNICA ODONTOLÓGICA

HC: ..... Fecha: ..... Hora: .....

ANAMNESIS

FILIACIÓN:

Nombres del paciente: ..... Edad: ..... Sexo: .....

Lugar y fecha de nacimiento: .....

Dirección: .....

Procedencia: ..... Ocupación: .....

Viajes en el último año: .....

Teléfono: .....

En caso de emergencia comunicarse a: .....

MOTIVO DE CONSULTA

.....

.....

ENFERMEDAD ACTUAL

.....

.....

Tiempo de enfermedad: .....

Signos y síntomas principales: .....

Relato cronológico: .....

Funciones biológicas:

.....

.....

ANTECEDENTES

Antecedentes familiares:

.....

Antecedentes personales:

EXAMEN CLÍNICO

Signos Vitales. P.A.: ..... Pulso: ..... Temp.: ..... F.C.: ..... F. Resp.: .....

Examen clínico general:

Examen clínico odontoestomatológico:

DIAGNÓSTICO (CIE 10)

Diagnóstico presuntivo:

Diagnóstico definitivo:

PLAN DE TRATAMIENTO

PRONÓSTICO

TRATAMIENTO / RECOMENDACIONES
(Nombre genérico del medicamento, dosis, vía de administración, tiempo de administración, cuidados, medidas higiénico- dietéticas, preventivas)

CONTROL Y EVOLUCIÓN

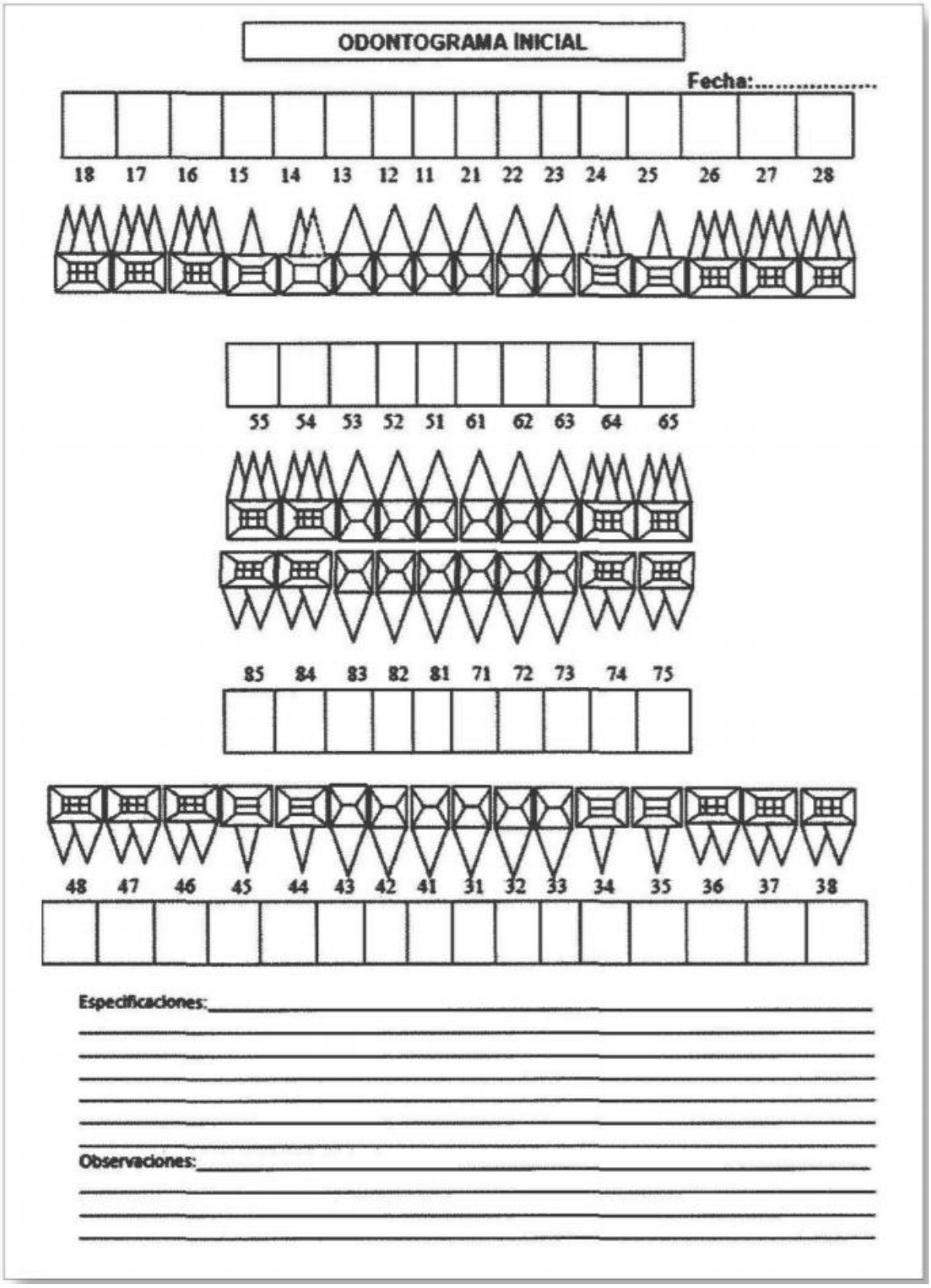
ALTA DEL PACIENTE

Nombres y apellidos del profesional

Sello y firma

Fuente: Colegio Odontológico del Perú (2019)

Anexo 3: Formato de Odontograma



Fuente: Colegio Odontológico del Perú (2019)

Anexo 4: Documentación de las API Rest

Microservicio Proveedor

Rutas finales

ProveedorServicio

- GET /api/ProveedorServicio/{id}
- PUT /api/ProveedorServicio/{id}
- POST /api/ProveedorServicio
- PUT /api/ProveedorServicio/cambiarPassword/{id}

Esquemas de entrada en Formato JSON

ProveedorServicioRequest {

aPaterno*	string maxLength: 20 minLength: 2
aMaterno*	string maxLength: 20 minLength: 2
nombre*	string maxLength: 20 minLength: 2
documento*	string maxLength: 15 minLength: 8
codigo	string maxLength: 50 minLength: 0 nullable: true
fechaNacimiento*	string($date-time)
sexo*	string maxLength: 1 minLength: 0
direccion	string maxLength: 100 minLength: 0 nullable: true
telefono	string maxLength: 15 minLength: 0 nullable: true
email*	string maxLength: 50 minLength: 5
password*	string maxLength: 50 minLength: 8

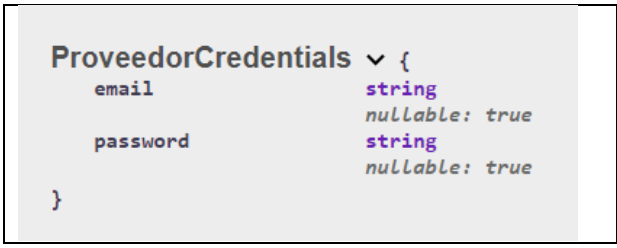
}

Esquemas de Salida en formato JSON

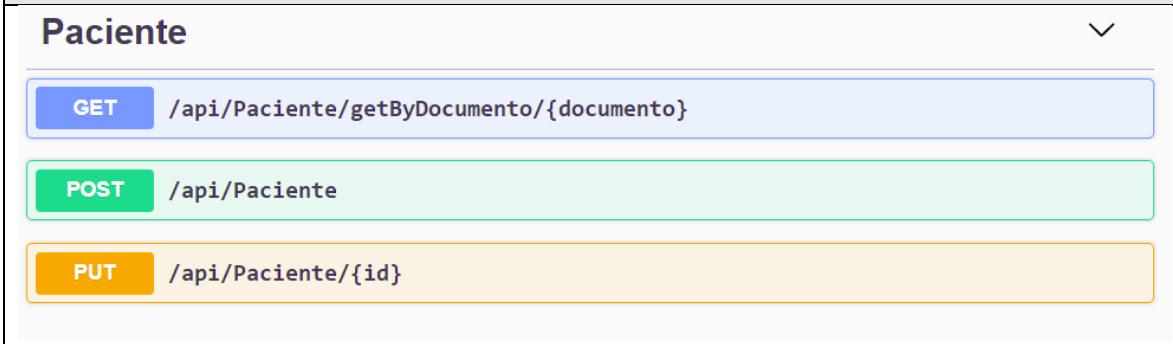
LoginResponseModel {

id	integer($int32)
aPaterno	string nullable: true
aMaterno	string nullable: true
nombre	string nullable: true
email	string nullable: true
token	string nullable: true

}



Microservicio Paciente
Rutas finales



Esquemas de entrada en Formato JSON	Esquemas de Salida en formato JSON
	PacienteResponse { id integer($int32), aPaterno string nullable: true, aMaterno string nullable: true, nombre string nullable: true, documento string nullable: true, fechaNacimiento string($date-time), sexo string nullable: true, lugarNacimiento string nullable: true, direccion string nullable: true, procedencia string nullable: true, ocupacion string nullable: true, telefono string nullable: true, email string nullable: true, contactoEmergencia string nullable: true }

PacienteRequest {

aPaterno*	string maxLength: 20 minLength: 2
aMaterno*	string maxLength: 20 minLength: 2
nombre*	string maxLength: 20 minLength: 2
documento*	string maxLength: 15 minLength: 8
fechaNacimiento*	string($date-time)
sexo*	string maxLength: 1 minLength: 0
lugarNacimiento	string maxLength: 50 minLength: 0 nullable: true
direccion	string maxLength: 100 minLength: 0 nullable: true
procedencia	string maxLength: 100 minLength: 0 nullable: true
ocupacion	string maxLength: 50 minLength: 0 nullable: true
telefono	string maxLength: 15 minLength: 0 nullable: true
email	string maxLength: 50 minLength: 0 nullable: true
contactoEmergencia	string maxLength: 100 minLength: 0 nullable: true

}

Microservicio Historia Clínica Odontológica

Rutas finales

Historia

GET	/api/Historia/{id}
GET	/api/Historia/getByPacienteDocumento /{documento}
POST	/api/Historia
PUT	/api/Historia

Esquemas de entrada en Formato JSON

HistoriaRequest {

fechaRegistro*	string($date-time)
motivoConsulta	string maxLength: 256 minLength: 0 nullable: true
enfermedadActual	string maxLength: 100 minLength: 0 nullable: true
tiempoEnfermedad	string maxLength: 30 minLength: 0 nullable: true
sintomas	string maxLength: 256 minLength: 0 nullable: true
relatoCronologico	string maxLength: 512 minLength: 0 nullable: true
funcionesBiologicas	string maxLength: 256 minLength: 0 nullable: true
antecedentesFamiliares	string maxLength: 256 minLength: 0 nullable: true
antecedentesPersonales	string maxLength: 256 minLength: 0 nullable: true
presionArterial	string maxLength: 12 minLength: 0 nullable: true
pulso	integer($int32)
temperatura	integer($int32)
frecuenciaCardiaca	integer($int32)
frecuenciaRespiratoria	integer($int32)
examenClinicoGeneral	string maxLength: 512 minLength: 0 nullable: true

Esquemas de Salida en formato JSON

HistoriaResponse {

id	integer($int32)
fechaRegistro	string($date-time)
motivoConsulta	string nullable: true
enfermedadActual	string nullable: true
tiempoEnfermedad	string nullable: true
sintomas	string nullable: true
relatoCronologico	string nullable: true
funcionesBiologicas	string nullable: true
antecedentesFamiliares	string nullable: true
antecedentesPersonales	string nullable: true
presionArterial	string nullable: true
pulso	integer($int32)
temperatura	integer($int32)
frecuenciaCardiaca	integer($int32)
frecuenciaRespiratoria	integer($int32)
examenClinicoGeneral	string nullable: true
examenClinicoOdontologico	string nullable: true
diagnosticoPresuntivo	string nullable: true
diagnosticoDefinitivo	string nullable: true
planTratamiento	string nullable: true
pronostico	string nullable: true
tratamiento	string nullable: true
evolucion	string nullable: true
fechaAlta	string($date-time)
estado	string nullable: true
ambito	boolean
pacienteId	integer($int32)
proveedorServicioId	integer($int32)

}

```
    examenClinicoOdontologico string
                              maxLength: 512
                              minLength: 0
                              nullable: true
    diagnosticoPresuntivo     string
                              maxLength: 256
                              minLength: 0
                              nullable: true
    diagnosticoDefinitivo     string
                              maxLength: 256
                              minLength: 0
                              nullable: true
    planTratamiento           string
                              maxLength: 256
                              minLength: 0
                              nullable: true
    pronostico                string
                              maxLength: 256
                              minLength: 0
                              nullable: true
    tratamiento               string
                              maxLength: 512
                              minLength: 0
                              nullable: true
    evolucion                 string
                              maxLength: 512
                              minLength: 0
                              nullable: true
    fechaAlta*                string($date-time)
    estado                    string
                              maxLength: 20
                              minLength: 0
                              nullable: true
    ambito*                   boolean
    pacienteId*               integer($int32)
    proveedorServicioId*      integer($int32)
}
```

Microservicio Odontograma	
Rutas finales	
DefinicionOperativa GET /api/DefinicionOperativa/getByNombre Odontograma GET /api/Odontograma/getByHistoriaId/{id} GET /api/Odontograma/getUltimo POST /api/Odontograma PUT /api/Odontograma/{id}	
Esquemas de entrada en Formato JSON	**Esquemas de Salida en formato JSON**

Fuente: Elaboración propia

Anexo 5: Configuración API Gateway con Ocelot

```
{
  "Routes": [
    {
      "DownstreamPathTemplate": "/api/Security/Login",
      "DownstreamScheme": "http",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 57351
        }
      ],
      "UpstreamPathTemplate": "/Login",
      "UpstreamHttpMethod": [ "POST" ]
    },
    {
      "DownstreamPathTemplate": "/api/ProveedorServicio/",
      "DownstreamScheme": "http",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 57351
        }
      ],
      "UpstreamPathTemplate": "/ProveedorServicio/",
      "UpstreamHttpMethod": [ "POST", "PUT" ]
    },
    {
      "DownstreamPathTemplate": "/api/Paciente/{everything}",
      "DownstreamScheme": "http",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 44324
        }
      ],
      "UpstreamPathTemplate": "/Paciente/{everything}",
      "UpstreamHttpMethod": [ "GET","POST", "PUT" ]
    },
```

```
{
      "DownstreamPathTemplate": "/api/Historia/{everything}",
      "DownstreamScheme": "http",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 53040
        }
      ],
      "UpstreamPathTemplate": "/Historia/{everything}",
      "UpstreamHttpMethod": [ "POST", "PUT", "GET" ]
    },
    {
      "DownstreamPathTemplate": "/api/Odontograma/{everything}",
      "DownstreamScheme": "http",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 53050
        }
      ],
      "UpstreamPathTemplate": "/Odontograma/{everything}",
      "UpstreamHttpMethod": [ "POST", "PUT", "GET" ]
    },
    {
      "DownstreamPathTemplate": "/api/DefinicionOperativa/{everything}",
      "DownstreamScheme": "http",
      "DownstreamHostAndPorts": [
        {
          "Host": "localhost",
          "Port": 53050
        }
      ],
      "UpstreamPathTemplate": "/DefinicionOperativa/{everything}",
      "UpstreamHttpMethod": [ "GET" ]
    }
  ]
}
```

Fuente: Elaboración propia

Anexo 6: Guía de uso del Cliente Web APP SPA

1. Generalidades

Este cliente web SPA se denomina IOdonto Web, en su versión 1.0, tiene las siguientes funcionalidades:

a. Registro y actualización de Proveedores de servicios Odontológicos

b. Registro y Actualización de Pacientes

c. Registro, actualización y Consulta de Historia Clínica Odontológica (Atenciones odontológicas) y Odontogramas

Se puede acceder a este sitio web a través de su URL pública.

Esta es una guía inicial para poder empezar a trabajar con IOdonto web.

2. Registro de datos de acceso

Los que pueden acceder al sistema de Software para la disponibilidad de HCO son los diferentes Proveedores de Servicios Odontológicos a través de una cuenta, para esto se ha considerado su previo registro. En la interfaz de inicio de sesión se tiene el botón de "Registrarse", al hacer click en ese botón se muestra el formulario de registro (Fig. 1).

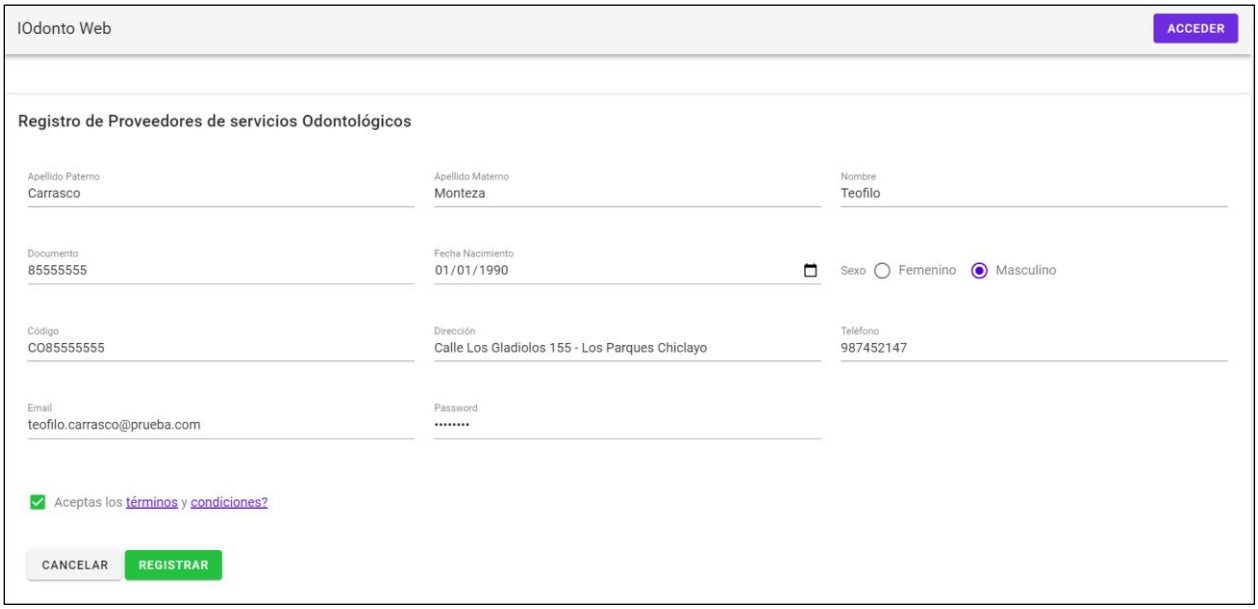


Figura 1. Formulario de Registro

Una vez ya registrado los datos de acceso el proveedor de servicios odontológicos puede acceder a IOdonto web.

3. Autenticación

Para poder acceder y consulta las HCO se debe tener una cuenta de Proveedor de Servicio Odontológico, en el formulario de acceso (Fig. 2) se ingresa el email y password correspondiente, si los datos son correctos se crea un token de acceso y se ingresa al dashboard para poder consultar las HCO y sus odontogramas, consultar pacientes.

Figura 2. Formulario de Acceso

4. Interfaz Principal

Al acceder correctamente a IOdonto web se muestra una interfaz de usuario (Fig. 3) con las siguientes opciones de menú:

Consultar HCO de Paciente: Permite consultar el Historial Odontológico del paciente a través de su documento.

Mi Perfil: Opción que permite actualizar los datos del Proveedor de Servicio Odontológico.

Ayuda: Opción que muestra la siguiente guía de ayuda.

Acerca de: Datos de contacto del desarrollador.

Cerrar sesión: Opción que elimina la sesión de acceso y el token de autenticación.

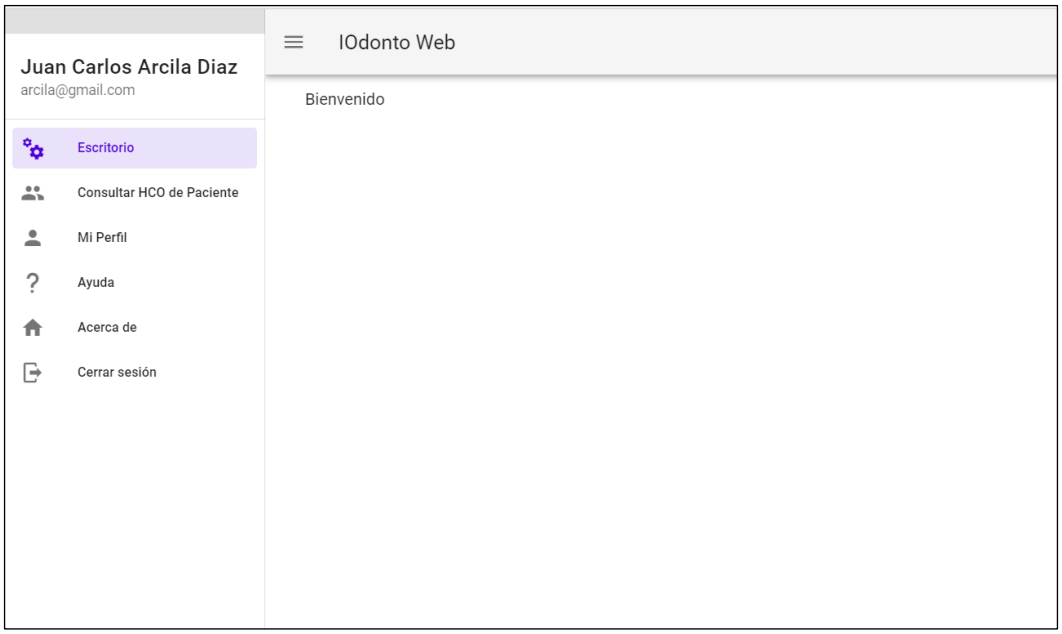


Figura 3. Interfaz principal

5. Consulta de Historial Clínico Odontológico

5.1.Consulta paciente

Para realizar la consulta del Historia Odontológico asociada a un paciente se debe seleccionar la opción de menú "Consultar HCO de paciente", en este interfaz se debe ingresar el documento de identificación (DNI, cédula, Passport entre otros) del paciente a consultar (Fig. 4).

Figura 4. Consultar Paciente

Si el paciente no está registrado y por ende no tiene Historial Odontológico nos aparece la opción de registrarlo previamente (Fig. 5).

Figura 5. Resultado consulta paciente

5.2.Registrar paciente

Al seleccionar el botón "Registrar paciente" nos aparece el formulario (Fig. 6) para registrar los datos del paciente.

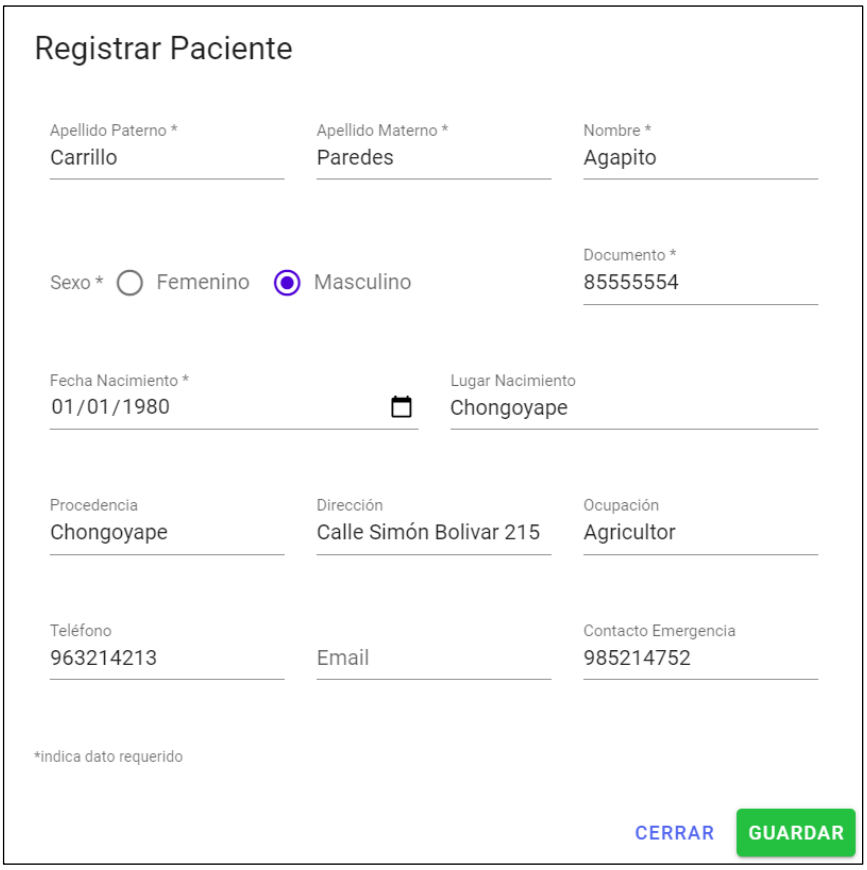


Figura 6. Registrar paciente

Una vez registrado el paciente se puede registrar su atención Odontológica, que formará parte de su historial odontológico.

5.3.Registrar atención odontológica

Para registrar una nueva atención simplemente se debe hacer click en el botón “Nueva atención”, esta opción aparece después de haber registrado a un paciente o después de consultar un paciente existente (Fig. 7).

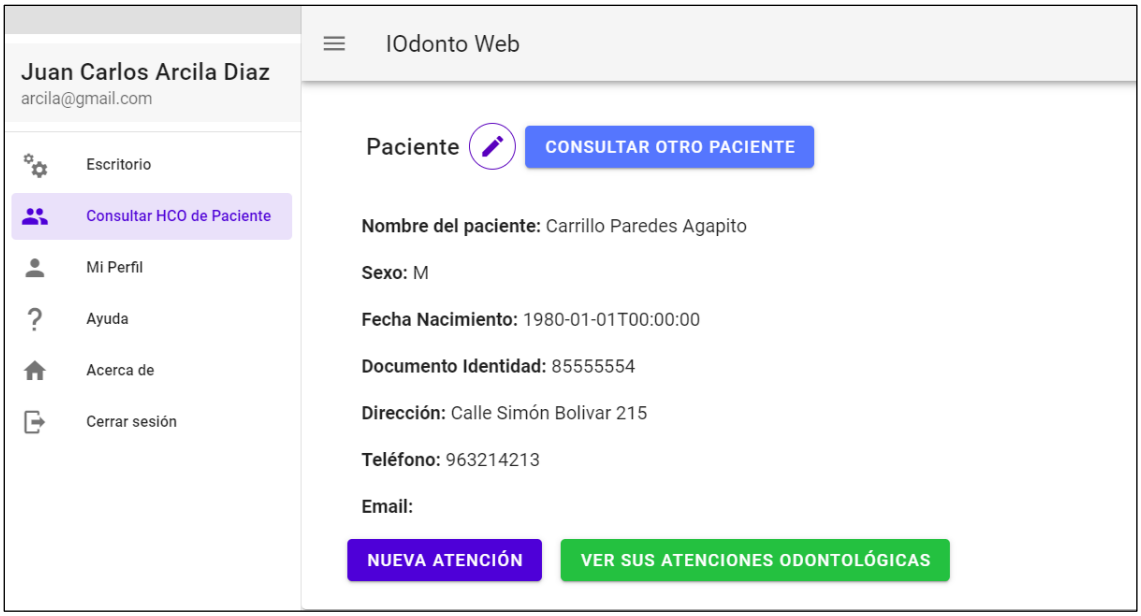


Figura 7. Datos de paciente

Al hacer click en el botón “Nueva atención” nos aparece el formulario de registro de la atención odontológica que formara parte de su HCO.

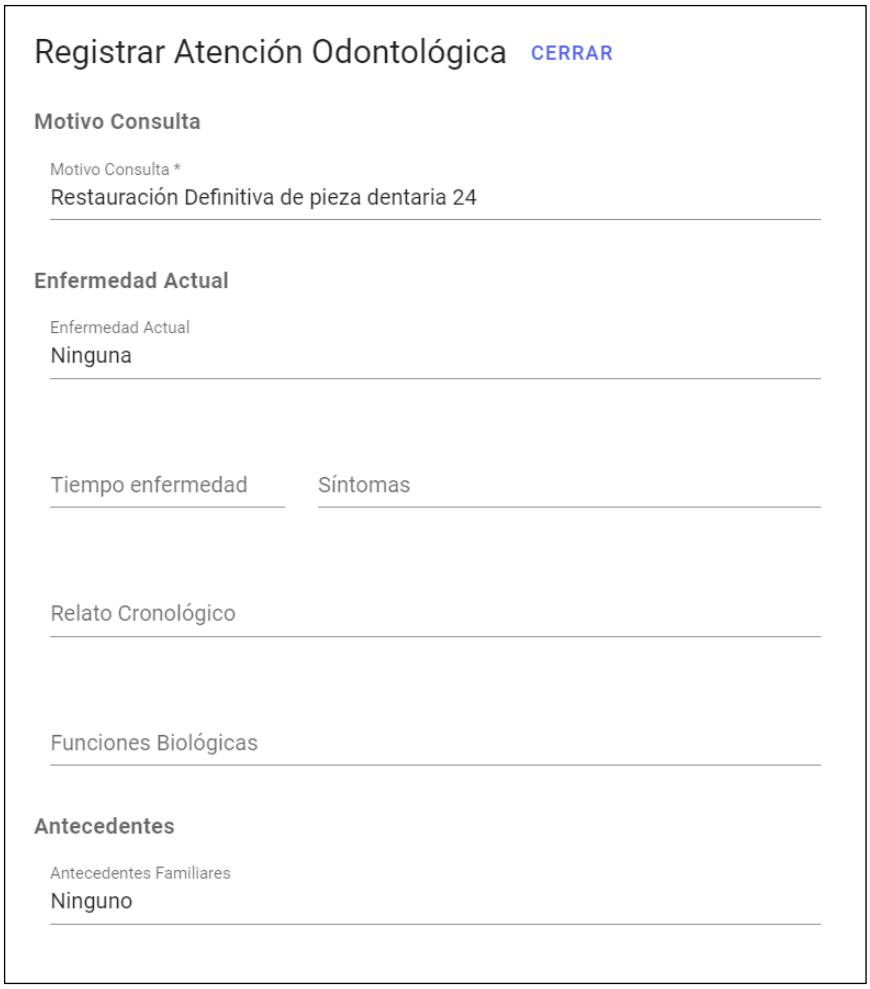


Figura 8. Registro de atención odontológica

Después del registro de la atención odontológica IOdonto web pedirá que se actualice el odontograma del paciente (Fig. 9).

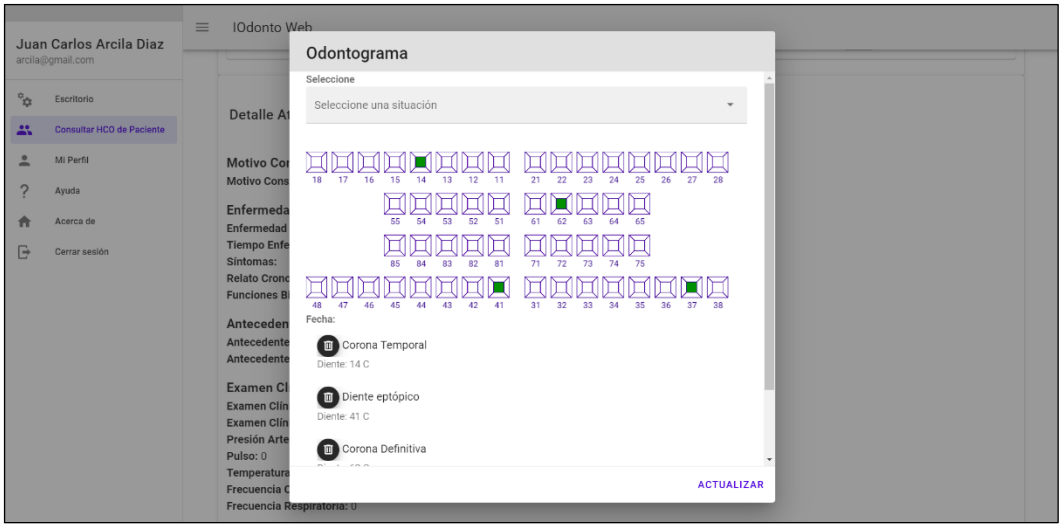


Figura 9. Actualizar Odontograma del Paciente

5.4.Consultar atenciones odontológicas previas

Existe la posibilidad de consultar atenciones odontológicas previas, para esto se consulta de igual manera el paciente como en el 5.1. pero esta vez como el usuario ya está registrado tendremos la opción para consultar si tiene atenciones odontológicas desde el botón “Ver sus atenciones odontológicas”, al hacer click en este botón nos aparecerá la lista de esas atenciones (Fig. 10).

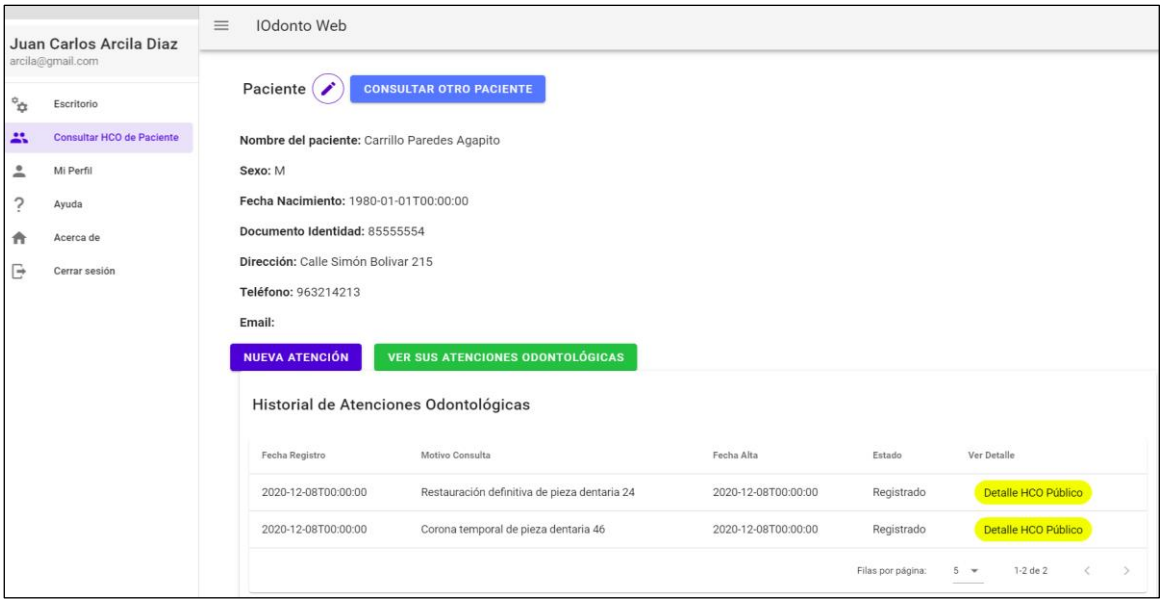


Figura 10. Historial de Atenciones Odontológicas

Al hacer click en la opción “Ver detalle” de cada atención odontológica podremos revisar el detalle de la atención y su odontograma respectivo desde el botón “Ver Odontograma” para esa atención Odontológica (Fig. 11).

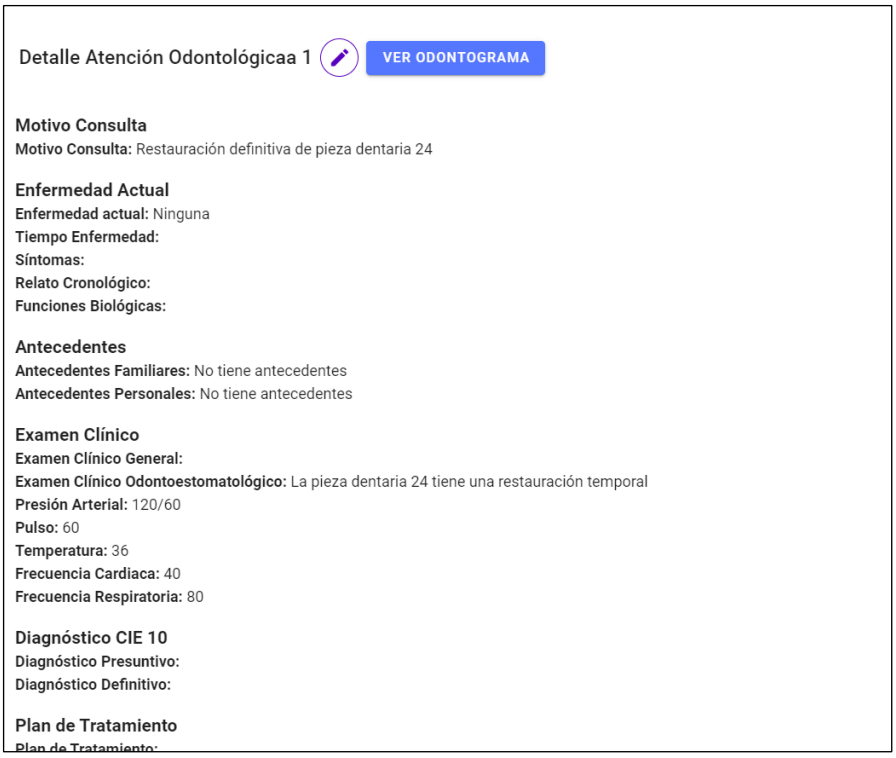


Figura 11. Detalle de la atención odontológica

Anexo 7: Ficha de prueba de carga – Test de carga

<table>
<tr><th colspan="2">Información global del caso de prueba: Arquitectura Software basada en Microservicios para mejorar la disponibilidad de HCO</th></tr>
<tr><td>Tipo de prueba</td><td>Prueba de carga</td></tr>
<tr><td>Nombre</td><td>Prueba de carga 01 – Consultar HCO por su identificador</td></tr>
<tr><td>Descripción de la prueba</td><td>Prueba de carga que garantiza que se obtienen los datos necesarios para analizar la eficiencia de la arquitectura de software basada en Microservicios.</td></tr>
<tr><td>Fecha de la prueba</td><td>10/12/2020</td></tr>
<tr><td>Escenario de negocio</td><td>Consultar HCO por su Identificador
Se va a consumir el API REST de HCO enviando el id de la HCO en la petición mediante el método GET, las peticiones consultarán HCO diferentes.

Número de usuario concurrentes: 2, 5, 10, 15, 20, 21, 22, 23 por segundo.
Tiempo de ejecución de la prueba: 60 Segundos por cada grupo de peticiones.</td></tr>
<tr><td>Prerrequisitos de la prueba</td><td>- El software que implementa la arquitectura basada en microservicios debe estar desplegado en el servidor descrito en insumos y se debe tener un token de acceso para realizar las pruebas.</td></tr>
<tr><td>Insumos de la prueba</td><td>Hardware
<table>
<tr><th colspan="2">CONTENEDOR DE DESPLIEGUE DEL SOFTWARE</th></tr>
<tr><th>CARACTERISTICA</th><th>VALOR</th></tr>
<tr><td>Nombre del Servicio</td><td>B1 APP Services – Microsoft Azure</td></tr>
<tr><td>Costo</td><td>14.60 USD mensual</td></tr>
<tr><td>Velocidad Procesador</td><td>1 Core 3,7 GHz</td></tr>
<tr><td>Memoria RAM</td><td>1.75GB</td></tr>
<tr><td>Almacenamiento</td><td>10GB</td></tr>
<tr><td>Región</td><td>South Central US</td></tr>
<tr><td>Sistema Operativo</td><td>Windows</td></tr>
</table>
Software
<table>
<tr><th colspan="2">SOFTWARE DE APLICACION</th></tr>
<tr><th>CARACTERISTICA</th><th>VALOR</th></tr>
<tr><td>Pila de entorno</td><td>.Net Core 3.1 (LTS)</td></tr>
</table>
<table>
<tr><th>SOFTWARE DE BASE DE DATOS</th></tr>
</table></td></tr>
</table>

	CARACTERISTICA	**VALOR**
	Nombre del servicio	SQL Database DTU 5
	Gestor de base de datos	SQL Server
	Seguridad	Azure defender
	SOFTWARE DE PRUEBAS	
	CARACTERISTICA	**VALOR**
	Nombre	JMETER
		Microsoft Azure informes
Indicadores de aceptación	**Tiempo respuesta esperado**	20 transacciones por segundo, tiempo de respuesta esperado es de 2 segundos con una tolerancia de 3 segundos, es decir nunca debe exceder los 5 segundos.
	Disponibilidad	99%
	Rendimiento	Máximo de peticiones por minuto

Anexo 8: Informe de prueba de carga - Test de carga

Informe de resultados: Prueba de carga 01 – Consultar HCO por su identificador	
Fecha de la prueba	10/12/2020
Escenario de negocio	**Consultar HCO por su Identificador** Se va a consumir el API REST de HCO enviando el id de la HCO en la petición mediante el método GET, las peticiones consultarán HCO diferentes. **Número de usuario concurrentes:** 2, 5, 10, 15, 20, 21, 22, 23 por segundo. **Tiempo de ejecución de la prueba:** 60 Segundos por cada grupo de peticiones.

Resultado de la prueba de carga

Peticiones/Segundo	Peticiones totales	Tiempo promedio (ms)	Min (ms)	Max (ms)	Error %
2	120	849	637	1354	0.00%
5	300	748	629	1562	0.00%
10	600	696	618	1629	0.00%
15	900	686	595	1566	0.00%
20	1200	680	590	1587	0.00%
21	1260	662	598	1126	0.00%
22	1320	**4393**	591	46414	0.16%
23	1380	**13765**	590	86685	**16.72%**

Conclusiones de la prueba

Esta prueba permite demostrar que la arquitectura puede responder hasta 20 peticiones en un segundo determinado cumpliendo de esta manera con el indicador de aceptación que indica que debe permite la disponibilidad de hasta 20 HCO en 1 segundo; vemos además que incluso puede responder hasta 21 peticiones dentro del tiempo de 1 segundo; cuando la arquitectura recibe más de 21 peticiones ya tarda más de 1 segundo en resolver estas peticiones.

El resultado del test de carga referente al indicador de disponibilidad logra determinar un 100% de disponibilidad de una HCO con la arquitectura de software propuesta para disponer 20 HCO cumpliendo de esta manera el indicador de aceptación de disponibilidad, además se puede verificar que la arquitectura puede disponer de hasta 21 HCO sin ningún margen de error y a partir de 22 HCO ya se tendría peticiones resueltas de manera incorrecta.

Teniendo en cuenta que la arquitectura de software propuesta permite disponer de hasta 21 HCO con una disponibilidad del 100% se puede determinar que el rendimiento de la arquitectura propuesta es de 1260 HCO por minuto.

Gráficas del Servidor

A continuación, se muestran las gráficas de las estadísticas del servidor luego de aplicar la prueba de carga.

Anexo 9: Ficha de prueba de escalabilidad – Test escalabilidad

Información global del caso de prueba: Arquitectura Software basada en Microservicios para mejorar la disponibilidad de HCO	
Tipo de prueba	Prueba de escalabilidad
Nombre	Prueba de escalabilidad 01 – Escalar el Microservicio Historia clínica odontológica
Descripción de la prueba	Prueba de escalabilidad del Microservicio Historia Clínica Odontológica para que pueda soportar el incremento de demanda de HCO por segundo en un tiempo determinado.
Fecha de la prueba	23/05/2021
Escenario de negocio	**Consultar 50 HCO por su Identificador** Se va a consumir el API REST de HCO enviando el id de la HCO en la petición mediante el método GET, las peticiones consultarán HCO diferentes. **Número de usuario concurrentes:** 20, 25, 30, 35, 40, 50, 55, 60, 65, 70 por segundo. **Tiempo de ejecución de la prueba:** 60 Segundos por cada grupo de peticiones.
Prerrequisitos de la prueba	- El software que implementa la arquitectura basada en microservicios debe estar desplegado en el servidor descrito en insumos y se debe tener un token de acceso para realizar las pruebas. - Se debe de configurar el escalado horizontal en el contenedor del Microservicio Historia Clínica Odontológica en el servicio de Microsoft Azure para que funcione hasta con 3 instancias cuando se tenga una alta demanda de peticiones.

Insumos de la prueba

Hardware

CONTENEDOR DE DESPLIEGUE DEL SOFTWARE	
CARACTERISTICA	**VALOR**
Nombre del Servicio	B1 APP Services – Microsoft Azure – Total ACU 100
Costo	54.75 USD mensual
Instancias máximo	3
Velocidad Procesador	1 Core 3,7 GHz
Memoria RAM	1.75GB
Almacenamiento	10GB
Región	North Central US
Sistema Operativo	Windows

Software

SOFTWARE DE APLICACION	
CARACTERISTICA	**VALOR**
Pila de entorno	.Net Core 3.1 (LTS)

SOFTWARE DE BASE DE DATOS	
CARACTERISTICA	**VALOR**
Nombre del servicio	SQL Database DTU 5
Gestor de base de datos	SQL Server
Seguridad	Azure defender

SOFTWARE DE PRUEBAS	
CARACTERISTICA	**VALOR**
Nombre	JMETER
	Microsoft Azure informes

Indicadores de aceptación	**Escalado esperado**	Se debe de escalar el Microservicio para soportar 50 peticiones por segundo, tiempo de respuesta esperado es de 2 segundos con una tolerancia de 3 segundos, es decir nunca debe exceder los 5 segundos, se debe tener una disponibilidad de al menos 99%. Se espera que se escale el microservicio hasta 3 instancias para responder todas las peticiones.

Anexo 10: Informe de prueba de escalabilidad – Test escalabilidad

Informe de resultados: Prueba de escalabilidad 01 – Escalar el Microservicio Historia clínica odontológica	
Fecha de la prueba	23/05/2020
Escenario de negocio	**Consultar 50 HCO por su Identificador** Se va a consumir el API REST de HCO enviando el id de la HCO en la petición mediante el método GET, las peticiones consultarán HCO diferentes. **Número de usuario concurrentes:** 20, 25, 30, 35, 40, 50, 55, 60, 65, 70 por segundo **Tiempo de ejecución de la prueba:** 60 Segundos por cada grupo de peticiones.

A. Resultado de la prueba de escalabilidad con 1 instancia del Microservicio

Se ha ejecutado la prueba con una sola instancia del Microservicio, para determinar en qué número de peticiones ya no se cumple con los indicadores de aceptación.

Peticiones/Segundo	Peticiones totales	Tiempo promedio (ms)	Min (ms)	Max (ms)	Error %
20	1200	680	590	1587	0.00%
25	1500	1122	918	3377	0.00%
30	1800	1102	909	3291	0.00%
35	2100	1160	896	3954	0.00%

40	2400	1069	899	3229	0.00%
50	3000	1122	908	2801	0.00%
55	3300	**1505**	899	**8345**	0.00%
60	3600	**1274**	917	**8625**	0.00%

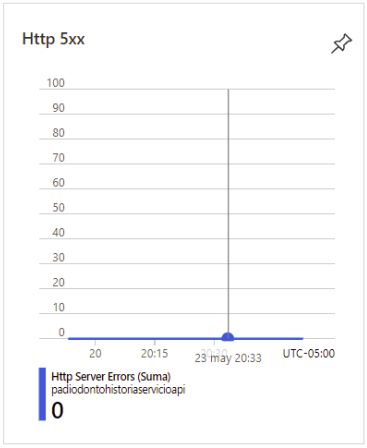
Conclusiones de la prueba

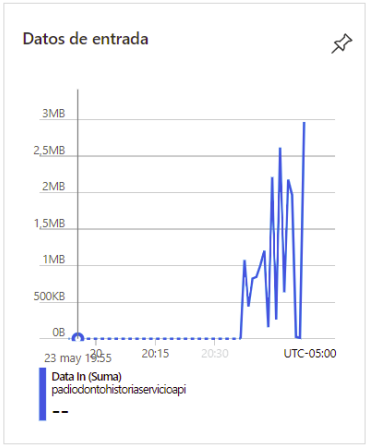
Esta prueba permite determinar en qué número de petición la arquitectura presenta inconvenientes para cumplir con los indicadores de aceptación

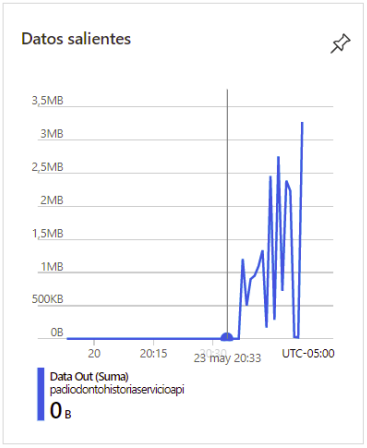
El resultado de esta primera prueba logro determinar que a partir de la petición número 25 se necesita más de un segundo para lograr resolver alguna petición, pero teniendo en cuenta el indicador de aceptación de hasta 2 segundos vemos que la arquitectura cumple este indicador; considerando el indicador de 5 segundos (teniendo en cuenta la tolerancia de 3 segundos) la arquitectura soporta hasta 50 peticiones por segundo, si deseamos resolver más de 50 peticiones con un retardo máximo de hasta 5 segundos es necesario escalar horizontalmente el microservicio de HCO.

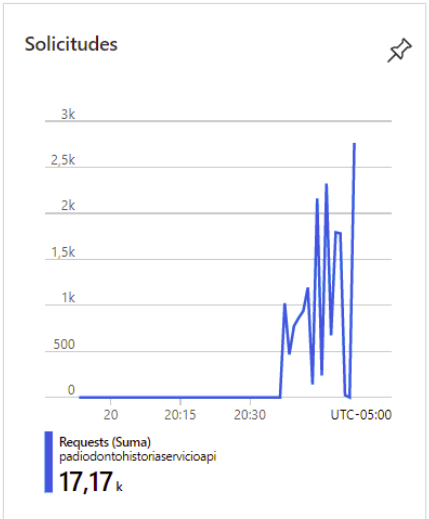
Gráficas del Servidor

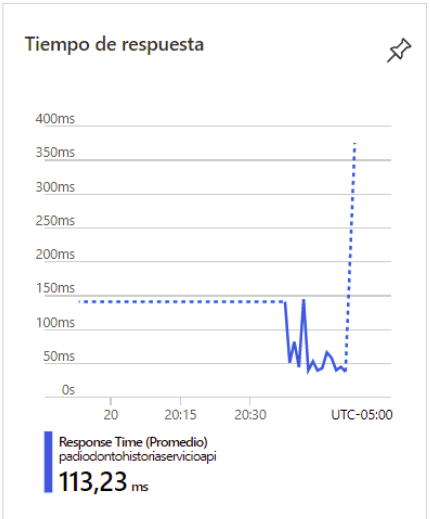
A continuación, se muestran las gráficas de las estadísticas del servidor luego de aplicar la prueba de escalabilidad para una instancia del microservicio.











B. Resultado de la prueba de escalabilidad con 2 instancias del Microservicio

Se ha ejecutado la prueba con dos instancias del Microservicio, para determinar en qué número de peticiones ya no se cumple con los indicadores de aceptación.

Peticiones/Segundo	Peticiones totales	Tiempo promedio (ms)	Min (ms)	Max (ms)	Error %
25	1500	1041	893	1776	0.00%
30	1800	1064	881	2801	0.00%
35	2100	1091	906	2428	0.00%
40	2400	1059	895	3262	0.00%
45	2700	1177	875	2374	0.00%
50	3000	1188	904	3563	0.00%
55	3300	1093	908	3332	0.00%
60	3600	1160	886	**4255**	0.00%
65	3900	1641	885	**21062**	**0.64%**
70	4200	**8339**	887	**21185**	**31.83%**

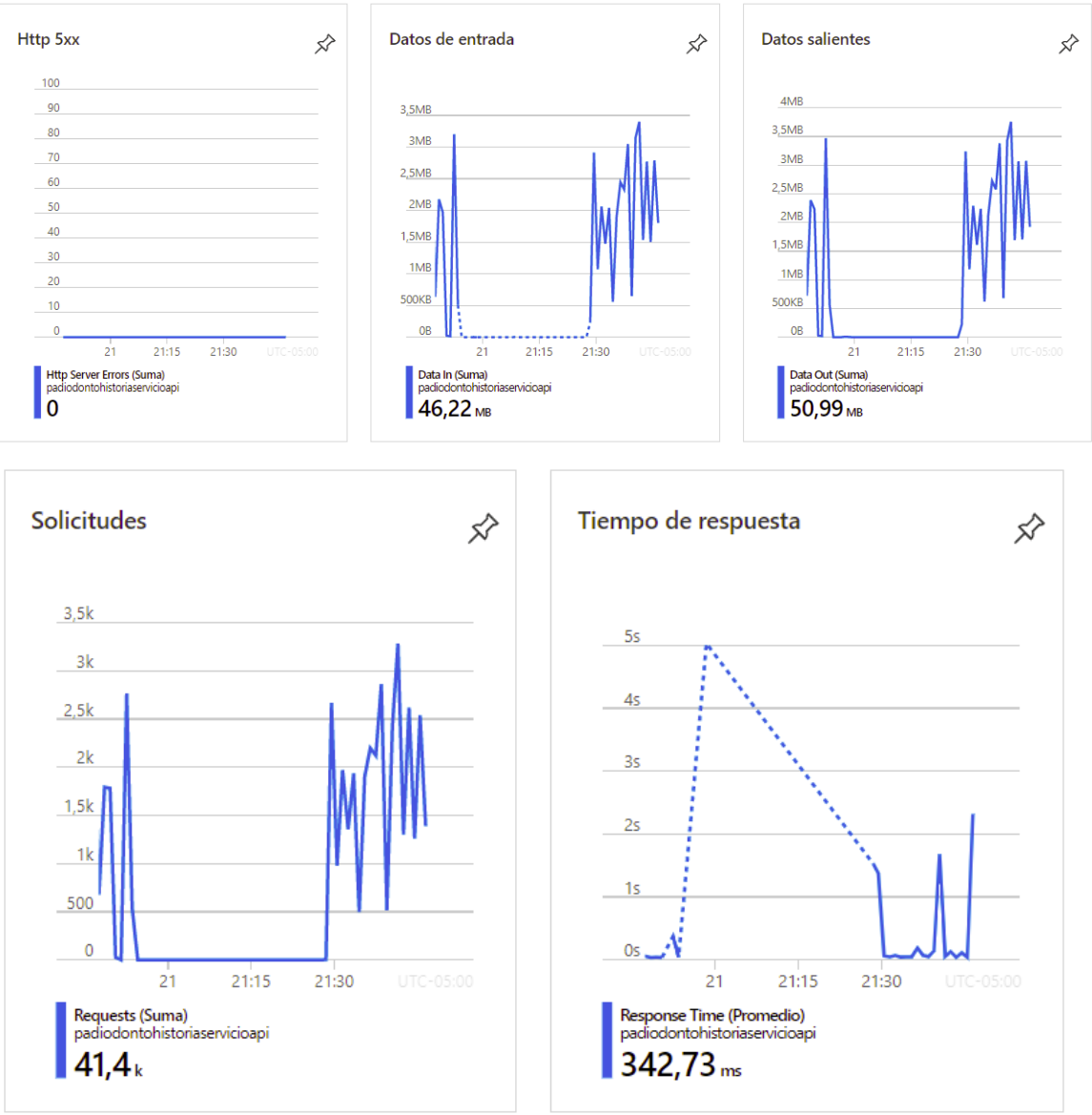
Conclusiones de la prueba

Esta prueba permite determinar hasta cuantas peticiones como máximo se pueden lograr resolver con dos instancias del microservicio de HCO cumpliendo con los criterios de aceptación.

El resultado de esta segunda prueba logro determinar que el microservicio de HCO con dos instancias permite resolver hasta 60 peticiones, considerando el indicador de 5 segundos (teniendo en cuenta la tolerancia de 3 segundos).

Gráficas del Servidor

A continuación, se muestran las gráficas de las estadísticas del servidor luego de aplicar la prueba de escalabilidad para dos instancias del microservicio de HCO.



Resultado de la prueba de escalabilidad con 3 instancias del Microservicio

Se ha ejecutado la prueba con tres instancias del Microservicio, para determinar en qué número de peticiones ya no se cumple con los indicadores de aceptación.

Peticiones/Segundo	Peticiones totales	Tiempo promedio (ms)	Min (ms)	Max (ms)	Error %
30	1800	1056	892	3045	0.00%
35	2100	1048	885	2655	0.00%
40	2400	1054	875	3143	0.00%
45	2700	1099	890	3379	0.00%
50	3000	1021	880	2628	0.00%

55	3300	1074	879	3061	0.00%
60	3600	1042	892	2779	0.00%
62	3720	1166	891	3974	0.00%
63	3780	1199	884	**6027**	0.00%
65	3900	2632	894	**21153**	**1.67%**
70	4200	**4638**	897	**21370**	**7.95%**

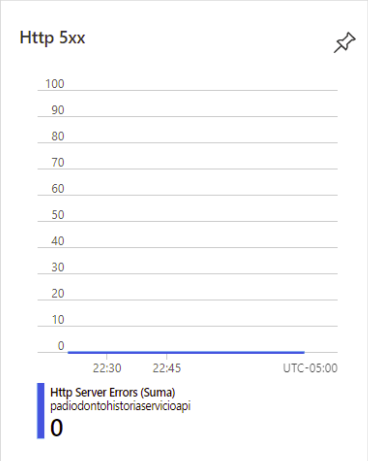
Conclusiones de la prueba

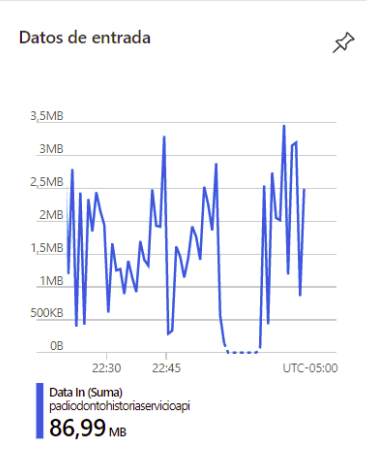
Esta prueba permite determinar hasta cuantas peticiones como máximo se pueden lograr resolver con tres instancias del microservicio de HCO cumpliendo con los criterios de aceptación.

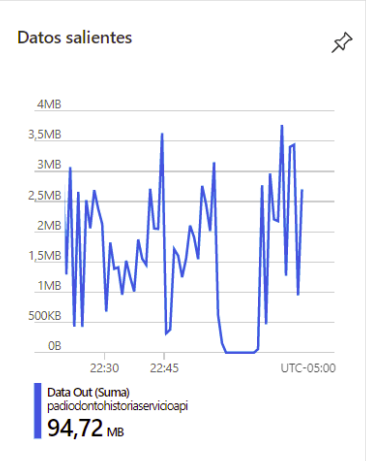
El resultado de esta tercera prueba logro determinar que el microservicio de HCO con tres instancias permite resolver hasta 62 peticiones, considerando el indicador de 5 segundos (teniendo en cuenta la tolerancia de 3 segundos).

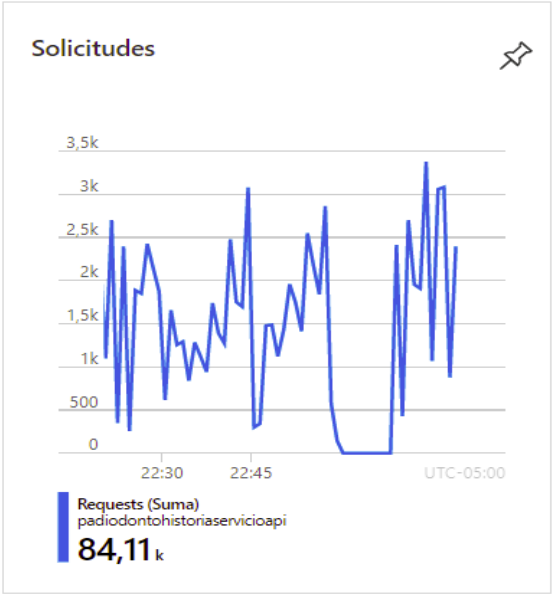
Gráficas del Servidor

A continuación, se muestran las gráficas de las estadísticas del servidor luego de aplicar la prueba de escalabilidad para 3 instancias.









Conclusiones finales

Con 3 instancias del microservicio de HCO se puede soportar 50 peticiones por segundo cumpliendo el indicador de aceptación de respuesta esperado de 2 segundos utilizando la tolerancia de 3 segundos, con una disponibilidad del 100%.

Si bien se puede soportar estas peticiones tan solo con dos instancias, es mejor escalar horizontalmente hasta 3 instancias para soportar de manera adecuada hasta 50 peticiones por segundo.

Anexo 11: Informe de prueba de caja negra - Test de caja negra

Se ha utilizado la prueba de caja negra para evaluar la Funcionalidad y Seguridad de la Arquitectura de Software desarrollada para mejorar la disponibilidad de HCO.

En la siguiente tabla mostramos los dos atributos de calidad evaluados.

FUNCIONALIDAD	La arquitectura debe permitir la gestión de HCO, odontogramas, paciente y proveedores de servicios odontológicos según formatos establecidos por el COP.
SEGURIDAD	El acceso a los datos de los pacientes, HCO y odontograma debe utilizar mecanismos de autenticación y autorización.

Para cada driver se han realizado las siguientes pruebas de caja negra.

SEGURIDAD	CP1. Acceso y generación de token
	CP2. Autorización para consultar HCO y odontograma
FUNCIONALIDAD	CP3. Gestión de Proveedores de Servicios Odontológicos
	CP4. Gestión de Pacientes
	CP5. Gestión de HCO y Odontograma según formato establecido

Casos de prueba.

Nombre	**Acceso y generación de Token**	**Código del caso de prueba**	**CP1**
Propósito	Verificar si un Proveedor de Servicios Odontológicos registrado puede obtener un Token de acceso.		
Pre requisitos	Datos de acceso de Proveedor de servicios odontológicos registrado.		
Ubicación	Consulta a Microservicio Proveedor Servicio Odontológico.		
Pasos	1. Acceder al prototipo de cliente Web desarrollado para poder realizar las peticiones al Microservicio Proveedor Servicio Odontológico. 2. Ir a la opción de “Acceder al sistema”.		
Ubicación	**Entrada/Acción del usuario/ Condición**	**Salida esperada**	**Estado**
/Security/Login **Método:** POST	El cliente Web envía las credenciales correctas de acceso del Proveedor de Servicios Odontológicos en formato JSON. `{ "email":"arcila@gmail.com", "password":"12345678" }`	Microservicio Proveedor Servicio Odontológico responde con código 200 y Token de Acceso	OK
/Security/Login **Método:** POST	El cliente Web envía las credenciales incorrectas de acceso del Proveedor de Servicios Odontológicos en formato JSON. `{ "email":"arcila@gmail.com", "password":"11111111" }`	Microservicio Proveedor Servicio Odontológico responde con código 404 y un mensaje Indicando que los datos de acceso son inválidos.	OK

Nombre	**Autorización para consultar HCO y odontograma**	**Código del caso de prueba**	**CP2**
Propósito	Validar la autorización mediante Token para que un Proveedor de Servicios Odontológicos puede consultar una HCO.		
Pre requisitos	Proveedor de servicios odontológicos accede al sistema, el Token de acceso se almacena en el cliente web que utiliza el Proveedor para acceder.		
Ubicación	Consulta a Microservicio HCO y Odontograma.		
Pasos	1. Acceder al prototipo de cliente Web desarrollado para poder realizar las peticiones al Microservicio Proveedor Servicio Odontológico. 2. Proveedor de Servicio Odontológico accede con sus credenciales. Se almacena el Token de acceso en el cliente web y se muestra la interfaz de trabajo del cliente web. 3. Ir a la opción de "Consultar HCO de Paciente". 4. Se consulta el paciente por su DNI y se muestra el listado de su Historial Odontológico.		
Ubicación	**Entrada/Acción del usuario/ Condición**	**Salida esperada**	**Estado**
/Historia/7 **Método:** GET	El cliente Web envía en su petición el id de la HCO del paciente a consultar, en la cabecera de la petición envía su token de acceso, este token debe ser válido y estar vigente. `let header={ "Authorization" : "Bearer " + this.$store.state.token };`	Microservicio Historia Odontológica responde con código 200 y la HCO en un formato JSON.	OK
/Historia/7 **Método:** GET	El cliente Web envía en su petición el id de la HCO del paciente a consultar, en la cabecera de la petición no se envía su token de acceso o bien este token es inválido o no está vigente. `let header={ };`	Microservicio Historia Odontológica responde con código 401 indicando que no está autorizado.	OK

Nombre	**Gestión de Proveedores de Servicios Odontológicos**	**Código del caso de prueba**	**CP3**
Propósito	Verificar el registro, actualización, cambio de password de los Proveedores de Servicios odontológicos.		
Pre requisitos	a. Para el registro de Proveedores de Servicios Odontológicos no se necesita tener un Token de acceso. b. Para la actualización y cambio de password de acceso del Proveedor de servicios odontológicos si es necesario que accede		

<table>
<tr><td></td><td colspan="3">al sistema, para generar el Token y este sea usado por el cliente web para realizar las peticiones.</td></tr>
<tr><td>Ubicación</td><td colspan="3">Consultas a Microservicio Proveedor Servicio Odontológico.</td></tr>
<tr><td>Pasos</td><td colspan="3">1a. Acceder al prototipo de cliente Web desarrollado y hacer click en la opción “Registrarse”.
1b. Acceder al prototipo de cliente Web desarrollado para poder realizar las peticiones al Microservicio Proveedor Servicio Odontológico.
2b. Proveedor de Servicio Odontológico accede con sus credenciales. Se almacena el Token de acceso en el cliente web y se muestra la interfaz de trabajo del cliente web.
3b. Ir a la opción “Mi perfil”.
4b. Seleccionar si desea cambiar sus datos o su password.</td></tr>
<tr><td>Ubicación</td><td>Entrada/Acción del usuario/ Condición</td><td>Salida esperada</td><td>Estado</td></tr>
<tr><td>/ProveedorServicio Odontologico

Método: POST</td><td>El Proveedor de Servicio Odontológico desea registrarse para esto ingresa sus datos necesarios en el formulario, el cliente Web envía una petición con los datos de registro de Proveedor de Servicio Odontológico en formato JSON.
<pre>{
"aPaterno":"Arcila",
"aMaterno":"Diaz",
"nombre":"Juan Carlos",
"documento":"47715777",
"codigo":"CO47715777",
"fechaNacimiento":"1990-01-07",
"sexo":"F",
"direccion":"Ca. Simón Bol 2267",
"telefono":"931742904",
"email":"arcila@gmail.com",
"password":"12345678"
}</pre></td><td>Microservicio Proveedor Servicio Odontológico responde con código 200 y muestra los datos del Proveedor registrado, la contraseña esta encriptada.</td><td>OK</td></tr>
<tr><td>/ProveedorServicio Odontologico/7

Método: PUT</td><td>El Proveedor de Servicio Odontológico desea actualizar sus datos para esto modifica los datos necesarios en el formulario, el cliente Web envía una petición con los datos actualizados de Proveedor de Servicio Odontológico en formato JSON, en la cabecera de la petición envía su token de acceso, este token debe ser válido y estar vigente.
<pre>{</pre></td><td>Microservicio Proveedor Servicio Odontológico responde con código 200 y muestra los datos del Proveedor actualizado.</td><td>OK</td></tr>
</table>

	<pre>"aPaterno":"Arcila", "aMaterno":"Diaz", "nombre":"Juan Carlos", "documento":"47715777", "codigo":"CO47715777", "fechaNacimiento":"1990-01-07", "sexo":"F", "direccion":"Ca. S.Bolívar 2267", "telefono":"931742904", "email":"arcila@gmail.com" }</pre> <pre>let header={ "Authorization" : "Bearer " + this.$store.state.token };</pre>	Si no se envía el Token responde con código 401.	
/ProveedorServicioOdontologico/CambiarPassword/7 **Método:** PUT	El Proveedor de Servicios Odontológicos desea actualizar su password para esto ingresa un nuevo password en el formulario, el cliente Web envía una petición con su password actualizado del Proveedor de Servicio Odontológico en formato JSON, en la cabecera de la petición envía su token de acceso, este token debe ser válido y estar vigente. <pre>{ "password":"12345679" }</pre> <pre>let header={ "Authorization" : "Bearer " + this.$store.state.token };</pre>	Microservicio Proveedor Servicio Odontológico responde con código 200. Si no se envía el Token responde con código 401.	OK

Nombre	**Gestión de Pacientes**	**Código del caso de prueba**	**CP4**
Propósito	Verificar el registro, actualización y consulta de los Pacientes teniendo en cuenta el formato establecido.		
Pre requisitos	Para el registro, actualización y consulta de pacientes es necesario que el Proveedor de servicios odontológicos acceda al sistema, para		

<table>
<tr><td></td><td colspan="3">generar el Token y este sea usado por el cliente web para realizar las peticiones.</td></tr>
<tr><td>Ubicación</td><td colspan="3">Consultas a Microservicio Paciente.</td></tr>
<tr><td>Pasos</td><td colspan="3">1. Acceder al prototipo de cliente Web desarrollado para poder realizar las peticiones al Microservicio Paciente.
2. Proveedor de Servicios Odontológicos accede con sus credenciales. Se almacena el Token de acceso en el cliente web y se muestra la interfaz de trabajo del cliente web.
3. Ir a la opción “Consultar HCO de paciente”.
4. Realizar la búsqueda de pacientes por su número de documento.</td></tr>
<tr><td>Ubicación</td><td>Entrada/Acción del usuario/ Condición</td><td>Salida esperada</td><td>Estado</td></tr>
<tr><td>/Paciente/getByDocumento/47715777

Método: GET</td><td>El Proveedor de Servicios Odontológicos consulta el documento de un paciente, para esto ingresa el documento del paciente a consultar en el formulario, el cliente Web envía una petición con el documento a consultar, en la cabecera de la petición envía su token de acceso, este token debe ser válido y estar vigente.
<pre>let header={
  "Authorization" : "Bearer "
  + this.$store.state.token
  };</pre></td><td>Si el paciente existe el Microservicio Paciente responde con código 200 y muestra los datos del paciente en formato JSON.
Si el paciente no existe el Microservicio Paciente responde con código 404.
Si no se envía el Token responde con código 401.</td><td>OK</td></tr>
<tr><td>/Paciente

Método: POST</td><td>El Proveedor de Servicios Odontológicos desea registrar los datos de un paciente para esto ingresa los datos necesarios en el formulario, el cliente Web envía una petición con los datos del formulario en formato JSON, en la cabecera de la petición envía su token de acceso, este token debe ser válido y estar vigente.
<pre>{
  "aPaterno":"Valdivia",
  "aMaterno":"Salazar",
  "nombre":"Carlos",
  "documento":"45698717",
  "fechaNacimiento":"1980-08-04",
  "sexo":"M",</pre></td><td>Microservicio Paciente responde con código 200 y devuelve los datos del paciente registrado en formato JSON.
Si no se envía el Token responde con código 401.</td><td>OK</td></tr>
</table>

	<pre>"lugarNacimiento":"Chiclayo", "direccion":"Miraflores 143", "procedencia":"Chiclayo", "ocupacion":"Ingeniero", "telefono":"963214853", "email":"cvaldivia@gmail.com", "contactoEmergencia":"956321423" }</pre> <pre>let header={ "Authorization" : "Bearer " + this.$store.state.token };</pre>		
/Paciente/8 **Método:** PUT	El Proveedor de Servicios Odontológicos desea actualizar los datos de un paciente para esto modifica los datos en el formulario, el cliente Web envía una petición con los datos del formulario en formato JSON, en la cabecera de la petición envía su token de acceso, este token debe ser válido y estar vigente. <pre>{ "aPaterno":"Valdivia", "aMaterno":"Salazar", "nombre":"Carlos Alberto", "documento":"43698717", "fechaNacimiento":"1980-08-04", "sexo":"M", "lugarNacimiento":"Chiclayo", "direccion":"Miraflores 143", "procedencia":"Chiclayo", "ocupacion":"Ingeniero", "telefono":"963214853", "email":"cvaldivia@gmail.com", "contactoEmergencia":"956321423" }</pre>	Microservicio Paciente responde con código 200 y devuelve los datos del paciente actualizado en formato JSON. Si no se envía el Token responde con código 401.	OK

	`` let header={ "Authorization" : "Bearer " + this.$store.state.token }; ```		

Nombre	**Gestión de HCO y Odontograma según formato establecido**	**Código del caso de prueba**	**CP5**
Propósito	Verificar el registro, actualización y consulta de HCO y odontogramas.		
Pre requisitos	Para el registro, actualización y consulta de HCO y odontogramas es necesario que el Proveedor de servicios odontológicos acceda al sistema, para generar el Token y este sea usado por el cliente web para realizar las peticiones.		
Ubicación	Consultas a Microservicio Historia Clínica Odontológica y Odontograma.		
Pasos	1. Acceder al prototipo de cliente Web desarrollado para poder realizar las peticiones al Microservicio HCO y Odontograma. 2. Proveedor de Servicio Odontológico accede con sus credenciales. Se almacena el Token de acceso en el cliente web y se muestra la interfaz de trabajo del cliente web. 3. Ir a la opción “Consultar HCO de paciente”. 4. Realizar la búsqueda de pacientes por su número de documento y consultas sus atenciones odontológicas.		
Ubicación	**Entrada/Acción del usuario/ Condición**	**Salida esperada**	**Estado**
/Historia/7 **Método:** GET	El Proveedor de Servicios Odontológicos Selecciona una HCO de un paciente para poder revisarla, las HCO tienen un estado que puede tener 3 valores posibles: Registre esta HCO, HCO privada y HCO pública; las HCO privadas solo pueden ser consultadas por el Proveedor que la registro. El cliente Web envía una petición con el código de la HCO a consultar, en la cabecera de la petición envía su token de acceso, este token debe ser válido y estar vigente. ``` let header={ "Authorization" : "Bearer " + this.$store.state.token }; ```	El Microservicio HCO responde con código 200, responde con los datos de la HCO y su odontograma actual en un formato JSON. Si no se envía el Token responde con código 401.	OK
/Historia/ /Odontograma	El Proveedor de Servicio Odontológico desea registrar los datos de atención del su Historial odontológico, para esto	Microservicio HCO responde con código 200	OK

Método: POST	ingresa los datos necesarios en el formulario siguiendo el formato establecido, además actualiza el odontograma del paciente siguiendo el formato establecido mediante una interfaz gráfica, el cliente Web envía una petición con los datos del formulario en formato JSON, en la cabecera de la petición envía su token de acceso, este token debe ser válido y estar vigente. <pre>{ "fechaRegistro":"2020-12-08", "motivoConsulta":"Corona temp oral de pieza dentaria 46", "enfermedadActual":"Ninguna", "tiempoEnfermedad":"", "sintomas":"", "relatoCronologico":"", "funcionesBiologicas":"", "antecedentesFamiliares":"No tiene antecedentes", "antecedentesPersonales":"No tiene antecedentes", "presionArterial":"120/60", "pulso":60, "temperatura":36, "frecuenciaCardiaca":40, "frecuenciaRespiratoria":80, "examenClinicoGeneral":"", "examenClinicoOdontologico":" La pieza dentaria 24 tiene una re stauración temporal ", "diagnosticoPresuntivo":"", "diagnosticoDefinitivo":"", "planTratamiento":"", "pronostico":"", "tratamiento":"", "evolucion":"", "fechaAlta":"2020-12-08",</pre>	y devuelve los datos de la HCO registrada en formato JSON. Microservicio Odontograma responde con un código 200. Si no se envía el Token responde con código 401.	

	<pre>"estado":"Registrado", "ambito":true, "pacienteId":2, "proveedorServicioId":1 } { "fechaRegistro":"2020-11-30", "historiaId":17, "detalles": [{ "diente":53, "ubicacion":"B", "situacionDiente":"Di ente extruído" }, { "diente":51, "ubicacion":"T", "situacionDiente":"Co rona Temporal" }] }</pre> <pre>let header={ "Authorization" : "Bearer " + this.$store.state.token };</pre>		
/Historia/17 /Odontograma/get ByHistoriaId/17 **Método:** PUT	El Proveedor de Servicio Odontológico actualiza los datos de un Historial odontológico para esto modifica los datos en el formulario, también puede actualizar los datos del odontograma desde una interfaz gráfica, el cliente Web envía una petición con los datos del formulario y de la interfaz en formato JSON, en la cabecera de la petición envía su token de acceso, este token debe ser válido y estar vigente. <pre>{ "fechaRegistro":"2020-12-08", "motivoConsulta":"Corona temp oral de pieza dentaria 46", "enfermedadActual":"Ninguna", "tiempoEnfermedad":"", "sintomas":"",</pre>	Microservicio HCO responde con código 200 y devuelve los datos de la HCO actualizada en formato JSON. Microservicio Odontograma responde con un código 200. Si no se envía el Token responde con código 401.	OK

```
    "relatoCronologico":"",
    "funcionesBiologicas":"",
    "antecedentesFamiliares":"No
tiene antecedentes",
    "antecedentesPersonales":"No
tiene antecedentes",
    "presionArterial":"120/60",
    "pulso":60,
    "temperatura":36,
    "frecuenciaCardiaca":40,
    "frecuenciaRespiratoria":80,
    "examenClinicoGeneral":"",
    "examenClinicoOdontologico":"
La pieza dentaria 24 tiene una re
stauración temporal ",
    "diagnosticoPresuntivo":"",
    "diagnosticoDefinitivo":"",
    "planTratamiento":"",
    "pronostico":"",
    "tratamiento":"",
    "evolucion":"Satisfactoria",
    "fechaAlta":"2020-12-08",
    "estado":"Registrado",
    "ambito":true,
    "pacienteId":2,
    "proveedorServicioId":1
}
{
    "fechaRegistro":"2020-11-30",
    "historiaId":17,
    "detalles": [
        {
            "diente":53,
            "ubicacion":"B",
            "situacionDiente":"Di
ente extruído"
        },
        {
            "diente":51,
            "ubicacion":"T",
            "situacionDiente":"Co
rona Temporal"
```

```
        },
        {
            "diente":50,
            "ubicacion":"T",
            "situacionDiente":"Co
rona Definitiva"
        }

    ]
}
```

```
let header={
  "Authorization" : "Bearer "
  + this.$store.state.token
  };
```

Anexo 12: Resultados de encuesta

En la siguiente tabla se muestra las preguntas que conforman la encuesta ejecutada para analizar el estado actual de la gestión de Historias Clínicas Electrónicas Odontológicas de la provincia de Chiclayo, revisar el detalle de la encuesta en el anexo 1.

ENCUESTA REALIZADA A LOS CIRUJANOS DENTISTAS DE LA PROVINCIA DE CHICLAYO
Preguntas
A. Almacenamiento de las Historias Clínicas Odontológicos
P1. En la clínica o establecimiento de salud donde usted labora se registra información de las historias clínicas Odontológicas de sus pacientes. P2. Almacena manualmente la información de las Historias Clínicas de sus pacientes. P3. Almacena digitalmente la información de las Historias Clínicas Odontológicas de sus pacientes, considere digitalmente a un sistema informático, documento digital de texto, hoja de cálculo o base de datos. P4. De la información de la Historia Clínica Odontológica que usted registra se entrega una copia al paciente.
B. Consulta de las Historias Clínicas Odontológicas
P5. ¿Cuándo asiste un paciente a atenderse a su clínica o establecimiento de salud ¿Es necesario consultar su historial médico? P6. El paciente le brinda la información adecuada para que usted pueda hacer su diagnóstico sin perder mucho tiempo.

P7. Se ha presentado inconvenientes porque no se tiene la información de Historia Clínica Odontológica completa de su paciente.

P8. ¿Estaría dispuesto a compartir la información de las Historias Clínicas Odontológicas de sus pacientes con otros proveedores de servicios odontológicos?

P9. Considera de utilidad que las Historias Clínicas Odontológicas se encuentren almacenadas en un repositorio digital seguro y confiable para que puedan ser consultadas por el especialista propietario y autorizado en el momento oportuno.

Escala
1) Nunca 2) A veces 3) Siempre

Resultados									
Encuestado	**P1**	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**
1	3	3	1	1	3	3	3	3	3
2	1	1	1	1	2	1	2	1	1
3	2	2	1	1	2	2	2	3	3
4	2	2	1	1	3	2	2	2	2
5	2	2	1	1	2	1	1	3	2
6	2	2	1	1	3	2	2	2	3
7	3	2	1	1	2	2	2	2	3
8	2	2	1	1	2	2	2	3	3
9	1	3	1	1	3	3	2	3	3
10	1	2	1	1	1	1	2	1	1
11	3	3	1	3	3	3	3	3	2
12	1	3	1	1	3	3	2	3	2
13	2	2	3	3	3	2	2	3	3
14	1	2	1	1	1	1	2	1	3
15	3	3	1	2	3	2	2	3	3
16	3	3	1	1	2	2	2	3	3
17	1	3	1	1	3	3	2	3	3
18	3	2	1	1	2	2	2	2	3
19	3	3	1	2	3	2	2	3	3
20	2	2	3	3	3	2	2	3	3
21	2	2	1	1	3	2	2	2	3
22	3	3	1	1	3	2	2	3	3
23	1	2	1	1	1	1	2	1	3
24	2	2	1	1	3	2	2	2	2
25	1	1	1	1	2	1	2	1	1
26	3	3	1	1	3	2	2	3	3
27	3	3	1	1	3	3	3	3	3

28	1	3	1	1	3	3	2	3	3
29	1	1	1	1	2	2	2	1	1
30	3	3	1	2	3	2	2	3	3
31	2	2	3	2	3	2	2	3	3
32	3	3	1	1	2	2	2	3	3
33	2	2	1	1	3	2	2	2	3
34	2	2	3	1	3	2	2	3	3
35	3	3	3	2	3	2	2	2	3
36	2	2	2	1	3	2	2	2	2
37	1	3	1	1	3	3	2	1	3
38	3	3	3	2	3	2	2	2	3
39	2	2	1	1	2	2	2	3	3
40	3	2	1	1	2	2	2	2	3
41	3	3	1	1	2	2	2	3	3
42	2	2	2	1	3	2	2	2	2
43	2	2	2	1	3	2	2	2	2
44	2	2	1	1	2	1	1	3	2
45	2	2	2	1	3	2	2	2	2
46	1	3	1	1	3	3	2	1	3
47	3	3	1	3	3	3	3	3	3
48	2	2	1	1	3	2	2	2	3
49	3	3	3	2	3	2	2	3	3
50	3	3	1	1	3	2	2	3	3
51	2	2	1	1	2	1	1	3	2
52	2	2	1	1	2	2	2	3	3
53	3	3	3	2	3	2	2	2	3
54	2	2	1	1	2	1	1	3	1
55	3	3	1	2	2	2	2	3	3
56	2	2	2	1	3	2	2	2	3
57	3	3	1	1	2	2	2	3	3
58	1	3	1	1	3	3	2	1	3
59	1	1	1	1	2	2	2	3	1
60	2	2	2	1	3	2	2	2	3
61	2	2	1	1	2	2	2	3	3
62	2	2	1	1	2	2	2	3	3
63	2	2	2	1	3	2	2	2	3
64	2	2	1	1	2	2	2	3	3
65	2	2	1	1	2	2	2	3	3
66	2	2	1	1	2	1	1	3	1
67	1	2	1	1	1	1	2	2	3
68	3	3	1	1	3	2	2	3	3
69	1	3	1	1	3	3	2	1	3
70	3	3	1	3	3	3	3	3	3
71	1	1	1	1	2	2	2	3	3
72	1	2	1	1	1	1	2	2	3
73	2	2	2	1	3	2	2	2	3

74	3	3	3	2	3	2	2	2	3
75	1	3	1	1	3	3	2	1	3
76	3	2	1	1	2	2	2	2	3
77	1	3	1	1	3	3	2	1	3
78	2	2	1	1	3	2	2	2	3
79	3	3	1	1	2	2	2	3	3
80	3	3	3	2	2	2	2	3	3
81	3	3	3	1	3	2	2	3	3
82	2	2	1	1	2	2	2	3	3
83	2	2	3	1	3	2	2	3	3

Anexo 13: Análisis de resultados de encuesta

Nombre encuesta	ENCUESTA REALIZADA A LOS CIRUJANOS DENTISTAS DE LA PROVINCIA DE CHICLAYO
Fecha	Noviembre 2020
Objetivo	Conocer la forma de trabajo de los proveedores de servicios odontológicos respecto al manejo de Historias Clínicas Odontológicas en la Provincia de Chiclayo.
Dirigido a	Cirujanos dentistas de la Provincia de Chiclayo
Preguntas	A. Almacenamiento de las Historias Clínicas Odontológicos
	P1. En la clínica o establecimiento de salud donde usted labora se registra información de las historias clínicas Odontológicas de sus pacientes. P2. Almacena manualmente la información de las Historias Clínicas de sus pacientes. P3. Almacena digitalmente la información de las Historias Clínicas Odontológicas de sus pacientes, considere digitalmente a un sistema informático, documento digital de texto, hoja de cálculo o base de datos. P4. De la información de la Historia Clínica Odontológica que usted registra se entrega una copia al paciente.
	B. Consulta de las Historias Clínicas Odontológicas
	P5. ¿Cuándo asiste un paciente a atenderse a su clínica o establecimiento de salud ¿Es necesario consultar su historial médico? P6. El paciente le brinda la información adecuada para que usted pueda hacer su diagnóstico sin perder mucho tiempo. P7. Se ha presentado inconvenientes porque no se tiene la información de Historia Clínica Odontológica completa de su paciente. P8. ¿Estaría dispuesto a compartir la información de las Historias Clínicas Odontológicas de sus pacientes con otros proveedores de servicios odontológicos? P9. Considera de utilidad que las Historias Clínicas Odontológicas se encuentren almacenadas en un repositorio digital seguro y

<table>
<tr><td></td><td>confiable para que puedan ser consultadas por el especialista propietario y autorizado en el momento oportuno.</td></tr>
<tr><td>Escala respuestas</td><td>1) Nunca 2) A veces 3) Siempre</td></tr>
<tr><td>Herramienta Utilizada</td><td>IBM SPSS Statistics 25</td></tr>
<tr><td>Nivel de fiabilidad</td><td>
Reliability Statistics
<table>
<tr><th>Cronbach's Alpha</th><th>N of Items</th></tr>
<tr><td>.755</td><td>9</td></tr>
</table>
El Alfa de Cronbach obtenido es de 0.755 demostrando una fiabilidad de la encuesta aplicada.
</td></tr>
<tr><td>Resultados</td><td>
P1.
<table>
<tr><th>Respuesta</th><th>Frecuencia absoluta</th><th>Frecuencia relativa</th></tr>
<tr><td>1</td><td>20</td><td>24.1</td></tr>
<tr><td>2</td><td>34</td><td>41.0</td></tr>
<tr><td>3</td><td>29</td><td>34.9</td></tr>
<tr><td>Total</td><td>83</td><td>100.0</td></tr>
</table>
Conclusión: El 34.9% de los encuestados almacena sus HCO siempre, el 41% lo almacena a veces y el 24.1% no almacena sus HCO.

P2.
<table>
<tr><th>Respuesta</th><th>Frecuencia absoluta</th><th>Frecuencia relativa</th></tr>
<tr><td>1</td><td>5</td><td>6.0</td></tr>
<tr><td>2</td><td>43</td><td>51.8</td></tr>
<tr><td>3</td><td>35</td><td>42.2</td></tr>
<tr><td>Total</td><td>83</td><td>100.0</td></tr>
</table>
Conclusión: El 42.2% de los encuestados almacena sus HCO manualmente siempre, el 51.8% lo almacena a veces y el 6% no almacena sus HCO manualmente.

P3.
<table>
<tr><th>Respuesta</th><th>Frecuencia absoluta</th><th>Frecuencia relativa</th></tr>
<tr><td>1</td><td>63</td><td>75.9</td></tr>
<tr><td>2</td><td>8</td><td>9.6</td></tr>
<tr><td>3</td><td>12</td><td>14.5</td></tr>
<tr><td>Total</td><td>83</td><td>100.0</td></tr>
</table>
Conclusión: El 14.5% de los encuestados almacena sus HCO digitalmente siempre, el 75.9% no almacena sus HCO digitalmente.
</td></tr>
</table>

P4.

Respuesta	Frecuencia absoluta	Frecuencia relativa
1	67	80.7
2	11	13.3
3	5	6.0
Total	83	100.0

Conclusión: El 80.7% de los encuestados nunca entrega una copia de su HCO al paciente, solo el 6% entrega una copia de su HCO.

P5.

Respuesta	Frecuencia absoluta	Frecuencia relativa
1	5	6.0
2	30	36.1
3	48	57.8
Total	83	100.0

Conclusión: El 57.8% de los encuestados indico que siempre es necesario consultar la HCO del paciente y solo el 6% indico que nunca es necesario consultar la HCO.

P6.

Respuesta	Frecuencia absoluta	Frecuencia relativa
1	12	14.5
2	56	67.5
3	15	18.1
Total	83	100.0

Conclusión: El 18.1% de los encuestados indico que los pacientes siempre brindan información necesaria sobre sus atenciones y el 14.5% indico que el paciente nunca brinda la información necesaria.

P7.

Respuesta	Frecuencia absoluta	Frecuencia relativa
1	5	6.0
2	73	88.0
3	5	6.0
Total	83	100.0

Conclusión: El 88.0% de los encuestados indico que a veces se tiene inconvenientes debido a que no se tiene la información de HCO de sus pacientes.

P8.

Respuesta	Frecuencia absoluta	Frecuencia relativa
1	12	14.5
2	25	30.1
3	46	55.4
Total	83	100.0

Conclusión: El 55.4% de los encuestados indico que estaría dispuesto a compartir la información de las HCO de sus pacientes siempre con otros proveedores de servicios odontológicos, el 30.1% indico que a veces podría compartir la información de sus pacientes y el 14.5% indico que nunca estaría dispuesto a compartir la información de HCO.

P9.

Respuesta	Frecuencia absoluta	Frecuencia relativa
1	7	8.4
2	11	13.3
3	65	78.3
Total	83	100.0

Conclusión: El 78.3% de los encuestados considera de utilidad que las HCO se encuentren almacenadas en un repositorio digital seguro y confiable para que puedan ser consultadas por el especialista propietario y autorizado en el momento oportuno.