Desarrollo de una arquitectura basada en microservicios para facilitar el uso y mantenibilidad del código de los proyectos de software del área de programación de la empresa Arest Consulting.

Tigse Pilla, Christian Mauricio

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnológica

Centro de Posgrados

Maestría en Ingeniería en Software

Trabajo de titulación, previo a la obtención de título de Magíster en Ingeniería en Software

Msc. Guevara Vega, Cathy Pamela

01 de diciembre del 2022

Latacunga

Ouriginal

Document Information

Analyzed document	Tesis_Ultima.docx (1).pdf (D151833504)
Submitted	12/2/2022 4:28:00 PM
Submitted by	
Submitter email	tigse.christian@gmail.com
Similarity	3%
Analysis address	cguevara.utn@analysis.urkund.com

Sources included in the report

SA	**UNIVERSIDAD TÉCNICA DEL NORTE / GuitarraZamia_Tesis.pdf** Document GuitarraZamia_Tesis.pdf (D149946814) Submitted by: zmguitarrad@utn.edu.ec Receiver: aquina.utn@analysis.urkund.com	1
SA	**UNIVERSIDAD TÉCNICA DEL NORTE / UTN_MIS_Tesis-DLH.docx** Document UTN_MIS_Tesis-DLH.docx (D31736650) Submitted by: daniel.lopez@asambleanacional.gob.ec Receiver: eamaya.utn@analysis.urkund.com	3
SA	**Tesis_Caiza_&_Guaigua_ver_3_3__full__-1.docx** Document Tesis_Caiza_&_Guaigua_ver_3_3__full__-1.docx (D109762340)	1
SA	**UNIVERSIDAD TÉCNICA DEL NORTE / TT Arquitectura de Software ESPE-Urkundok.pdf** Document TT Arquitectura de Software ESPE-Urkundok.pdf (D115414135) Submitted by: jrcaiza@espe.edu.ec Receiver: aquina.utn@analysis.urkund.com	1
W	URL: http://repositorio.utn.edu.ec/bitstream/123456789/10292/2/04%20ISC%20546%20TRABAJO%20GRADO.pdf Fetched: 4/10/2022 5:30:50 PM	9
SA	**UNIVERSIDAD TÉCNICA DEL NORTE / ProyectoTitulacion_Urkund.pdf** Document ProyectoTitulacion_Urkund.pdf (D100246656) Submitted by: paul.milcape@gmail.com Receiver: aquina.utn@analysis.urkund.com	3

Entire Document

MSc. Guevara Vega, Cathy Pamela
Director

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología

Centro de Posgrados

Certificación

Certifico que el trabajo de titulación: "**Desarrollo de una arquitectura basada en microservicios para facilitar el uso y mantenibilidad del código de los proyectos de software del área de programación de la empresa Arest Consulting**" fue realizado por el/los señor/señores **Tigse Pilla, Christian Mauricio;** el mismo que cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, además fue revisado y analizado en su totalidad por la herramienta de prevención y/o verificación de similitud de contenidos; razón por la cual me permito acreditar y autorizar para que se lo sustente públicamente.

Latacunga, 01 diciembre del 2022

MSc. Guevara Vega, Cathy Pamela

Director

C.C.: 100233483-5

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología
Centro de Posgrados

Responsabilidad de Autoría

Yo **Tigse Pilla, Christian Mauricio**, con cédula de ciudadanía n° **050287132-0**, declaro que el contenido, ideas y criterios del trabajo de titulación:"**Desarrollo de una arquitectura basada en microservicios para facilitar el uso y mantenibilidad del código de los proyectos de software del área de programación de laempresa Arest Consulting**" es de mí autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Latacunga, 01 diciembre del 2022

Ing. Tigse Pilla, Christian Mauricio
C.C.: 050287132-0

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnología

Centro de Posgrados

Autorización de Publicación

Yo **Tigse Pilla Christian Mauricio** con cédula/cédulas de ciudadanía n° **050287132- 0**, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: "**Desarrollo de una arquitectura basada en microservicios para facilitar el uso y mantenibilidad del código de los proyectos de software del área de programación de la empresa Arest Consulting**" en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi/nuestra responsabilidad.

Latacunga, 01 diciembre del 2022

Ing. Tigse Pilla, Christian Mauricio
C.C.: 050287132-0

DEDICATORIA

El presente trabajo le dedico primeramente a Dios quien me dio sabiduría y el entendimiento para seguir con mis objetivos, a mi Madre que fue mi motor fundamental que me impuso a seguir esforzándome y quien ha confiado en mí, a mi Padre por el apoyo que me dio para seguir creciendo profesionalmente, a mis Hermanos y a toda mi familia quienes me ayudaronen todo momento a tomar decisiones que fueron y serán importantes en mi vida los cuales me ha permitido conseguir mis objetivos.

A mis Abuelitas María Pascuala Velasque y Jesús Moposita, quienes siempre serán mi motivo para seguir adelante y no desmayar ante las adversidades y los problemas y cumplir con mis objetivos planteados.

Christian

AGRADECIMIENTO

Agradezco, primeramente, a Dios, por brindarme toda la sabiduría y fortaleza para seguir de pie ante toda adversidad. A mis Padres, quienes me brindaron la fortaleza para seguir adelante y poder cumplir una nueva etapa de mi vida, siendo siempre un apoyo incondicional y por haberme enseñado que con esfuerzo y perseverancia todo se consigue.

A toda mi familia por estar siempre a mi lado, apoyándome y aconsejándome.

Al Centro de Posgrados de la Universidad de las Fuerzas Armadas sede Latacunga quien me permitieron continuar con mis estudios, a mis profesores y compañeros que me acompañaron a concluir esta etapa y lograr especializarse

Y por último a la persona que ha depositado su entera confianza y me impulsó a seguir adelante para cumplir mi objetivo y meta, la cual me enseño que con esfuerzo, trabajo y constancia todo se consigue y no desfallecer en medio camino, la que me impuso a seguir creyendo en mí.

Christian

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Resumen

Las arquitecturas basadas en microservicios hacen que las aplicaciones sean más fáciles de escalar y más rápidas para el desarrollo y están compuestas con varias aplicaciones pequeñas de servicios, los cuales se ejecutan por su propio proceso y se comunicancon mecanismos ligeros, ya que son servicios independientes, reduciendo a los programadores una gran administración comprimiendo el tiempo de ejecución para toda la aplicación y para todo el equipo de desarrollo. Los microservicios son los responsables de obtener las propiedades de los archivos de lado a lado, esto es diferente del modelo tradicional o conocido como metodología monolítica tiene como características el uso de una base única para sus funciones y servicios donde este proceso con lleva a que se debe actualizar toda la aplicación para que tenga las actualizaciones necesarias causando molestias al cliente. Una metodología de microservicios tiene algunas ventajas, por ejemplo: A) Agilidad de implementación, puesto que todo es de forma independiente y esoresulta más fácil para la administración y corrección de errores. B) Equipos pequeños y organizados, por el hecho de que un microservicio tiene menor complejidad para ser compilado, probado e implementado. C) Base de código pequeño que permita la modificación en cualquier momento, ya que minimiza las dependencias y resulta más fácilde agregar nuevas cosas. D) Escalabilidad porque son independientes, permite escalar horizontalmente con subsistemas dentro de un microservicio. E) Mezcla de tecnologías se apartan a cualquier combinación de tecnología. Es un método de arquitectura que se basa en una serie de servicios se pueden ejecutarse de forma independiente y poseen su propia lógica empresarial y base de datos con un objetivo en común.

Palabras clave: Arquitectura de software, Microservicios, Mantenibilidad de software, Proyectos de software.

Abstract

Microservices-based architectures make applications easier to scale and faster to develop and are composed of several small applications of services, which are executed by their own process and communicate with lightweight mechanisms, as they are independent services, reducing programmers a large administration compressing the execution time for the entire application and for the entire development team. Microservices are responsible for obtaining the properties of the files from side to side, this is different from the traditional model or known as monolithic methodology has as characteristics the use of a single base for its functions and services where this process leads to the fact that the entire application must be updated to have the necessary updates causing inconvenience to the customer. A microservices methodology has some advantages, for example: A) Agility of implementation, since everything is independent and that is easier for the administration and correction of errors. B) Small and organized teams, due to the fact that a microservice has less complexity to be compiled, tested and implemented. C) Small code base that allows modification at any time, since it minimizes dependencies and it is easier to add new things. D) Scalability because they are independent, allowing to scale horizontally with subsystems within a microservice. E) Mix of technologies, they are apart to any combination of technology. It is a method of architecture that is based on a series of services that can be run independently and have their own business logic and database with a common goal.

Keywords: Software architecture, Microservices, Software maintainability, Software projects.

Capítulo I

Introducción

En el presente capítulo es para dar a conocer la introducción del proyecto de investigación, donde se manifiesta que se mantiene una arquitectura monolítica, en donde se genera problemas con la mantenibilidad del código fuente de los proyectos de sus aplicaciones.

Planteamiento del problema

La Empresa Arest Consulting de la Ciudad de Latacunga presta servicios de consultoría enel área informática y desarrolla e implementa software a medida. En la actualidad cuenta con aplicaciones monolíticas, en las cuales sean han observado dificultades en identificar y solucionar problemas concretos para el área de programación, como por ejemplo la dificultad de trabajar en varios ambientes de manera simultánea, además el crecimiento del código permite una sobrecarga de las aplicaciones informáticas limitando su agilidad y usabilidad.

En el área de programación se evidencia la dificultad en el costo alto de mantener el código fuente de las aplicaciones porque al momento de realizar un crecimiento o agregar varias funciones al mismo, se genera errores, los cuales resulta abrumador para los programadores, sabiendo que cualquier paso en falso puede comprometer el código en su conjunto.

Basándonos en las limitaciones antes descritas se plantea el siguiente problema: ¿Cómo contribuir en la facilidad de uso y mantenibilidad del código de los proyectos de software del área de programación de la Empresa Arest Consulting?

Antecedentes

Según Martínez, C. (2020) expresa que “La definición que le damos a la arquitectura viene del mundo de la construcción, pero viene a significar lo mismo en nuestro mundo, solo que nosotros no construimos edificios, construimos software”.

Villamizar, M, et al. (2015) manifiesta que “El estudio de las diferentes arquitecturas de software son un punto crucial dentro de la Ingeniería de Software, por lo cual el enfoque de arquitectura orientada a microservicios permite mejorar las dificulta desque atraviesa un enfoque monolítico tradicional”.

Balalaie, A., et al. (2016) muestra cómo se utilizan los microservicios y sus diferentes componentes en una arquitectura distribuida de una manera detallada con la utilización de un DevOps.

Salah, T., et al. (2016) muestra un análisis claro de cómo ha sido la evolución de este enfoque y los tipos de herramientas que se han venido usando, como Remote Procedure Call (RPC), Simple Object Access Protocol (SOAP), Representational StateTransfer (REST).

Según Levcovitz.R, (2016) expresa que “Las arquitecturas monolíticas son metodologías que contienen limitaciones y ocasionan problemas al momento de una implementación, mediante la aplicación de una arquitectura de microservicios se puede solventar la estabilidad, confiabilidad, escalabilidad, tolerancia a fallos, preparación para catástrofes, desempeño y monitoreo”.

De lo antes expuesto se puede mencionar que una arquitectura de microservicios es un enfoque de vanguardia que facilita el desarrollo de aplicaciones mediante un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y mecanismo ligeros de comunicación.

Objetivos

Objetivo General

Desarrollar una arquitectura basada en microservicios para facilitar el uso y mantenibilidad del código de los proyectos de software del área de programación de la empresa Arest Consulting.

Objetivos Específicos

- ✓ Establecer el marco teórico que fundamente la arquitectura orientada a microservicios.
- ✓ Desarrollo de una Arquitectura basada en Microservicios.
- ✓ Consumir la arquitectura de microservicios mediante un cliente móvil.
- ✓ Validar los resultados mediante las normas ISO/IEC 25000 de calidad externa

Justificación e importancia

En la actualidad, las empresas del sector público y privado se dedican a desarrollar sistemas de software, en su mayor parte mediante arquitecturas monolíticas, en donde se han generado inconvenientes, especialmente en la mantenibilidad del código fuente de las

aplicaciones informáticas, para suplir esa necesidad se desarrolla una arquitectura de microservicios para disminuir el tiempo de manipulación de datos y de esta manera automatizar los procesos internos de las mismas.

En el área de programación de la empresa Arest Consulting se desarrollan módulos, los mismos que están orientados con una arquitectura tradicional o monolítica, la cual incide en diferentes aspectos tecnológicos. Los problemas se visualizan cuando las aplicaciones entran a una etapa de mantenimiento, la cual conlleva un proceso de cambio o una nueva funcionalidad, lo que provoca diversos fallos dentro de las aplicaciones debido a que tienen todas las funcionalidades dentro de un componente común.

El presente proyecto tiene como objetivo desarrollar una arquitectura basada en microservicios para facilitar el uso y mantenibilidad del código de los proyectos de software del área de programación de la empresa Arest Consulting en el año 2020.

Hipótesis

Si se desarrolla una arquitectura basada en microservicios entonces se contribuye a la facilidad de uso y mantenibilidad del código de los proyectos de software del área de programación de la Empresa Arest Consulting.

Variable de la investigación

Dependiente

Se contribuye a la facilidad de uso y mantenibilidad del código de los proyectos de software del área de programación de la Empresa Arest Consulting.

Operacionalización de la variable dependiente (Indicadores)

- ✓ Madurez
- ✓ Disponibilidad
- ✓ Tolerancia a fallos
- ✓ Recuperabilidad

Independiente

Se desarrolla una arquitectura basada en microservicios, conceptualización de la variable independiente, la arquitectura en microservicios es el procedimiento del desarrollo de aplicaciones construidas con pequeños servicios independientes, con procesos bien definidos e independientes entre ellos.

Capítulo II

Marco teórico

En el presente capítulo se describen los siguientes temas, antecedentes históricos, los cuales describen la importancia de la calidad de software con el uso de las arquitecturas en microservicios. Los antecedentes Conceptuales se describen las arquitecturas de desarrollo de software y los Antecedentes Referenciales clasifica cada una de las arquitecturas tanto en microservicios o monolíticas.

Los Antecedentes Contextuales evidencia el problema que se encuentra en la empresa Arest Consulting con la utilización de una arquitectura monolítica en la utilización de sus módulos en el departamento de programación y facilitar el desarrollo, una arquitectura basada en microservicios para facilitar el uso y mantenibilidad del código de los proyectos de software del área de programación de la empresa Arest Consulting.

Antecedentes históricos

Según el estudio de Moreno, Jorge Jair, Bolaños, Liliam Paola, Navia, Manuel Alejandro(2010) La importancia de la calidad de software en la disciplina de la ingeniería del software es ampliamente reconocida en la actualidad; sin embargo, desde el punto de vista de los modelos y estándares hacia el producto, el desarrollo de estos durante décadas, la sobreabundancia de información, el alto costo y el acceso limitado a esta información, impiden un acercamiento de estos a los ingenieros de software en pro de la calidad del producto software al interior de la organización, (Moreno, Jorge Jair, Bolaños, Liliam Paola, Navia, Manuel Alejandro.2010,págs. 39-53).

De lo anterior expuesto se dice que el proceso de ingeniería de software permite el desarrollo de aplicaciones a largo tiempo de una manera ordenada, coherente, metodológica, teniendo como ejes principales el diseño arquitectónico de un sistema.

Antecedentes conceptuales y referenciales

Arquitecturas de software

Según Bass, Clements y Kazman mencionan que la arquitectura de software mediante el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) se define como una organización fundamental de un sistema, incorporando en sus componentes y relaciones entre sí en el medio ambiente y los principios que rigen su diseño y evolución. En la literatura, existen

diferentes definiciones respecto a la arquitectura de software, sin embargo, la definición que se usa con mayor frecuencia es que la arquitectura describe aspectos estructurales de un sistema software en particular, que comprende elementos importantes para el desarrollo de software, las propiedades son visibles de esos elementos y las relaciones entre ellos. (Bass, Clements, y Kazman, 2018).

Después de haber analizado la información de los autores, se puede recalcar que la arquitectura describe varios aspectos estructurales de un sistema, al igual que las restricciones sobre las arquitecturas como son Representational State Transfer (REST).

Arquitectura monolítica

La arquitectura monolítica permite que el software se estructure de forma que todos los aspectos funcionales quedan acoplados y sujetos en un mismo programa. En este tipo de sistema, toda la información está alojada en un servidor, por lo que no hay separación entre módulos y las diferentes partes de un programa están muy acopladas. (Astorga ,2020).

Al analizar la información se define una arquitectura monolítica donde se acopla a una sola base de codificación que se sujeta a varios módulos, ver Figura 1

Figura 1.

Esquema de arquitectura monolítica

Nota. La Figura represen como se compone una arquitectura monolítica, la cual describeuna sola estructura. Tomado de (Apiservice, 2020)

Arquitectura orientada a servicios (SOA)

Se denomina Arquitectura Orientada a Servicios (Service Oriented ArchitectureSOA) a un marco conceptual de arquitecturas informáticas de negocios que se caracteriza porofrecer las funcionalidades básicas de los Sistemas de Información de una empresa a través de servicios reutilizables, (Apiservice, 2020).

Desde el punto de vista de negocio, se define SOA como un conjunto de componentes informáticos que se integran de forma flexible para configurar distintos procesos de negocio, desde una perspectiva técnica, estas arquitecturas constan de servicios que se pueden invocar para realizar operaciones específicas, (Mezo, 2008).

El análisis de la información sobre la Arquitectura Orientada a Servicios (SOA) es aportar con la flexibilidad y la automatización de las infraestructuras donde en el diseño y desarrollo de las aplicaciones de sistemas ayudando a la gestión de grandes datos de información.

Arquitectura de microservicio

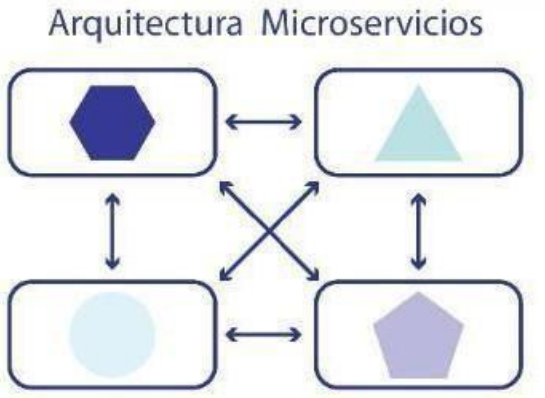
Es un enfoque para el desarrollo de una aplicación única como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y mecanismos ligeros de comunicación, a menudo un recurso de una interfaz de programación de aplicaciones (API) sobre protocolo de transferencia de hipertexto HTTP, (Maya, Edgary López, Daniel, 2018).

Estos servicios están construidos alrededor de las capacidades del negocio y con independencia de despliegue e implementación totalmente automatizada. Existe un mínimo de gestión centralizada de estos servicios, los que pueden estar escritos en lenguajes de programación diferentes y utilizar diferentes tecnologías de almacenamiento de datos, (Maya, Edgar y López, Daniel, 2018).

Un microservicio es un conjunto de servicios pequeños, cada uno de ellos se ejecutade manera independiente, es decir, cada uno tiene su propio proceso, la cual mejora los recursos de comunicación y de la programación, ver Figura 2.

Figura 2.

Esquema de arquitectura de microservicio.



Nota. Tomado de (Apiservice, 2020).

Representational state transfer (rest)

Representational State Transfer (REST) es una serie de restricciones que, al momento de ser plasmada en el diseño de software, crea un estilo arquitectónico. REST emplea la arquitectura cliente-servidor que se caracteriza por él envió de las peticiones hacia el servidor por parte del cliente y brinda su respuesta por parte del servidor. Además, cada solicitud ocupa servicios independientes brindando la información necesaria para ser utilizada, así el servicio no guarda un estado; los recursos son obtenidos por medio de una dirección única y válida (Sandoval, 2009).

La información recolectada respecto a REST nos menciona que utiliza la arquitectura cliente-servidor que se caracteriza por él envió de peticiones hacia el servidor porparte del cliente y respuestas por parte del servidor, (Sandoval. J,2009).

Con esto en mente, mira la aplicación distribuida de documentos arquitectónicos, estilos que comienzan con lo que él llama el espacio nulo, que representa la disponibilidad de cada tecnología y cada estilo de desarrollo de aplicaciones sin reglas, o termina con las siguientes restricciones que definen un sistema REST:

- ✓ Tiene que soportar un sistema de almacenamiento en caché, la infraestructura de la red debe apoyar el caché a diferentes niveles
- ✓ Tiene que ser accesible de manera uniforme
- ✓ Cada recurso debe tener una dirección única y un punto de acceso válido
- ✓ Tiene que ser en capas, debe apoyar la escalabilidad

Las abstracciones que hacen un sistema REST:

Las abstracciones de un sistema REST están enfocados en los siguientes aspectos, Según José Sandoval, (Sandoval. J,2009):

- ✓ **Recursos:** Un recurso de descanso es cualquier cosa que se pueda dirigir a través de la web. Posteriormente, un recurso es un mapeo lógico y temporal de un concepto en el problema para el cual estamos implementando una solución.

- ✓ **Representación:** Una representación es un flujo binario junto con sus metadatos que describe cómo debe ser consumido el flujo, ya sea por el cliente o el servidor. Por lo tanto,una representación puede adoptar diversas formas, como una imagen, un archivo de texto o un flujo XML o un flujo JSON, pero tiene que estar disponible a través del mismo URI. En el caso de las solicitudes generadas por el ser humano a través de un navegadorweb, una representación suele ser de manera de una página HTML. En el caso de las solicitudes automatizadas de otros servicios web, la legibilidad no es tan importante y se puede utilizar una representación más eficiente como XML, (Sandoval. J,2009).

- ✓ **UrI:** Un Identificador Uniforme de Recursos, o URI, en un servicio web de RESTFULes un hipervínculo a un y es el único medio para que los clientes y servidores intercambienrepresentaciones. El conjunto de restricciones de REST no dictan que las URIs deban serhipervínculos. En un sistema REST, la URI no está destinada a cambiar con el tiempo, yaque la arquitectura implementación es lo que gestiona los servicios, localiza los recursos,negociarlas representaciones, y luego envía respuestas con los recursos solicitados, (Sandoval. J,2009).

Interfaz de peticiones en HTTP

Son representaciones negociadas entre clientes y servidores a través del protocolo de comunicación en tiempo de ejecución a través de HTTP. En esta sección, miramos en detalle lo que significa intercambiar estas representaciones, y lo que significa para los clientes y servidores tomar acciones sobre estos recursos.

El desarrollo de los servicios web de RESTFUL es similar a lo que hemos estado haciendo hasta ahora con nuestras aplicaciones web. Sin embargo, la diferencia fundamental entre el desarrollo de aplicaciones web modernas y tradicionales es como pensamos en las acciones tomadas sobre nuestras abstracciones de datos.

En el desarrollo de aplicaciones web modernas limitamos el diseño y ambigüedad en la aplicación, porque tenemos cuatro medidas específicas que podemos tomar sobre los recursos - Crear, Recuperar, Actualizar y Borrar (CRUD) ver Figura3.

Figura 3.

Equivalencia de acciones de datos y operaciones HTTP

Data action	HTTP protocol equivalent
CREATE	POST
RETRIEVE	GET
UPDATE	PUT
DELETE	DELETE

Nota. Tomado de (Sandoval, 2009).

Detalle los cuatro tipos de peticiones HTTP y veamos cómo se utiliza cada uno de ellos para intercambiar representaciones para modificar el estado de los recursos que según (Sandoval, 2009) se definen:

- ✓ **Get/Retrieve:** El método get se utiliza para recuperar los recursos.
- ✓ **Post/Create:** El método POST se utiliza para crear recursos.
- ✓ **Put/Update:** El método put se utiliza para actualizar los recursos.
- ✓ **Delete/Delete:** El método borrar se utiliza para borrar representaciones, finalmente,al borrar un recurso se utiliza la misma URI que hemos usado en los otros tres casos.

Interfaz de programación de aplicaciones (API)

El término API es una abreviatura de Application Programming Interfaces, que en español significa interfaz de programación de aplicaciones. Se trata de un conjunto de programas que trabajan mediante protocolos de conexión y que se ocupan para el desarrollo de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas (Fernández, 2019)

API se podría definir como interfaces de programación que permiten la comunicación entre varias aplicaciones de software.

Una API puede proporcionar un gancho para los colegas, socios, o terceros desarrolladoresa acceso a datos y servicios a construir aplicaciones tal como iPhone, aplicaciones rápidamente (Jacobson et al., 2011).

Según un nuevo estudio (Slate, 2019) las API pueden incluirse en una de las dos categorías.

Api privadas: Una API privada solo pueden acceder los desarrolladores y los usuarios de la organización. Estas API normalmente conectan procesos internos de los equipos con el fin de reducir el trabajo aislado y mejorar la colaboración (Jacobson et al., 2011)

Api abiertas: Las API abiertas proporcionan a los desarrolladores externos unmodo de acceder fácilmente a la información e integrar entre herramientas (Jacobson et al., 2011)

Una API abierta o pública ahorra a los desarrolladores tiempo, pues les permite conectar su plataforma con herramientas que ya tienen, lo que reduce la necesidad de crear funciones totalmente nuevas (Jacobson et al., 2011).

Protocolos de servicio web y API

Según un nuevo estudio (Slate, 2019), estos estándares, llamados protocolos de servicio web, son conjuntos de prácticas que dictan el modo de comunicar los datos y cómo se accede a las API.

Api soap : Protocolo Simple de Acceso a Objetos (SOAP) es un protocolo basado en XML para el intercambio de información entre ordenadores. Aunque SOAP se puede utilizar en una variedad de sistemas de mensajería y puede ser entregado a través de una variedad de protocolos de transporte, el enfoque inicial de SOAP es el de las llamadas de procedimientos remotos transportados a través de HTTP. Por lo tanto, SOAP permite que las aplicaciones cliente se conecten fácilmente a servicios remotos e invocar métodos remotos (Cerami, 2002b).

SOAP según la información obtenida, podemos definir como un protocolo de mensajería cuya función nos permite la comunicación de las aplicaciones basada en el esquema de XML a través de HTTP.

Api rest: Sus siglas en inglés de Transferencia de Estado Representacional, es el recién llegado a los protocolos de servicio web y admite un mayor número de formatos de datos, además de que ofrece tiempos de carga más breves y un mejor rendimiento (Slate, 2019).

Una API REST es un backend capaz de contestar a las llamadas a una serie de URLs en formato JSON y que también es capaz de recibir JSON para gestionarla información que le

enviemos. La principal ventaja de las API REST es que podemos desarrollar una API en el backend y utilizarla en cualquier dispositivo, ahorrando así mucho tiempo de desarrollo (Theory, s. f.).

Con esta información podemos definir que una Api Rest es un protocolo de servicio web que puede ser consumida o usada por cualquier dispositivo, mejorando el rendimiento en el desarrollo.

Api graphql : GraphQL es un lenguaje de manipulación de datos que me permite realizar consultas en tiempo de ejecución, para las distintas interfaces de aplicaciones denominadas “API”, dentro de sus funciones es de brindar a los clientes solo los datos que necesitan (El concepto de GraphQL, s. f.).

Lenguaje de consultas GraphQL

Esta es la especificación para GraphQL, un lenguaje de consulta y motor de ejecución creado originalmente en Facebook en 2012 para describir las capacidades y requisitos de los modelos de datos para aplicaciones cliente-servidor (GraphQL, 2018).

GraphQL proporciona una interfaz común entre las aplicaciones cliente y servidor para la obtención y manipulación de los datos (Buna, 2016).

La consulta para su API y un tiempo de ejecución del lado del servidor para ejecutar consultas mediante el uso de un sistema de tipos que defina para sus datos. GraphQL no está vinculado a ninguna base de datos o motor de almacenamiento específico, sino que está respaldado por su código y datos existentes (GraphQL: A query language for API., s. f.).

Según estas investigaciones podemos mencionar que GraphQL en un lenguaje de consultay manipulación de datos que me permite definir de una forma sencilla e independiente a los mismos, esta tecnología lo ha creado Facebook y la cual está siendo utilizada.

Consultas y mutaciones

Operaciones: GraphQL le permite tener al cliente mucha libertad en los datos, si los requerimientos de datos del cliente cambian en algún momento y necesitan recuperar otros campos, simplemente tienen que modificar sus las estructuras de la consulta y enviarla al mismo punto final. GraphQL también proporciona mecanismos para filtrar los resultados.

También permite las cuatro funciones básicas del CRUD: crear, leer, actualizar y borrar. A diferencia de REST, donde las funciones de CRUD se especifican en el método HTTP (PUT, POST, GET y DELETE), graphql define tres tipos de operaciones independientes dela petición HTTP (Vázquez-Ingelmo et al., 2017):

- ✓ **Consultas:** recuperación de sólo lectura (ya explicado).
- ✓ **Mutación:** una modificación en el backend seguida de una respuesta con resultados.
- ✓ **Suscripciones:** peticiones en respuesta a eventos de origen.

En las operaciones podemos mencionar que GraphQl tiene la capacidad de realizar las funciones principales como es el CRUD las cuales son especificadas, mediante una consulta de http las cuales son independientes, hay que tomar en cuenta que GraphQl permite al cliente la libertad de presentación de datos.

- ✓ ***Campos:*** El tipo de objeto tiene un nombre y campos; estos campos los datos concretosresueltos son de tipo escalar y el lenguaje de esquema GraphQL soportalos tipos escalaresde String, Int, Float, Boolean e ID. (Ghebremicael, 2017).

- ✓ **Argumentos:** La Aplicación de GraphQL ejecuta múltiples API que incluye en la implementación de las transformaciones de datos una vez en el servidor, en lugar de cada cliente (GraphQL: A query language for APIs., s. f.)

  Los argumentos se definen como la relación que poseen un campo al realizar la consulta, talcomo la relación entre clave y valor, además de los mismos permiten realizar él filtró de las consultas (Porcello & Banks, 2018).

- ✓ **Variables:** Las variables son valores dinámicos que pueden ser reasignados a los argumentos y son usadas para generar consultas, para enviar el valor dinámico en la variablegeneré un archivo en formato JSON (Porcello & Banks, 2018).

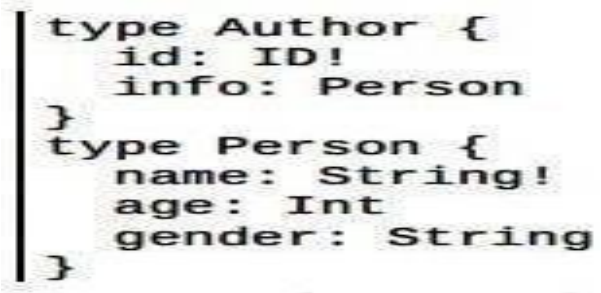
La estructura de una variable dentro de una consulta lo define (GraphQL: A querylanguage for APIs., s. f.)

- ✓ Se reemplaza el valor estático en la consulta con $variableName (Porcello & Banks, 2018)
- ✓ Se declara $variableName como una de las variables aceptadas por la consulta. (Porcello & Banks, 2018)
- ✓ Se pasa variableName: valué el diccionario devariables separado (Porcello & Banks, 2018)

Esquemas y tipo de datos

- **Esquema:** Al igual que los esquemas de las bases de datos, los esquemas GraphQL definenla estructura de nuestros datos. Esto incluye el tipo de datos y el tipo de operaciones que pueden realizar con los datos (Kimokoti, 2018) ver Figura 4.

Figura 4.
Esquema de GraphQL.



Nota. Tomado de (Kimokoti, 2018).

En esta investigación se define que el esquema es él

- **Campos y tipos de objetos:** Un esquema GraphQL está construido por tipos, los cuales se define como el esquema central, también tenemos la libertad de definir nuestros tipos escalares personalizados proporcionando implementaciones de su serialización, deserialización y validación (Kimokoti, 2018).

Según (Kimokoti, 2018), GraphQL viene con un conjunto de tipos de escalares por defecto, son los siguientes:

- ✓ **Float:** Un valor de punto flotante de doble precisión firmado
- ✓ **Int:** Un entero de 32 bits firmado
- ✓ **String:** Una secuencia de caracteres codificados en UTF-8
- ✓ **Boolean:** verdadero o falso
- ✓ **ID:** Identificador único que puede ser un entero o una cadena

- **Argumentos:** Permiten obtener datos filtrados mediante el uso del mismo valor, tienen diferentes tipos en un solo esquema (Porcello & Banks, 2018).

En la Figura 5. Se observa que, dado que las consultas proporcionan argumentos, nuestra función de resolución necesitará tener al menos dos parámetros. También necesitamos incluir la consulta como parte del esquema.

Figura 5.

Argumentos.

```
...
const typeDefs = `{
...
type Query {
getAuthors: [Author]
retrieveAuthor(id: ID!): Author
}
`;
const resolvers = {
  Query: {
    getAuthors: () => authors ,
    retrieveAuthor: (obj, { id }) =>
     authors.find(author => author.id === id)
  },
};
...
```

Nota. Tomado de (Kimokoti, 2018).

- **Tipos consulta y mutación (type query, type mutation):** Se puede observar diferentes tipos de objetos y cada uno tiene diferentes funcionalidades importantes para ejecutar una consulta y mutación (Linux-Fundation, 2019).

En la Figura 6 se puede evidenciar cada punto de entrada para una API.

Figura 6.

Tipo query y Tipo mutation.

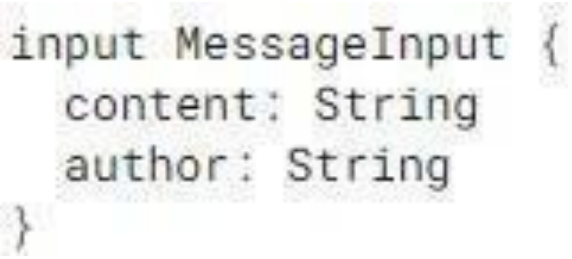
```
type Mutation {
  setMessage(message: String): String
}

type Query {
  getMessage: String
}
```

Nota. Tomado de (GraphQL, 2018).

En los tipos de consulta y mutación en un esquema GraphQL, las conexiones de entrada están compuestas por diferentes campos y tipos de variables de resultados, los cuales se conectan con aplicaciones denominadas “Demonios” (Kimokoti,2018).

- **Los tipos de entrada o Input type:** Dentro de GraphQL, se establece como palabra apartada TYPE, que son prácticamente las salidas de datos con el fin de obtener la información basado en instrucciones INPUT, que es una instrucción de entrada de datos (Linux-Fundation, 2019).A continuación, se describe en la Figura 7. Los tipos de entradas que tienen estainstrucción.

Figura 7. Tipo de Entrada.



Nota. Tomado de (GraphQL, 2018).

Dentro del esquema de GraphQL al escribir un asentencia de entrada, permite tener una ventaja ya que estos tipos son utilizados para el mejoramiento de consulta en la base de datos tanto en el filtro como la paginación mejorando la documentación y la utilización de las API haciendo que sus procesos sean más fáciles de entender y aplicable (Porcello & Banks, 2018).

La validación: Existen un conjunto de normas que me permiten validar una consulta, la cual garantiza que la información obtenida sea correcta y de fácil uso, mediante los tipos de errores donde se puede validar la consulta y realizar comprobaciones en tiempo real (Linux-Fundation, 2019).

Podemos detallar a continuación las siguientes características:

- Fragmento Repetido: Esto se ocasiona cuando el fragmento se llama así mismo, estoprovoca un ciclo infinito, como consecuencia se sobrecarga la información consultadaprovocando que el procesamiento sea ineficiente.
- Campos no existentes: La determinación de los campos dentro de una consulta, los mismos, tiene que existir dentro de la definición del esquema, si el tipo de campo no existe esto provocará un error y se visualiza como campo no válido.
- Sin Campos para devolver: Este error es provocado por un tipo de objeto que no existedentro del acoplamiento de la consulta, lo cual muestra un error debido a que poseeun grado de tipo escalar.
- Campo escalable: Verifica que no haya niveles inferiores en la composición de un tipo, lo cual al existir esto provocaría un error.

Una de las validaciones en común es la inserción forzosa de campos, lo cual al implementar en la consulta se deberá especificará con el signo “!”.

Ejecución: Es importante la validación de campos dentro de las consultas o en las mutaciones, los resultados de las misma nos permitirán realizar estas operaciones denominados los resolutores. Cada campo devolverá el valor referente al tipo, si dicho valor pertenece a un campo escalar finalizará el proceso caso contrario seguirá con el proceso. (Linux-Fundation, 2019).

Resolutores: Son utilizadas para la definición de campos dentro de los esquemas las cuales me permite obtener los datos, donde cada campo se encuentre definido dentro del mismo posee una funcionalidad (Ghebremicael, 2017).

Los resolutores tienen como indicadores la operación que realiza los datos, cada uno tiene su propia funcionalidad y esto permite su resolución, estas funcionalidades se las conoce como mapa de resolución y poseen 4 argumentos que son las siguientes (Apollo GraphQL,2019).

a) **Parent (obj):** Son parámetros anteriores que son inservibles dentro de las consultascon especificaciones de campos raíz debido al anidamiento de la misma.

b) **Args:** Son parámetros que se envía dentro de la consulta.

c) **Context:** Permiten compartir la información sobre un estado de las solicitudes con elfin de ser compartido con los resultados.

d) **Info:** Es el estado de la consulta donde contiene toda la información, argumentos yrutas de los campos de forma jerárquica.

e) **Respuesta:** Se presenta al término de un proceso que se genera al realizar una consulta, cuyos resultados se presentan mediante un mapa donde contienen cada campo definido como la clave y el valor en un formato JSON (Linux-Fundation, 2019).

Al generar la respuesta puede tomar distintas estructuras provocando en ocasiones que la consulta contenga errores por el contenido de la misma (ApolloGraphQL,2019).

Los posibles errores que se puedan producir dentro de la consulta son los siguientes.

a) **Valor escalar:** Cuando la respuesta no posee ningún significado.

b) **Matriz:** Se define el esquema en forma de listas.

c) **Promesa:** Es provocada al momento de generar las operaciones y estas seana sincrónicas.

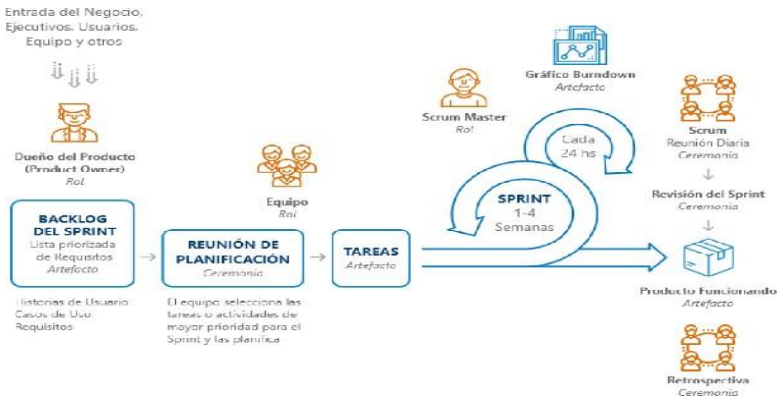
d) **Indefinido o nulo:** Esto es provocado cuando el objeto se encuentra vacío o ala vez no existe.

Marco de trabajo Scrum

Según (Marco de trabajo Scrum, 2018) menciona que se: La metodología se genera en las reuniones son los sprint y se realiza al inicio y al final de un proceso se realiza reuniones diarias como se puede observar en la Figura 8.

Figura 8.

Marco de trabajo Scrum.



Nota. Tomado de (Pablo, 2018).

Roles de Scrum

Está conformado por un equipo de trabajo donde todos los miembros poseen un rol específico, dichos roles son los siguientes: el dueño del producto (Product Owner), el jefe del plan (Scrum Master), el cual integra también el equipo de trabajo (Team) conjuntamente con el equipo de desarrollo (Greene & Stellman, 2014).

Los roles que ponen el Scrum son los siguientes según (Palacio, Menzinsky, & López, 2016):

a. El Product Owner toma las decisiones finales del producto, así como definir partes principales que debe ir al producto ya que se encuentra en constante relación con el ScrumMaster.

b. El Scrum Master es el líder de proyecto, el mismo que conoce cómo va el desarrollo de producto, una de sus funciones es delegar tareas o actividades y dar seguimiento de

lo que se ha cumplido dentro del proyecto ya que tiene constante comunicación entre el equipode trabajo y el dueño del proyecto.

c. El equipo de trabajo está compuesto por personas previamente calificadas en las necesidades que tiene el proyecto a desarrollarse, el equipo realiza las actividades establecidas dentro del proyecto siempre trabaja conjuntamente con el Scrum Master.

Artefactos de scrum

Según (Djandrw, 2019), en Scrum existen 3 artefactos que se refieren a elementos físicos que se producen como resultado de la aplicación de Scrum. Estos son:

El Product backlog. - Es la fuente principal de información sobre el producto, es el resultado del trabajo del Product Owner con los distintos Stakeholders y contiene cualquier tipo de tarea en cualquier formato y su única condición es que esté priorizado con aquellos ítems que tienen más valor en ese momento.

a) El Sprint backlog. - Es un factor para visualizar el trabajo del Sprint y está gestionadopor el equipo de desarrollo, el que se ocupa de mantenerlo actualizado y transparente a lo largo de toda la iteración, en especial por mediode los Daily Scrums. Posibilita examinar hasta donde se ha cumplido el propósito en cada Sprint y que se podría borrar.

b) El incremento. - Es la suma de cada una de las labores, casos de uso, user stories y cualquier factor que se haya desarrollado a lo largo del Sprint y que va a ser puesto a disposición del cliente final en forma de programa finalmente del mismo.

Así se hace una aplicación de manera iterativa e incremental.

Eventos de scrum

Según (Greene & Stellman, 2014) se detallan los siguientes eventos:

a) Sprint Planning es la planificación y el inicio del sprint donde se determina las tareaso actividades realizadas por todos los miembros que conforman el Scrum, mediante reuniones donde definen la dificultades y prioridades que tiene el producto con el fin de entregar el producto.

b) Daily Scrum tiene como finalidad conocer las inquietudes y el avance del proyecto mediante reuniones que no superan los 15 minutos dentro de esta reunión se tomó como principio realizar la siguiente pregunta “¿Qué actividades realizó el día anterior?”,

"¿Qué actividades realizar hoy?" y "¿Qué inconvenientes tuvo al realizar las actividades?", la persona a cargo de dirigir estas reuniones es el Scrum Master.

c) Sprint Review es una reunión previa a la reunión final, se entrega una versión del proyecto mediante la demostración del mismo, donde se revisa los procesos funcionales del proyecto realizadas en la planificación si no cumples dichos procesos se entregará al propietario del producto una versión funcional que sea aprobado por el mismo.

d) La retrospectiva es una reunión donde se realiza un análisis sobre el proceso de las actividades del proyecto donde se puede dar a conocer mejoras que se pueden implementardentro del proyecto en esta reunión intervienen todos los miembros del Scrum.

ISO /IEC 25000

Según el sitio web oficial (Portal ISO 25000, 2017) la familia ISO/IEC 25000 constituye un conjunto de normas basadas en ISO/IEC 91261 y en ISO/IEC 145982 cuyo objetivo principal es proporcionar una guía para el uso de la nueva serie de estándares internacionales llamada Requisitos y Evaluación de Calidad de Productos de Software (SQuaRE - System and Software Quality Requirements and Evaluation).

Las normas ISO/IEC 25000 se compone de cinco fases como se muestra en la Figura 9, (ISO 25000,2019).

Figura 9.

Divisiones de la Familia ISO/IEC 25000.

ISO/IEC 25010

En la calidad se necesita evaluar el producto mediante un modelo de calidad, se puede generar un punto de calidad bajo ciertas condiciones que se acoplan al modelo de calidad las cuales formarán características y subcaracterísticas que representaran la clasificación de un modelo de calidad entre ellas se destacan las subcaracterísticas como adecuación servible, eficiencia de funcionamiento, compatibilidad, usabilidad, fiabilidad, estabilidad, mantenibilidad y portabilidad (ISO 25000, 2019).

Modelo de calidad en uso.

Las características de la calidad de uso son las siguientes: eficiencia, efectividad, satisfacción, independencia de peligro y cobertura de entorno como se observa en la Figura 10, estas permiten evaluar el nivel producto a utilizar la norma ISO/IEC 25010,2011.

Figura 10.

Características de la Calidad de Uso.

Eficacia
Eficiencia
Satisfacción
Utilidad
Confianza
Placer
Comodidad
Libertad de riesgo
Mitigación de riesgos económicos.
Mitigación de riesgos de salud y seguridad.
Mitigación de riesgos ambientales.
Cobertura de contexto
Completitud del contexto
Flexibilidad

Nota. Tomado de (ISO/IEC 25010, 2011).

La (ISO/IEC 25010, 2011) define a la usabilidad como una agrupación de calidad que se utiliza para radicar la efectividad, eficiencia y satisfacción y conservar su significado, esta calidad tiene una característica que se nombra usabilidad, por lo que va a ser evaluado.

ISO/IEC 25022

Medición de la calidad de uso: Es necesario realizar un modelo de calidad de uso y es necesario la aplicación de la normativa ISO/IEC 25022 ya que permite la evaluación de la calidad de software en los sistemas informáticos, la evaluación es acorde al criterio del

usuario con la finalidad de gestionar la calidad del funcionamiento e implementación.

ISO/IEC 25040

Evaluación de calidad en uso : Dentro del estándar ISO/IEC 25040, donde se define conceptos generales y los requisitos para la evaluación de los procesos, en conjunto con el estándar ISO/IEC 25010 e ISO/IEC 25022 se establece los modelos y las métricas de medición para la calidad de uso, esto permite que la evaluación del producto a partir de un punto de vista de calidad interna y externa, al utilizar esta norma los afectados son las personas que está en el desarrollo del producto, la persona que adquiere el producto y la evaluadora o certificadores quienes son autorizados para evaluar el proceso (ISO/IEC 25040, 2011).

Antecedentes contextuales

La empresa Arest Consulting brinda servicios de asesoramiento, orientación y asistencia operativa a cualquier tipo de empresa, y se encuentra ubicada en la ciudad de Latacunga.

El objetivo es asesorar profesional e independientemente a las organizaciones, aportando con soluciones efectivas en áreas: arquitectura empresarial, tecnologías de información, legales, contables, tributarias, administrativas, entre otros, ofreciendo soluciones prácticas de alto nivel que aporten con el progreso de nuestros clientes y socios estratégicos a través del cumplimiento de objetivos y metas planteados por nuestros líderes, logrando ser una empresa de éxito en un mercado competitivo.

Para el diagnóstico del problema se aplicó una encuesta, como instrumento de investigación,a los cinco empleados del área de programación de la empresa Arest Consulting, de la cualse obtuvo el siguiente resultado:

1. **¿En el área de programación se tiene definida una arquitectura para todos sus módulos?**

Si	No
28.51%	71.43%

El 71.43% del personal del departamento de programación de la empresa Arest Consulting dice que no tienen definida una arquitectura para el desarrollo de sus módulos y el 28.51% dicen que sí cuentan con una arquitectura.

2. ¿En el desarrollo de los módulos se utilizan arquitecturas híbridas enversiones actuales?

Si	No
0%	100%

El 100% del personal del departamento de programación de la empresa Arest Consulting menciona que no se usa una arquitectura híbrida para el desarrollo de sus módulos.

3. ¿Qué problemas presenta el mantenimiento de las aplicaciones con laarquitectura monolítica?

Tiempo	Integración	Tecnología usada	Otros Personal
57.14%	28.51%	28.57%	0%

El 57.14% del personal del departamento de programación de la empresa Arest Consulting menciona que el tiempo es el mayor problema para el mantenimiento de sus módulos y el 28.51% con respecto a la integración, seguido del 28.57% sobre la tecnología empleada.

4. ¿Durante el mantenimiento se presentó alguna falla en el aplicativo que fuera tomado como algo crítico dentro de los procesos, cuál fue el tiempo que se toma para llegar a una solución?

Minutos	Horas	Días
0%	28.57%	71.43%

El 71.43% del personal del área de programación de la empresa Arest Consulting dicen que se demoran días para poder resolver un cambio y el 28.57% dicen que se demoran horas, dependiendo de la gravedad del cambio en la aplicación.

5. ¿Se usan estándares de calidad para el desarrollo de software?

Si	No
0%	100%

El 100% del personal del departamento de programación de la empresa Arest Consulting

menciona que no se utiliza una arquitectura híbrida para el desarrollo de sus módulos.

6. ¿Le gustaría aplicar una arquitectura híbrida orientada a microservicios para medir la usabilidad de los datos desde aplicaciones móviles?

Si	No
100%	0%

El 100% del personal del departamento de programación de la empresa Arest Consulting menciona que si les gustaría aplicar una arquitectura híbrida en microservicios que sea de mucha utilidad para sus aplicaciones móviles.

7. ¿Para usted, qué porcentaje de importancia tiene el desarrollo de la arquitectura orientada a microservicios?

Alta	Media	Baja
100%	0%	0%

El 100% del personal del área de programación de la empresa Arest Consulting menciona que es de muy alta importancia contar con una arquitectura orientada en microservicios.

8. ¿En qué porcentaje considera usted que, la implementación de una Arquitectura de Microservicios ayudará con la disminución del tiempo para el despliegue de las aplicaciones?

1% - 10%	11% – 20%	21% - 30%	Mayor al 30%
0 %	0%	0%	100%

El 100% del personal del área de programación de la empresa Arest Consulting menciona que la arquitectura híbrida ayudará la realización de los cambios, siendo que trabaja por servicios.

Análisis General

Analizando los resultados de la encuesta aplicada, el 80% del personal de la empresa Arest Consulting coinciden que la arquitectura monolítica retrasa el trabajo para los programadores porque les lleva horas o días realizar un cambio si se dañó un módulo.

De los resultados obtenidos en la encuesta, el 100% del personal piensa que la aplicación de un modelo arquitectónico orientado a microservicios facilitará el uso y mantenibilidad del

código de los proyectos de software del área de desarrollo de la empresa Arest Consulting y se llega a la conclusión que si existe la factibilidad del proyecto.

Capítulo III

Desarrollar e implementar una arquitectura de microservicios

Para la implementación de la arquitectura basada en microservicios API-Rest y GraphQL, la cual va a ser desarrollada con la metodología de trabajo Scrum, ya que es una metodología ágil que permite el desarrollo y la optimización del tiempo y recursos.
A continuación, se describe el equipo de trabajo, etapas de los artefactos que son contemplan el desarrollo del proyecto.

Roles del proyecto

Para el adecuado seguimiento del marco de trabajo en Scrum, se define en tres roles importantes para la metodología. El primer rol es el Scrum Máster que tiene un papel muy relevante y que contribuye como enlace de todo el equipo de trabajo y está pendiente en cada una de las etapas del Scrum. El segundo rol son los Development Teams, son los encargados del desarrollo y de la implementación del software en general y Finalmente el tercer rol es el Product Owner quien hace la función de ser el representante del cliente frente al Development Teams.

Se detallan los roles definidos para el desarrollo de la arquitectura de microservicios.

Tabla 1

Roles del Equipo de Trabajo de la arquitectura de microservicios

Scrum Máster: Alex Abarca
Coordinador del Proyecto Zeus en la empresa Arest Consulting, con conocimientos técnicos y metodológicos en el ámbito de software, sus funciones: ✓ Ayuda al equipo en la organización y planificación del proyecto Zeus ✓ Motiva al Scrum Teams a crear un trabajo colaborativo.
Development Teams Christian Tigse
Estudiante del programa de maestría de software de la Universidad de las Fuerzas Armadas con un sólido conocimiento para el desarrollo de sistemas.
Product Owner: Clientes del Zeus
Cliente de la aplicación Zeus donde evalúa cada uno de sus procesos y conoce claramente las necesidades de su empresa, sus actividades son las siguientes:

Product Owner: Clientes del Zeus
✓ Representar a todos su equipo de trabajo para evaluar la aplicación. ✓ Validar y verificar las funcionalidades implementadas en la aplicación

Etapas del proceso de desarrollo de Software

El marco de trabajo Scrum fue desarrollado especialmente para proyectos con altas probabilidades de cambio y con equipos de trabajo pequeños mediante plazos de tiempo reducidos. Una de las razones de la metodología de desarrollo de software es por la división del trabajo en un intervalo de tiempo de 2 a 4 semanas donde se desarrollan funciones específicas que contribuyen con los subproyectos o módulos que permitan terminar todo el proceso que conforman la arquitectura de microservicios.

Se detalla las etapas a realizar para el desarrollo de la arquitectura de microservicios:

- ✓ Planificación del Sprint
- ✓ Etapas de Desarrollo
- ✓ Revisión del Sprint
- ✓ Retroalimentación

Planificación del Sprint

El sprint inicia con el conjunto de requerimientos necesarios para la implementación y son analizados por parte del equipo de trabajo, prevaleciendo la opinión del Product Owner, de igual forma cada una de las funcionalidades establecidas que van a ser priorizadas según la importancia como son alta, media, baja.

A continuación, se presenta el formato con el que se procede a detallar los requerimientos de los usuarios para el desarrollo de la arquitectura de microservicios.

Tabla 2

Formato para definir historias de usuario

HISTORIA DE USUARIO	
Número: 01	**Usuario:** Usuario Abc
Descripción de la Historia:	Historia de pruebas

HISTORIA DE USUARIO			
Prioridad en Negocio:	Alta	**Iteración Asignada:**	
Programador Responsable:	Programador 1		

Las historias de usuario son recogidas mediante un diálogo con el Product Owner, luego de su redacción deben ser analizadas por todo el equipo de trabajo y generan un listado de tareas que reflejan las necesidades detectadas por el Scrum Máster. El Product Backlog también cumple la función de ser un elemento visual, el cual está presente para todo el equipo y permite tener una visión más clara para todas las tareas.

Uno de los factores importantes dentro del proceso de priorización de requerimientos es iniciar el trabajo con aquellas tareas que se encuentran en la pila en la parte superior.

El formato diseñado para la elaboración del Product Backlog es el siguiente:

Tabla 3

Formato del Product Backlog de la arquitectura de microservicio

ID	TAREA	RESPONSABLE	PRIORIDAD
1	Tarea 1	Responsable 1	Alta
2	Tarea 1	Responsable 1	Baja

El Product Backlog puede construir una lista amplia de requerimientos por el cual la metodología Scrum establece que se divida en subtareas para los avances continuos específicos en los artefactos conocidos como sprint o pila. El formato establecido para redactar el sprint backlog se presenta a continuación:

Tabla 4

Formato para redactar los sprints de la arquitectura de microservicio

DATOS PARA EL SPRINT	
Número:	001
Fecha Inicio:	02/01/2021

DATOS PARA EL SPRINT	
Fecha de Culminación:	02/01/2021

TAREAS POR DESARROLLAR			
PRIORIDAD	**DESCRIPCIÓN**	**RESPONSABLE**	**ESTADO**
Alta	Tarea 1	Desarrollador 1	Realizado
Alta	Tarea 2	Desarrollador 1	Realizado
Alta	Tarea 3	Desarrollador 1	Realizado

Etapa de Desarrollo

Para el desarrollo de la arquitectura basada en microservicios Api-Rest y GraphQL se utiliza las herramientas de programación de última generación, puesto que se busca generar una arquitectura para un ambiente móvil capaz de funcionar sin ninguna novedad de una manera correcta dentro del equipo de trabajo de la Empresa Arest Consulting, para lo cual se utiliza el lenguaje de programación C#, el editor de código Visual Studio y la interfaz de programación de apelaciones Zeus.

Mediante el lenguaje de programación se puede deducir que es un lenguaje muy fácil y ejecuta del lado del cliente, actuando como procesos de petición de datos para analizar cada uno de los procesos en la aplicación Zeus y obtener una comparación óptima con la arquitectura de microservicios.

Revisión del Sprint

Cuando se finaliza un sprint se procede a la verificación que tiene como objetivo la verificación de cada una de las funcionalidades se hayan desarrollado mediante la planificación establecida en el Product Owner, Para ello se utiliza la técnica de Checklist que sirve para la validación de todas las actividades realizadas por el equipo de trabajo y posterior por el experto en el área de software.

Una vez verificada la implementación y el desarrollo basado en la lista de requerimientos especificados por el Product Backlog se procede a hacer las pruebas para garantizar la calidad del software.

Cada una de las pruebas que se efectúan sirve para revisar el correcto funcionamiento del sistema que se ha basado en parámetros como variables de entrada, resultados.

Tabla 5

Evaluación de la arquitectura en microservicios

N.º	Variables de Entrada	Resultado Esperado	Estado
1	Variable A	Resultado A	Aprobado () Rechazada ()
2	Variable B	Resultado B	Aprobado () Rechazada ()

La metodología Scrum es importante para la colaboración de todo el equipo, ya que permite analizar conjuntamente con personas externas al grupo de una manera que complemente la revisión de los requerimientos, el equipo manifiesta todo lo que ha ido desarrollando y el resto de equipo procede a la revisión estableciendo los resultados para la implementación.

Retroalimentación

Después que el sprint haya sido sometido a verificación es fundamental recibir un feedback no solo del equipo de trabajo, sino también por los usuarios del sistema, para ello se le coloca en producción para que el cliente pueda analizar los cambios establecidos. La retroalimentación se evidencia en una bitácora de trabajo en el cual se califican todos los puntos satisfactorios y los aspectos que han mejorado con el desarrollo de la arquitectura de microservicios.

Artefactos de entrada y salida

Scrum sigue los manifiestos de una metodología ágil donde se valora el software y sus funcionalidades mediante una documentación, el marco de trabajo abarca varios artefactos esenciales para la implementación. Para el presente proyecto se considéralos siguientes con las etapas descritas anteriormente.

Tabla 7

Artefactos de entrada y salida

N.º	Etapa	Artefacto de entrada	Artefacto de Salida
1	Planificación del Sprint	✓ Historias de Usuarios	✓ Product Backlog ✓ Sprint Backlog
2	Desarrollo	✓ Sprint Backlog	✓ Funcionalidades Implementadas
3	Revisión	✓ Funcionalidades Implementadas	✓ Checklist de requerimientos ✓ Pruebas Realizadas
4	Retroalimentación	✓ Checklist de Requerimientos ✓ Pruebas realizadas	✓ Bitácora de Requerimientos

Arquitectura de microservicios

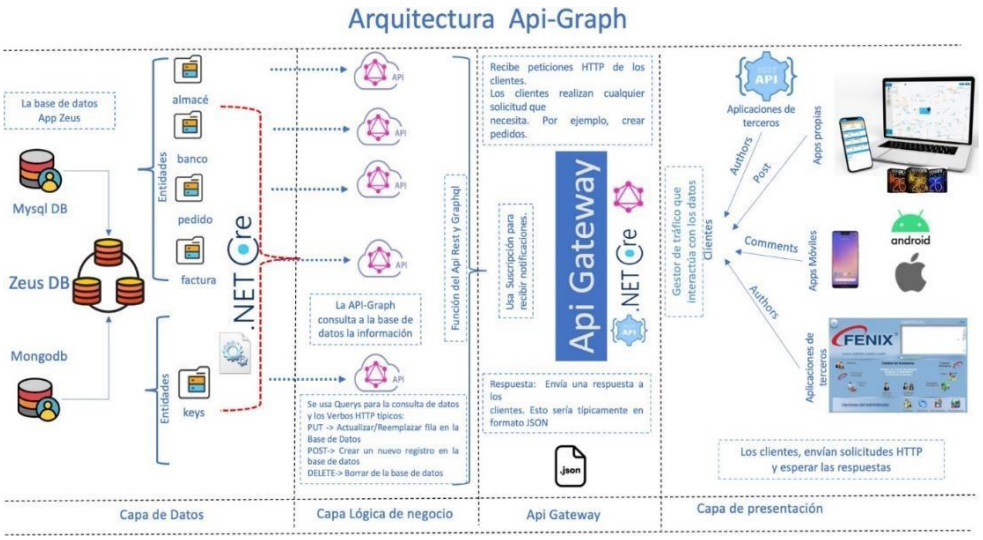
El desarrollo de la arquitectura de microservicios Api-Rest y GraphQL será desarrollado mediante el patrón de diseño arquitectónico de software Modelo, Vista y Controlador (MVC), el cual sirven para clasificar la información de la base de datos, el diseño y las relaciones que existen entre los componentes de la aplicación. Este punto es importante para la definición, desarrollo de la arquitectura y depende de los requerimientos y las expectativas que el Product Owner haya tenido el diálogo con las historias de usuario.

Uno de los factores esenciales para determinar una arquitectura es la iteración con otras aplicaciones informáticas, puesto que la aplicación móvil Zeus tiene como finalidad la interacción con otras plataformas informáticas.

En el **Figura 11** se detallan todos los factores considerados para establecer la arquitectura óptima para el desarrollo de la arquitectura de microservicios.

Figura 11

Arquitectura de microservicios Api-Rest y Graph del Sistema Zeus



La arquitectura API-REST y GRAPH está diseñada para acoplarse al desarrollo en multicapas como actualmente se maneja para el desarrollo en la empresa Arest Consulting, la cual se define de la siguiente manera:

- **Tenemos la capa de datos. -** Donde se encuentra las diferentes bases de datos, ya seaen MySQL, MongoDB o SQL Sever, la API-graph permite consultar la información independiente a la base, en que las entidades se encuentren, para la manipulación de los datos se los realiza mediante un módulo de biblioteca que me permite la administración de todas las acciones que se realiza a la base de datos tales como la consulta de datos y parala realización de CRUD las cuales permitirá crear, actualizar o eliminar.

- **Capa lógica. -** En esta capa interactúa con el módulo de biblioteca, la cual se gestionalos métodos y procesos que van a interactuar dentro con los distintos servicios (Apis) y la base de datos, las cuales recibirán las peticiones que el cliente necesite mediante solicitudes https del cliente y esta enviará una respuesta en formato JSON.

- **Api gateway .-** Permitirá administrar el tráfico de las interacciones que tiene el clientecon los datos solicitados, a la vez la API Gateway se encarga de usar suscripciones que permite enviar notificaciones de acuerdo a lo solicitado, esto permite tener un control de las solicitudes que se envía y recibe la API-graph, esto también interactuar con las suscripcioneslas cuales enviarán la notificación correspondiente además nos permitirá las correctas administraciones de cada uno de los servicios (Apis) creados.

- **Capa de presentación. -** Los clientes enviarán solicitudes mediante HTTPS, ya sea de aplicaciones propias de la empresa como aplicaciones de terceros, aquí se toma en cuenta las API de terceros, las cuales se consumirá de forma transparente, ya que la API-graph permite el uso de las mismas, las solicitudes pueden ser de consulta o manipulación de los datos.

En sí la arquitectura API-GRAPH me permite interactuar con diferentes lenguajes de programación porque está hecho de acuerdo a las necesidades de los programadores para su fácil uso y comprensión de Código, también me permitirá la expansión de desarrollar aplicaciones de entorno web o de escritorio y móvil porque actualmente solo se maneja aplicaciones de escritorio en la empresa Arest Consulting, esta API-GRAPH permite sobresalir con sus productos, ya que se encuentra con grandes estándares de calidad.

Normas ISO /IEC 250000

System and Software Quality Requirements and Evaluation (SQuaRE) pertenece a la familia de las normas ISO/ IEC y ayuda con la creación de un marco de trabajo para la evaluación de las propiedades de la norma. La ISO/IEC 250000 es la evaluación de otras normas y describe las particularidades de un modelo de calidad del producto.

Calidad de Uso – 25010

Los requisitos aportados por los usuarios para una calidad del software se interpretan mediante la funcionalidad, rendimiento, mantenibilidad, entre otros, los cuales son representadores del modelo de calidad de sus características y sub características, las cuales tiene un grado de satisfacciones al momento de recibir los requerimientos.

La ISO/IEC 25010 de modelo de calidad define las siguientes características:

Eficiencia: Es la interpretación de la calidad de los recursos utilizados y se subdivide de la siguiente manera:

Comportamiento Temporal: Evalúa los tiempos de respuesta y los procedimientos de unaaplicación de software cuando lleva a cabo sus funciones bajo condiciones mediante las pruebas establecidas.

Utilización de Recursos: Permite la evaluación de la arquitectura en los tiempos y la cantidad de datos que se obtiene en el sistema.

Capacidad: Evalúa el límite de parámetros que una aplicación de software cumple con cada uno de los requisitos.

Usabilidad: Supervisa el proceso de calidad de una aplicación de software evaluando elaprendizaje y su utilidad del mismo.

Capacidad para reconocer su adecuación. Permite al usuario evaluar si el desarrollo dela arquitectura en microservicios es adecuado para el sistema.

Capacidad de aprendizaje. Permite al usuario evaluar a la arquitectura para el aprendizaje.

Capacidad para ser usado. Permite al usuario tener el manejo completo de la arquitecturapara ver su operatividad y control de los datos.

Protección contra errores de usuario. Permite al usuario evaluar a la arquitectura y laprotección de errores en la información.

Estética de la interfaz de usuario. Permite al usuario evaluar la capacidad de aprendizajede la interfaz.

Accesibilidad. Permite al usuario determinar las características de la arquitectura

Métricas de calidad internas y externas

Métricas internas

Se evalúa mediante las medidas medibles a partir de las características intrínsecas del propio producto de software.

- ✓ Aplican a un producto de software no ejecutable
- ✓ Aplica durante las etapas de su desarrollo
- ✓ Permite medir la calidad de los entregables intermedios
- ✓ Permite predecir la calidad del producto final
- ✓ Permite al usuario iniciar acciones correctivas temprano en el ciclo de desarrollo

Métricas externas

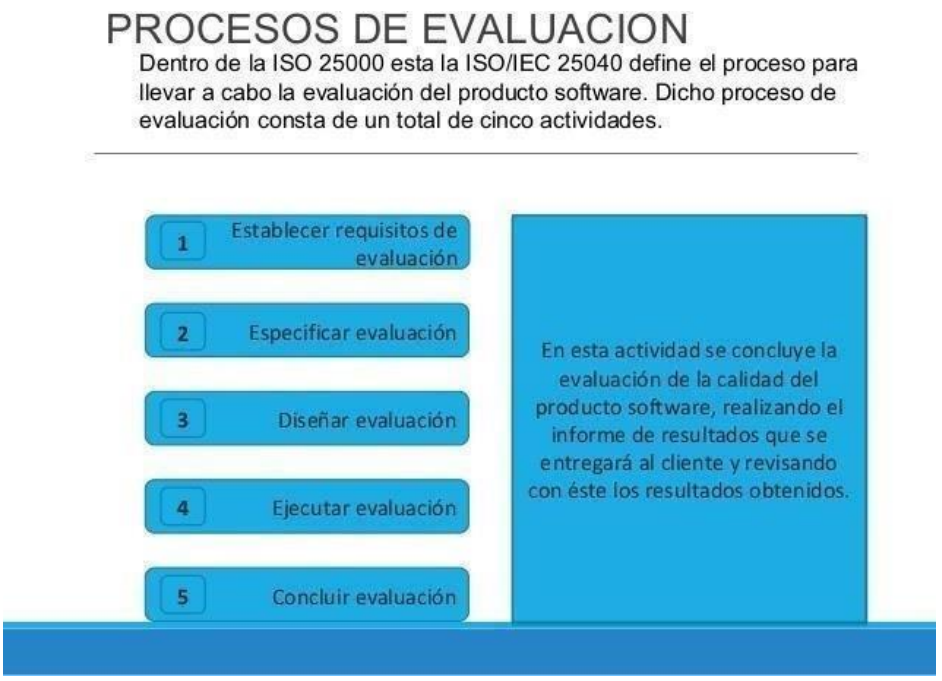
- ✓ Son medibles a partir del comportamiento del software
- ✓ Aplican a un producto de software ejecutable
- ✓ Permite medir la calidad del producto final.

Criterios de Calidad ISO 25000

ISO/IEC 25040 define el proceso para llevar a cabo la evaluación del producto software. Dicho criterio de evaluación consta de un total de cinco actividades.

Figura 12

Figura sobre los procesos de Evaluación



Desarrollo api-graph

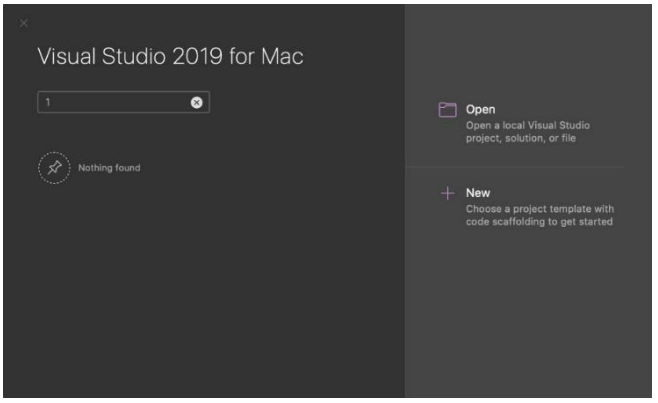
Para el desarrollo de esta nueva arquitectura se planteó realizar una api - Graph la cuales una herramienta adecuada para la adaptación de la empresa donde se aplica el proyecto, debido a que los programadores tienen experiencia en estos tipos de lenguajes.

Herramientas

Visual Studio versión 2019.- Es una aplicación de desarrollo de sistemas integrados que acoge a la mayoría de lenguajes de programación compatibles, su amplia robustez permite interactuar con los diferentes lenguajes de programación que existen en la actualidad.

Figura 13

Figura Entorno de Visual Studio 2019.



C#.- Es un lenguaje multiparadigmas que prácticamente se encuentra orientado a objetos, y al ser un lenguaje que evoluciona como C + + y C, las cuales permiten crear paquetes, bibliotecas o librerías. La cual permite trabajar de una manera robusta, y a que se interactúa con los diferentes lenguajes existentes.

Asp Net.- Es una herramienta que permite consumir la librería creada en C# con la finalidad de desarrollar las API, pues esta herramienta realiza ese proceso de la API.

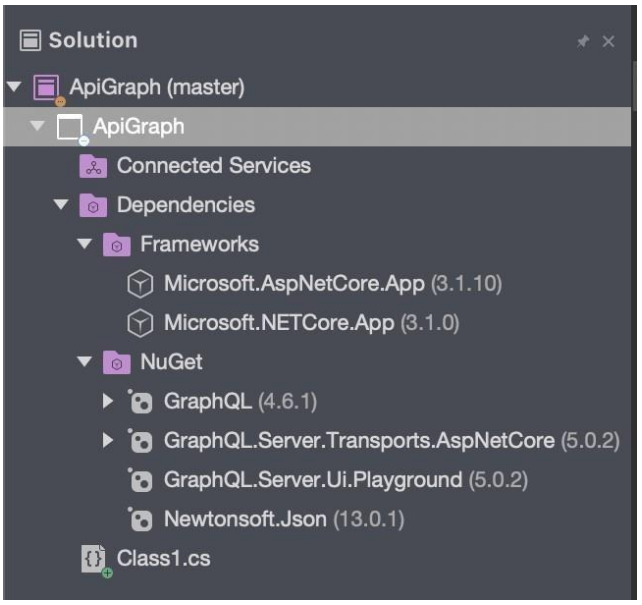
Configuraciones del proyecto

Una vez creado el proyecto se necesita las siguientes dependencias:

- ✓ **GraphQL -Versión 4.6.1.-** Este paquete proporciona a .net clases que definen el esquema de GraphQL y ejecutar las consultas.
- ✓ **GraphQL. Server. Transports. AspNetCore -Versión 5.0.2.-** proporciona un middleware ASP.NET Core que expone la API a través de HTTP.
- ✓ **GraphQL. Server. Ui. Playground -Versión 5.0.2.-** Le brinda un editor en el navegador donde puede escribir consultas GraphQL en su servidor y ver cómo responde.
- ✓ **Newtonsoft. Json - Versión 13.0.1.-** Permite descomponer los datos y dar formato en un archivo JSON y se utiliza más para la interacción de los datos.

Figura 14

Figura sobre los paquetes usados.



Desarrollo del Zeus – Api-Graph

Estructura del desarrollo del Zeus-api-Graph

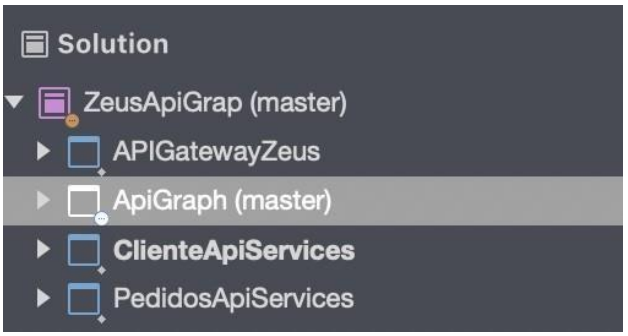
Para la siguiente estructura se tomó como ejemplo servicios que sean básicos y servicios transaccionales

- ✓ **APIGatewayZeus.-** Proyecto que se encarga de gestionar el tráfico de laspeticiones del cliente

- ✓ **PedidosApiServices.-** Es un servicio que permite consumir al clientepeticiones mediante HTTP.
- ✓ **ApiGraph.-** proyecto que contiene la nueva bibliotecas para la creación de losservicios.

Figura 15

Figura sobre los paquetes utilizados.



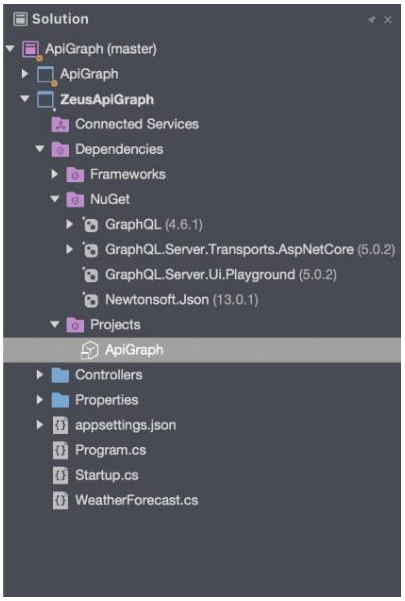
Dependencias

Api-Graph.- Una vez desarrollada la biblioteca nueva, se procederá a realizar los servicios,para esto se ocupará la herramienta de Visual Studios, donde se añadirá una nueva dependencia al proyecto que es Api-Graph, esto permitirá ejecutar las clases, métodos y funciones.

Package Ocelot.- Permitirá manipular los objetos, la cual es especificado en las configuraciones del middleware para las peticiones http request que son las solicitudes de envío y respuesta que se realiza al servicio.

Figura 16

Figura sobre los paquetes usados



Implementación

Para la implementación se realizó en una máquina virtual creada en Amazon machine, adquirida por parte de la empresa Arest Consulting.

- ✓ Para publicar el servicio se realiza los siguientes pasos:
- ✓ En el Panel de control se coloca en la opción desinstalar programas y se coloca en Activar o desactivar las características de Windows.

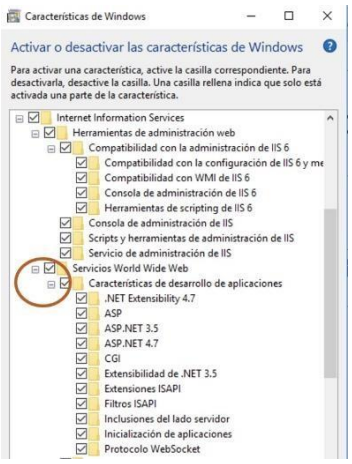
Figura 17

Figura sobre la interfaz de panel de control

Una vez ahí activar las siguientes opciones.

Figura 18

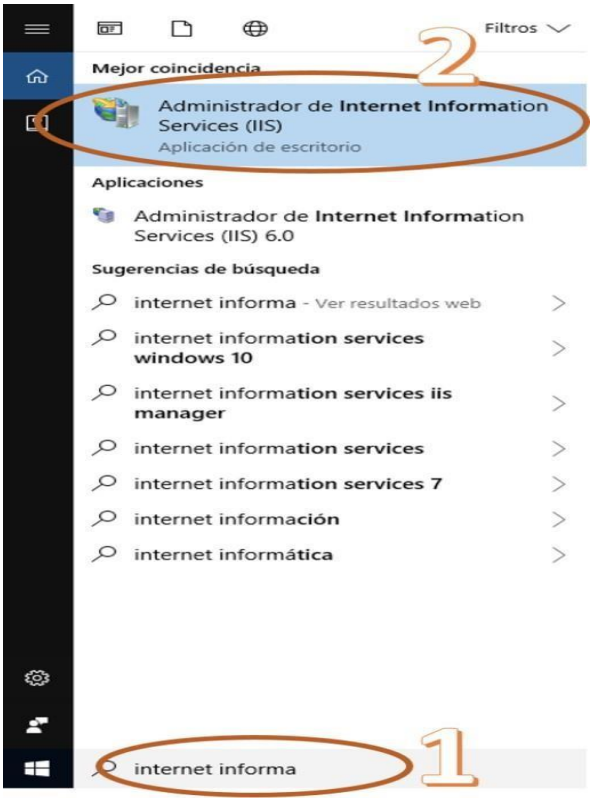
Figura las Características de Windows



- ✓ Aceptamos los cambios
- ✓ Abrimos el buscador de Windows y buscamos IIS

Figura 19

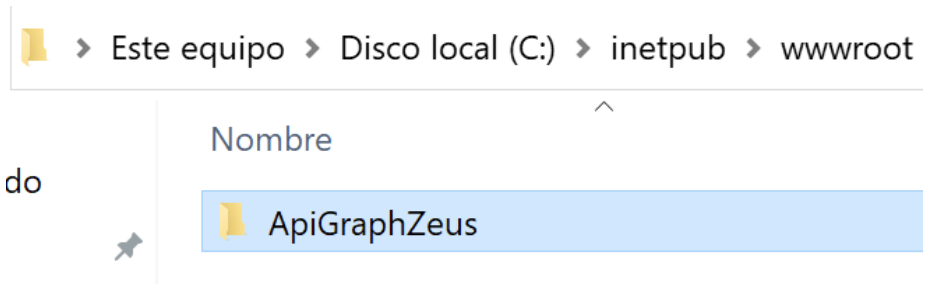
Figura sobre la interfaz de panel de control



- ✓ Una vez configurado el IIS se coloca la siguiente ruta C:\inetpub\wwwroot
- ✓ Creamos una carpeta nueva.

Figura 20

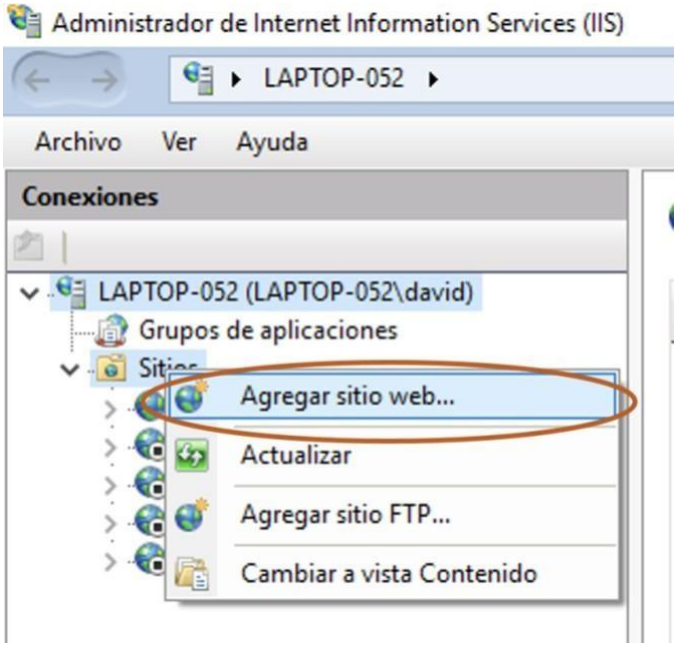
Figura sobre la Api que están en el root



- ✓ Una vez realizada la creación de la carpeta ir al IIS, en la parte del sitio ygeneramos un sitio nuevo.

Figura 21

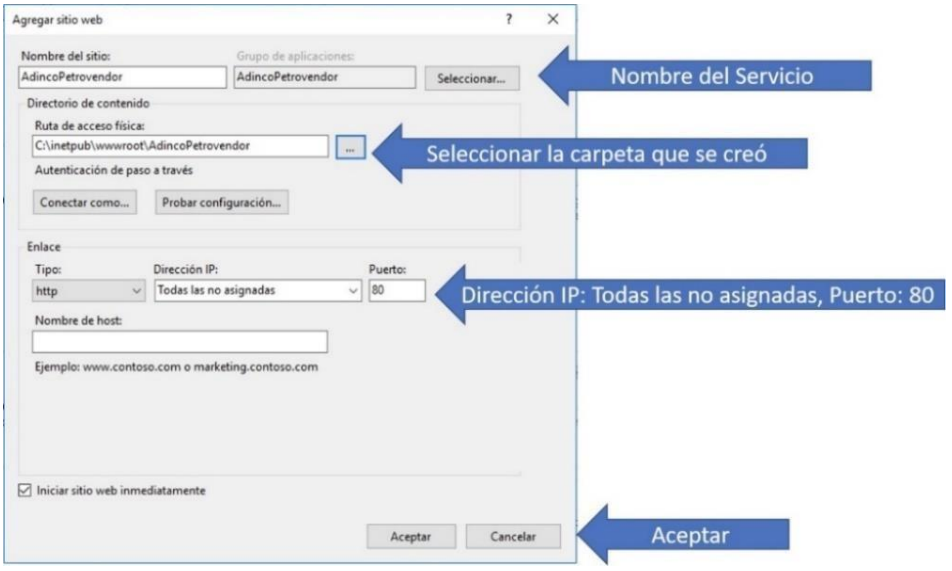
Figura sobre la adm. IIS



Nota. Muestra un formulario lo cual llenaremos con la información solicitada

Figura 22

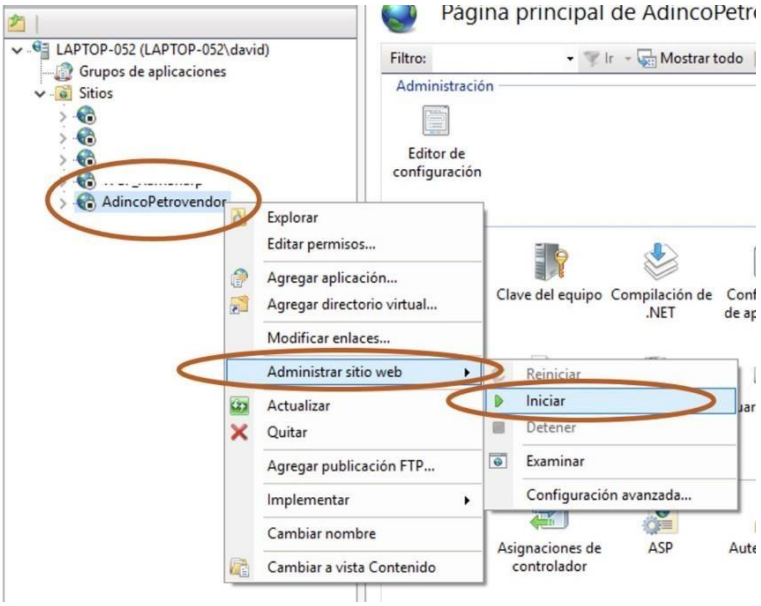
Figura sobre sitio Web



Nota. Una vez creado el sitio nuevo activar el sitio

Figura 23

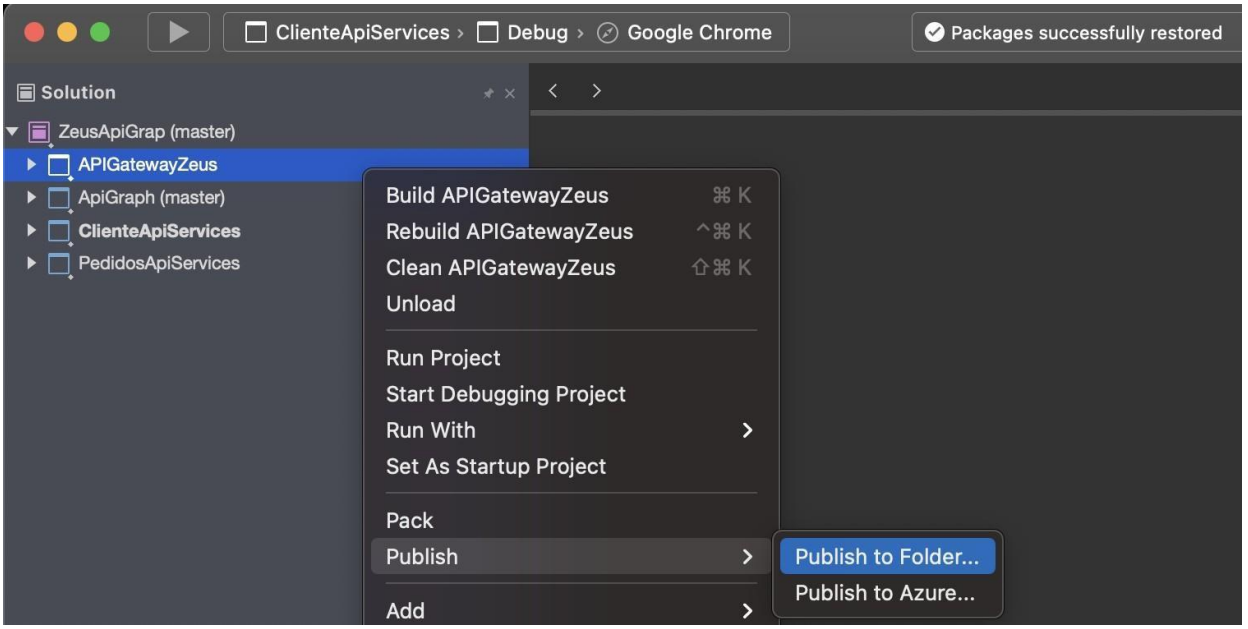
Figura sobre ejecución de un sitio web



Nota. Ahora desde el visual buscamos el apigatewayzeus damos clic derecho yseleccionamos publicar

Figura 24

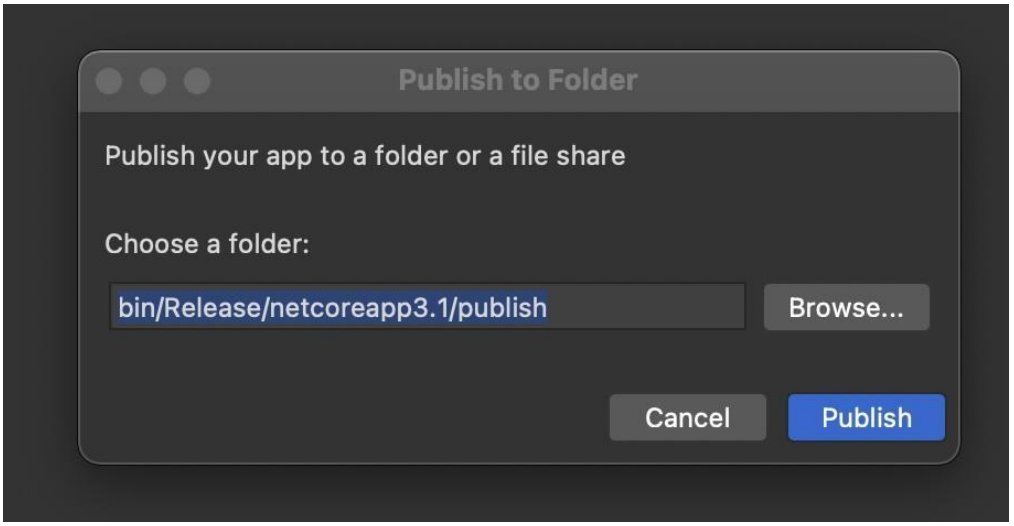
Figura sobre programación de la arquitectura



Nota. Seleccionar publicar desde un folder y colocaremos la ruta de donde se van aguardarlos archivos publicados y presionar en publicar.

Figura 25

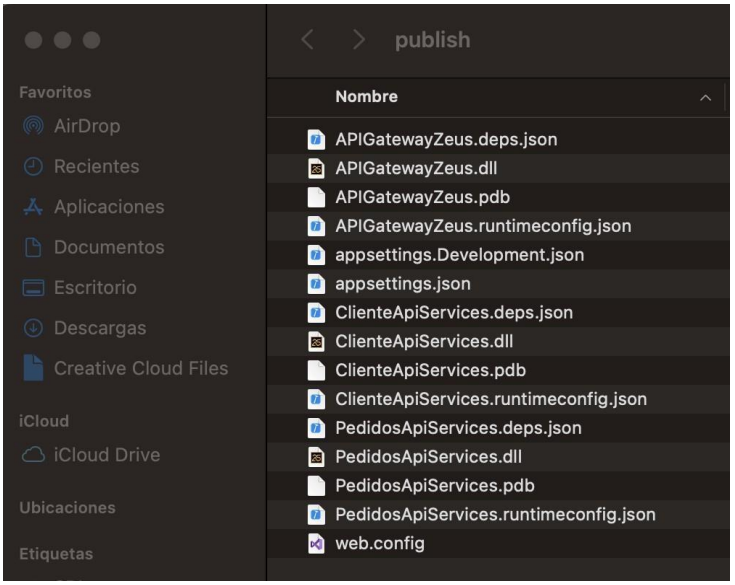
Figura sobre la publicación



- ✓ Terminada la publicación copiar los archivos de la carpeta publish y de pegarenla ruta C:\inetpub\wwwroot en la carpeta que creamos

Figura 26

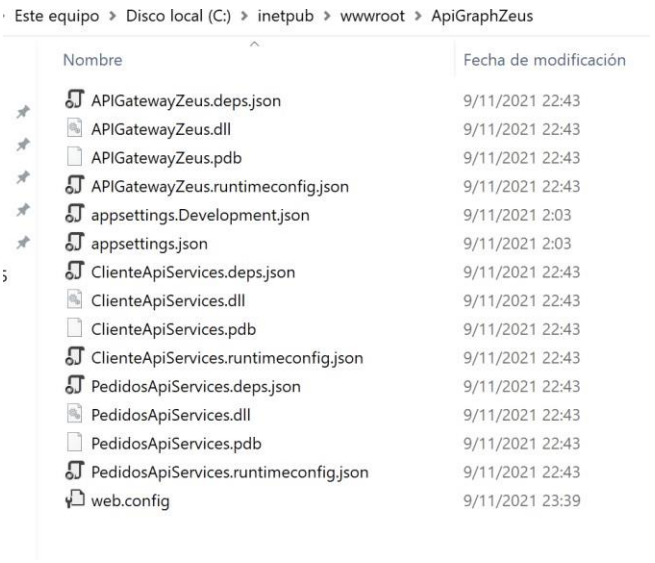
Figura sobre publicación



Nota. En la siguiente imagen están los archivos que permiten ejecutar los servicios del api. Es donde se aloja los complementos para que funcionen los servicios con lo cual va a interactuar con las aplicaciones que tengan acceso a dichos servicios, es decir, que son como los archivos de instalación del servicio.

Figura 27.

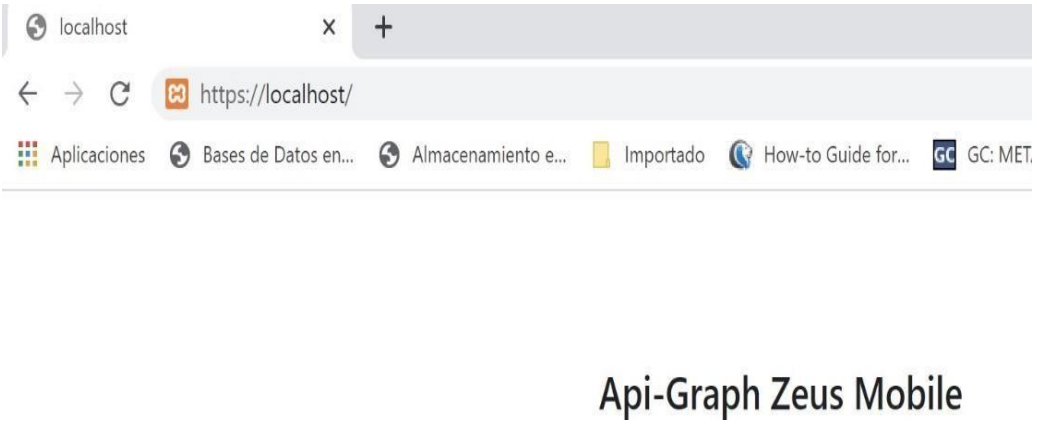
Figura sobre ApiZeus



Nota. Una vez realizado el proceso se debe reiniciar el servicio del nuevo sitio y probar.

Figura 28.

Figura sobre ApiZeus



Nota. Aquí se puede evidenciar como ya se tiene acceso con el api y el sistema móvil Zeus de la empresa Arest Consulting.

Capítulo IV

Consumir la arquitectura de microservicios mediante un cliente móvil

La aplicación del presente proyecto consiste en el diseño de una arquitectura en microservicios Api-Rest y GraphQL, el cual servirá para facilitar la usabilidad y mantenibilidad del código de los proyectos de software en el área de programación de la empresa Arest Consulting. Una vez desarrollada la arquitectura permite la rapidez en la consulta de los datos de la aplicación móvil Zeus, se deja a un lado la utilización de la arquitectura tradicional monolítica que ocupaban actualmente en el área de programación de la empresa Arest Consulting, esta arquitectura no era la adecuada para la aplicación Móvil porque al momento de realizar las peticiones hacia la base de datos tenía un retraso para visualizar la información y eso causabaproblemas con el cliente. Por ellos se opta con el desarrollo de la nueva arquitectura para lo cual se utiliza las herramientas de programación necesarias para la aplicación móvil Zeus, esto permitirá a la aplicación la mejora de sus funciones y procesos como es la gestión de los pedidos, cobranza, creación de rutas y visitas que generan al cliente. Esta aplicación permite integrar toda la información obtenida por el cliente con el Sistema Contable Fénix, la cual gestiona el proceso de cartera.

Para la Validación de la propuesta se usará el método de criterios de expertos para que inspeccionen el proceso de desarrollo de la arquitectura basada en microservicios Api-Rest yGraphQL bajo la aplicación de métodos adecuados y pertinentes se presentan los resultados.

Resultado del diagnóstico del problema

El diagnóstico del problema se ejecuta mediante las opiniones de los programadores de la Empresa Arest Consulting a través de una encuesta ejecutada mediante la plataforma de Google forms que tiene como objetivo facilitar la recolección de información para después seranalizada y tabulada.

Para la cual se va a ocupar la Ley de Pareto, que tiene como primera instancia el análisis de la tabla de frecuencia de los aspectos más importantes para la realización dela arquitectura basada en microservicios API-Rest y GraphQL.

A continuación, se presentará los resultados de la tabla 6

Tabla 7.

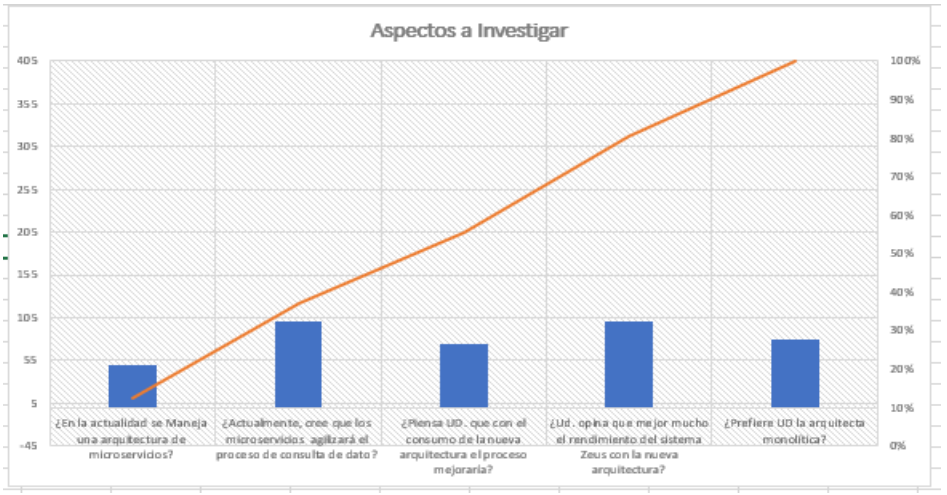
Frecuencia y Porcentaje de los aspectos para la investigación

N.º	Aspecto Analizar	Frecuencia	%	% Acumulado
1	¿En la actualidad se maneja una arquitectura de microservicios?	50%	12%	50%
2	¿Actualmente, cree que los microservicios agilizarán el proceso de consulta de datos?	100%	37%	150%
3	¿Piensa UD. ¿Con el consumo de la nueva arquitectura el proceso mejoraría?	75%	56%	225%
4	¿Ud. ¿Opina que mejoró mucho el rendimiento del sistema Zeus con la nueva arquitectura?	100%	80%	325%
5	¿Prefiere UD la arquitectura monolítica?	80%	100%	405%
	Total		100%	405%

Después de haber obtenido los resultados de los cálculos de la frecuencia y el porcentaje de cada pregunta realizada a los programadores de la empresa Arest Consulting se realiza el análisis aplicando la Ley de Pareto para poder evidenciar de una mejor manera cada aspecto mencionado.

Figura 29.

Diagrama de Pareto de los aspectos de la arquitectura de microservicios



Mediante el análisis de diagrama, se puede observar que es muy útil el consumo de la arquitectura en microservicios Api-Rest y GraphQL para que la aplicación móvil Zeus obtiene una mejora al hacer la consulta de la información que sea más rápida, al igual que la manipulación de los datos la cual permite que los clientes no tengan inconvenientes al momento de pasar la información a fénix.

Métodos específicos

El proceso del diseño de la arquitectura Api-Rest y GraphQL, es guiado por la metodología ágil Scrum la cual busca optimizar los tiempos del área de programación de la empresa Arest Consulting, esto permite mantener un contacto constante con el cliente representado por el Product Owner para especificar los requerimientos necesario para el desarrollo de la arquitectura de microservicios Api-Rest y GraphQL implementada con la aplicación móvil Zeus. La metodología Scrum se divide en las siguientes etapas: Planificación, desarrollo, revisión, retroalimentación, cada una de sus etapas son elaboradas con documentos técnicos, que detallan cada uno de sus procesos y sirva para el futuro poder dar mantenimiento a la arquitectura realizada.

Encuesta Semiestructurada: Para la recolección de las tareas de la arquitectura de microservicios Api-Rest y GraphQL implementada en la aplicación móvil Zeus se ejecuta mediante una entrevista semiestructurada donde el Product Owner describe las necesidades que existen en el sistema móvil Zeus.

Product backlog

Después de haber procedido a la encuesta se define el conjunto de tareas para el desarrollo,ya que son útiles para el equipo de trabajo, mediante el consumo de la arquitectura Api-Rest y GraphQL, el que muestra en la siguiente tabla:

Tabla 8.

Product Backlog del sistema

ID	TAREA	RESPONSABLE	PRIORIDAD
1	Analizar la base de datos que trabaja en la aplicación Zeus móvil	El investigador	Alta
2	Definición de la estructura del software	El investigador	Alta
3	Gestión del recurso de cliente	El investigador	Alta
4	Gestión del recurso de pedidos	El investigador	Alta
5	Autentificación de la Api	El investigador	Alta
6	Gestión de cliente y pedido	El investigador	Alta

Sprint backlog

En el Product Backlog se establece las actividades, las cuales van a hacer realizadas para eldesarrollo de la arquitectura de microservicios Api-Rest y GraphQL, siguiendo la metodologíaScrum, se define el sprint Backlog y el tiempo establecido para cada sprint implementando unsubconjunto de tareas el cual se detalla a continuación.

Sprint N.º 1

El diseño de la arquitectura en microservicios Api-Rest y GraphQL permite facilitar la usabilidady mantenibilidad del código de la aplicación móvil Zeus desarrollada en el área de programación de la empresa Arest Consulting, el desarrollo del primer sprint tiene como finalidad analizar cada uno de los procesos que se encuentra dentro de la estructura de la base de datos, teniendo claro sus entidades y atributos los cuales definen sus procesos, en la capa de negocio de la aplicación móvil Zeus. A continuación, se detalla el sprint 1.

Tabla 9.

Sprint N.º 1 Analizar la base de datos que trabaja en la aplicación Zeus móvil

DATOS DEL SPRINT			
NÚMERO	1		
FECHA DE INICIO:	17/septiembre/2021		
FECHA DE CULMINACIÓN:	24/septiembre/2021		
TAREAS POR DESARROLLAR			
PRIORIDAD	**DESCRIPCIÓN**	**RESPONSABLE**	**ESTADO**
Alta	Analizar la estructura de la base de datos de Zeus	El investigar	Realizada
Alta	Definir el conjunto de definición que se va a utilizar para la arquitectura	El investigar	Realizada
Alta	integrar la API con el sistema Zeus con los microservicios	El investigar	Realizada

Sprint N.º 2

El sprint 2 es principal porque analiza cada una de las funcionalidades a desarrollar, la arquitectura de microservicios Api-Rest y GraphQL con el consumo de la Api en la aplicación móvil Zeus. El desarrollo de este sprint tiene como finalidad detallar cada uno de los servicios que se va a utilizar para procesar la información de sus módulos. A continuación, se detalla el sprint 2.

Tabla 10

Sprint N.º 2 Definición de la estructura del software

DATOS DEL SPRINT	
NÚMERO	2
FECHA DE INICIO:	24/septiembre/2021
FECHA DE CULMINACIÓN:	01/octubre/2021

TAREAS POR DESARROLLAR			
PRIORIDAD	**DESCRIPCIÓN**	**RESPONSABLE**	**ESTADO**
Alta	Establecer las principales funcionalidades del contenedor de arquitectura Api-Rest y GraphQL.	El investigar	Realizada
Alta	Verificar los servicios que se obtienen y así estructurar la base técnica para el desarrollo del software.	El investigar	Realizada
Alta	Descripción detallada de los servicios que se van a usar.	El investigar	Realizada

Sprint N.º 3

El sprint 3 gestiona toda la información de los pedidos de la aplicación móvil Zeus por la cual se diseña el desarrollo de la arquitectura basada en microservicios Api-Rest y GraphQL. Este sprint permite la realización del CRUD mediante la manipulación de la información, realizandolas validaciones respectivas al momento de colocar los clientes. Este ítem se desarrolla en el transcurso del Sprint No. 3 con la siguiente planificación:

Tabla 11

Sprint N.º 3 Gestión del recurso de cliente

DATOS DEL SPRINT	
NÚMERO	3
FECHA DE INICIO:	01/octubre/2021
FECHA DE CULMINACIÓN:	08/octubre/2021

TAREAS POR DESARROLLAR			
PRIORIDAD	**DESCRIPCIÓN**	**RESPONSABLE**	**ESTADO**
Alta	Se requiere tener un servicio API que me permita gestionar los clientes.	El investigar	Realizada

TAREAS POR DESARROLLAR

PRIORIDAD	DESCRIPCIÓN	RESPONSABLE	ESTADO
Alta	Se requiere tener un servicio que me permita crear, editar y eliminar los clientes.	El investigar	Realizada

TAREAS POR DESARROLLAR

PRIORIDAD	DESCRIPCIÓN	RESPONSABLE	ESTADO
Alta	Se requiere que el servicio me permita validar si el cliente ya se encuentra registrado, al igual que la validación como la cédula y el RUC.	El investigar	Realizada

Sprint N.º 4

El sprint 4 se relaciona con el sprint anterior de manera independiente con la gestión de los pedidos de la aplicación móvil Zeus, lo cual se incluyen la realización del CRUD para la manipulación de la información, con sus validaciones respectivas al momento de colocar los clientes. Este ítem se desarrolla en el transcurso del Sprint No.4 con la siguiente planificación:

Tabla 12

Sprint N.º 4 Gestión del recurso de pedidos

DATOS DEL SPRINT	
NÚMERO	
FECHA DEINICIO:	08/octubre/2021
FECHA DE CULMINACIÓN:	15/octubre/2021

TAREAS POR DESARROLLAR

PRIORIDAD	DESCRIPCIÓN	RESPONSABLE	ESTADO
Alta	Se requiere tener un servicio API que me permita gestionar los pedidos.	El investigar	Realizada

TAREAS POR DESARROLLAR			
PRIORIDAD	**DESCRIPCIÓN**	**RESPONSABLE**	**ESTADO**
Alta	Se requiere tener un servicio que me permita crear, editar y eliminar los pedidos.	El investigar	Realizada
Alta	Se requiere que el servicio me permita validar si el pedido ya se encuentra registrado.	El investigar	Realizada

Sprint N.º 5

El sprint 5 permite al cliente tener una autentificación para el sistema y tener controlado el acceso al mismo y poder tener un consumo adecuado de la API para la consulta de la información. Este ítem se desarrolla en el transcurso del Sprint No. 5 con la siguiente planificación:

Tabla 13

Sprint N.º 5 autentificación de la API

DATOS DEL SPRINT	
NÚMERO	5
FECHA DE INICIO:	15/octubre/2021
FECHA DE CULMINACIÓN:	22/octubre/2021

TAREAS POR DESARROLLAR			
PRIORIDAD	**DESCRIPCIÓN**	**RESPONSABLE**	**ESTADO**
Alta	Se requiere un servicio de API de autenticación para permitir que ya estén registrados para conectarse.	El investigar	Realizada
Alta	El servicio debe permitir ingresar el usuario o correo electrónico y la contraseña.	El investigar	Realizada

TAREAS POR DESARROLLAR			
PRIORIDAD	**DESCRIPCIÓN**	**RESPONSABLE**	**ESTADO**
Alta	Si la validación de las credenciales es correcta permitirá acceder a los servicios restantes, caso contrario se visualizará un mensaje de error.	El investigar	Realizada

Sprint N.º 6

El sprint 6 En lista cada una de las gestiones, este es un sprint de menor importancia parael desarrollo del api. Este ítem se desarrolla en el transcurso del Sprint No. 6 con la siguiente planificación:

Tabla 14

Sprint N.º 6 Gestión de cliente y pedido

DATOS DEL SPRINT	
NÚMERO	6
FECHA DE INICIO:	22/octubre/2021
FECHA DE CULMINACIÓN:	29/octubre/2021

TAREAS POR DESARROLLAR			
PRIORIDAD	**DESCRIPCIÓN**	**RESPONSABLE**	**ESTADO**
Alta	Se requiere en listar la lista de pedidos.	El investigar	Realizada
Alta	Al ingresar la lista de pedidos me debe permitir crear, editar o eliminar un cliente.	El investigar	Realizada
Alta	Al generar un pedido debe validar si el producto existe	El investigar	Realizada

TAREAS POR DESARROLLAR

PRIORIDAD	DESCRIPCIÓN	RESPONSABLE	ESTADO
Alta	Al ingresar la lista de clientes me debe permitir generar, editar o eliminar un cliente.	El investigar	Realizada
Alta	Al generar un cliente debe validar si existe, al igual que la validación de la cédula o RUC	El investigar	Realizada
Alta	al eliminar el cliente se debe validar que no existan pedidos registrados con el cliente	El investigar	Realizada

Requisitos Específicos

Requerimientos Funcionales

Tabla 15
Requerimiento Funcional 01

N°. Requerimiento:	N°01
Requerimiento:	Inicio de Sesión.
Cualidad:	El administrador debería identificarse para acceder a cualquier parte del sistema.
Detalle del requerimiento:	El sistema podrá ser utilizado únicamente por los usuarios que se encuentren dentro de la base de datos.
Requerimientos no funcionales:	✓ N°01 ✓ N°02 ✓ N°05 ✓ N°06
Prioridad: Alta	

Tabla 16

Requerimiento Funcional 02

N°. Requerimiento:	N°02
Requerimiento:	Generar lista de datos Cliente.
Cualidad:	El usuario debe acceder con sus credenciales y depende a los permisos, puede ingresar a cualquier parte de la aplicación
Detalle del requerimiento:	El sistema podrá ser usado por los usuarios que tengan permisos de visualización de los datos.
Requerimientos no funcionales:	✓ N°01 ✓ N°03 ✓ N°04 ✓ N°06
Prioridad: Alta	

Tabla 17

Requerimiento Funcional 03

N°. Requerimiento:	N°03
Requerimiento:	Crear, editar y eliminar Cliente.
Cualidad:	El usuario debe acceder con sus credenciales y tener permisos de edición y eliminación de datos
Detalle del requerimiento:	El sistema podrá ser empleado por los usuarios que tengan permisos de edición y eliminación de datos.
Requerimientos no funcionales:	✓ N°01 ✓ N°03 ✓ N°05 ✓ N°06
Prioridad: Alta	

Tabla 18

Requerimiento Funcional 04

N°. Requerimiento:	RF04
Requerimiento:	Generar lista de datos Pedido.
Cualidad:	El usuario debe acceder con sus credenciales y depende a los permisos, puede ingresar a cualquier parte de la aplicación

N°. Requerimiento:	RF04
Detalle del requerimiento:	El sistema podrá ser utilizado por los usuarios que tengan permisos de visualización de los datos.
Requerimientos no funcionales:	✓ N°01 ✓ N°02 ✓ N°05 ✓ N°06
Prioridad: Alta	

Tabla 19.

Requerimiento Funcional 05

N°. Requerimiento:	N°05
Requerimiento:	Crear, editar y eliminar Pedido.
Cualidad:	El usuario debe acceder con sus credenciales y tener permisos de edición y eliminación de datos
Detalle del requerimiento:	El sistema podrá ser empleado por los usuarios que tengan permisos de edición y eliminación de datos.
Requerimientos no funcionales:	✓ N°01 ✓ N°02 ✓ N°05 ✓ N°06
Prioridad: Alta	

Requerimientos No Funcionales

Tabla 20.

Requerimiento No Funcional 01

N°. Requerimiento:	N°.01
Requerimiento:	Interfaz del sistema.
Cualidad:	La aplicación móvil Zeus deberá presentar una interfaz al usuario que sea de manera sencilla con la finalidad de que sea manejable para los usuarios, tomando en cuenta los permisos que tienen los mismos.

Detalle del requerimiento:	El sistema debe ser una interfaz de uso amigable y sencilla, la cual se debe ajustarse a los procesos de la aplicación.
Prioridad:	Alta

Tabla 21.

Requerimiento No Funcional 02

N°. Requerimiento:	N°.02
Requerimiento:	Mantenimiento de la arquitectura
Cualidad:	La arquitectura debe tener una buena estructura para facilitar al cliente su mantenimiento.
Detalle del requerimiento:	La Arquitectura debe tener una buena documentación para el mantenimiento de la arquitectura.
Prioridad del requerimiento:	**Alta**

Tabla 22

Requerimiento No Funcional 03

N°. Requerimiento:	N°.03
Requerimiento:	Desempeño
Cualidad:	La Arquitectura en microservicios garantiza un buen desempeño en la alimentación de la información mediante el sistema.
Detalle del requerimiento:	La Arquitectura garantiza el desempeño de toda la información y registro de procesos.
Prioridad:	**Alta**

Tabla 23

Requerimiento No Funcional 04

N°. Requerimiento:	N°.04
Requerimiento:	Usabilidad de Usuario
Cualidad:	La Arquitectura garantiza al cliente el acceso de la información con los procesos correctos.
Detalle del requerimiento:	Acceso a la información únicamente con los clientes permitidos a la aplicación realizando correctamente el proceso de CRUD.
Prioridad:	**Alta**

Tabla 24

Requerimiento No Funcional 05

N°. Requerimiento:	N°.05
Requerimiento:	Confiabilidad del Sistema.
Cualidad:	La arquitectura en microservicios tendrá un funcionamiento de las 24 horas los 7 días de la semana. ya que tiene toda la información de los datos para gestionar los procesos.
Detalle del requerimiento:	La Arquitectura de microservicios tiene la disponibilidad de los servicios que los clientes deben tener la disponibilidad de 24 horas en los 7 días, garantizando las posibles fallas.
Prioridad:	**Alta**

Tabla 25

Requerimiento No Funcional 06

N°. Requerimiento:	N°.06
Requerimiento:	Seguridad en los Datos
Cualidad:	La arquitectura garantiza al cliente la seguridad necesaria para la ejecución de los procesos.
Detalle del requerimiento:	La arquitectura en microservicios garantiza que toda la Información de los datos no pierda su integridad en el momentode la utilización.
Prioridad	**Alta**

Casos de Usos

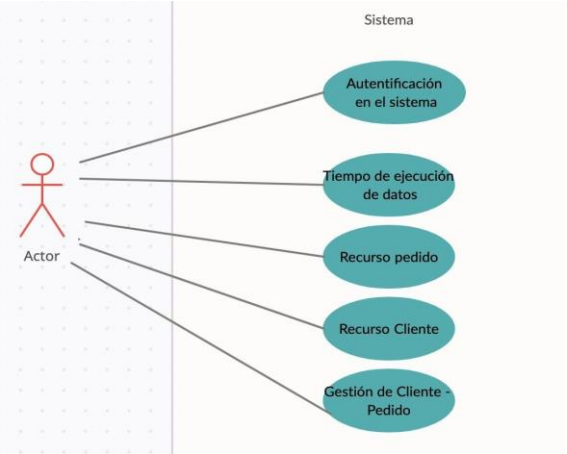
El diseño de la arquitectura en microservicios API-Rest y GraphQL es usado para la aplicación móvil Zeus del área de programación de la empresa Arest Consulting, desarrollan aplicaciones de software con una arquitectura tradicional monolítica y necesitanla rapidez en el manejo de toda la información que se encuentra en la aplicación Zeus paraque se pueda ejecutar de una manera más óptima y rápida.

Diagrama de Especificación de Casos de Usos

A continuación, se presenta un diagrama de casos de usos donde se detalla cada una de las funcionalidades que son útiles para el desarrollo del proyecto.

Figura 30

Diagrama de Casos de Usos de la Arquitectura de microservicios



En la Figura 30 se puede visualizar el diagrama de caso de usos, el cual está relacionado con requerimiento recogido por el Product Owner para el desarrollo de la arquitectura basada en microservicios Api-Rest, GraphQL, ya que esta parte es de suma importancias para el desarrollo del proyecto.

Detalle de los casos de usos de la aplicación

En la tabla 26 se detalla cada uno de los casos de usos que fueron necesarios para el desarrollo de la arquitectura basada en microservicios para la aplicación Zeus.

Tabla 26.

Detalle de los Casos de Uso

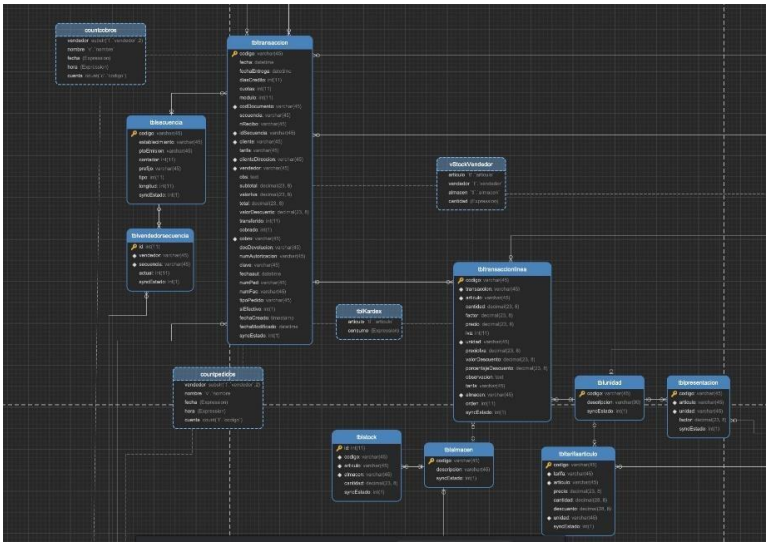
Casos de Usos (CU)	Descripción	Usuarios
CU001	Autentificación al sistema	✓ Usuario ✓ Investigador
CU002	Tiempo de Ejecución de los datos	✓ Usuario ✓ Investigador
CU003	Gestión de la información de los pedidos	✓ Usuario ✓ Investigador
CU004	Gestión de la información de los clientes	✓ Usuario ✓ Investigador
CU005	Gestión de la información de pedidos y clientes	✓ Usuario ✓ Investigador

Diagrama de entidad relación

Para el desarrollo de la arquitectura en microservicios API-Rest y GraphQL es necesario el diseño del modelo entidad relación para tener identificado cada uno de sus procesos en las entidades y atributos mediante las relaciones de cada uno en la base de datos. A continuación se presenta el modelo de la base de datos con el cual se va a trabajar.

Figura 31

Diagrama Entidad Relación

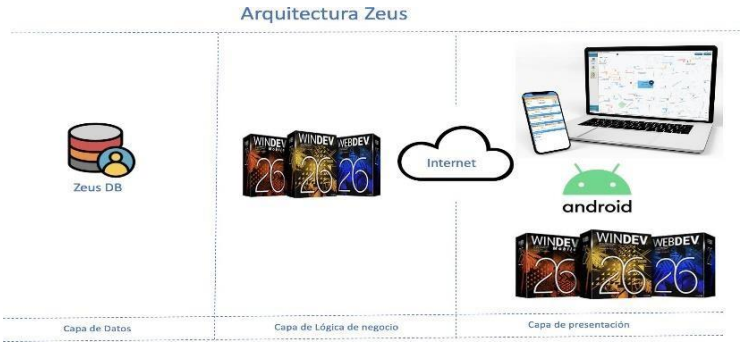


Nota. En la siguiente figura se presencia el modelo de la base de datos que tiene el desarrollo de los servicios para el cliente móvil Zeus.

Arquitectura en API-Graph - aplicación móvil Zeus

Figura 32

Arquitectura de microservicios



Nota. aquí se describe el modelo de la arquitectura basada en microservicios la cual se divide en 3 capas.

La aplicación Zeus móvil se desarrolló utilizando una arquitectura monolítica y servicios web,la cual permite interactuar con la base de datos, y permite que la aplicación funcione mediante capas e interactúa con la herramienta de Windev como capa de negocio y de presentación, como se visualiza en la Figura

Desarrollo de aplicación móvil Zeus

La arquitectura de microservicios Api-Rest y GraphQL será consumida por la aplicación móvil Zeus y tiene como finalidad que su funcionamiento sea de la misma manera por capas, pero en esta ocasión, cada capa se trabajará individualmente es decir que la capade presentación solo va a concentrarse en presentar las pantallas por distintas tecnologías como Windev, y Ionic, para acceder a la base de datos debe pasar por un Api-Gateway la cual permite realizar las peticiones de las API, en donde se va a alojar la capa de negocios en esta capa permitirá estructurar las reglas de negocios, también la capa de negocios permite interactuar con la capa de base de datos la cual obtendrá todas las entidades de la aplicación Zeus y de ahí se podrá extraer toda la información que necesita el cliente.

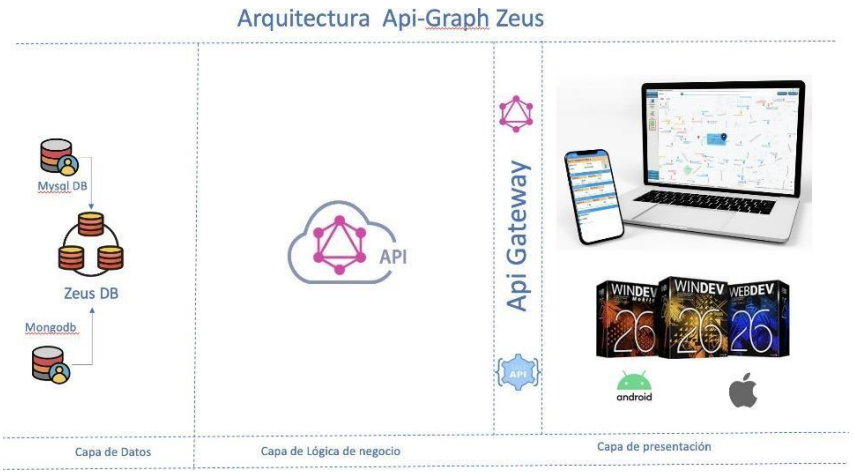
Implementación de la arquitectura Api-Rest y GraphQL en la aplicación Móvil Zeus

El desarrollo de la arquitectura en microservicios Api-Rest y GraphQL tiene como finalidad la implementación en la aplicación móvil Zeus, utilizando la metodología Scrum y aplicandolos sprints desarrollados.

A continuación, se presentarán las pantallas de la aplicación Zeus con el consumo de la Api-Rest y GraphQL.

Figura 33

Diagrama Arquitectura de microservicios



Pruebas de Arquitectura en microservicio Api-Rest y GraphQL

En esta etapa se procede a efectuar las pruebas de funcionalidad de la nueva arquitectura de microservicios Api-Rest y GraphQL, mediante la implementación de la misma en la aplicación móvil Zeus. Donde se aplicará una plantilla donde se describen todas las pruebas efectuadas a la aplicación para conocer la funcionalidad de los procesos y mejoras que se puede obtener al implementar de microservicios Api-Rest y GraphQL en la aplicación.

A continuación, se presentan los resultados de las pruebas efectuadas:

Tabla 27

Pruebas, funcionalidad de la arquitectura de microservicios Api-Rest y GraphQL

N.º	Variables de Entrada	Resultados Esperados	Estado
1	Analizar la base de datos que trabaja en la aplicación Zeus móvil	El sistema permite consultar la información de la base de datos de una manera muy ágil y eficaz para el cliente	Aprobado
2	Definición de la estructura del software	Una vez analizada la estructura y obtenida la información de los datos, se procede a las consultas con la nueva arquitectura que permite a la aplicación Zeus una mejor muy útil para los clientes del sistema	Aprobado
3	Gestión del recurso de cliente	Una vez realizada la API con la nueva arquitectura se verifica la eficacia del mismo y el entendimiento de cómo se va a realizar la manipulación de los datos	Aprobado
4	Gestión del recurso de pedidos	Una vez procesada la API con la nueva arquitectura, sé comprueba la eficacia del mismo y el entendimiento de cómo se va a ejecutar la manipulación de los datos	Aprobado

N.º	Variables de Entrada	Resultados Esperados	Estado
5	Autentificación de la Api	el sistema recibe una petición para verificar las credenciales de ingreso a las APIS, mediante la consulta de la base de datos, permitiendo acceder a los diferentes servicios	Aprobado
6	Gestión de cliente y pedido	El sistema presenta pantalla de los distintos formularios en las cuales se puede ejecutar procesos de creación, edición, listado y eliminación de los datos.	Aprobado

Capítulo V
Evaluación normas ISO/IEC 25000 de calidad externa.

En este capítulo se va a evaluar las características de usabilidad y mantenibilidad de la arquitectura basada en la arquitectura de microservicios Api-Rest y GraphQL aplicada en el sistema móvil Zeus, mediante la aplicación las normas ISO/IEC 25000 de calidad externa y se va a ocupar la ISO/IEC 25040 permite la evaluación de los requisitos parala aplicación y se puede evaluar la evaluación de la calidad interna y externa.

Modelo de calidad en uso.

El departamento de programación de la empresa Arest Consulting son los encargados de la evaluación del modelo de calidad con las normas ISO, en este caso se escogió al Product Owner y al Scrum Máster, los cuales evaluaron las características de usabilidad y eficiencia colocadas sub-características y evaluando por porcentajes cada una de las mismas.

A continuación, se presenta una tabla con la evaluación de los modelos de calidad con las características escogidas por los encargados.

Tabla 28.

Modelo de calidad

Caracteristicas	Subcaracterísticas	Peso - características	Peso - sub características	Criterio de medición
Usabilidad	Inteligibilidad/Capacidad de adecuación	40%	10%	Respuesta satisfactoria
	Aprendizaje		15%	Respuesta satisfactoria
	Operabilidad/Capacidad para serutilizado		15%	Respuesta satisfactoria
Eficacia	Modularidad	60%	20%	Respuesta satisfactoria
	Reusabilidad		20%	Respuesta satisfactoria
	Capacidad para ser modificado		20%	Respuesta satisfactoria

Medición del modelo de calidad externa de software

En esta etapa de implementación de la aplicación, Zeus se realiza a través de los métodos empíricos de la encuesta, el cual permitirá evaluar el conocimiento de los modelos de calidad.

Encuesta: Mediante la Encuesta planteada se puede analizar los aspectos de medición de la usabilidad y la eficacia y está basada en preguntas referentes al consumo de la API con el sistema móvil Zeus.

La encuesta utiliza una escala del 1 siendo bajo y 5 satisfactorios.

1. Qué tan complejo encontró al trabajar con la API Rest - GraphQL
2. Al desarrollar el código es más entendible.
3. El código mejora la usabilidad entre desarrolladores.
4. El tiempo de ejecución del servicio es óptimo para la API Rest - GraphQL.
5. Las funciones de la API Rest - GraphQL se encuentran integradascorrectamente.
6. Cree usted que el consumo de la API Rest - GraphQL es beneficioso para el sistema.
7. Piensa usted que la arquitectura basada en micro servicio API Rest - GraphQL esdemasiado complicada para el sistema Zeus.
8. El equipo de programación encuentra difícil el uso de la arquitectura basada en microservicio API Rest - GraphQL para la implementación de otros proyectos.
9. El desarrollo de la arquitectura de microservicios API Rest - GraphQL tiene todas lasfuncionalidades para una buena ejecución del sistema móvil Zeus.
10. En el área de programación les gusta trabajar más con una arquitectura monolítica ouna arquitectura de microservicios.

Tabla 29.
Porcentajes de valores

Criterio	Valor
Excelente	**5**
Muy Bueno	**4**
Bueno	**3**
Regular	**2**
insuficiente	**1**

Población y muestra

No se calcula la muestra debido a que se utiliza un muestreo no probabilístico debido a que solo es un equipo de 4 programadores en la empresa Arest Consulting y se trabajara con todos.

Medición del modelo de evaluación

Medición de las normas de calidad basada en la ISO/IEC 25000 aplicando la ISO2540,la cual evalúa la calidad interna y externa de la aplicación móvil Zeus con la implementaciónde la arquitectura basada en microservicios Api-rest, GraphQL las cuales se detalla en la tabla siguiente.

Tabla 30.

Medición de los modelos de calidad

Cualidades de la norma ISO /IEC 25000	Sub-Cualidades de la norma ISO /IEC 25000	Medición de las cualidades de la norma ISO /IEC 25000
Usabilidad	Inteligibilidad/Capacidad de adecuación	**Y=** C/E; C= Actividades complementarias del proceso, E =Actividades realizadas intentadas.
	Aprendizaje	**X=** [(ΣCi) * 0,25] / U; Ci= Objetivos

Cualidades de la normaISO /IEC 25000	Sub-Cualidades de la norma ISO /IEC 25000	Medición de las cualidades de lanorma ISO /IEC 25000
		errores, E= Actividades propuestas.
Eficacia	Modularidad	**Y=** C/E;
	Reusabilidad	C= Actividades complementarias del proceso,
		E =Actividades realizadas intentadas.
	Capacidad para ser Modificado	**X=** [(ΣCi) * 0,25] / U; Ci= Objetivos planteados por el usuario del sistema.
		U= Número de clientes.
		U= Número de clientes.
		X= 1-(C/E); C= Actividades con errores, E= Actividades sugeridas.

Resultados de la encuesta

Con la finalidad de validar el desarrollo de la arquitectura de microservicios, se realizó la aplicación de una encuesta a cuatro usuarios que tienen títulos profesionales referentes a la carrera de software con una experiencia mínima de 4 años en adelante.

La encuesta está formulada con preguntas de selección dividida en las siguientes fases cinco preguntas referentes a la usabilidad de la arquitectura de microservicios y su utilidad. Evaluando las preguntas con una escala de 1 es insuficiente a 5 es Excelente de manera ascendente.

Tabla 31.

Evaluación de expertos

Expertos	Media Aritmética	Moda X
Experto N°1	4.5	4.5
Experto N°2	5	5
Experto N°3	5	5
Experto N°4	5	5

Una vez aplicadas las encuestas se evalúa que el experto 4, 3 , 2 tiene una valoración enla media aritmética están de acuerdo en el desarrollo de la arquitectura basada en microservicios.
Mediante que el experto 2 da como resultado una media aritmética de 4.5 obteniendo como resultado media aritmética y moda de 4.5 es decir que la propuesta a desarrollarseestá en un rango de muy buena con una tolerancia de casi excelente.

El experto 1 obtiene un resultado de una media aritmética de 4.5 dando como óptimo el desarrollo de la arquitectura y su aplicación en el sistema Zeus.

Evaluación de las características de la usabilidad

Es la capacidad de evaluar a un productor de software usando y resaltando el atractivo que es para el usuario.es la capacidad que permite que sea utilizado y determina las características y discapacidades que pueda obtener un sistema.

Sub-característica de Inteligibilidad/Capacidad de adecuación

La Inteligibilidad/Capacidad de adecuación mide la capacidad de la arquitectura basada en microservicios Api-rest y GraphQL, permitiendo al cliente saber si el uso de la misma es adecuado para sus necesidades.

Tabla 32.
Especificación de métricas sobre la Sub-característica de Inteligibilidad/Capacidad de adecuación.

Métrica	Inteligibilidad/Capacidadde adecuación	Valor
ΣAi	Sumatoria del número de tareaspor usuarios.	15,00

Evaluación de las características de la eficacia

La medición de esta característica se mide mediante la acción que se tiene empleado por cada usuario y es medido mediante la sumatoria en cada acción que realice se puede evidenciar en la siguiente tabla.

Tabla 33.

Métricas de la eficacia

Métrica	Inteligibilidad/Capacidadde adecuación	Valor
ΣAi	Relación del tiempo empleado en cada tarea con el tiempo experimentado por cada uno de los usuarios.	12,00 milisegundos

Evaluación de los resultados en la aplicación Zeus Móvil con la arquitectura de GraphQL y Rest

Para esta evaluación se toma en cuenta los siguientes criterios mediante un cuadro donde se detalla el tipo de arquitectura y el tipo de datos que se va a evaluar.

Tabla 34.

Evaluación de resultados de las arquitecturas

Acción	Arquitectura	Proceso	Tiempo Respuesta
Consulta de datos	**Api-Rest**	Clientes	163
		Pedidos	187
	GraphQL	Clientes	133
		Pedidos	108

Los resultados obtenidos en los procesos de consultas para Clientes y Pedidos ocupando la Arquitectura del Api- Rest lleva mayor tiempo de cantidad 163 y 187 milisegundos, mientras que el mismo proceso fue evaluado con la arquitectura de GraphQL, el tiempo es de 133 y 108 milisegundos eso quiere decir que la arquitectura no consume mucha memoria RAM para la consulta de datos y efectúa un mejor rendimiento para la aplicación móvil Zeus.

Capítulo VI

Conclusiones y recomendaciones

Conclusiones

- ✓ Una vez analizado todo el marco teórico sobre las dos arquitecturas monolíticas y microservicios se puede identificar sus desventajas y se llega a la conclusión que una empresa no debería utilizar arquitecturas monolíticas para el cambio de actualizaciones de las aplicaciones, si no implementar una arquitectura de microservicios con la utilización de las normas ISO/IEC 25000 de Calidad Externa para la implementación de sus cambios.
- ✓ Se aplica al desarrollo de la nueva arquitectura de microservicios ocupando servicios independientes para el momento de realizar un cambio no afecte a la aplicación del cliente y que todo el proceso sea de una manera transparente y conla aplicación de las normas ISO/IEC 25000 de Calidad Externa para la evaluaciónde la calidad de la aplicación Zeus de la empresa Arest Consulting.
- ✓ En el consumo de la arquitectura basada en microservicios se obtiene como resultado la determinación de los componentes necesarios que se denominan servicios para la utilización de la arquitectura y tiene una mejor respuesta con la aplicación Zeus ya que para los clientes es transparente los cambios y no tiene ninguna afectación en toda la aplicación y se optimiza los tiempos de los programadores para subir un cambio.
- ✓ Mediante la validación de la arquitectura se aplicó las normas ISO/IEC 25000 de calidad externa evaluando las métricas de Eficacia para la evaluar el comportamiento del tiempo y la utilización de los recursos y la usabilidad para saber el aprendizaje y la operatividad de la aplicación Zeus del Cliente.

Recomendaciones

- ✓ Para el manejo de la arquitectura en microservicio se debe tener en cuenta las ventajas y desventajas que puede brindar la arquitectura y de igual manera el manejo de los servicios que se van a realizar dentro de una empresa.

- ✓ La recopilación de los requerimientos con las personas encargadas de las empresas para el desarrollo de los servicios que se van a implementar para las diferentes aplicaciones tanto de escritorio como Web.

- ✓ El desarrollo de la nueva arquitectura basada en microservicios es ejecutado mediante el Lenguaje de Programación C# con la conexión de la base de datos MySQL y la utilización de la metodología ágil SCRUM para el desarrollo.

- ✓ Se recomienda la utilización de las normas ISO/IEC 25000 de calidad externa para evaluar la calidad de la aplicación con la implementación de los servicios y tener claro si es funcional o no donde el cliente.

Bibliografía

- ✓ Balalaie, A. Heydarnoori, and P. Jamshidi, "Migrating to cloud-native architectures using microservices: an experience report," in European Conference on Service-Oriented and Cloud Computing, 2015, pp. 201-215: Springer.
- ✓ Apiservice. (2020, 22 julio). Microservicios. Apiservice. https://apiservice.cl/servicios/implementar-arquitecturas-digitales/microservicios/
- ✓ Apollo graphql. (2019). Apollo Docs. Retrieved May 12, 2019, from https://www.apollographql.com/docs/
- ✓ Astorga, P. P. (2020, 27 marzo). Arquitectura de microservicios: qué es, ventajas y desventajas. Decide Soluciones. Https://decidesoluciones.es/arquitectura-de-microservicios/
- ✓ Buna, S. (2016). Learning graphql and Relay. Van Haren Publishing.
- ✓ Cerami, E. (2002b). Web Services Essentials. Van Duuren Media.
- ✓ Djandrw, A. (2019, 13 abril). ¿Qué es Scrum? - Andrew Djandrw. Medium. Https://medium.com/@andrewdjandrw/qu%C3%A9-es-scrum-674c6b791af4
- ✓ El concepto de graphql. (s. F.). Https://www.redhat.com/es. Https://www.redhat.com/es/topics/api/what-is-graphql
- ✓ Fernández, Y. (2019, 23 agosto). API: qué es y para qué sirve. Xataka. https://www.xataka.com/basics/api-que-sirve
- ✓ Ghebremicael, E. S. (2017). Transformation of REST API to graphql for opentosca. University of Stuttgart. Https://doi.org/10.18419/opus-9352
- ✓ Ghebremicael, E. S. (2017). Transformation of REST API to graphql for opentosca. University of Stuttgart. Https://doi.org/10.18419/opus-9352
- ✓ Graphql. (2018, junio). Http://spec.graphql.org/. Http://spec.graphql.org/June2018/
- ✓ Graphql: A query language for apis. (s. F.). Graphql.org. Https://graphql.org/learn/
- ✓ Greene, J., & Stellman, A. (2014). Learning Agile. O'Reilly Media, Inc. Retrieved from https://learning.oreilly.com/library/view/learning-agile/9781449363819/

- ✓ ISO 25000. (2019). PORTAL ISO 25000. Retrieved December 4, 2019, from https://iso25000.com/
- ✓ ISO/IEC 25010. (2011). ISO/IEC 25010:2011(en), Systems and software engineering Systems and software Quality Requirements and Evaluation (square) — System and Software quality models. Retrieved December 4, 2019, from https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en
- ✓ ISO/IEC 25022. (2016). ISO/IEC 25022;2016, Systems and software engineering — Systems and software quality requirements and evaluation (square) — Measurement of quality in use. Retrieved from https://www.iso.org/obp/ui/#iso:std:iso-iec:25022:ed-1:v1:en 91
- ✓ ISO/IEC 25040. (2011). ISO/IEC 25040:2011, Systems and software engineering — Systems and software Quality Requirements and Evaluation (square) — Evaluation process.
- ✓ Jacobson, D., Brail, G., & Woods, D. (2011). Apis: A Strategy Guide. Van Duuren Media.
- ✓ Kimokoti, B. (2018). Beginning graphql. Van Haren Publishing
- ✓ Linux-Fundation. (2019). Introduction to graphql | graphql. Retrieved February 21, 2019,from https://graphql.org/learn/
- ✓ M. Villamizar et al., "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," in Computing Colombian Conference (10CCC), 2015 10th, 2015, pp. 583-590: IEEE.
- ✓ Maya, Edgar & López, Daniel. (2018). Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web.
- ✓ Mezo, B. M. (2008). main characteristics of companies using service-oriented architecture (soa) and of their operation context. jistem Journal of Information Systems and Technology Management, 5(2), 269-304.
    - Https://doi.org/10.4301/s1807-17752008000200005
- ✓ Molina, B., Vite, H., & Dávila, J. (2018). Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software.

- ✓ Moreno, Jorge Jair, Bolaños, Liliam Paola, Navia, Manuel Alejandro (2010). Exploración de Modelos y Estándares de Calidad Para el Producto Software. Revista UIS Ingenierías, 39-53. Https://www.redalyc.org/articulo.oa?Id=553756877003
- ✓ Pablo, J. (2018, 6 marzo). Marco de trabajo Scrum. elConspirador.com. https://www.elconspirador.com/2013/08/16/marco-de-trabajo-scrum/
- ✓ Porcello, E., & Banks, A. (2018). Learning graphql. Van Duuren Media.
- ✓ Sandoval, J. (2009). Restful Java Web Services. Packt Pub.
- ✓ Slate, A. (2019, 8 julio). Qué es una API: todo lo que necesitas saber. Https://www.wrike.com/es. Https://www.wrike.com/es/blog/que-es-una-api-necesitas-saber/
- ✓ T. Salah, M. J. Zemerly, C. Y. Yeun, M. Al-Qutayri, and Y. Al-Hammadi, "The evolution of distributed systems towards microservices architecture," in Internet Technology and Secured Transactions (ICITST), 2016 11th International Conference, 2016, pp. 318-325: IEEE.
- ✓ Theory, G. (s. F.). Qué es una API REST y para qué se utiliza. Geeky Theory. Https://geekytheory.com/que-es-una-api-rest-y-para-que-se-utiliza
- ✓ Vázquez-Ingelmo, A., Cruz-Benito, J., & García-Peñalvo, F. J. (2017). Improving the OEEU's data-driven technological ecosystem's interoperability with graphql. Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality - TEEM 2017, 1-10.Https://doi.org/10.1145/3144826.31454.

Anexos