

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**  
**Facultad de Ingeniería en Electricidad y Computación**



“MIGRACIÓN DE UN SISTEMA ELECTRÓNICO ORIENTADO A  
SERVICIOS DE COBROS Y PAGOS DE UNA ENTIDAD  
BANCARIA DESDE UNA ARQUITECTURA DE CAPAS A UNA  
ARQUITECTURA DE MICROSERVICIOS BASADOS EN  
DOMINIO (DDD), PARA ESTABLECER UN SISTEMA  
DESCENTRALIZADO, DE ACOPLAMIENTO FLEXIBLE Y  
SOSTENIBLE EN EL TIEMPO”

**EXAMEN DE GRADO (COMPLEXIVO)**

Previo a la obtención del Título de:

**MAGISTER EN SISTEMAS DE INFORMACIÓN  
GERENCIAL**

GLADYS MARIELLA LINDAO TIGRERO

GUAYAQUIL - ECUADOR

2021

## AGRADECIMIENTO

Agradezco a Dios por darme la salud, fortaleza y sabiduría para culminar una meta más en mi vida, a mi madre Lic. Gladys Tigreiro González, a mis hijos Ronald Alexander y Sebastián Emanuel, por todo ese tiempo prestado, y que al verlos me daban el impulso para no desistir, a mi ángel mi papi Ing. Domingo Lindao (+).

A mi familia por siempre estar pendiente, a mis amigas, y a todas aquellas personas que me decían que sí podía.

A mis compañeros de proyectos de la maestría.



## **DEDICATORIA**

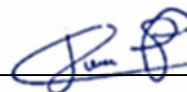
Dedico este trabajo a mis hijos Ronald Alexander y Sebastián Emanuel, porque ellos son los motorcitos de mi vida, a mi mami Gladys que es mi pilar, y que juntos de la mano de Dios seguiremos cosechando éxitos.

## TRIBUNAL DE SUSTENTACIÓN

A handwritten signature in blue ink, appearing to be 'Lenin Freire Cobo', written over a horizontal line.

MSIG. Lenin Freire Cobo.

COORDINADOR MSIG

A handwritten signature in blue ink, appearing to be 'Juan García Plúa', written over a horizontal line.

MSIG. Juan García Plúa

PROFESOR MSIG

## RESUMEN

Una Empresa Proveedor de Sistemas de Soluciones Financieras, situada en la Ciudad de Guayaquil, cuenta con un producto que es el Sistema Electrónico Orientado a Servicios de Cobros y Pagos (SEOSCP), dirigido para una entidad bancaria, dicho sistema con el pasar de los años ha sido migrado a diferentes plataformas y cuya última versión se categoriza como una aplicación monolítica.

EL SEOSCP se encuentra desarrollado en ASP.NET de Microsoft .NET Framework versión 2.0, y este dejó de tener soporte por parte del proveedor, lo que generó para la entidad bancaria un riesgo para los clientes debido a la falta de actualización de las características y funcionalidades del sistema, quedando un tanto obsoletas con relación al avance de nuevas tecnologías.

Para esto, como solución se optó por migrar la arquitectura de capas de los módulos de Cobros y Pagos a una arquitectura de MicroServicios, en esta arquitectura en lugar de clases son reemplazadas por controladores y los métodos por capacidades, que han sido desarrollados en diferentes lenguajes de programación, Angular, C# de .Net Core, así mismo contienen APIs REST y el repositorio de base de datos se mantiene en Oracle.

## ÍNDICE GENERAL

AGRADECIMIENTO .....	II
DEDICATORIA .....	III
TRIBUNAL DE SUSTENTACIÓN.....	IV
RESUMEN .....	V
ÍNDICE GENERAL.....	VI
ABREVIATURAS Y SIMBOLOGÍA .....	IX
ÍNDICE DE TABLAS .....	X
ÍNDICE DE FIGURAS.....	XI
INTRODUCCIÓN .....	xiii
CAPÍTULO 1 .....	1
GENERALIDADES.....	1
1.1 DESCRIPCIÓN DEL PROBLEMA .....	1
1.2 SOLUCIÓN PROPUESTA .....	3
CAPÍTULO 2 .....	4
METODOLOGÍA PARA EL DESARROLLO DE LA SOLUCIÓN .....	4
2.1. DEFINICIONES .....	4
2.2. ARQUITECTURA MONOLÍTICA .....	4
2.3. ARQUITECTURA MICROSERVICIOS .....	6
2.3.1 DDD (Domain Driven Design) .....	6

2.4.	API REST .....	7
2.5.	DEVOPS.....	8
2.6.	DMZ (ZONA DESMILITARIZADA).....	9
2.7.	SITUACIÓN TÉCNICA ACTUAL.....	9
2.7.1	DISEÑO ARQUITECTÓNICO.....	9
2.7.2	ESPECIFICACIONES DE HARDWARE.....	11
2.7.3	DISEÑO DE LA APLICACIÓN .....	11
2.8.	SOLUCIÓN IMPLEMENTADA .....	14
2.8.1	DISEÑO ARQUITECTÓNICO EN MICROSERVICIOS.....	14
2.8.2	ESPECIFICACIONES DE HARDWARE.....	16
2.8.3	SELECCIÓN DEL FRONT - END .....	17
2.8.4	SELECCIÓN DEL BACK – END .....	18
2.8.5	TECNOLOGÍAS Y HERRAMIENTAS USADAS.....	19
2.8.6	DESARROLLO DE LA SOLUCIÓN .....	20
2.8.7	DESARROLLO DEL FRONT-END .....	21
2.8.8	DESARROLLO DEL BACK-END .....	21
2.8.9	SEGURIDAD EN SERVICIOS .....	23
2.9.	DOCUMENTACIÓN TÉCNICA .....	24
2.10.	PRUEBAS DE FUNCIONAMIENTO .....	24
CAPÍTULO 3	.....	25

ANÁLISIS DE RESULTADOS .....	25
3.1 MEJORA EN LA INTERFAZ DEL USUARIO .....	25
3.2 MEJORA EN TIEMPO DE RESPUESTA .....	27
3.3 INDICADOR DE AVANCES .....	28
CONCLUSIONES Y RECOMENDACIONES .....	30
BIBLIOGRAFÍA .....	32



## ABREVIATURAS Y SIMBOLOGÍA

<b>API</b>	Interfaz de Programación de Aplicaciones
<b>CAD</b>	Capa de acceso a base de datos
<b>CIU</b>	Capa de interface al usuario
<b>CRN</b>	Capa de reglas de negocio
<b>DDD</b>	Domain Driven Design
<b>DMZ</b>	Zona desmilitarizada
<b>MSA</b>	Micro Services Architecture
<b>REST</b>	Representational State Transfer
<b>SEOSCP</b>	Sistema Electrónico Orientado a Servicios de Cobros y Pagos
<b>OAS</b>	Open Automation Software

## ÍNDICE DE TABLAS

<b>Tabla 1.</b> Características del servidor web y de componentes del SEOSCP 4.0.....	11
<b>Tabla 2.</b> Características de servidores SEOSCP 5.0 .....	17
<b>Tabla 3.</b> Herramientas usadas para el desarrollo de la migración .....	20
<b>Tabla 4.</b> Tiempo de respuesta por versión. ....	28
<b>Tabla 5.</b> Tiempo de respuesta minutos vs segundos .....	28

## ÍNDICE DE FIGURAS

<b>Figura 2.1.</b>	Representación de una arquitectura monolítica .....	5
<b>Figura 2.2.</b>	Solución del proyecto SEOSCP 4.0 (ASP.NET) .....	5
<b>Figura 2.3.</b>	Representación de una arquitectura de MicroServicios .....	6
<b>Figura 2.4.</b>	Representación de metodología DevOps .....	8
<b>Figura 2.5.</b>	Representación de la arquitectura SEOSCP 4.0.....	10
<b>Figura 2.6.</b>	Opción de Ingreso de Orden Rápida SEOSCP 4.0.....	12
<b>Figura 2.7.</b>	Llamado de un método transacción Ingreso de orden rápida.....	13
<b>Figura 2.8.</b>	Definición de un método transacción Ingreso de orden rápida.....	13
<b>Figura 2.9.</b>	Representación de la arquitectura SEOSCP 5.0.....	16
<b>Figura 2.10.</b>	Resultados de estadísticas de Marcos web .....	18
<b>Figura 2.11.</b>	Resultados de estadísticas de Otros marcos, bibliotecas y herramienta .....	18
<b>Figura 2.12.</b>	Controladores y capacidades dentro de un proyecto .....	22
<b>Figura 2.13.</b>	Pruebas de petición mediante herramienta Postman .....	22
<b>Figura 2.14.</b>	Estructura de un servicio del SEOSCP 5.0 .....	23
<b>Figura 2.15.</b>	Diagrama para acceder a una API.....	23
<b>Figura 3.1.</b>	Pantalla de SEOSCP 5.0, ingreso de “Orden Rápida”, módulo de Pagos. ....	26
<b>Figura 3.2.</b>	Figura 4.2: Pantalla de SEOSCP 5.0, “Ingreso de Archivos”, módulo de Pagos. ....	27

**Figura 3.3.**    Indicador de avances ..... 29

## INTRODUCCIÓN

En la actualidad está en la mira por parte de las entidades bancarias actualizar sus plataformas y no quedarse estancados en una arquitectura monolítica, es por ello que para esta entidad bancaria quien cuenta con un amplio portafolio de aplicaciones entre heredadas y/o soluciones cerradas, su visión trasciende para uno de sus productos: SEOSCP 4.0.

El presente proyecto muestra como objetivo la migración que bajo análisis y por necesidad técnica se optó de una arquitectura de capas a una Arquitectura de MicroServicios basados en dominio (DDD), debido a que éste ofrece beneficios como agilidad, escalabilidad independiente y fácil mantenimiento.

La empresa proveedora es responsable del desarrollo de los servicios y mantener la funcionalidad del SEOSCP 4.0 a excepción de las opciones establecidas como unificadas.

Este proyecto lo hemos dividido en 2 fases: desarrollo e implementación del SEOSCP 5.0 aplicando refactoring en su código fuente del front-end y back-end, entrega de documentación, pruebas respectivas de ambas partes. Como parte de la segunda fase se tiene contemplado el desarrollo e implementación del producto nuevo SEOSCP móvil; mismo que no forma parte del objetivo del presente documento.

# CAPÍTULO 1

## GENERALIDADES

### 1.1 DESCRIPCIÓN DEL PROBLEMA

Una Empresa Proveedorora de Sistemas de Soluciones Financieras, situada en la Ciudad de Guayaquil, que se mantiene con más de quince años de experiencia en aplicaciones bancarias, cuenta con un Sistema SEOSCP, que en su versión 2.0, fue desarrollado e implementado en el año 2004, en la plataforma ASP clásico, y base de datos Microsoft SQL y Oracle, en una entidad bancaria pionera en nuestro país. En el año 2011, se actualizó el SEOSCP a la versión 3.0, bajo el entorno de ASP.NET de Microsoft .NET Framework versión 2.0. Dicho sistema permite administrar órdenes bancarias, tanto de forma masiva como individual y contiene los siguientes módulos:

*Módulo de cobros.*- Transacciones concernientes a los cobros que ordenan los Clientes (Proveedores, contratistas, etc.) la entidad bancaria a sus terceros.

*Módulo de pagos.*- Transacciones concernientes a los pagos que ordena el Cliente para que la entidad bancaria ejecute.

El SEOSCP está categorizado como una aplicación monolítica, basada en una arquitectura de capas, puesto que todos sus módulos se empaquetan como una única unidad que se puede implementar y ejecutar; en este caso ASP.Net se ejecuta en el Servidor de Internet de Información (IIS) utilizando un diseño de capas [1] independiente entre la interfaz de usuario, lógica de aplicación y acceso a datos, que con el tiempo tiende a crecer para satisfacer las necesidades empresariales [2]. Por ende se presentan los siguientes problemas:

- Los elementos individuales del sistema no son independientemente escalables.
- Las pruebas se hacen más complicadas, aumentando vulnerabilidades.
- Se complica el mantenimiento del código.
- Se obstaculiza el crecimiento y la estabilidad futura.

Además, a inicios del año 2019 el SEOSCP, se migró a la versión 4.0 utilizando la misma plataforma de desarrollo de ASP.Net, pero optimizando tecnologías y mejorando procesos, a pesar de que el soporte y mantenimiento brindado por Microsoft para .NET Framework 2.0 y Visual Studio 2005 finalizó en 2016 [3].

## 1.2 SOLUCIÓN PROPUESTA

Siendo uno de los sistemas principales para la entidad bancaria, en el año 2020 se propone como solución migrar la arquitectura de capas de los módulos de Cobros y Pagos a una arquitectura de MicroServicios, mediante el uso de Api Rest, separando el Front-End del Back-End y descomponiendo y desacoplando la solución monolítica en una serie de MicroServicios aplicando los principios de diseño de dominios (DDD).

Para ello se eligió un lenguaje de programación adecuado y común para todas las partes interesadas. Los módulos más relevantes de la aplicación monolítica fueron identificados para luego poder definir una transformación de los mismos basados en la arquitectura de MicroServicios. Las API REST fueron creadas en Microsoft .Net Core, tecnología que ya no solo es compatible con servidores Windows (IIS), sino también se puede desplegar en servidores Linux.



## **CAPÍTULO 2**

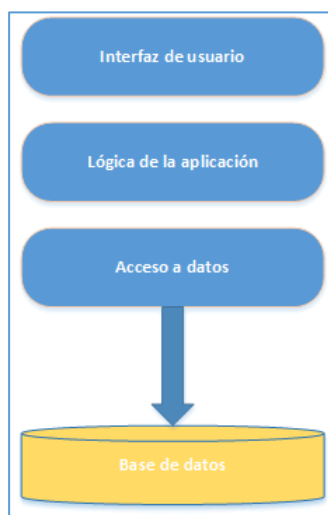
### **METODOLOGÍA PARA EL DESARROLLO DE LA SOLUCIÓN**

#### **2.1. DEFINICIONES**

Para un mejor entendimiento del presente proyecto, se expone breves definiciones de la arquitectura monolítica y de microServicios, así como también de las plataformas a utilizarse.

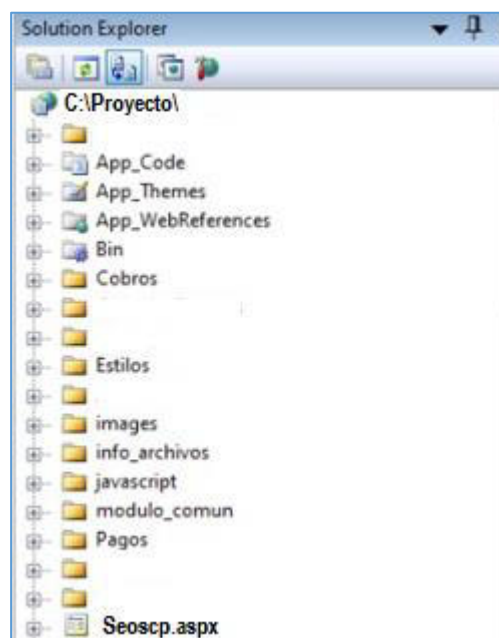
#### **2.2. ARQUITECTURA MONOLÍTICA**

La arquitectura monolítica integra la interface de usuario, la lógica de negocio y acceso a datos, que al momento de compilarse se empaqueta en un solo ejecutable (.exe), todos los módulos y librerías se empaquetan junto con la aplicación principal. [4]



**Figura 2.1.** Representación de una arquitectura monolítica  
Fuente: El autor

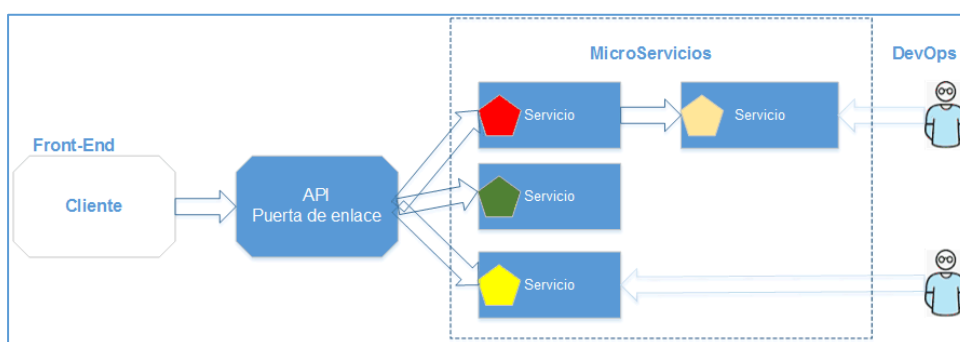
Dentro del IDE de desarrollo de Microsoft ASP.Net, al abrir un proyecto así se visualiza una arquitectura monolítica:



**Figura 2.2.** Solución del proyecto SEOSCP 4.0 (ASP.NET)  
Fuente: El autor

## 2.3. ARQUITECTURA MICROSERVICIOS

La arquitectura de MicroServicios, en siglas MSA (Micro Services Architecture), son pequeños servicios que se ejecutan de manera independiente y autónoma. Cada microServicio es un código que consigue estar en un lenguaje de programación diferente; es la matriz de una funcionalidad de negocio específica y única. Los microServicios se comunican entre sí mediante APIs, y cuentan con sistemas de almacenamiento propios, lo que evita la sobrecarga y caída de la aplicación. Permite también hacer un cambio de negocio de forma fácil, escalable sin que afecte a las otras funcionalidades del sistema. [5] [6]



**Figura 2.3.** Representación de una arquitectura de MicroServicios  
Fuente: El autor

En la figura 2.3, el cliente (front-end) y los microservicios, son conectados por una puerta de enlace de API. También vemos que existe la conexión entre servicios que tienen la lógica de negocio.

### 2.3.1 DDD (Domain Driven Design)

Ayuda a diseñar un modelo de dominio desde la lógica del negocio o basándonos en la problemática que se necesita resolver, enfocando el

desarrollo de manera más eficiente en las tareas y requerimientos que realmente generan valor. [6]

## 2.4. API REST

**API** (Interfaz de Programación de Aplicaciones), es un conjunto de peticiones que consiste en comunicar datos entre distintas aplicaciones, utiliza el protocolo HTTP para el envío de datos; como operaciones básicas tenemos el método GET (lee registro del host), el método POST (crea registro en el servidor), el método DELETE (borra registro) y el método PUT (actualiza el registro). [7]

**REST** (Representational State Transfer), contiene conjunto de restricciones para que las solicitudes del protocolo HTTP den cumplimiento a las directrices dadas en dicha arquitectura. [7]

Ventajas que brinda REST:

- **Separación entre el cliente y el servidor.**- Como su propia palabra lo dice separa la interfaz de usuario y el acceso a datos. [8]
- **Visibilidad, fiabilidad y escalabilidad.**- Se da al momento que un equipo de desarrollo escala un producto sin problema alguno, la flexibilidad es porque tanto fron-end y back-end se pueden encontrar en tecnologías diferentes, y permite modificar a nivel de base datos, toda vez que se envíe de forma correcta los datos. [8]
- **Independiente del tipo de plataformas o lenguajes.**- Los servicios pueden ser desarrollados en diferentes lenguajes, comunicados entre sí a través de

REST, pero considerando que debe haber el intercambio de información entre los lenguajes escogidos. [8]

Es decir API Rest, es utilizar una API para acceder a aplicaciones back-end, y que dicha comunicación se efectúe con estándares de arquitectura Rest ya definidos. [7].

Las peticiones del protocolo HTTP realizadas en API Rest o también denominadas para este proyecto capacidades, devuelven los datos en formato JSON.

## 2.5. DEVOPS

Es una metodología que cambia el modo en el que se maneja el ciclo de desarrollo de software, a nivel técnico pero en particular a nivel cultural. Los equipos de desarrollo y de Operaciones trabajan de una manera colaborativa y bidireccional. Este equipo cumple con el ciclo completo de desarrollo de software, con el fin de garantizar procesos seguros y rápidos, para que sus entregas sean confiables y de calidad. [9]



**Figura 2.4.** Representación de metodología DevOps  
Fuente: El autor

## **2.6. DMZ (ZONA DESMILITARIZADA)**

Es una red aislada estructurada dentro de la red interna de una empresa. Aquí se encuentran ubicados todos los equipos y/o recursos de la empresa que deben ser accesibles desde la Internet, por ejemplo un servidor web. [10]

## **2.7. SITUACIÓN TÉCNICA ACTUAL**

### **2.7.1 DISEÑO ARQUITECTÓNICO**

El SEOSCP 4.0 está compuesto de 2 partes:

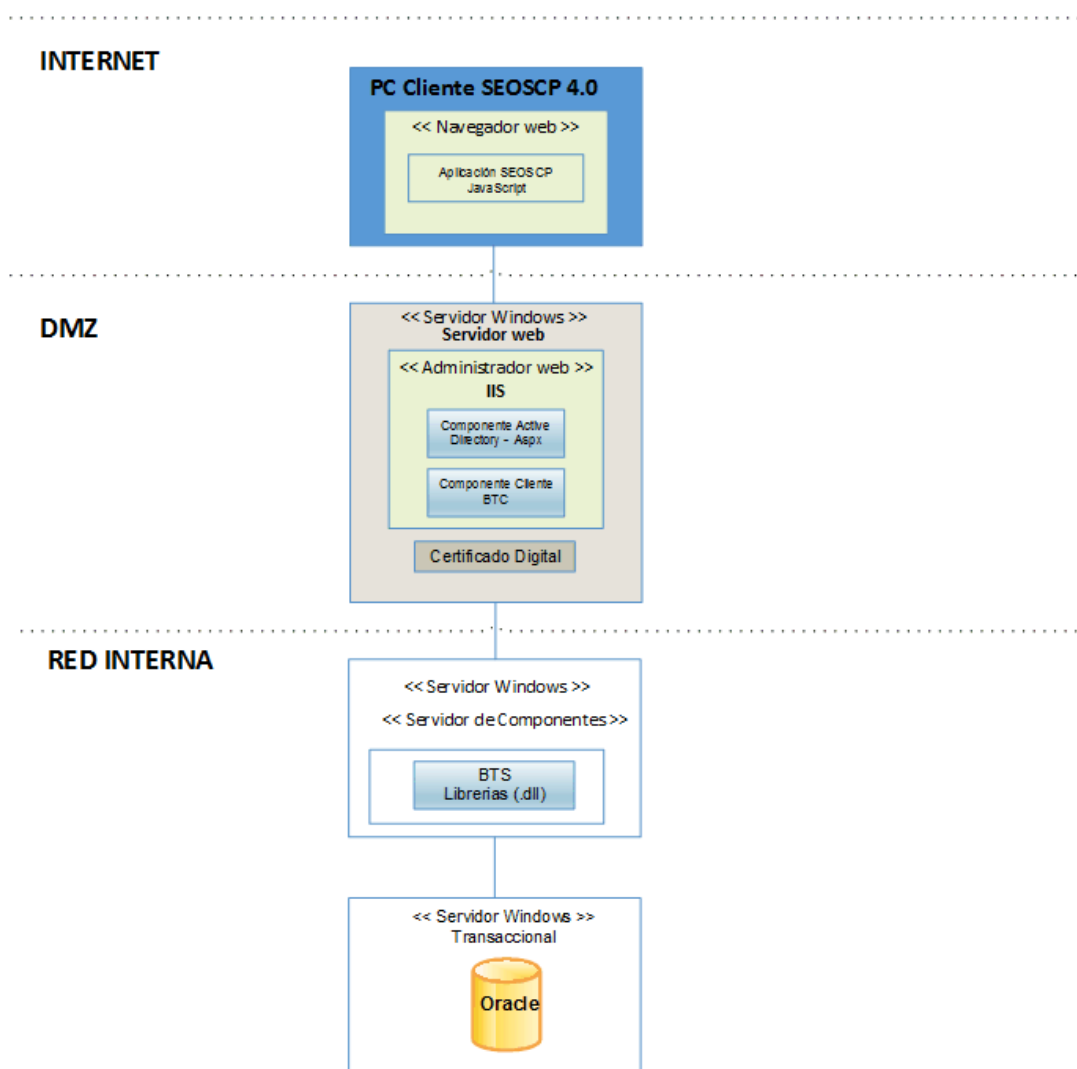
- 1) El software construido en .net Framework.
- 2) Un conjunto de bases de datos Oracle.

El estilo arquitectónico del Software esta a su vez compuesto de varias capas:

- a. capa de formularios,
- b. capa de código común,
- c. capa de seguridad

Dentro de los formularios se encuentran los medios visuales que permiten al usuario interactuar con la aplicación. El código escrito en c# (clases) se enlaza con la capa de código de común en forma de controles personalizados para ejecutar lógica de programación como validaciones y algunas reglas de negocios, finalmente se comunica con la base de datos por medio de una capa de seguridad (servidor de componente) que mediante librerías (.dll) ejecuta la autorización de cada transacción. La autenticación se realiza considerando técnicas de multifactor MFA

(Multi Factor Autentication) que lo realiza el componente BTS, es la seguridad que se aplica para las operaciones de autorización y autenticación contra la base de datos.



**Figura 2.5.** Representación de la arquitectura SEOSCP 4.0  
Fuente: El autor

## 2.7.2 ESPECIFICACIONES DE HARDWARE

A continuación se detalla las características del servidor web, en ambiente desarrollo, pre - producción y producción:

Equipo	Descripción
Servidor Web	<ul style="list-style-type: none"> <li>• Sistema Operativo: Windows Server 2016 Standard</li> <li>• Procesador: Intel(R) Xeon(R) CPU X5650 @2.67GHz (4 processors)</li> <li>• Memoria (RAM): 16,0 GB</li> <li>• Tipo Sistema: Sistema operativo de 64 bits</li> <li>• Espacio en disco duro: 500 GB</li> </ul>

**Tabla 1.** Características del servidor web y de componentes del SEOSCP 4.0  
Fuente: El autor

## 2.7.3 DISEÑO DE LA APLICACIÓN

Como ejemplo está la pantalla de una de las opciones que forma parte del caso del proyecto como es la de Ingreso Orden Rápida:







## 2.8. SOLUCIÓN IMPLEMENTADA

En esta sección detallaremos el diseño arquitectónico, la selección de las tecnologías a usarse en el front-end y back - end, así como también el desarrollo del SEOSCP 5.0.

### 2.8.1 DISEÑO ARQUITECTÓNICO EN MICROSERVICIOS

La arquitectura del SEOSCP 5.0 se divide en 3 partes: Internet, DMZ y red interna bancaria.

En la parte inicial del proceso tenemos la pc del cliente que realiza la petición accediendo al sistema web, url del SEOSCP 5.0; esta petición llega a la **DMZ** (Zona desmilitarizada) de la entidad bancaria expuesta al mundo y hace que en el servidor procese este requerimiento HTTP, luego retorna al navegador del cliente la página web inicial estructurada básicamente por varios html, JavaScript, entre otros elementos.

Todos los requerimientos posteriores, a esta primera formación de la página web dentro del navegador, es decir desde el login hasta que sale de la aplicación incluida las transacciones, van a ser ejecutadas a través del APIM Externo, en otras palabras el servidor web solo sirve para cargar la página en su totalidad, mientras que las peticiones de cualquier transacción son a través del APIM externo.

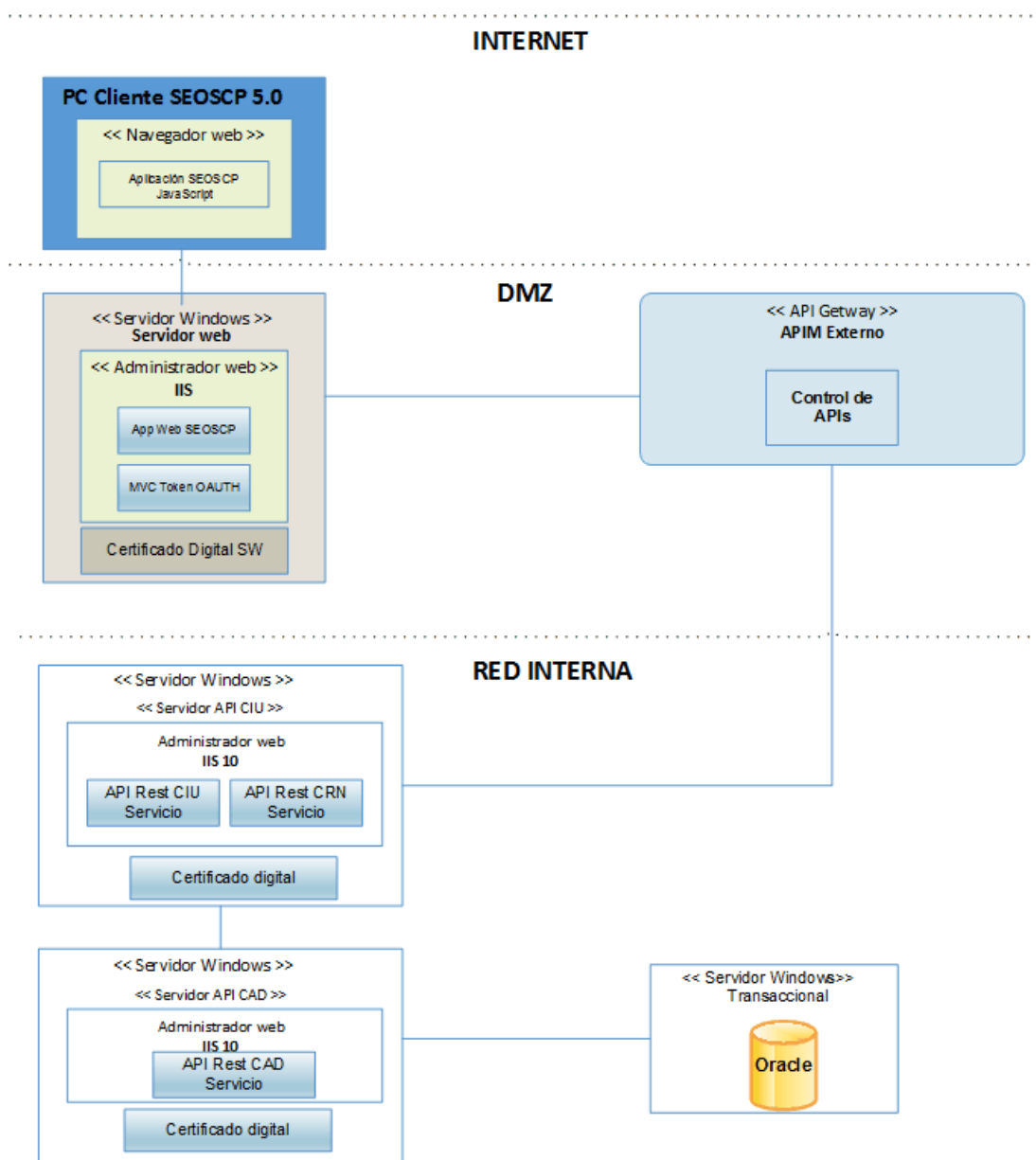
**APIM Externo** lo que hace es recibir todas las invocaciones de todos los servicios de API Rest que son ejecutados desde el navegador de la aplicación web, estos requerimientos son validados desde el APIM tanto

en su composición, en su estructura, y también a nivel de seguridad se hace filtro para detectar vulnerabilidades. Una vez que ha pasado los niveles de control, la petición es ruteada a la red interna.

En la **red interna** esta petición es atendida por el servicio CIU respectivo que está implementado dentro de un servidor IIS (Internet Information Services), este servicio es consumido, solo se dedica a recibir el requerimiento y a validar los datos de entrada, es decir hace una segunda validación de datos de entrada, un poco más especializado en lo que se refiere a la validación de datos elementales, como cédula, ruc, etc.

Luego de validar los datos se procede con la invocación al API Rest de la capa de acceso a datos. Este es un servicio CAD también implementado en otro Servidor IIS, que valida los parámetros de entrada y salida para comunicarse con la base de datos Oracle.

Una vez que se realiza esa ejecución la base de datos devuelve los resultados y en el mismo flujo que llegó retorna los resultados hacia el computador del usuario, es decir el ruteo del servidor del CAD al CIU, del CIU al APIM Externo y este hacia la computadora del cliente.



**Figura 2.9.** Representación de la arquitectura SEOSCP 5.0  
Fuente: El autor

## 2.8.2 ESPECIFICACIONES DE HARDWARE

Como requisitos mínimos para los equipos con la nueva arquitectura, la entidad bancaria consideró las siguientes características:

Equipo	Descripción
Servidor Web	<ul style="list-style-type: none"> <li>• Sistema Operativo: Windows Server 2019 Standard</li> <li>• Procesador: Intel(R) Xeon(R) Gold 6348</li> <li>• Cantidad de núcleos: 6</li> <li>• Memoria (RAM): 16,0 GB</li> <li>• Tipo Sistema: Sistema operativo de 64 bits</li> <li>• Espacio en disco duro: 500 GB</li> </ul>
Servidor API CIU  Servidor API CAD	<ul style="list-style-type: none"> <li>• Sistema Operativo: Windows Server 2019 Standard</li> <li>• Procesador: Intel(R) Xeon(R) Gold 6348</li> <li>• Cantidad de núcleos: 28</li> <li>• Memoria (RAM): 32,0 GB</li> <li>• Tipo Sistema: Sistema operativo de 64 bits</li> <li>• Espacio en disco duro: 1 TB</li> </ul>

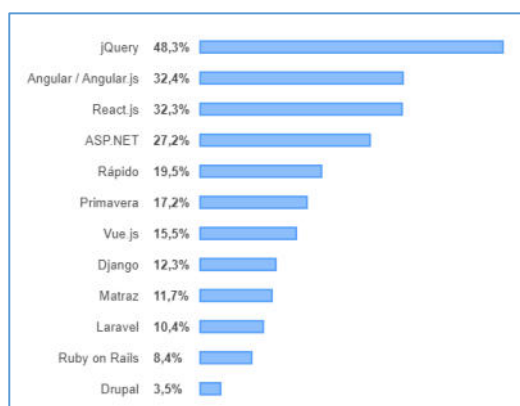
**Tabla 2.** Características de servidores SEOSCP 5.0

Fuente: El autor

### 2.8.3 SELECCIÓN DEL FRONT - END

Bajo el nuevo modelo de Arquitectura de MicroServicios para la migración del SEOSCP 5.0, el Arquitecto de Software para la selección de la tecnología del Front - End, analizó las estadísticas del año 2019 en **Stack Overflow**, es un sitio público que se utiliza para que los programadores y profesionales aprendan, impartan conocimientos, colaboren y desarrollen sus carreras [11]; en la Figura 2.7, se muestran

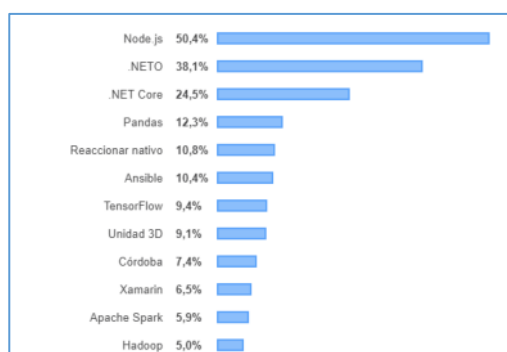
los resultados dando en primer lugar con un 48,3% lo ocupaba JQuery y en segundo lugar Angular / Angular.js con un 32,4%.



**Figura 2.10.** Resultados de estadísticas de Marcos web [12]

#### 2.8.4 SELECCIÓN DEL BACK – END

Por lo consiguiente para la selección de la tecnología del Back - End, se analizó las estadísticas del año 2019 en Stack Overflow, en primer lugar Node .js con 50,4%, Neto con un 38,1% y .Net Core 24,5%.



**Figura 2.11.** Resultados de estadísticas de Otros marcos, bibliotecas y herramienta [12]

Ante la documentación y análisis presentados, la entidad bancaria decidió que la migración de la nueva versión del SEOSCP sea en

Angular, porque es una tecnología desacoplada, y por tener un Framework robusto. Así también con base en la experiencia obtenida por parte del proveedor, trabajar con C# de .Net Core.

## 2.8.5 TECNOLOGÍAS Y HERRAMIENTAS USADAS

Lo que se quiere es la disponibilidad y aumentar la usabilidad del SEOSCP, por lo que en el siguiente cuadro se muestran las tecnologías y herramientas usadas para el desarrollo del proyecto:

Herramienta	Versión	Definición	Motivo
Visual Studio Code	1.49.0	Editor de código fuente	Se ha utilizado las extensiones Angular 8 Snippets - TypeScript, Html, Angular Material JS-CSS-HTML
Visual Studio 2019 Community y .Net Core	-	Herramienta para la creación de aplicaciones web y servicios en la nube.	Se ha utilizado para el desarrollo del <b>back-end</b> .
Angular Cli	10.1.3	Herramienta de interfaz de línea de comando, se utiliza para mantener aplicaciones angulares directamente desde un shell de comandos.	Se ha utilizado para el desarrollo de una single-page application (SPA), utilizando como framework de diseño MaterialDesign.
Postman	8.1.0	Herramienta diseñada para realizar las pruebas de las API REST.	Se ha utilizado esta herramienta para probar las API REST creadas.
OAS Service		Herramienta para diseñar, construir, documentar, y utilizar servicios web Rest.	Se ha utilizado esta herramienta para probar desde un navegador las API REST creadas.



Google Drive	Online	Repositorio que permite el almacenamiento de archivos y se encuentra disponible las 24 horas.	Se ha utilizado este programa para compartir documentación propia del proyecto, referente a los diseños, desarrollos, QA y demás, para el conocimiento del equipo de trabajo. También como parte de copia de seguridad de nuestro trabajo.
GitHub	Online	Es una plataforma para alojar proyectos y sirve como control de versionamiento.	Se ha utilizado como herramienta para la actualización de cambios del proyecto tanto la parte de front-end y back-end.
Skype	Online	Es una plataforma para comunicarse entre dos o más personas.	Se ha utilizado como herramienta para mantener comunicación tanto vía chat como video llamadas para reuniones, se activó esta modalidad desde inicios de pandemia.
Microsoft Teams	Online	Plataforma de colaboración y comunicación, reuniones mediante videollamadas.	Se ha utilizado como herramienta para reuniones mantenidas con el personal de la entidad bancaria, se activó esta modalidad desde inicios de pandemia.

**Tabla 3.** Herramientas usadas para el desarrollo de la migración  
Fuente: El autor

### 2.8.6 DESARROLLO DE LA SOLUCIÓN

Se cuantificó en el módulo de Pagos 35 transacciones y en el módulo de Cobros 30 transacciones en producción.

Se elaboró una matriz de referencia, como bitácora en la que se registró por cada transacción del SEOSCP 4.0, información como nombre del módulo, nombre de transacción, nombre de formulario, nombre del control, nombre de clase, nombre del método, esto con la finalidad de asignar las tareas del back - end como del front - end.

Así mismo se estableció unificación de 2 de las opciones principales de los módulos, como es el caso del ingreso manual de Orden Rápida con la Carga de archivo para generar órdenes.

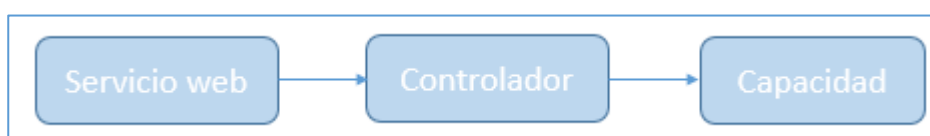
### 2.8.7 DESARROLLO DEL FRONT-END

Para el desarrollo del Front-End se migraron las transacciones en Angular, realizando los ajustes en el diseño y mejora de funcionalidades de las pantallas definidas con la entidad bancaria, de aquí se va a realizar el llamado a los servicios web.

El diseño de las pantallas se cambió en color de etiquetas, el estilo de los objetos de textos y botones, a nivel del detalle los textos para el ingreso de registros ahora son dinámicos. La funcionalidad principal de la web es realizar peticiones *post*, *get*, y *delete* mostrando dichos resultados en pantalla.

### 2.8.8 DESARROLLO DEL BACK-END

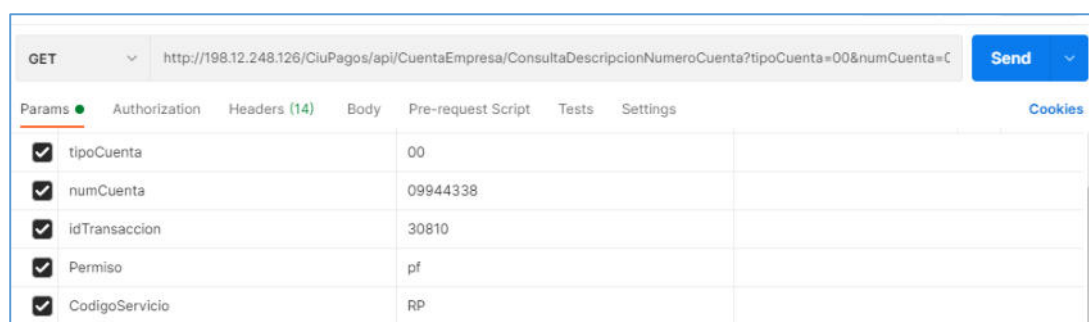
Las clases con sus métodos fueron migradas a controladores con sus capacidades pero esto dentro de varios servicios web, como política de la entidad bancaria estableció que 1 servicio web puede tener hasta 10 capacidades relacionadas a una o varias transacciones.



**Figura 2.12.** Controladores y capacidades dentro de un proyecto  
Fuente: El autor

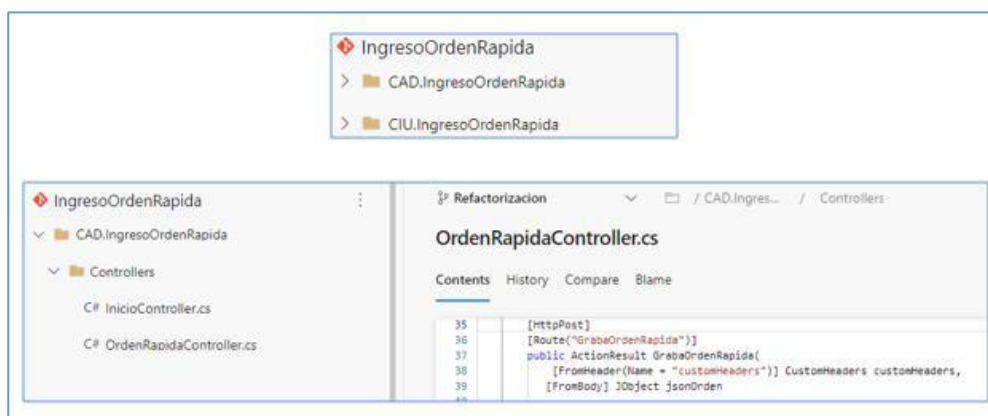
Las capacidades fueron desarrolladas en C# creando las capas CIU (Capa de interface con el usuario) y CRN (capa de reglas de negocio) o CAD (Capa de acceso a base de datos) según la transacción.

- a) Cada developer realizó prueba locales con el programa Postman, los mismos que debían devolver información.



**Figura 2.13.** Pruebas de petición mediante herramienta Postman  
Fuente: El autor

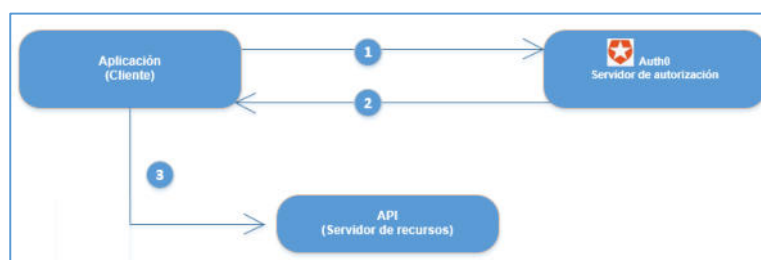
Aplicando arquitectura de microServicios, como ya se definió anteriormente son proyectos independientes, por lo que ahora es servicio web “Ingreso Orden Rapida”, controlador “OrdenRapidaController” y capacidad “GrabaOrdenRapida”:



**Figura 2.14.** Estructura de un servicio del SEOSCP 5.0  
Fuente: El autor

## 2.8.9 SEGURIDAD EN SERVICIOS

Se utiliza en el desarrollo de las API el JSON Web Token (JWT) que define una forma compacta y autónoma de transmitir información de forma segura entre las partes como un objeto JSON. Esta información se puede verificar y confiar porque está firmada digitalmente. [13]



**Figura 2.15.** Diagrama para acceder a una API  
Fuente: El autor

Los servicios fueron entregados a la entidad bancaria y ellos a su vez aplicaron sus propias seguridades para evitar vulnerabilidades en páginas o servicios.

## **2.9. DOCUMENTACIÓN TÉCNICA**

Realizar la documentación técnica fue un requisito primordial para la entrega de los servicios a la entidad bancaria; por lo que se hicieron despliegues de los servicios en un servidor asignado como desarrollo, y también se utilizó la herramienta Postman apuntando a dicho servidor, para documentar todos los parámetros de entrada y salida de cada servicio.

## **2.10. PRUEBAS DE FUNCIONAMIENTO**

El área de QA elaboró los casos de usos y su respectiva matriz de pruebas.

Se establecieron fechas ajustadas de pruebas. Cuando surgía alguna novedad, eran notificadas al developer, teniendo éste un tiempo de 4 a 8 horas para su resolución, según la dimensión y urgencia de la novedad.

Una vez culminadas las pruebas de QA de la empresa proveedora, se gestionó con la entidad bancaria el pase a pre-producción para las pruebas respectivas de su área de QA.

## **CAPÍTULO 3**

### **ANÁLISIS DE RESULTADOS**

#### **3.1 MEJORA EN LA INTERFAZ DEL USUARIO**

En la opción de “Orden rápida”, se debe seleccionar el servicio, cuenta, fecha de inicio de pago, fecha de vencimiento, referencia, la forma de pago, el total es la suma del valor de la orden, a nivel de detalle según el servicio seleccionado, debe ingresar el tipo de identificación, identificación, nombre, código, concepto y valor, una vez ingresado en pantalla todos los datos, se debe dar clic en “Procesar”, esto llama a los servicios de validar información, grabar cabecera de orden, grabar detalle de orden y totales procesados, finalmente muestra mensaje de *“La orden # fue generada con éxito”*.

ORDEN RÁPIDA    INGRESO DE ARCHIVOS

Servicio: OP-ORDEN DE PAGO    Cuenta: 10-33346852-    Inicio de pago: 1/9/2021    Fecha de vencimiento: 1/10/2021

Referencia: REF 2    Forma de pago: EFECTIVO    Total: \$ 2,236.33

[Procesar](#)    [Limpiar](#)

**Detalle de la orden**

Tipo	Identificación	Nombre	Código	Concepto	Valor
Cédula	0930867239	Jonathan Villa	00001	Orden pago 01	\$ 112.12
Cédula	3050564032	María Perez	00002	Orden pago 02	\$ 2,124.21

La orden 56983 fue generada con éxito. [Cerrar](#)

**Figura 3.1.** Pantalla de SEOSCP 5.0, ingreso de “Orden Rápida”, módulo de Pagos.

Fuente: El autor

En la opción de “Ingreso de Archivos”, se debe seleccionar el servicio, cuenta, referencia, fecha de inicio de pago, fecha de vencimiento, y permite buscar y elegir el archivo plano, una vez ingresado en pantalla todos los datos se da clic en “Procesar”, para invocar a un servicio que hace el traspaso del archivo a un servidor de archivos y luego a los servicios que se encarga de validar información, grabar cabecera de orden, grabar detalle de orden y totales procesados. Si la carga fue un éxito, se podrá visualizar un recuadro de resumen; en el cuadro de resultados se mostrará el número de registros ordenados y montos totalizados.

El usuario al verificar que no se encuentra novedad alguna debe dar clic en “Cargar”, aquí se genera la cabecera, detalle y totalizados de la orden, así

mismo mostrando en pantalla “La orden # fue generada con éxito” y mostrando un resumen del envío de información.

Forma Pago	# Reg. Ordenados	Monto Ordenado	# Reg. Errados	Monto Errado
CUENTA AHORRO	3	\$ 406.13	0	\$ 0.00
CUENTA CORRIENTE	4	\$ 425.88	0	\$ 0.00
TOTAL	7	\$ 832.01	0	\$ 0.00

**Figura 3.2.** Figura 4.2: Pantalla de SEOSCP 5.0, “Ingreso de Archivos”, módulo de Pagos.

Fuente: El autor

### 3.2 MEJORA EN TIEMPO DE RESPUESTA

Si bien es cierto, la parte de la interfaz de usuario es amigable, como resultado para el presente proyecto se ha considerado la terminología de “Medida derivada”, tomando un archivo plano (.txt) con información de 7,030 registros (filas), con un peso de 3.05 MB (3,198.976 bytes), la carga se realizó 4 veces en tiempos diferentes y se obtiene el promedio del tiempo en minutos, desde la opción de “Ingreso archivo” en el módulo de Pagos del **SEOSCP 4.0** en comparación con el ambiente de pruebas se realiza la carga con el mismo archivo en la opción “INGRESO DE ARCHIVOS” en el módulo de Pagos del **SEOSCP 5.0** dando los siguientes resultados:



Promedio = total del tiempo (minutos) / número de pruebas.

**Peso Archivo:** 3.05 MB (3,198.976 bytes)

**# Registros** 7,030

	SEOSCP 4.0	SEOSCP 5.0
Archivo	Minutos	Minutos
Prueba_1	2.509	2.324
Prueba_2	2.508	2.321
Prueba_3	2.506	2.322
Prueba_4	2.507	2.32
<b>Promedio</b>	<b>2.508</b>	<b>2.322</b>

**Tabla 4.** Tiempo de respuesta por versión.

Fuente: El autor

En promedio la diferencia que se da en tiempo es la siguiente:

Minutos	0.186
Segundos	11.16

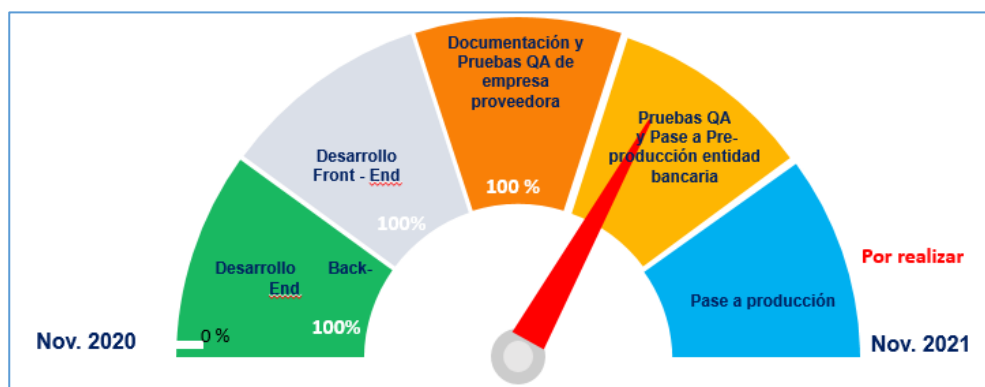
**Tabla 5.** Tiempo de respuesta minutos vs segundos

Fuente: El autor

Como se puede evidenciar en el SEOSCP 5.0 en el ambiente de pruebas se obtiene un tiempo menor que en el SEOSCP 4.0 de **11.16 segundos** en promedio, mostrando de esta manera la mejora por la implementación de la nueva arquitectura de microServicios.

### 3.3 INDICADOR DE AVANCES

El desarrollo del back y front-end se encuentra desarrollados en un 100%, así también la documentación, sin embargo aún se encuentra en pruebas QA.



**Figura 3.3.** Indicador de avances  
Fuente: El autor

# CONCLUSIONES Y RECOMENDACIONES

## CONCLUSIONES

Formar parte del equipo de migración de un sistema para una entidad bancaria, teniendo como objetivo personal el aprendizaje de una nueva tecnología, fue un reto. De igual manera, se logró tener buenos resultados, sobre todo por el empeño y profesionalismo de cada miembro del equipo.

Como conclusiones generales se puede mencionar las siguientes:

1. SEOSCP 5.0 tiende a una integración inmediata de los canales electrónicos para la implantación de nuevos servicios.
2. Con la implementación se atraerá más clientes que conozcan el sistema por referencias.
3. Se le da un valor agregado a los clientes actuales, mediante un sistema que reporta menos incidencias generando más confianza.
4. Se optimiza recursos del área técnica gracias a la escalabilidad del sistema.
5. Disminuirá el riesgo de pérdida de código en producción relacionado a algún cambio por mantenimiento o incidencia reportada.
6. Con la Arquitectura de microServicios se podrá implantar toda la plataforma en la nube.
7. Los recursos técnicos utilizados en la implementación del SEOSCP 5.0 son de última tecnología lo que hace que el sistema sea flexible, escalable y sostenible en el tiempo.

## **RECOMENDACIONES**

1. Con este esquema innovador de accesos a los servicios se pueden hacer uso para que sean consumidos desde los dispositivos móviles.
2. Una vez que las pruebas por parte de la entidad bancaria se encuentren validadas, se recomienda continuar con el despliegue del pase en el ambiente producción.
3. Se recomienda a la entidad bancaria en un futuro cercano la implementación del SEOSCP 6.0 In Cloud.

## BIBLIOGRAFÍA

- [1] E. Acosta Gonzaga, J. A. Álvarez Cedillo y A. Gordillo Mejía, «Arquitecturas en n-Capas: Un Sistema Adaptivo,» pp. 34-37, 2006.
- [2] DocsMicrosoft, «Monoliths to microservices using domain-driven design,» 4 Noviembre 2019. [En línea]. Available: <https://docs.microsoft.com/es-es/azure/architecture/microservices/migrate-monolith>.
- [3] N. Richards, «This Is The End: .NET Framework 2.0 & Visual Studio 2005,» 29 03 2016. [En línea]. Available: <https://www.entranceconsulting.com/end-net-framework-2-0-visual-studio-2005/>.
- [4] O. Blancarte, «reactiveprogramming,» 02 marzo 2020. [En línea]. Available: <https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/monolitico>.
- [5] S. Decide Soluciones, «Arquitectura de microservicios: qué es, ventajas y desventajas,» 03 09 2019. [En línea]. Available: <https://decidesoluciones.es/arquitectura-de-microservicios/>.
- [6] R. B. C. F. José Arteaga, «Propuesta arquitectónica para la migración de un sistema monolítico a un sistema distribuido utilizando domain driven design y microservicios: caso de estudio,» *Universidad Don Bosco*, pp. 1-10, 2011.
- [7] I. d. Souza, «API rest: conoce la importancia de ese recurso para el desempeño de una página web,» 17 marzo 2020. [En línea]. Available: <https://rockcontent.com/es/blog/api-rest/>.

- [8] B. API\_Market, «API REST: qué es y cuáles son sus ventajas en el desarrollo de proyectos,» 03 2016. [En línea]. Available: <https://www.bbvaapimarket.com/es/mundo-api/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos/>.
- [9] Xeridia, «¿Sabes realmente qué es DevOps?,» 17 01 2017. [En línea]. Available: <https://www.xeridia.com/blog/sabes-realmente-que-es-devops>.
- [10] INCIBE, «Qué es una DMZ y cómo te puede ayudar a proteger tu empresa,» 19 09 2019. [En línea]. Available: <https://www.incibe.es/protege-tu-empresa/blog/dmz-y-te-puede-ayudar-proteger-tu-empresa>.
- [11] S. E. Inc, «Empoderar al mundo para desarrollar tecnología a través del conocimiento colectivo,» 2021. [En línea]. Available: <https://stackoverflow.com/company>.
- [12] S. Overflow, «Resultados de la encuesta para desarrolladores,» 2019. [En línea]. Available: <https://insights.stackoverflow.com/survey/2019#technology>.
- [13] JWT, «Introduction to JSON Web Tokens,» 2021. [En línea]. Available: <https://jwt.io/introduction>.