# Arquitectura de software para el desarrollo de aplicaciones web orientada a micro-servicios en TecNM campus Escárcega

## Software architecture for the development of web applications oriented to microservices at TecNM campus Escárcega

Damián Uriel Rosado Castellanos, Ivette Stephany Pacheco Farfán, Iván Humberto Fuentes Chab, Julio Cesar Cantun Páez

Tecnológico Nacional de México. Instituto Tecnológico Superior de Escárcega
damianrc@itsescarcega.edu.mx

**PALABRAS CLAVE:**

Micro-servicios, Aplicaciones Web, Desarrollo de Software, Angular, Spring Boot, Arquitectura de Software

**RESUMEN**

En TecNM Campus Escárcega, los docentes y alumnos pertenecientes a la carrera de Ingeniería en Sistemas Computacionales, utilizan el paradigma de desarrollo monolítico. Sin embargo, las necesidades actuales apuntan a nuevos enfoques de desarrollo como son los micro-servicios. Este trabajo presenta una arquitectura de software para el desarrollo de aplicaciones web orientada a micro-servicios con la finalidad de tener una herramienta que mantenga o aumente los niveles de productividad y disminuya los tiempos de desarrollo a partir de los niveles de usabilidad de los docentes y alumnos de este campus, como cuando crean aplicaciones monolíticas. Con la finalidad de lograr lo antes mencionado, se diseñó y creó una aplicación web en Angular para el consumo de micro-servicios en Spring Boot. La integración de las dos partes permite tener una arquitectura reutilizable para la creación de software orientado a micro-servicios al momento de iniciar un nuevo proyecto institucional. El cálculo de los niveles de satisfacción de la arquitectura propuesta por parte de los docentes y alumnos evaluados, consistió en medir dos métricas con base en la literatura: productividad y usabilidad. Los resultados demuestran que los alumnos y docentes de la institución, que al menos tienen un proyecto relacionado a una aplicación web, están conformes con el uso de la arquitectura propuesta al momento de crear un nuevo proyecto, ya que muestra un nivel superior de 4.36 puntos de 5 en la escala Likert, en contraste con el desarrollo de una aplicación monolítica que indica un nivel de conformidad inferior con un valor de 4.24 puntos de 5 en la escala Likert. Es decir, que el uso de la arquitectura propuesta reduce los tiempos de desarrollo a partir de los niveles de usabilidad y aumenta los niveles de productividad en comparación con el paradigma monolítico. Con esto, se permite que los docentes y alumnos de TecNM campus Escárcega se encuentren vigentes en conocimientos tecnológicos y puedan migrar al uso de nuevas tecnologías y paradigmas de desarrollo con base en las nuevas tendencias tecnológicas, como es el caso de los micro-servicios.

**KEYWORDS:**

Micro-services, Web Applications, Software Development, Angular, Spring Boot, Software Architecture.

**ABSTRACT**

At TecNM Campus Escárcega the students and professors that are part of Computer Systems engineering, works with the monolithic development paradigm. However, current needs aim to new development approaches like microservices. This work presents a software architecture to develop web applications focused on microservices in order to have a tool that mantains or increases productivity levels and reduces the development time based on the usability levels of the students and professors on this campus, as if they create monolithic applications. In order to achieve the mentioned above, a web application was designed and coded using Angular and microservices through Spring Boot. The integration of two elements allows to have a reusable architecture for the creation of a microservice-oriented software when starting a new institutional project. The calculation of the satisfaction levels for the architecture proposed by the teachers and students evaluated, consisted in measuring two metrics based on the literature: productivity and usability. The results demonstrate that students and professors at TecNM Campus Escárcega, with at least a project related to web applications development, are satisfied with the usage of the proposed architecture when creating a new project, as it shows a high level of 4.36 points of 5 on Likert scale than the monolithic applications development which has a lower level with 4.24 points of 5 on Likert scale. As mentioned, it means that the usage of the proposed architecture reduces the development time based on the usability level and increases the level of the productivity compared to monolithic paradigm, this allows the students and professors at TecNM Campus Escárcega to have current technology knowledge and migrate to new technologies and development paradigms based on new technology trends, in this case microservices.

**• Recibido: 13 de agosto 2021 • Aceptado: 20 de abril 2022 • Publicado en línea: 23 de junio 2023**

*ProgMat*, 2023, 15, 2; https://progmat.uaem.mx/progmat

DOI: 10.30973/progmat/2023.15.2/2

## 1. INTRODUCCIÓN

El desarrollo de aplicaciones web orientada a micro-servicios se ha vuelto un estándar en el desarrollo de software, esto se debe a las diferentes ventajas que ofrece como la interoperatividad entre diferentes productos de software, el desarrollo modular de la lógica de negocios del proyecto y la escalabilidad de los proyectos de software.

Para este tipo de desarrollo, requerimos un flujo de comunicación integrado por dos partes, como se aprecia en la figura 1, el patrón de arquitectura de micro-servicio establece construir una aplicación como un conjunto de servicios, donde cada uno de estos es independiente del otro y hasta pueden ser escritos en lenguajes diferentes y mantenidos por equipos diferentes [1]. Y la aplicación web que es un tipo especial de aplicación cliente/servidor, donde tanto el cliente (el navegador, explorador o visualizador) como el servidor (el servidor web) y el protocolo de comunicación (HTTP) están estandarizados y no han de ser creados por el programador [2].

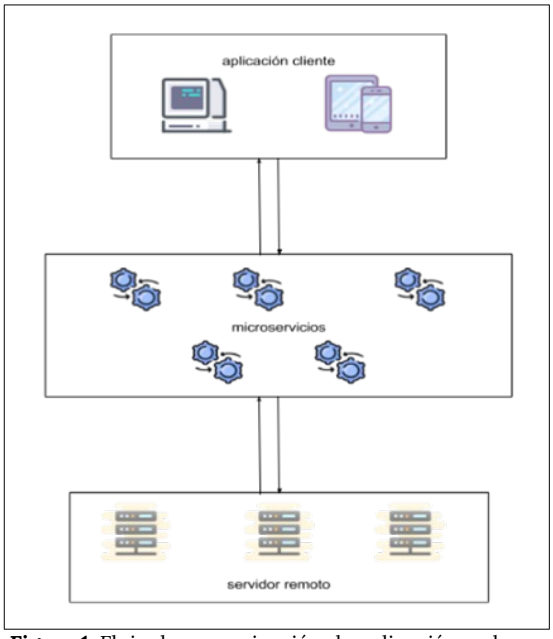


**Figura 1.** Flujo de comunicación de aplicación web para el consumo de micro-servicios. Fuente propia.

Para el desarrollo del patrón de la arquitectura del micro-servicio, existen diferentes herramientas tecnológicas que permiten su creación (ver tabla 1). Para determinar qué herramienta es más recomendable, debemos considerar principalmente los requerimientos funcionales y no funcionales basados en los patrones de diseño y desempeño, latencia y consumo de recursos [3]. Sin embargo, uno de los más destacados es Spring Boot, que tiene un nivel de madurez de años comparado con otras tecnologías, es más robusto en cuestiones de seguridad y permite la integración con base de datos transaccionales [4].

**Tabla 1.** Herramientas tecnológicas de uso común para el desarrollo de micro-servicios.

| Nombre | Descripción |
|---|---|
| Spring Boot | Es una de las herramientas más populares escrito en el lenguaje de Programación Java. |
| Vert.X | Herramienta creada por Eclipse Foundation. Soporta diversos lenguajes como Java, Javascript y Kotlin. |
| Helidon | Herramienta desarrollada por Oracle compuesta de una colección de librerías escritas en Java. |
| Molecular | Es una moderna herramienta basada en JavaScript que puede ser integrada con NodeJS. |
| Quarkus | Herramienta desarrollada por Red Hat basada en el lenguaje de programación Java. |
| Flask | Herramienta minimalista que busca agilizar el proceso de desarrollo con el menor número de líneas de código y se basa en el lenguaje de programación Python. |
| NodeJS | Herramienta que corre en un entorno de ejecución para JavaScript y permite el desarrollo ágil de micro-servicios. |
| Sinatra | Herramienta para construir micro-servicios en el lenguaje de programación Ruby. |
| Laravel | Herramienta de código abierto para el desarrollo de aplicaciones y servicios basados en el lenguaje de programación PHP. |
| ASP.NET | Herramienta creada por Microsoft que permite la construcción de micro-servicios basada en C#. |

Para el desarrollo de la aplicación web contamos con diferentes herramientas tecnológicas, pero las más populares en el

mercado son tres (tabla 2). Para determinar qué herramienta es más recomendable, debemos considerar principalmente el tamaño de la organización y el marco de desarrollo [5]. Sin embargo, Angular es una de las herramientas que tiene una curva de aprendizaje mayor y un rendimiento inferior comparado con otros, pero lo compensa mediante la usabilidad de la aplicación web y los marcos de construcción para aplicaciones a gran escala [6].

**Tabla 2.** Herramientas tecnológicas de uso común para el desarrollo de aplicaciones web.

| Nombre | Descripción |
|---|---|
| Angular | Herramienta para el desarrollo de aplicaciones web basado en TypeScript, es de código abierto y es mantenido por la empresa Google. |
| React | Herramienta de código abierto basada en JavaScript que facilita el desarrollo de aplicaciones y es mantenido por la empresa Facebook. |
| Vue.js | Herramienta de código abierto basada en JavaScript que fue creado por Evan You, el cual es el responsable del mantenimiento junto con su equipo de miembros activos de la empresa Netlify y Netguru. |

En TecNM campus Escárcega, los alumnos y los docentes pertenecientes a la carrera de Ingeniería en Sistemas Computacionales (ISC) tienen un enfoque de desarrollo monolítico, como se aprecia en la figura 2, es decir, el desarrollo de los sistemas de información tiene un enfoque de Modelo-Vista-Controlador (MVC), pero las necesidades actuales de desarrollo de software institucional y de sectores públicos y privados apuntan a la implementación de nuevos paradigmas de desarrollo, precisando los micro-servicios. Por tanto, el desarrollo de aplicaciones monolíticas apunta a largo plazo a ser obsoleto, en comparación a las nuevas tendencias tecnológicas que buscan el desarrollo de aplicaciones más interactivas por parte del usuario.

Uno de los principales motivos por el que los docentes y alumnos de esta institución (TecNM Campus Escárcega) siguen utilizando el paradigma de desarrollo monolítico, son los módulos preestablecidos que se crean al momento de iniciar un nuevo proyecto. Sin embargo, dado las nuevas necesidades (aplicaciones con mayor grado de interacción para el usuario) y tendencias de desarrollo tecnológico, esto conlleva a que los docentes y alumnos a largo plazo tengan conocimientos obsoletos y no cuenten con el dominio de nuevos paradigmas de desarrollo, como es el caso del desarrollo de aplicaciones web con mayor facilidad de interacción para el usuario final basado en el consumo de micro-servicios.

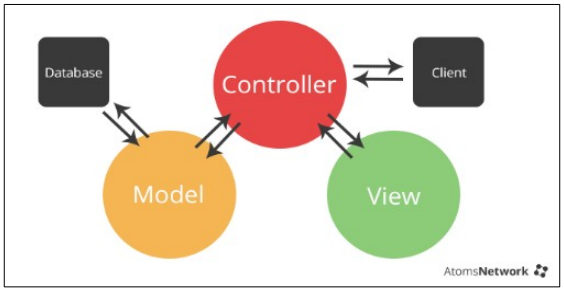


**Figura 2.** Flujo de comunicación de desarrollo monolítico. Fuente AtomNetwork.

Esta resistencia al cambio se debe a que, bajo el enfoque de micro-servicios, no hay un catálogo de repositorios preestablecidos para la gestión de usuarios (autenticación, roles, permisos de usuario, etc), seguridad y otros módulos de las diferentes tecnologías de desarrollo existentes en comparación con las tecnologías de desarrollo de aplicaciones monolíticas. Lo anterior se traduce en un aumento en los tiempos de desarrollo a partir de los niveles de usabilidad y disminución en los niveles de productividad, al tener que desarrollar los módulos desde cero al iniciar un nuevo proyecto con el enfoque de micro-servicios. Es decir, los alumnos y docentes de TecNM Campus Escárcega carecen de una arquitectura de software orientada al desarrollo de aplicaciones web bajo el paradigma de micro-servicios que cuente con diferentes módulos preestablecidos al iniciar un nuevo proyecto, por lo que prefieren utilizar un paradigma de desarrollo monolítico.

En este trabajo, presentamos una arquitectura de software reutilizable que

permita la integración de diferentes módulos relacionados a la gestión de usuarios, roles, inicio de sesión y otros elementos, que reduzca el tiempo de desarrollo, aumente la productividad y permita el uso de nuevas tecnologías con base en los paradigmas de desarrollo actual para los alumnos y docentes de TecNM campus Escárcega que tengan proyectos relacionados al desarrollo de software. La finalidad es que los alumnos y docentes de la institución puedan desarrollar aplicaciones web bajo el enfoque de micro-servicios sin afectar la media aritmética del tiempo de desarrollo a partir de los niveles de usabilidad y el nivel de productividad de la creación de aplicaciones web bajo el enfoque monolítico, paradigma actual utilizado en los proyectos de la institución. Para ello, en la sección de metodología, explicamos el diseño y la codificación de la arquitectura propuesta con base en los paquetes desarrollados; de igual forma, detallamos el proceso para definir la herramienta de medición de la arquitectura propuesta con base en las métricas de productividad y usabilidad. En la sección de resultados, detallamos los valores obtenidos al aplicar la herramienta de medición a los docentes y alumnos pertenecientes a la carrera de ISC de TecNM campus Escárcega, que usaron la arquitectura propuesta, y describimos los resultados obtenidos basados en los niveles de conformidad de los involucrados. Posteriormente, comparamos el nivel de conformidad de desarrollo monolítico con el nivel de conformidad de la arquitectura de software propuesta, en donde clarificamos la reducción en los tiempos de desarrollo a partir del nivel de usabilidad y el aumento en la productividad utilizando la arquitectura de software propuesta, por parte de los docentes y alumnos de TecNM campus Escárcega, que al menos tiene un proyecto de desarrollo de software orientado a aplicaciones web. En la sección de conclusiones explicamos los beneficios e importancia del uso de la arquitectura de software propuesta en la institución, específicamente en los docentes y alumnos de la carrera de ISC, así como las áreas de oportunidad y trabajo futuro del proyecto.

## 2. METODOLOGÍA

Para el desarrollo del proyecto, como se muestra en la figura 3, primero realizamos un levantamiento y análisis de requerimientos para definir los módulos de desarrollo de la arquitectura de software propuesta. El segundo paso fue el diseño y codificación de micro-servicios, posteriormente como tercer paso, realizamos el diseño y codificación de la aplicación cliente. Una vez finalizada la codificación de ambas partes, continuamos con el cuarto paso que fue la integración de los micro-servicios con la aplicación cliente. El quinto paso fue la definición de métricas de evaluación de software. Finalmente el sexto paso aplica el instrumento de evaluación propuesto para conocer el nivel de conformidad de los usuarios a partir de los tiempos de desarrollo con base en los niveles de usabilidad y niveles de productividad bajo el paradigma de desarrollo monolítico y de la arquitectura propuesta de micro-servicios, por parte de los alumnos y docentes de la carrera de ISC en TecNM campus Escárcega, para determinar si la arquitectura propuesta ofrece los mismos niveles de usabilidad y productividad o los mejora en comparación con el paradigma de desarrollo utilizado en la institución (paradigma monolítico).

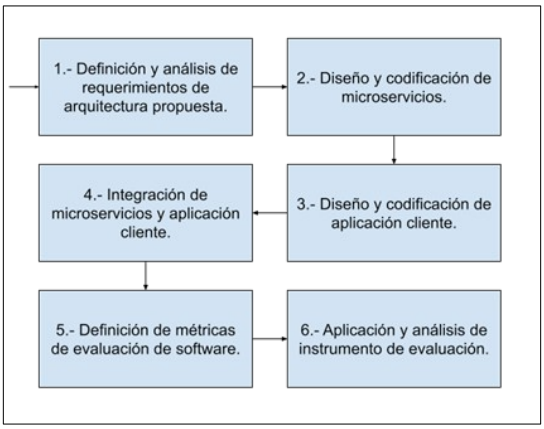


**Figura 3.** Metodología de desarrollo para la arquitectura de software propuesta.

## 3. DEFINICIÓN Y ANÁLISIS DE REQUERIMIENTOS DE ARQUITECTURA PROPUESTA

En esta etapa definimos las tecnologías a utilizar con base en la literatura, así como los diferentes módulos de desarrollo de los micro-servicios y la aplicación cliente. Para el desarrollo de micro-servicios utilizamos Spring Boot en su versión 4.3.18 con el lenguaje de programación Java en su versión 8 y la base de datos relacional PostgreSQL en su versión 9.6. Para el desarrollo de la aplicación cliente utilizamos Angular en su versión 9.1. Con base en el análisis de requerimientos, definimos cinco módulos de desarrollo, como se aprecia en la tabla 3, que componen la arquitectura de software propuesta. Para la definición de estos cinco módulos, aplicamos una encuesta con preguntas abiertas a los alumnos y docentes de TecNM campus Escárcega para conocer los diferentes módulos preestablecidos cuando crean un nuevo proyecto orientado al paradigma monolítico. De la encuesta se obtuvo que los módulos sugeridos son: seguridad, gestión de usuarios, autenticación y registro de usuarios, comunicación, y para finalizar, temas y notificaciones.

**Tabla 3.** Módulos de desarrollo propuestos en la arquitectura de software.

| Nombre del módulo | Descripción |
|---|---|
| Módulo de seguridad | Contiene todas las configuraciones necesarias para proporcionarle seguridad a los micro-servicios, la aplicación cliente y el canal de comunicación de peticiones y respuestas. |
| Módulo de gestión de usuarios | Permite la gestión y administración de usuarios; así como la asignación de roles, privilegios y otras modificaciones del usuario. |
| Módulo de autenticación y registro de usuarios | Contiene los métodos de autenticación preestablecidos en la arquitectura de software propuesta y la creación de nuevos usuarios. |
| Módulo de comunicación | Contiene un chat interno que permite la comunicación general y privada entre los usuarios. |
| Módulo de tema y notificaciones | Gestiona la paleta de colores de la aplicación y las notificaciones con base en las diferentes acciones del usuario. |

## 4. DISEÑO Y CODIFICACIÓN DE MICRO-SERVICIOS

Los micro-servicios fueron creados con base en el estilo de arquitectura *RESTful*, el cual consiste en la infraestructura Representational State Transfer (REST), con la diferencia de que el Localizador Uniforme de Recursos (URL), funciona como un identificador del recurso de los métodos de acceso estándar (GET, POST, PUT y DELETE), bajo el estándar de operaciones *HTTP* [7].

La creación de los paquetes del proyecto, siguen un patrón de diseño basado en capas, como se muestra en la figura 4, que consiste en dividir la aplicación en diferentes capas, con la intención de que cada capa tenga un rol definido [8]. Las entidades, son los modelos que tienen una relación directa con la base de datos; los repositorios, son una clase abstracta que gestiona el acceso a los datos; los servicios, son la capa de la lógica de negocios; y la capa de control, permite manejo de peticiones y respuestas.

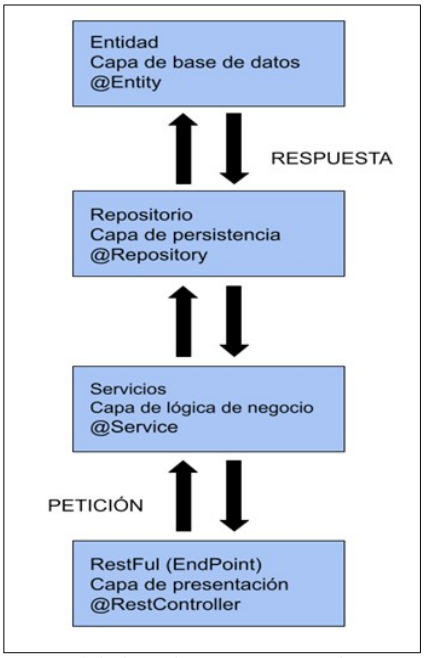


**Figura 4.** Modelo basado en capas para la creación de micro-servicios. Fuente propia.

El manejo de dependencias del proyecto está basado en Maven, el cual es un plugin

que soporta Apache Maven y permite ejecutar archivos JAR o WAR en aplicaciones desarrollas con Spring Boot [9]. Las dependencias utilizadas en este proyecto, como se aprecia en la tabla 4, permiten el funcionamiento adecuado de los micro-servicios ya que están relacionados al control de origen de datos, seguridad, protocolos de comunicación, servidores embebidos y otros.

**Tabla 4.** Dependencias del proyecto utilizadas para crear los micro-servicios.

| Dependencia | Descripción |
|---|---|
| gson | Permite la conversión de cadenas de texto a objetos en formato JSON. |
| spring-boot-starter-web | Iniciador para la construcción de aplicaciones web incluidas RESTful. Utiliza el contenedor Tomcat. |
| spring-boot-starter-security | Iniciador de seguridad para aplicaciones desarrolladas con Spring. |
| guava | Conjunto de bibliotecas centrales y amplias que incluyen clases de utilidad, de E/S y otras más. |
| spring-boot-starter-data-jpa | Iniciador que permite utilizar Spring Data JPA con Hibernet para la manipulación de datos relacionales. |
| postgresql | Controlador JDBC de PostgreSQL. |
| jjwt-api<br>jjwt-impl<br>jjwt-jackson | Plugins para la creación y control de tokens que permiten la propagación de identidad y privilegios en las aplicaciones. |
| spring-boot-starter-websocket | Iniciador para construcción de aplicaciones que trabajen con conexión bidireccional. |

A partir de las dependencias mencionadas, codificamos los paquetes de desarrollo de los micro-servicios, como se aprecia en la figura 5, con el lenguaje de programación de alto nivel Java versión 8. El *paquete de* "***autenticación***" pertenece al módulo de autenticación y registro de usuario, este contiene las 4 capas de los micro-servicios en las diferentes clases codificadas para definir sus modelos, sus métodos de acceso, sus servicios en la lógica de negocios y sus end-points, y su objetivo es llevar el control de las sesiones, el registro y la gestión de usuarios. El paquete "***jwt***", la clase "***ApplicationSecurityConfig***" y la clase "***WebSocketConfig***", son archivos de configuración y no recaen en alguna capa de los micro-servicios. Estos pertenecen al módulo de seguridad y su principal tarea es la creación de tokens y su manipulación para la comunicación de la aplicación cliente con el consumo de los micro-servicios relacionados al inicio de sesión, intento de autenticación, autenticaciones exitosas, entre otras tareas relacionadas a la seguridad y evasión de robo de información.

El paquete "***roles_permisos***", pertenece al módulo de gestión de usuarios, las clases definidas contienen los 4 niveles de capa de micro-servicios y se relacionan con modelos, métodos de acceso, servicios en la lógica de negocios y endpoints. Su principal tarea es la asignación de privilegios, roles y otras operaciones relacionados a los usuarios, como edición y eliminación de un usuario, mediante los endpoints de los micro-servicios. El último paquete "***Websocket***", tiene relación con los módulos: comunicación, temas y notificaciones. Este paquete permite tener comunicación e interacción en tiempo real por parte de la aplicación cliente al momento de consumir los micro-servicios. El sub-paquete "***Chat***", se basa en la capa de base de datos (@Entity) y contiene los modelos necesarios a partir de la definición de sus atributos para el almacenamiento histórico de las conversaciones entre usuarios. La clase "***SocketController***", pertenece a la capa de presentación (@RestController) y contiene los endpoints para tener el canal de comunicación mediante el chat integrado a partir de los diferentes métodos de acceso y consumo de los micro-servicios; mientras que, la clase "***SocketService***" pertenece a la capa de lógica de negocios (@Service) y tiene la responsabilidad de la manipulación de los mensajes enviados desde la aplicación web cliente. Finalmente, el último paquete considerado en la arquitectura de software propuesta es "***util***", que contiene todas las clases de utilería para definir de manera global los encabezados de las peticiones de los micro-servicios, cuestiones de encriptación, conversión de tipos de dato y otros elementos utilizados de manera general en todas las clases del proyecto.

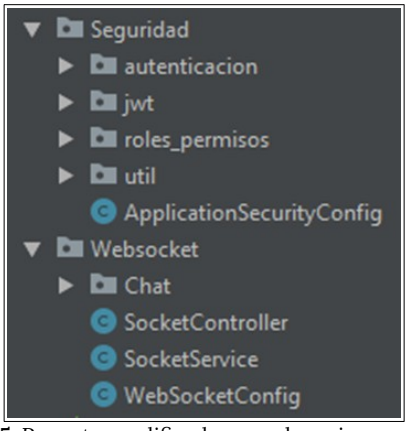


**Figura 5.** Paquetes codificados para los micro-servicios.

## 5. DISEÑO Y CODIFICACIÓN DE APLICACIÓN CLIENTE

El desarrollo de la aplicación cliente fue codificado con Angular Versión 9.1, el cual es una herramienta de diseño y plataforma de desarrollo que permite crear aplicaciones de una sola página [10]. Las dependencias utilizadas en la aplicación son las preestablecidas por Angular al momento de crear el proyecto en conjunto de las dependencias de terceros, como se muestra en la tabla 5, que facilitan el desarrollo de los componentes visuales, animaciones y funcionalidad, así como mejorar la interacción de la aplicación cliente con el usuario final.

**Tabla 5.** Dependencias de desarrollo de la aplicación web.

| Dependencia | Descripción |
|---|---|
| Material Design | Sistema de diseño creado por Google para crear experiencias digitales de alta calidad. |
| JWT-Decode | Librería que permite la decodificación y extracción de información de tokens basados en estándar JSON Web Token (JWT). |
| Boostrap con JQuery y Popper.js | Herramienta multiplataforma de código abierto para diseño e interacción de sitios y aplicaciones web. |
| SweetAlert2 | Herramienta de alertas personalizadas en tus aplicaciones web. |
| SockJS y StompJS | Herramienta que trabaja bajo el Protocolo Simple de Mensajería Orientado a Texto (STOMP) que permite a un navegador transmitir información a un servidor. |

A partir de la instalación de las dependencias, generamos los componentes visuales con su funcionalidad, como se aprecia en la figura 6, las cuales están conformadas por los siguientes paquetes: “chat”, “clases”, “partials”, “providers”, “sesion”. Cada paquete que debe interactuar con el servidor contiene el paquete de servicios, en donde creamos los métodos de accesos necesarios para poder establecer un flujo de comunicación.

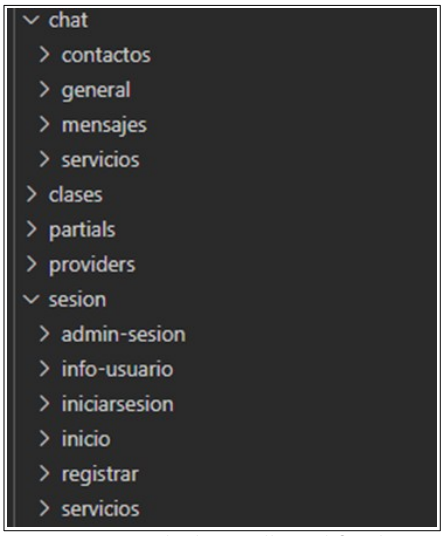


**Figura 6.** Paquetes de desarrollo codificadas para la aplicación web.

El paquete “***chat***”, contiene los componentes visuales que permiten la gestión de los contactos, enviar y recibir mensajes personales con otros usuarios o globales. Este paquete pertenece al módulo de comunicación y se compone de diferentes sub-paquetes; donde “***contactos***”, permite visualizar a los usuarios registrados a través de una lista de elementos gráficos; el sub-paquete “***general***” y “***mensajes***”, contienen todos los elementos gráficos de una ventana de chat para una conversación global o personal. Finalmente, el sub-paquete “***servicios***” pertenece al módulo de seguridad y contiene toda la configuración y endpoints orientados al servidor para tener el canal de comunicación en tiempo real aplicando la tecnología de WebSockets en combinación

con los micro-servicios, es decir establecer un canal de comunicación bidireccional entre la aplicación cliente y el servidor.

El paquete "***clases***", consiste en todas las clases globales que se comparten en el proyecto (por ejemplo: la clase Usuario o la clase Chat) y que pueden ser utilizados como modelos para la representación de datos de las respuestas de las peticiones de los micro-servicios, entre otras tareas. El paquete "***partials***", contiene todos los componentes visuales que pueden ser reutilizados en la aplicación web, algunos elementos son: notificaciones, diálogos, alertas, barras de navegación y otros. El paquete "***providers***", contiene todas las clases con sus atributos y funciones que pueden ser utilizados por otras clases para sobrescribir funciones o agregar nuevas funcionalidades a través de la inyección de dependencias. Los paquetes mencionados no pertenecen directamente a ningún módulo de desarrollo y son de utilería para la reutilización de código.

El paquete "***sesion***", contiene los componentes visuales para la gestión de sesiones y usuarios. Esta se compone de diversos sub-paquetes que están relacionados a elementos gráficos como la ventana principal, información de usuario, página de registro, entre otros elementos. Los sub-paquetes se componen de diferentes gráficos con base en su interacción. El sub-paquete "***admin-sesion***", pertenece al módulo de gestión de usuarios y contiene todos los componentes visuales para gestionar a los usuarios registrados. Los sub-paquetes "***info-usuario***" e "***inicio***", están relacionados a los módulos de temas y notificaciones, y contienen los elementos gráficos para la configuración del tema claro u oscuro, y alertas de notificaciones. Los sub-paquetes "***iniciarsesion***" y "***registrar***", pertenecen al módulo de desarrollo de autenticación y registro de usuarios, este se compone de elementos gráficos que van a permitir al usuario iniciar sesión en la aplicación web o generar un nuevo usuario mediante un formulario de registro. Finalmente, el sub-paquete "***servicios***" pertenece al módulo de seguridad y contiene todos los endpoints de los micro-servicios para el funcionamiento de los componentes visuales comentados previamente y el mecanismo de comunicación es asíncrono mediante un objeto observable.

## 6. INTEGRACIÓN DE MICRO-SERVICIOS Y APLICACIÓN CLIENTE

La integración de los micro-servicios con la aplicación cliente nos permite tener una aplicación funcional basada en la arquitectura de software propuesta, como se muestra en la figura 7, los componentes visuales permiten la interacción con el usuario y su funcionalidad es ejecutada por medio de una interfaz basada en el protocolo HTTP y STOMP (simple text-orientated messaging protocol, por sus siglas en inglés) para la interacción de los sistemas, es decir la comunicación de la aplicación cliente con los micro-servicios.

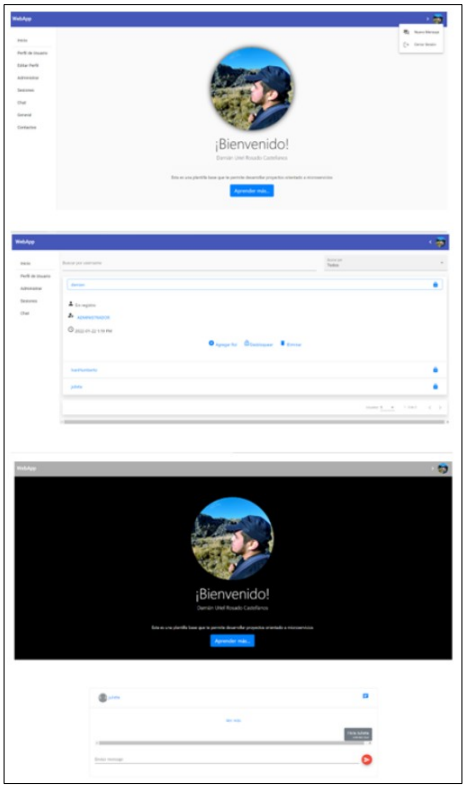


**Figura 7.** Componentes visuales de la arquitectura de software propuesta.

Para establecer este canal de comunicación y decir que ambas partes están integradas, desde la aplicación cliente, utilizamos el servicio de la clase inyectable HttpClient ofrecida por Angular, que contiene los métodos de acceso del protocolo HTTP mediante una interfaz comunicación asíncrona conocida como Observable, Así como configuramos el canal de comunicación bidireccional (mejor conocido como Full-Dúplex) con la librería SockJS y el protocolo STOMP. Desde los micro-servicios configuramos la falsificación de solicitud entre sitios (CSRF, por sus siglas en inglés) para no tener acciones no deseadas por usuarios ajenos al proyecto, definimos los endpoints públicos y los endpoints que requieren algún tipo de autenticación de usuario y sus filtros de validación con base en los tokens antes de procesar una petición, tanto en el protocolo HTTP como en el protocolo STOMP, con la finalidad de que sólo la aplicación web pueda acceder a estos. Posteriormente, recuperamos los endpoint obtenidos al momento de desplegar el proyecto en un archivo WAR, en el contenedor Apache Tomcat en su versión 8.5.6 y estos fueron declarados en la aplicación cliente con sus diferentes métodos de acceso (GET, POST, PUT y DELETE).

## 7. DEFINICIÓN DE MÉTRICAS DE CALIDAD DE SOFTWARE

Una vez realizado el desarrollo e integración de la aplicación cliente con los servicios, el siguiente paso fue la definición de las métricas de calidad de software, como se aprecia en la figura 8, que consiste en medir el grado de desempeño de las principales características por medio de una revisión sistemática de un producto de software [11].

Para ello, definimos dos criterios de calidad en el uso de métricas, a partir de la literatura propuesta [12], con la finalidad de obtener un factor de conformidad cuantitativo de la arquitectura de software para la creación de aplicaciones web orientada a micro-servicios. Los criterios propuestos son niveles de productividad y niveles de usabilidad, medidos en alumnos y docentes de la carrera de ISC en TecNM campus Escárcega, que tengan proyectos de colaboración, académicos o investigación relacionados directamente en el desarrollo de aplicaciones web.

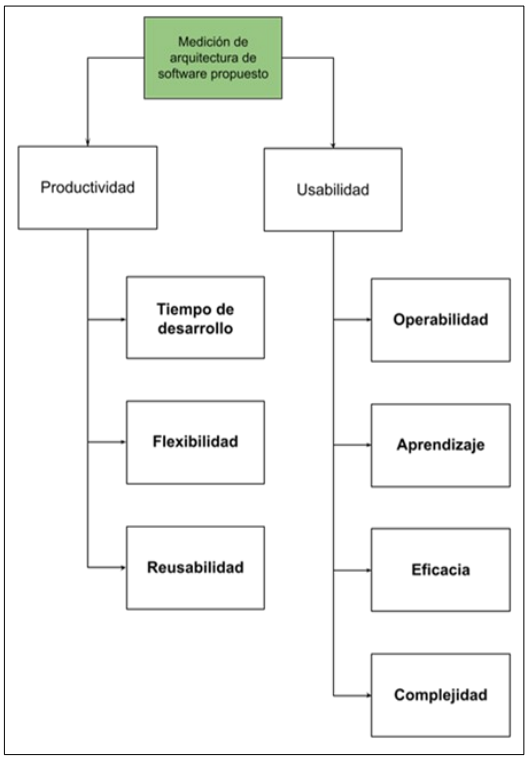


**Figura 8.** Métricas propuestas para medir el factor de conformidad de la arquitectura propuesta.

El nivel de productividad es el rendimiento del proceso de desarrollo y se compone de: tiempo de desarrollo, que es la administración del tiempo para la entrega; la flexibilidad que es la facilidad de hacer modificaciones al proyecto; y la reusabilidad, definida como la facilidad de reutilización de software [13] [14]. El nivel de usabilidad es el grado en el que el software es fácil de usar y se compone por: la operabilidad, que significa la capacidad de manejo y control sobre el software; el aprendizaje, es el nivel de conocimiento requerido para el uso del sistema; la eficacia, es la precisión y la exhaustividad para alcanzar objetivos

específicos; y la complejidad, es el nivel de esfuerzo para comprender el uso del software [15] [16] [17].

Para lograr la medición, utilizamos la metodología propuesta en [18], la cual se encuentra basada en el protocolo ***Personal Opinion Surveys*** [19]. El desarrollo del instrumento de medición, como se visualiza en la figura 9, consiste en una serie de etapas, en donde; la definición de objetivos, establece el nivel de relevancia de las métricas; la etapa de diseño de la encuesta, contiene el conjunto de preguntas para recopilar información con base en las métricas propuestas; la etapa de elaboración de instrumento, relaciona las preguntas con el atributo a partir de un esquema de valoración; la etapa de evaluación del instrumento, mide la validez del instrumento a aplicar; la etapa de documentación del instrumento, categoriza cada pregunta a partir de las medidas de valores y métricas propuestas, y para finalizar la aplicación de la encuesta recolecta los valores de las métricas propuestas.

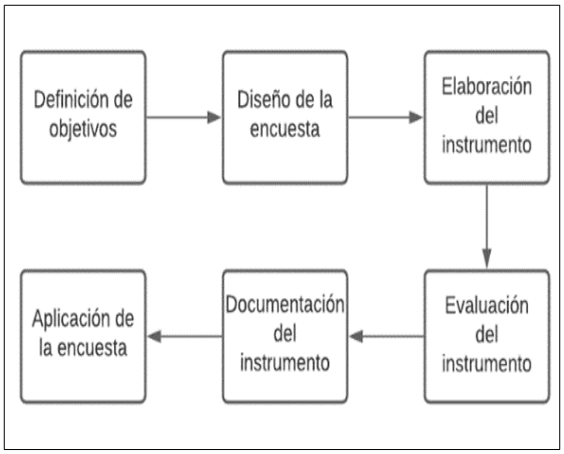


**Figura 9.** Métodología para la definición del instrumento de medición propuesto.

## 6. APLICACIÓN Y ANÁLISIS DE INSTRUMENTO DE EVALUACIÓN

Esta etapa consiste en la aplicación y análisis de resultados del instrumento de evaluación propuesto, como se muestra en la tabla 6, la aplicación de esta herramienta permitirá conocer los niveles de conformidad de los alumnos y docentes de TecNM Campus Escárcega que utilizaron la arquitectura de software propuesta al menos en algún proyecto de desarrollo de software orientado a aplicaciones web.

**Tabla 6.** Herramienta de medición propuesta.

| Pregunta | Medida | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| 1. ¿La relación de las horas de desarrollo con base en la codificación generada es adecuada utilizando la arquitectura del proyecto propuesta? | | | | | |
| 2. ¿La cantidad de modificaciones realizadas a la arquitectura propuesta retrasó los tiempos de desarrollo del proyecto del docente? | | | | | |
| 3. ¿La reutilización de elementos del proyecto permitió reducir el tiempo de desarrollo? | | | | | |
| 4. ¿La arquitectura propuesta permite una fácil implementación de metodologías ágiles de desarrollo del proyecto del docente? | | | | | |
| 5. ¿La arquitectura propuesta es adecuada a las necesidades del proyecto a desarrollar por parte del docente? | | | | | |
| 6. ¿La arquitectura propuesta proporciona un fácil acoplamiento con nuevas funcionalidades del sistema? | | | | | |
| 7. ¿La cantidad de modificaciones necesarias para agregar funcionalidad no prevista en la arquitectura propuesta a partir de las necesidades del proyecto a desarrollar fue baja? | | | | | |
| 8. ¿La arquitectura propuesta cumple con un grado de usabilidad con base en los objetivos específicos propuestos por el alumno/docente? | | | | | |
| 9. ¿El nivel de aprendizaje requerido de las tecnologías para el uso de la arquitectura propuesta es alta? | | | | | |
| 10. ¿El nivel de comprensión e intuición para el uso de la arquitectura propuesta fue sencillo? | | | | | |

La herramienta de medición propuesta se compone del esquema de valoración con base en la escala Likert, que permite conocer el grado de acuerdo/desacuerdo sobre una afirmación, ítem o reactivo [20]. Los niveles de escala utilizados en este proyecto son los siguientes: 1 = Totalmente en desacuerdo, 2 = En desacuerdo, 3 = Indiferente, 4 = De

acuerdo y 5 = Completamente de acuerdo. Con base en el esquema de valoración, podemos obtener conclusiones del nivel de aceptación y qué tan conformes están en utilizar la arquitectura propuesta con base en su experiencia orientado a los tiempos de desarrollo con base en los niveles de usabilidad y niveles de productividad en el uso de nuevas tecnologías, ya que el enfoque actual de desarrollo es el monolítico.

## 7. RESULTADOS

El código de la arquitectura de software propuesta está almacenado en el repositorio de *github*, el cual es un sistema descentralizado de control de versiones de proyectos [21]. Para la implementación del proyecto, seleccionamos a 25 alumnos y 5 docentes de ISC que al menos tienen proyecto de desarrollo de software definido en clases, eventos académicos o colaboración en algún proyecto de investigación de una población total de 121 alumnos y 14 docentes de ISC. De igual forma, realizamos una capacitación en cuestión de tecnologías de desarrollo e implementación de la arquitectura de software propuesta. Para la implementación, asignamos el rol de colaborador a los sujetos bajo estudio con la finalidad de acceder al repositorio https://github.com/DamianUriel para la clonación de los recursos del proyecto y su implementación de manera local.

El instrumento de medición propuesto (regresar a tabla 6) fue aplicado a los sujetos bajo estudio evaluando primeramente el paradigma de desarrollo monolítico utilizado en la institución. Posteriormente, el mismo instrumento fue aplicado para evaluar la arquitectura de software propuesta basada en micro-servicios. Los resultados obtenidos con base en las métricas de productividad y usabilidad, como se aprecia en la figura 9 y 10, indican que la media aritmética en la métrica de productividad es de 4.36 de 5 puntos bajo el enfoque monolítico, en comparación con la media de 4.46 bajo el enfoque de micro-servicios, lo cual quiere decir que la arquitectura propuesta aumentó ligeramente los niveles de productividad de los alumnos y docentes de la carrera de ISC en TecNM Campus Escárcega. La métrica de usabilidad alcanzó una media aritmética de 4.13 de 5 puntos bajo el enfoque monolítico, mientras que, la arquitectura propuesta alcanzó una media de 4.26 de 5 puntos, indicando un ligero aumento en la facilidad de uso de la arquitectura propuesta en contraste con el desarrollo monolítico.

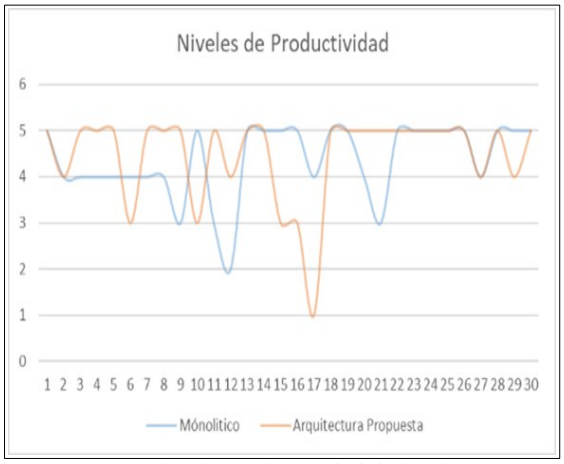


**Figura 9.** Niveles de productividad de los docentes y alumnos evaluados en TecNM Campus Escárcega.

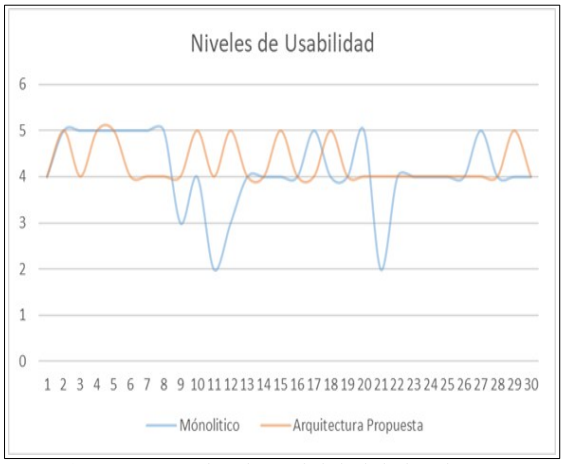


**Figura 10.** Niveles de usabilidad de los docentes y alumnos evaluados en TecNM Campus Escárcega.

A partir de los valores de las métricas evaluadas, como se aprecia en la figura 11, la media general del paradigma monolítico es de 4.24 de 5 puntos, mientras que la arquitectura propuesta alcanza un valor de 4.36 de 5 puntos. Traduciendo los valores a la escalar

de Likert, estos nos indican un nivel: "***De acuerdo***". Este nivel sugiere que los alumnos y docentes evaluados tienen un alto nivel de conformidad en la creación de aplicaciones monolíticas; sin embargo, el uso de nuevas tecnologías y paradigmas de desarrollo como el de micro-servicios mejora el nivel de conformidad, es decir, los alumnos y docentes de la carrera de ISC en TecNM Campus Escárcega están de acuerdo con la implementación de la arquitectura propuesta para el desarrollo de aplicaciones web, como reemplazo del paradigma monolítico, dado que las respuestas de los sujetos de estudio muestran valores equivalentes o ligeramente superiores utilizando nuestra arquitectura en contraste con la creación de aplicaciones monolíticas.

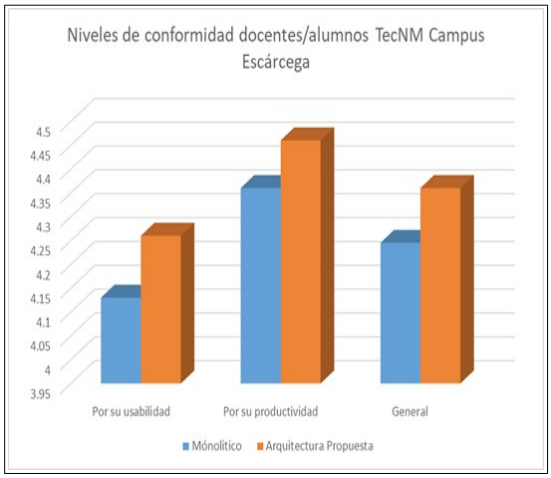


**Figura 11.** Niveles de conformidad de la arquitectura propuesta.

La arquitectura de software propuesta aumenta los niveles de productividad y reduce los tiempos de desarrollo a partir del nivel de usabilidad que indican un fácil uso al momento de crear una nueva aplicación web bajo el enfoque de micro-servicios de los docentes y alumnos de ISC en TecNM campus Escárcega que al menos tienen un proyecto de software a desarrollar orientado a la creación de aplicaciones web. Esto se debe a que la arquitectura propuesta contiene diferentes módulos: seguridad, gestión de usuarios, autenticación/registro de usuarios, temas/notificaciones y comunicación; que permiten enfocar al estudiante o docente en la lógica de negocios de su proyecto, y evita que tenga que generar desde cero módulos de uso común en una aplicación web orientada a micro-servicios. Así como, favorece la facilidad de implementación (clonación de proyecto), configuración (puertos, origen de datos y otras propiedades del proyecto) y controles de seguridad (métodos de autenticación, origen de peticiones, configuración de encabezados y otros aspectos de seguridad) al momento de generar un nuevo proyecto. De igual forma, permite tener todas las ventajas de este paradigma de desarrollo de micro-servicios como lo es la interoperabilidad, modularidad y escalabilidad en los proyectos. Para finalizar, el uso de esta arquitectura propuesta, permite mantener a los docentes y alumnos de esta institución vigente ante los nuevos paradigmas de desarrollo, como lo es el enfoque de micro-servicios; y tendencias de desarrollo tecnológico, como el caso de SpringBoot y Angular en contraste con el desarrollo monolítico basado en modelos, vistas y controladores.

## 8. CONCLUSIÓN

En este trabajo desarrollamos una arquitectura de software que tiene como objetivo ser una base de desarrollo de proyectos orientados a la creación de aplicaciones web bajo el paradigma de micro-servicios para alumnos y docentes de la carrera de ISC en TecNM campus Escárcega. La arquitectura, se compone de diferentes módulos como el de seguridad, gestión de usuarios, autenticación/registro de usuarios, temas/notificaciones y comunicación que permiten la reusabilidad al momento de crear un nuevo proyecto.

Para el desarrollo de la arquitectura dividimos el proyecto en dos partes: El frontend, consiste en la aplicación cliente desarrollada en Angular versión 9.1; el otro apartado es el backend, que contiene los micro-servicios, lógica de negocios y base de datos desarrollados con Java versión 8, Spring Boot 4.3.18 y PostgreSQL versión 9.6. Para la

medición generamos una herramienta basada en 2 métricas: productividad y usabilidad. La integración de ambas partes permite tener una base de arquitectura de software reutilizable para el desarrollo de aplicaciones web orientada a micro-servicios al momento de generar un nuevo proyecto por parte de los alumnos y docentes de la carrera de ISC en TecNM Campus Escárcega.

Los resultados obtenidos muestran un alto nivel de conformidad en el uso de la arquitectura propuesta para el desarrollo de aplicaciones web en comparación con el paradigma de desarrollo monolítico por parte de los 25 alumnos y 5 docentes evaluados que al menos tienen un proyecto de desarrollo web en TecNM Campus Escárcega. Es decir, los docentes y alumnos de esta institución están de acuerdo en sustituir los paradigmas y tecnologías de desarrollo utilizados actualmente en la institución, con el enfoque de micro-servicios a partir de la arquitectura de software propuesta, ya que les permite mejorar sus niveles de productividad y sus tiempos de desarrollo a partir de los niveles de usabilidad. Esto permite a los alumnos y docentes migrar a nuevos paradigmas de desarrollo y dejar gradualmente el paradigma de desarrollo monolítico, ya que el principal motivo que genera resistencia al cambio, son los diferentes módulos preestablecidos al iniciar un nuevo proyecto bajo el enfoque monolítico, problemática que se resuelve con el paradigma orientado a micro-servicios a partir de la arquitectura de software propuesta para el desarrollo de aplicaciones web.

## REFERENCIAS


[1] Barrios Contreras, D. A. Arquitectura de Microservicios. *Tecnología Investigación y Academia*. 2018, 6(1), 36–46.

[2] Mora, Sergio. *Programación de aplicaciones web: historia, principios básicos y clientes web*. San Vicente Alicante: Editorial Club Universitario, 2002.

[3] Dinh-Tuan, H., Mora-Martínez, M., Beierle, F., Garzón, S. R. Development Frameworks for Microservice-based Applications. *Proc. of the 2020 European Symposium on Software Engineering*. 2020, 12-20. doi: 10.1145/3393822.3432339.

[4] Haro, E., Guarda, T., Peñaherrera, A. O. Z., Quiña, G. N. Desarrollo backend para aplicaciones web, servicios web Restful: Node.js vs Spring boot. *Iberian Journal of Information Systems and Technologies*, 2019, (E17), 309-321.

[5] Wohlgethan, E. Supporting Web Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue.js [Tesis Doctorado]. Hamburgo, Alemania: Hamburg university, 2018. https://reposit.haw-hamburg.de/bitstream/20.500.12738/8417/1/BA_Wohlgethan_2176410.pdf.

[6] Saks, E. JavaScript Frameworks: Angular vs React vs Vue [Tesis Licenciatura]. Helsinki, Finlandia; Haaga-Helia University of Applied Sciences, 2019. https://urn.fi/URN:NBN:fi:amk-2019110720797.

[7] Asnika S., Vasudeva, P., Sudhindra R. RESTful Web Services. *International Journal of Advanced Information Science and Technology*. 2014, 3(4), 13-16.

[8] Iturralde, B. O. J. *Introducción a los patrones de diseño: Un enfoque práctico*. Ciudad de México: Createspace Plataforma de Publicación Independiente, 2016.

[9] Nicoll, S., Wilkinson, A., Frederick, S. Spring Boot Maven Plugin Documentation. Recuperado el 20 de febrero de 2021, de https://docs.spring.io/spring-boot/docs/2.6.0-SNAPSHOT/maven-plugin/reference/pdf/spring-boot-maven-plugin-reference.pdf.

[10] Google LLC. Angular. Angular. Recuperado 6 de febrero de 2021, de https://angular.io/docs.

[11] Callejas-Cuervo, M., Alarcón-Aldana, A. C., Álvarez-Carreño, A. M. Modelos de calidad del software, un estado del arte. ENTRAMADO, 2017, 13(1), 236–250. doi: 10.18041/entramado.2017v13n1.25125.

[12] Redrován Castillo, F. F., Loja Mora, N. M., Correa Elizaldes, K. D., Piña Orozco, J. I. Comparación de métricas de calidad para el desarrollo de aplicaciones web. *3C Tecnología_Glosas de innovación aplicadas a la pyme*, 2018, 7(3), 94–113.

[13] Constanzo, M. A., Casas, S. I., Marcos, C. A. Comparación de modelos de calidad, factores y métricas. *Informes Científicos Técnicos - UNPA*, 6(1), 1–36. doi: 10.22305/ict-unpa.v6i1.89.

[14] Cantú-Mata, J. L., Torres-Castillo, F., Alcaraz-Corona, S., Banda-Muñoz, F. Calidad, tiempo y costo en proyectos de desarrollo de software. *Interciencia*, 2018, 43(10), 707-710.

[15] Moumane, K., Idri, A., Abran, A. Usability evaluation of mobile applications using ISO 9241 and ISO 25062 standards. *SpringerPlus*, 2016,5(1), 1-15. doi: 10.1186/s40064-016-2171-z.

[16] Velasco Balazs, R. A. Análisis de la calidad de una aplicación para el desarrollo de un plan de mejora de acuerdo a las normas ISO [Tesis Licenciatura]. Valparaiso, Chile: Universidad Técnica Federico Santa María. 2017. http://hdl.handle.net/20.500.12010/4683.

[17] González-Disla, R.. Métrica de complejidad del software y procesos cognitivos. Working paper, 2016, doi: 10.13140/RG.2.2.21170.20166/2.

[18] Hernández, G., Martínez, Á., Jiménez, R., Jiménez, F. ¿Cómo los profesionales perciben la relevancia de las métricas de productividad para un equipo ágil de desarrollo de software? *Iberian Journal of Information Systems and Technologies*, 2020, (E32), 596-609.

[19] Kitchenham B. A., Pfleeger S.L. Personal Opinion Surveys. In: Shull F., Singer J., Sjøberg D.I.K. (eds) *Guide to Advanced Empirical Software Engineering*. Springer, London. 2008, 63-92. doi: 10.1007/978-1-84800-044-5_3.

[20] Matas, A. Diseño del formato de escalas tipo Likert: un estado de la cuestión. *Revista electrónica de investigación educativa*, 2018, 20(1), 38-47. doi: 10.24320/redie.2018.20.1.1347

[21] Cosentino, V., Cánovas Izquierdo, J. L., Cabot, J. A Systematic Mapping Study of Software Development With GitHub. *IEEE Access*. 2017, (5), 7173-7192. doi: 10.1109/ACCESS.2017.2682323.

## ACERCA DE LOS AUTORES

**Damián Uriel Rosado Castellanos**. Ingeniero en Sistemas Computacionales por el Instituto Tecnológico de Campeche en 2017 con especialidad en Desarrollo Web. Obtuvo el grado de Maestro en Ciencias de la Computación por el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) en 2019 con especialidad en Inteligencia Artificial. Profesor Investigador de la Línea de Investigación de Ingeniería de Software de la Ingeniería en Sistemas Computacionales de TecNM Campus Escárcega. Fundador de la empresa de desarrollo DENSCODE, Escárcega, Campeche. Ha realizado diferentes softwares institucionales, el más destacado es el sistema trayectoria escolar de TecNM Campus Escárcega. Así como ha participado en diferentes eventos nacionales e internacionales de innovación tecnológica.

**Ivette Stephany Pacheco Farfán**. Ingeniero en Sistemas Computacionales por la Universidad Autónoma de Campeche (2008) con Maestría en Informática de la Universidad Hispanoamericana (2017) y actualmente es estudiante de Doctorado en Proyectos por la Universidad Internacional Iberoamericana. Docente Investigador del Instituto Tecnológico Nacional de México campus Escárcega de la Ingeniería en Sistemas Computacionales. En el 2018 obtuvo el reconocimiento como Docente con Perfil Deseable por el Programa de Desarrollo Profesional Docente. Líder de la Línea de Investigación de Ingeniería de Software. Ha participado en diversos congresos con exposición de carteles científicos y ponencias nacionales e internacionales. Asesor de proyectos del Evento Nacional Estudiantil de Innovación Tecnológica y la Feria Nacional de Ciencias.

**Iván Humberto Fuentes Chab**. Ingeniero en Sistemas Computacionales por el Instituto Tecnológico de Campeche en 2017. Obtuvo el grado de Maestro en Ciencias de la Computación por el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) en 2019 y actualmente estudiante del Doctorado en Sistemas Computacionales en la Universidad Da Vinci. Ha participado en carteles científicos en la 2ª Jornada de Ciencia y Tecnología del CENIDET, seminarios de investigación con el Grupo ARKADIUS de la Universidad de Medellín. Docente Investigador del Instituto Tecnológico Superior de Escárcega de la Licenciatura de Ingeniería en Sistemas Computacionales.

**Julio Cesar Cantun Páez.** Estudiante de séptimo semestre de la carrera de Ingeniería en Sistemas Computacionales en el Instituto Tecnológico Superior de Escárcega. Ha participado en eventos como Evento Nacional Estudiantil de Innovación Tecnológica (ENEIT) en su fase local y regional, así como en diferentes concursos de programación a nivel regional y nacional como el Coding Cup TecNM.