Artículo **Revista de Tecnologías Computacionales**
Junio 2020 Vol.4 No.13 9-13

Uso de la arquitectura de mini servicios: gestión de servicios

Using the mini service architecture: service management

BENÍTEZ-QUECHA, Claribel†*, ALTAMIRANO-CABRERA, Marisol, SÁNCHEZ-CHÁVEZ, Jorge Edgar y MÉNDEZ-LÓPEZ Minerva Donají

Tecnológico Nacional de México Campus Oaxaca, México.

ID 1er Autor: *Claribel, Benítez-Quecha* / **ORC ID:** 0000-0001-6516-5760, **CVU CONACYT ID:** 657582

ID 1er Coautor: *Marisol, Altamirano-Cabrera* / **ORC ID:** 0000-0001-5800-9655, **CVU CONACYT ID:** 657390

ID 2do Coautor: *Jorge Edgar, Sánchez-Chávez* / **ORC ID:** 0000-0002-4737-091X

ID 3er Coautor: *Minerva Donají, Méndez-López* / **ORC ID:** 0000-0001-5336-1772

DOI: 10.35429/JOCT.2020.13.4.9.13 Recibido: Abril 15, 2020; Aceptado: Junio 30, 2020

Resumen

Existen diversas arquitecturas que permiten un sistema distribuido que han sido ampliamente probadas en un entorno empresarial, con el propósito de tener alta disponibilidad. Esta infraestructura nos permite un despliegue rápido y una menor carga en la red, desarrollo rápido entre otros. El presente trabajo se avoca a construir módulos de software para una arquitectura de mini servicios, bajo una API REST gestionada por NestJs, compartiendo una base de datos Mongo.

Objetivo General: Implementar el uso de la arquitectura de miniservicios en el desarrollo de los módulos de gestión de recursos, control y seguimiento de proyectos para el sistema de vinculación entre incubadoras del TecNM y sociedad en general.

Específicos:

1. Representar los procesos mediante BPMN (Business Management Process).
2. Analizar los requerimientos funcionales del módulo de recursos y seguimiento de servicios.
3. Diseñar la arquitectura de miniservicios.
4. Diseñar los miniservicios para la gestión de la base de datos del módulo de recursos y seguimiento de servicios.
5. Desarrollar el miniservicio para la gestión de la base de datos del módulo de recursos y seguimiento de servicios.
6. Desarrollar el miniservicio para la gestión del módulo de recursos y seguimiento de servicios.

Metodología. Para la realización de este proyecto se utiliza la metodología SCRUM, ya que para su desarrollo se dividió a los participantes en dos equipos, debido a la extensión del sistema computacional. Y SCRUM, se caracteriza por ofrecer un marco metodológico de trabajo que permite gestionar la colaboración entre varios equipos. Así como ofrece una gran adaptabilidad a modificaciones sobre diseños o codificación previa, punto muy importante en este caso, ya que se trabaja sobre un tema que es actual y no se cuenta con experiencia previa, lo que genera constantes pruebas y cambios. SCRUM tiene otra característica esencial en éste desarrollo, de que está diseñado para proyectos de alto nivel de incertidumbre, como es nuestro caso, donde se probará la arquitectura de miniservicios y se irá identificando el grado de mejoras que aporta. Por todo lo anterior el presente trabajo se ha llevado a cabo siguiendo las etapas marcadas por SCRUM, a saber:

1.- Planificación del sprint.
2.- Etapa de desarrollo.
3.- Revisión del sprint.
4.- Retroalimentación.

Contribución. Con este trabajo se aporta la experiencia en la configuración de una arquitectura que soporta un desarrollo rápido, desacoplado, reutilizable y con la posibilidad de migrar a una arquitectura de microservicio.

Miniservicio, Microservicios

Abstract

There are various architectures that allow a distributed system that have been extensively tested in a business environment, with the purpose of having high availability. This infrastructure allows us a rapid deployment and a lower load on the network, rapid development among others. The present work aims to build software modules for a mini-service architecture, under a REST API managed by NestJs, sharing a Mongo database.

General Objective: Implement the use of the architecture of miniservices in the development of the modules of resource management, control and monitoring of projects for the linking system between TecNM incubators and society in general.

Specific:

1. Represent the processes using BPMN (Business Management Process).
2. Analyze the functional requirements of the resource module and service monitoring.
3. Design the architecture of miniservices.
4. Design the mini-services for the management of the resource module database and service monitoring.
5. Develop the mini-service for managing the resource module database and service monitoring.
6. Develop the mini-service for the management of the resource module and service monitoring.

Methodology. To carry out this project, the SCRUM methodology is used, since for its development the participants were divided into two teams, due to the extension of the computer system. And SCRUM is characterized by offering a methodological framework of work that allows managing collaboration between various teams. As well as offering great adaptability to modifications on designs or previous coding, a very important point in this case, since we are working on a topic that is current and there is no previous experience, which generates constant tests and changes. SCRUM has another essential characteristic in this development, that it is designed for projects with a high level of uncertainty, as is our case, where the architecture of miniservices will be tested and the degree of improvements it provides will be identified. For all the above, this work has been carried out following the stages set by SCRUM, namely:

1.- Sprint planning.
2.- Stage of development.
3.- Sprint review.
4.- Feedback.

Contribution. With this work, experience is provided in the configuration of an architecture that supports rapid development, decoupled, reusable and with the possibility of migrating to a microservice architecture..

Miniservices, Microservices

Citación: BENÍTEZ-QUECHA, Claribel, ALTAMIRANO-CABRERA, Marisol, SÁNCHEZ-CHÁVEZ, Jorge Edgar y MÉNDEZ-LÓPEZ Minerva Donají. Uso de la arquitectura de mini servicios: gestión de servicios. Revista de Tecnologías Computacionales. 2020. 4-13:9-13.

† Investigador contribuido como primer autor.

Introducción

Para este trabajo se realizó un análisis del proceso actual de las incubadoras del TecNM a nivel nacional, en el cual se visualizó que los procesos de las incubadoras al impartir un servicio se hace de manera muy similar y que además comparten un catálogo de servicios el cual es difícil de conocer, dado que no cuentan con un sistema que automatice estas tareas. Por tanto surge la necesidad de realizar un sistema que automatice y estandarice estas tareas, por tanto se requiere una arquitectura que sea capaz de soportar los usuarios a nivel nacional, y proporcione la alta disponibilidad que se espera. Este trabajo consiste en buscar e implementar una arquitectura que proporcione desacoplamiento, rapidez y un despliegue fácil en la nube.

Planteamiento del problema

Las incubadoras del TecNM carecen de un sistema el cual haga posible que la sociedad interesada en trabajar con una incubadora de proyectos se ponga en contacto y a su vez se realice todo el proceso de vinculación y seguimiento de los proyectos que se lleguen a concretar, además de estandarizar la forma de trabajo, ya que las incubadoras del TecNM llevan de diferente manera el control de sus reportes, ya se mediante documentos elaborados a mano, hojas de excel, etc.

Dado que el sistema es para un público a nivel nacional se requiere una arquitectura que sea capaz de controlar el nivel de usuarios que se pueden presentar de manera concurrente.

Marco teórico

La arquitectura de miniservicios nos da como alternativa el no ser tan restrictivos con las directrices en el diseño y reduce las inconsistencias, obteniendo beneficios similares a los de los microservicios.

La arquitectura de miniservicios puede ser implementada con diferentes tecnologías dado que solo se tiene que exponer un API bajo una especificación existente por ejemplo: REST o GraphQl.

NestJS es un framework para Node.js, que permite el uso de multi paradigmas de programación, principalmente enfocado al paradigma asíncrono, pero fácilmente se puede incluir programación reactiva, funcional u orientada a objetos.

Mongo es una base de datos no relacional, que permite guardar la información en formato JSON, muy amigable para trabajar con API's además que mediante GridFS se pueden gestionar archivos sin la necesidad de contratar un servicio de ficheros adicional.

Desarrollo

Dadas las necesidades del sistema de incubadoras del TecNM se decidió optar por una arquitectura de miniservicios, por las ventajas que estos tienen. La arquitectura cumple con un desarrollo ágil, desacoplamiento, fácil despliegue en la nube, migración sencilla a una arquitectura de microservicios, compartir un origen de datos (base de datos Mongo).

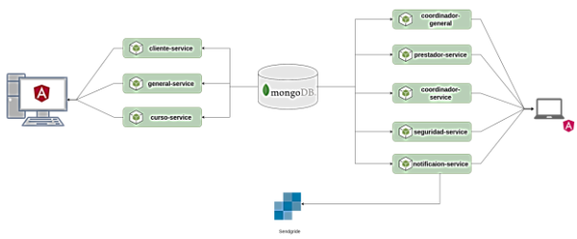

Figura 1 Arquitectura del sistema
Fuente: Elaboración propia

Las tecnologías por ocupar deben ser capaces de utilizar una arquitectura de miniservicios, tener fácil despliegue en la nube, por tal motivo se decidió trabajar con NestJS, un framework que permite realizar este trabajo de manera sencilla, junto con la base de datos Mongo, NestJS provee una manera sencilla al trabajar los objetos del mismo modo que lo hace mongo, la representación de la información en formato JSON.

La arquitectura plantea dos clientes con Angular, uno de ellos dedicado al backoffice, es decir la parte administrativa de este sistema, coordinadores de incubadora, coordinadores generales y prestadores de servicio. La aplicación del cliente, se encarga de la lógica de negocio del sistema.

ISSN 2523-6814

La seguridad del conjunto de miniservicios está gestionado por el miniservicio de seguridad, el cual gestiona los jwt-tokens de acuerdo a los roles, es decir coordinador de incubadora, coordinador general, prestador de servicio y cliente, este miniservicio se encarga de gestionar el tiempo de vida de los tokens, denegar acceso de tokens inválidos, etc.

Figura 2 Estructura de un token
Fuente: Elaboración propia

Como se muestra en la arquitectura los clientes (angular) son los encargados de la comunicación con los miniservicios, dependiendo de la información que se necesite se encargan de hacer peticiones con el respectivo miniservicio.

El desarrollo de las interfaces para los módulos de gestión de recursos y seguimiento de proyectos se muestran a continuación.

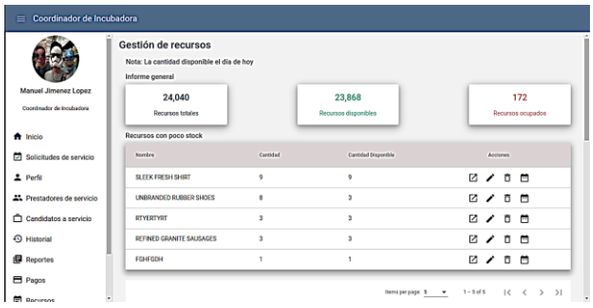

Figura 3 Gestión de recursos
Fuente: Elaboración propia

La gestión de recursos muestra un resumen de los recursos que tenemos que tenemos el dia de hoy la cantidad total, disponible y ocupada de los recursos, categoriza la información de acuerdo al stock de los recursos con los que se cuente.

Figura 4 Servicios en curso
Fuente: Elaboración propia

En el módulo de seguimiento muestra los servicios contratados por incubadora, el estado de los servicios contratados y los tipos de servicios que están contratados.

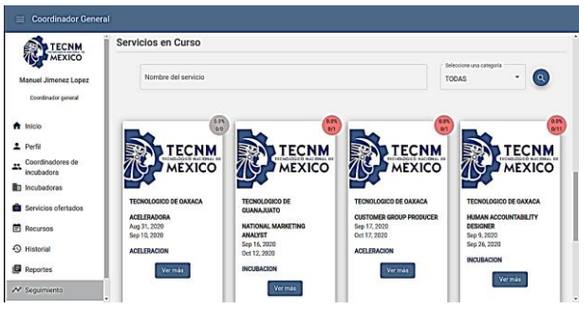

Figura 5 Servicios en curso, lista
Fuente: Elaboración propia

Se muestra una lista de todos los servicios que están en curso, mostrando información básica de estos, por ejemplo, nombre, estado, incubadora que lo ofertó, periodo de contratación del servicio, además de opciones de filtrado y búsqueda.

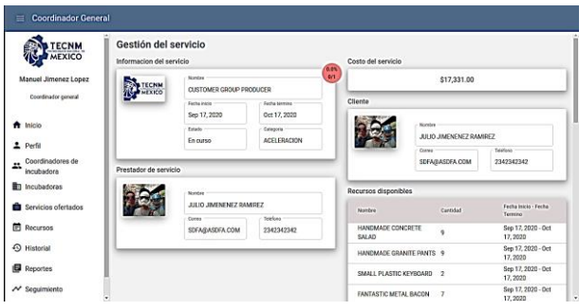

Figura 6 Detalle de un servicio
Fuente: Elaboración propia

Al seleccionar un servicio se muestran los detalles de este, tales como. prestador de servicio, estado, precio del servicio, cliente, recursos asignados al servicio, cronograma de actividades, entre otros.

ISSN 2523-6814

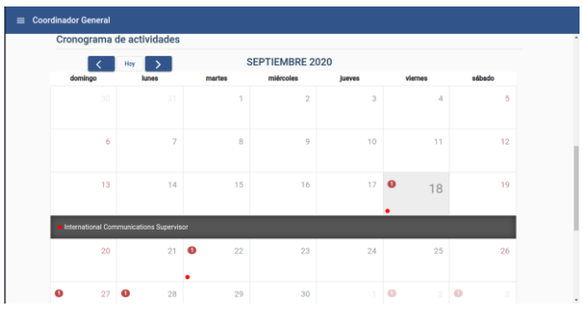

Figura 7 Cronograma del servicio
Fuente: Elaboración propia

Se detallan las actividades por día de un servicio contratado y el estado de estas es decir si se realizaron, o no se han realizado aún, en base al estado de estas actividades se obtiene el estado de servicio.

Figura 8 Monitoreo, base de datos
Fuente: Elaboración propia

El servicio de alojamiento MongoDB Atlas ofrece herramientas de monitoreo de la base de Datos. La herramienta permite observar métricas del rendimiento de la base de datos, se puede observar en la gráfica que a pesar de tener una arquitectura de miniservicios el uso de red y las conexiones a la base de datos son estables y no representan un riesgo en el rendimiento del sistema.

Resultados

Se pudo ver que la implementación de miniservicios mejora sustancialmente la posibilidad de acoplamiento entre diferentes procesos, ya que éstos a su vez se encuentran encapsulados en miniservicios, lo que proporciona una mayor independencia, al momento de hacer cambios, ya que no se amarran a un diseño, si no que permite la movilidad como piezas de cubos. Y se puede modificar el engranaje o acoplamiento de los mismos, según vayan variando las necesidades del sistema. Y la comunicación, se da de forma más rápida entre los miniservicios a través de mensajes.

Además, se logra una rápida interacción entre los procesos, ya que la comunicación es a través de mensajes entre los microservicios, eliminando un mayor número de fallas en la comunicación. Y descargando el tráfico en la red.

Por otra parte, pueden interactuar con una misma base de datos, soportando un gran número de accesos al mismo tiempo.

Dados los constantes cambios en las tecnologías en el área de sistemas computacionales, la implementación de desarrollos computacionales se ve enormemente beneficiada con el uso de arquitecturas de miniservicios y microservicios, ya que permiten hacer modificaciones constantes sin afectar a otros módulos como ocurre en sistemas monolíticos que, aunque estén divididos en módulos a final de cuentas pertenecen a un solo ejecutable que es necesario estar generando cada vez que se hace un cambio.

Lo cual no ocurre con el uso de arquitecturas de mini y microservicios ya que sólo se modifica el mini o microservicio que contiene el módulo y se hace el linking con los demás servicios.

Agradecimiento

Agradecemos al Tecnológico Nacional de México Campus Oaxaca, por su apoyo para la realización de este trabajo.

Conclusiones

El uso de miniservicios permite desarrollar Sistemas Computacionales más adaptables a futuras mejoras, con la implementación de reingeniería. Ya que encapsula procesos dentro de servicios, permitiendo la independencia entre los procesos. Permite el trabajo colaborativo entre grupos de programadores, cuestión muy importante en sistemas muy grandes.

Y los miniservicios permiten pasar rápidamente a una arquitectura de microservicios, conforme el sistema va creciendo. Sin necesidad de implementar microservicios desde un principio cuando el sistema es aún pequeño, con los miniservicios la complejidad es menor pero ya queda listo para la migración a microservicios.

ISSN 2523-6814

Referencias

Álvarez Caules, Cecilio. *Qué es un microservicio. Arquitectura Java*. 23 Enero 2015. Recuperado de https://www.arquitecturajava.com/que-es-un-microservicio/ el 6 de Febrero del 2020.

Berenguel Gómez, José Luis. *Desarrollo de aplicaciones web distribuidas*. Ed. Ediciones Paraninfo. 2016.

Casado, Sergio. *¿Cómo llevar al siguiente nivel tu backend en Node? Introducción a NestJS.* Recuperado de https://www.paradigmadigital.com/dev/introduccion-nestjs/ el 7 de Mayo del 2020.

Gupta, Aashis, Thomas, Anne. *Perspectiva de Innovación para Miniservicios.* Ed. Gartner. 21 de febrero de 2017.

Llopis, Joan. *Aligerar el Monolito: Miniservicios y Microservicios*. Recuperado de https://clouddistrict.com/aligerar-monolito-miniservicios-microservicios/ el día, 11/05/2020.

Ortega Candel, José Manuel. *Tecnologías para arquitecturas basadas en microservicios: Patrones y soluciones para aplicaciones desplegadas en contenedores.* Ed. José Manuel Ortega. 30 de Junio 2020.

ISSN 2523-6814