Migración backend a una arquitectura orientada a eventos: una aplicación para la creación y gestión de planes institucionales.

Anderson Estiven Villa Sierra

Informe de práctica presentado para optar al título de Ingeniero de Sistemas

Asesor
Luz Viviana Cobaleda Estepa, Doctora en Ingeniería Electrónica

Universidad de Antioquia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas
Medellín
2024

Cita		Villa Sierra [1]
Referencia Estilo IEEE (2020)	[1]	A. E. Villa Sierra, “Migración backend a una arquitectura orientada a eventos”: una aplicación para la creación y gestión de planes institucionales”, práctica empresarial, Ingeniería de Sistemas, Universidad de Antioquia, Medellín, 2023.

Dedicatoria

A mi madre que siempre me apoyó durante toda mi carrera. A mis amigas y amigos por quererme.

Agradecimientos

Quiero expresar mi más sincero agradecimiento a la empresa Pragma S.A por brindarme la invaluable oportunidad de realizar mi práctica profesional. Agradezco la confianza que han depositado en mí, así como el apoyo constante y la orientación proporcionada por todos los miembros del equipo, especialmente a Gilberto Gil, su experiencia y guía han sido invaluables para el aprendizaje y continua mejora. Finalmente, a la Universidad de Antioquia, agradezco la sólida formación que ha sido clave en mi experiencia en Pragma. Quiero expresar mi gratitud a Luz Viviana Cobaleda Estepa, mi asesora de prácticas, por su orientación invaluable.

Tabla de contenido

LISTA DE FIGURAS

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

PO	Product owner
MVP	Minimum viable product

RESUMEN

El presente proyecto busca enfocarse en la migración tecnológica del aplicativo para la Gestión de contratos de planes para empresas (antiguo) que tiene el cliente, más específicamente, migrar todo el backend y extender el alcance del mismo. Ya que el aplicativo actual que se tiene, presenta muchos inconvenientes que se transforman en manualidades para el usuario final y extiende así el tiempo de operación de las personas encargadas. Este aplicativo se encarga de la creación, la consulta y la gestión de contratos de planes para empresas, los cuales funcionan como un acuerdo para las personas involucradas con ciertas condiciones como lo pueden ser restricciones, tiempos, personas encargadas y demás, por lo cual, los largos tiempos de respuesta y los errores del aplicativo anterior es que se decide migrar dicho aplicativo, ya que es un aplicativo que carga con una responsabilidad bastante alta, y su buen funcionamiento debe ser esencial en el día a día. Se emplearon metodologías de desarrollo ágil como lo son SCRUM, con el fin de entregar funcionalidades de manera constante y periódicas, también haciendo uso del stack de tecnologías propuesto por la compañía. Todo esto con el objetivo de garantizar la entrega de un producto que cumpla con todos los requerimientos planteados y que responda a las necesidades del usuario final y que tecnológicamente hablando sea escalable, lo cual permita que el proyecto siga creciendo a medida que el alcance del negocio lo haga de igual manera.

Palabras clave **— Migración, Backend, AWS, Eventos, Arquitectura limpia, Mensajería**

ABSTRACT

This project focuses on the migration of an old application that the client has, more specifically, migrate the entire backend and extend the scope of it. Since the current application has many drawbacks that become manual tasks for the end user and thus extends the operation time of the people in charge. This application is responsible for the creation, consultation and management of plan contracts for companies, which work as an agreement for the people involved with certain conditions such as restrictions, times, people in charge and others, therefore, the long response times and errors of the previous application is that it is decided to migrate this application, since it is an application that carries a fairly high responsibility, and its proper functioning must be essential in the day to day. Agile development methodologies such as SCRUM were used, in order to deliver features constantly and periodically, also making use of the stack of technologies proposed by the company. All this in order to ensure the delivery of a product that meets all the requirements proposed and that responds to the needs of the end user and that technologically speaking is scalable, which allows the project to continue to grow as the scope of the business does likewise.

Keywords **— Migration, Backend, AWS, Events, Clean Architecture, Messaging**

I. INTRODUCCIÓN

Este proyecto surge de la necesidad de migrar un aplicativo que ya poseía varios años de haber sido desarrollado, el cual estaba presentando notables fallas. Su funcionalidad era la de crear unos contratos de planes para las empresas pero no estaba respondiendo de la manera adecuada o esperada, adicionalmente el área de operaciones se encontraban haciendo más trabajo del que se debería por el funcionamiento del anterior aplicativo, uno de los ejemplos es que, en el anterior aplicativo, a las personas encargadas de generar los contratos, no se les permitía añadir o ajuntar ciertos formatos, descargar pdf, enviar por correo electrónico el contrato, etc, entonces el área de operaciones tenía que realizar dichas tareas de manera manual, por lo cual se les incrementaba de manera considerable la carga laboral, ya que primero tenían que revisar el contrato y adicional, las tareas que el aplicativo no estaba supliendo. Por lo anterior mencionado, se tomó la decisión de migrar el aplicativo completamente, aumentar el alcance para evitar y borrar manualidades desde operaciones, tener un nuevo aplicativo más escalable y mantenible, utilizar una arquitectura de referencia que ayuda a aplicar buenas prácticas de desarrollo y a que el código sea mejor estructurado, también a utilizar una arquitectura basada en eventos, usando servicios de AWS como colas de mensajería y tópicos, lo cual permite al nuevo aplicativo soportar de mejor manera la concurrencia de muchos usuarios haciendo uso de los servicios. Esta migración se abordó por módulos, que representan una pestaña para el usuario que crea los contratos teniendo como total 8 módulos, que cada uno contiene información específica y segregada de los ítems o cláusulas que tendrán los contratos a generar. Esto con el fin de tener un orden en el momento de la construcción del aplicativo, ya que cada módulo tiene sus integraciones con terceros u operaciones distintas, por eso se decidió esta forma de abordar la migración. Es de suma importancia aclarar que no se presentan diagramas ni pantallas de la solución, ya que es información confidencial de la empresa para quien se realizó el desarrollo.

II. OBJETIVOS

A. Objetivo general

Migrar el backend del aplicativo antiguo para facilitar la creación y parametrización de los contratos del cliente y optimizar las tareas del área de operaciones, implementando una arquitectura orientada a eventos.

B. Objetivos específicos

1. Refinar las historias de usuario y las tareas necesarias para la construcción del producto.
2. Implementar la arquitectura de referencia en los 4 microservicios necesarios.
3. Desarrollar la migración de manera incremental, empleando metodologías ágiles para así tener soluciones funcionales en pequeños intervalos de tiempo.
4. Validar salud de cada microservicio de manera independiente
5. Crear escenarios de pruebas donde se pruebe el flujo correcto del aplicativo.

III. MARCO TEÓRICO

El problema que se busca solucionar con el cliente, nace de una necesidad que generó el aplicativo anterior que poseían, ya que presentaba constantes fallos a la hora de cumplir con sus objetivos, y estos fallos generaban muchas manualidades adicionales al área de operaciones.

La solución más viable para el cliente, es hacer una migración completa de los servicios backend, entendiéndose como migración el proceso trasladar aplicaciones de software de un entorno informático a otro, y en su proceso mejorar sus funcionalidades y actualizar sus tecnologías [1]. En este caso la migración estará centrada en backend, que se refiere a las partes de una aplicación informática que permiten su funcionamiento y a las que el usuario no puede acceder, como por ejemplo datos, la manera en la que funciona el almacenamiento, y la lógica gruesa del aplicativo [2]. Esta migración se hará empleando los servicios de AWS, como lo son colas de mensajería, tópicos, servicios para almacenar documentos y demás, siendo AWS un proveedor de servicios basados en la nube, que le permite a las aplicaciones tener una mayor escalabilidad y flexibilidad a demanda, también permite disponer de recursos a los que usualmente no se tiene acceso, como lo son servidores físicos, pero en este caso, todo se encuentra en un entorno de nube.

Para este reto la migración fue específicamente construida con una arquitectura basada en eventos, que internamente está compuesta por una arquitectura de referencia, que es básicamente un estructuración predeterminada para la construcción del software, esto basándose en los conceptos de arquitectura limpia que indican una serie comportamientos deseables que debe tener el software a construir, como lo son la independencia a la base de datos, la independencia a las tecnologías, la capacidad para ser probado y para ser escalado, esto ayuda a que los aplicativos que estén construidos con estas arquitecturas sea más fácil de extender su alcance, mantenerlo y actualizarlo frente a nuevas tecnologías [3].

La arquitectura por eventos permite aprovechar la asincronía que es permitirle al programa solicitar que una tarea se realice al mismo tiempo que la tarea original, sin detenerse a esperar a que la primera se haya completado [4] y que el aplicativo responda cuando esté preparado para desencolar los mensajes que encoló [5]. Esto es una buena estrategia para darle

estructura a la solución, ya la solución se planteó con 4 microservicios, y un microservicio es un enfoque arquitectónico para el desarrollo de software, que básicamente está compuesto por pequeños servicios independientes, y se comunican entre sí a través de APIS, para así separar de mejor manera las responsabilidades y permitir la escalabilidad y la rapidez al desarrollar, y estos 4 microservicios estarán sometidos a una gran carga de trabajo, y con disponibilidad 24/7, entonces esta arquitectura permite adoptar características como la resiliencia, comunicación asíncrona y la escalabilidad de los microservicios [6]. La facilidad y agilidad en la implementación de una arquitectura por eventos, es tener claro varios patrones como lo son el CQRS, que es identificar y dividir las operaciones en lo que son comandos, y consultas, es decir qué operaciones mutan los datos o la parte de comando de un sistema, y qué parte son consultas [7]. Con esto identificado se puede conocer qué será enviado como evento, y qué será construido como query. De esta manera podremos tener un producto más estructurado con la arquitectura planteada y ya dar paso al desarrollo de la migración, que está compuesta por operaciones de guardado, consultas, actualizaciones, eliminar datos, consumo a servicios maestros, generación de pdf, envío por correo electrónico y demás. De esta forma, se alcanza a abordar todo el alcance planeado y así reduciendo los errores y las manualidades actualmente presentes.

Adicionalmente al desarrollo de la solución, estuvo presente una tarea fundamental que permite asegurar de cierta manera las funcionalidades desarrolladas y fueron las pruebas unitarias y la salud del código, donde las pruebas unitarias juegan un papel muy fundamental en lo que es el aseguramiento del código o de la solución planteada, esto porque ayudan a explorar distintos flujos del aplicativo, distintos escenarios que se puedan presentar, la manera en la que se puede comportar un servicio o cómo puede responder a diversas solicitudes o parámetros. Ya que estas pruebas se centran en validar las funcionalidades en la menor unidad de diseño de software, es decir a nivel de módulo (Métodos de una clase, o una clase) se puede detectar errores en cada uno de esos módulos, ya que son ejecutadas de manera independiente a los demás [8]. Estas pruebas unitarias brindan algo llamado cobertura de código, que es la cantidad de código que se encuentra cubierto por las pruebas realizadas, en este caso se tenía como cobertura mínima un 70% por microservicio. Siendo esta métrica la que nos ayudó a validar el código desarrollado en los 4 microservicios.

IV. METODOLOGÍA

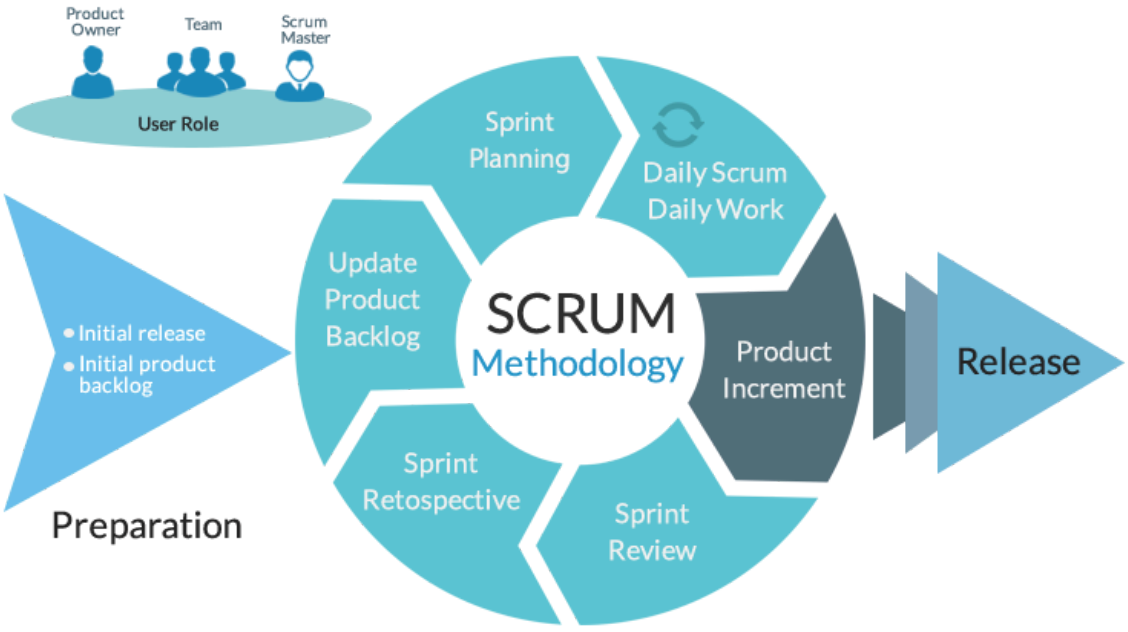

Fig. 1. Flujo de la metodología Scrum, tomado de[9]

La metodología implementada, es una metodología ágil, en este caso se usó Scrum ya que la idea era tener avances funcionales periódicamente, en este caso cada 15 días (Este periodo de 15 días se le conoce como Sprint) se presentarán los avances realizados en las ceremonias correspondientes. Ya que esta metodología tiene como finalidad la entrega continúa de valor, se centra en dividir la solución en tareas o historias de usuario, que serán los entregables correspondientes a una funcionalidad que al ser completada estaría agregando valor al proyecto y a la solución.

Las historias de usuario, son aquellas tareas y funcionalidades que al ser finalizadas en el sprint, agregan valor al producto. Durante el proceso de desarrollo, estas historias de usuario fueron definidas en una etapa temprana del proyecto, ya que al definir esta historias de usuario previamente se tiene mayor control sobre la planeación de las actividades que se llevarán durante los sprint, y permite organizar de mejor manera la capacidad a comprometer para dicho periodo.

Para este proceso, el registro de las historias de usuario se llevó en GitLab, donde se tuvo en cuenta un tablero con distintas columnas como lo son “Refinada, Aprobada planning, en curso, terminada, en QA” para así llevar un control del trabajo desarrollado, ya que refleja el flujo por el cual pasa una historia de usuario en el transcurso del sprint, se podría evidenciar si estuvo bloqueada, si tuvo impedimentos, y demás información que genera valor para el equipo de desarrollo.

En primera instancia, se buscó desarrollar un producto mínimo viable o “MVP”, que es una versión del producto que contiene las funcionalidades básicas o indispensables para funcionar de la mejor manera, cuando se tiene el MVP, lo que se buscará es seguir agregando características o funcionalidades que mejoran el producto final.

En general la metodología para este proyecto, funcionó de la siguiente manera. Inicialmente se definió un Backlog que es el conjunto de historias de usuario que se tienen hasta el momento, generalmente se van creando en orden de prioridad o de dependencia con otras historias. Ya con un banco de historias el siguiente paso era analizarlas y comprenderlas, más conocido como la fase de refinamiento, que el espacio donde el equipo de negocio socializa las historias de usuario con el equipo de desarrollo, esto con el fin de resolver dudas o problemas que puedan surgir de una historia de usuario, como por ejemplo, “Qué debe pasar si el usuario borra esta sección de información?”, es el espacio adecuado para tener esas discusiones, ya que el negocio es el encargado de resolver todas las inquietudes respecto a la información otorgada. Luego el enfoque estaba en planear las actividades que serán comprometidas para trabajar en el sprint correspondiente, esto se hizo en nuestro caso definiendo cuánta capacidad de trabajo podría realizar una persona, y con base en eso se hicieron las asignaciones correspondientes de historias de usuario.

La planeación fue una de las ceremonias más importantes durante todo este proceso, junto con los refinamientos, pero también hubo espacios que eran completamente necesarios para trabajar con esta metodología, y fueron los espacios de *review,* que básicamente se tienen finalizando los sprints, que es donde se le presentan los avances o el valor agregado alcanzado en el sprint, a los interesados del proyecto, es decir, PO, Team members que son personas que también conforman el equipo de trabajo, y les interesa el avance del proyecto, Scrum y equipo de

desarrollo. De estos espacios, es donde surgían cambios por parte del equipo del negocio, dado que algo cambiara en el momento en que se les presente la entrega, por eso tanta importancia en estos espacios, ya que es donde el equipo de negocio podría ver las funcionalidades agregadas y dado el caso de no estar conforme con algo, era el momento en el que se le indicaba al equipo de desarrollo de los futuros cambios. También se contaba con la ceremonia de las retrospectivas, que era el espacio donde se analizaba qué se hizo bien, qué se puede mejorar, qué no se hizo tan bien y qué medidas tomaremos como equipo para atacar esas fallas que se presentaron durante el sprint.

Al tener un backlog definido, y un horizonte claro sobre lo que sería el alcance del aplicativo, y cuáles serían los primeros entregables, es decir el MVP, se socializó la arquitectura a implementar para los 4 microservicios y los recursos necesarios como para su correcto funcionamiento, como lo son: base de datos, proyecto en GitLab, colas y tópicos de mensajería, bucket s3, y demás recursos iniciales.

Con respecto a las tecnologías usadas para dar solución a este problema, fueron Java como lenguaje de programación, haciendo uso del framework Spring Boot, que nos facilitó un gran conjunto de herramientas para los soportes de los microservicios, ya que proporciona una gran facilidad de configuración, desarrollo y escalabilidad. Con respecto a la base de datos, se decidió usar una base de datos, en este caso relacional, siendo esta Postgresql, y para el versionamiento de código, se usó Git, y para el alojamiento de los repositorios, también se hizo uso de GitLab, que contaba con una configuración para despliegue continuo, haciendo uso de la herramienta Jenkins para estos despliegues. Todo lo relacionado a los despliegues continuos, se pudo ver de manera integrada en un proceso llamado DevSecOps, que es una práctica que busca la integración de pruebas de seguridad en cada etapa del proceso de desarrollo de software, donde se tienen herramientas y procesos que ayudan a comprometer la colaboración entre varias partes, como lo son el equipo de desarrollo, seguridad y área de operaciones. Donde el equipo de desarrollo se encarga de la codificación, creación y pruebas, el equipo de seguridad de introducir una etapa de seguridad temprana, y operaciones que se encarga de la publicación, supervisión y corrección de cualquier problema que se pueda desprender del software. Esta práctica se implementó ya que es de mucha ayuda para todo el proceso de desarrollo, ya que permite identificar y abordar problemas de seguridad de una manera más eficaz y en una etapa más temprana identificando vulnerabilidades del software [10]. También ayudando a que los procesos

de despliegue se encuentren estandarizados con ciertas etapas y flujos de trabajo, para así agilizar más estos procesos tan importantes en la etapa de desarrollo.

El inicio del proceso de desarrolló empezó desde el sprint 2, ya que en el primer sprint se enfocó en el aprovisionamiento de los microservicios, la definición de los modelos de base de datos y demás fases iniciales, en los siguientes sprint restantes se consiguió tener un aplicativo estable, que contaba con las funciones bases de cada microservicio como lo son: Crear una empresa, actualizarla, borrarla, consultarla, con respecto a los contratos, se creó la funcionalidad de crear los contratos por empresa con sus 8 ítems independientes que conforman todas las cláusulas del contrato, generar un archivo pdf del contrato que se ha ido parametrizando, también se consumieron diversos servicios de integración importantes para el correcto funcionamiento y correcto flujo de negocio, alguno de ellos era consumir un servicio que habilitaba un trámite por contrato, y permitía al usuario cargar documentos necesarios para la validación del contrato. Y también se logró la parte final del flujo, es que enviar toda la información del contrato al core del cliente, que es donde se genera la tarea al equipo de operaciones con toda la información del contrato, y cuando se genera dicha tarea, se le notifica al usuario vía correo electrónico, que su contrato ya está en proceso de aprobación. Todo esto conformaba el MVP, que era lo mínimo con lo que se podía salir, de igual manera se continuó agregando características y mejorando la solución entregada.

V. RESULTADOS

Se alcanzó el objetivo de migrar el aplicativo anterior, a un aplicativo separado por microservicios, y haciendo uso de una arquitectura de referencia, que estuviese orientada a eventos. Donde se logró tener microservicios separados e independientes, los cuales responden a responsabilidades muy específicas, es decir, se logró separar la responsabilidad que se tenía unificada en el aplicativo anterior, lo cual generaba una gran robustez en el mismo y que fuera más difícil de comprender, extender y mantener. También el implementar una arquitectura basada en eventos, garantiza varios elementos importantes como lo es la concurrencia, que al microservicio le lleguen múltiples peticiones al tiempo, con esta arquitectura, irá tomando mensajes por mensajes de la cola de mensajería de AWS, y los irá procesando a medida que los recursos se vuelven a encontrar disponibles

Microservicio 1	Microservicio 2	Microservicio 3	Microservicio 4
Proceso de creación, consulta, eliminación y actualización de las empresas a registrar en el aplicativo	Proceso de creación de contratos de planes, eliminación, consulta, y actualización	Creación del trámite correspondiente a los contratos (Cargue de distintos documentos dependiendo de la parametrización)	Descarga de PDF, Envío de correo electrónico, Envío de información al CORE
Consumo de servicios de integración necesarios para el manejo de las empresas en el contexto del cliente	En este microservicio se exponen todos los posibles servicios que están relacionados con los planes	Este microservicio cuenta con un JOB, es decir un trabajo o tarea automática programada, para conocer el estado de los documentos que sube el usuario, para así darle continuidad al proceso	Este microservicio también cuenta con un JOB, con el fin de obtener los contratos que se encuentren en un estado específico, para así enviar automáticamente los contratos al core, es decir, para que el equipo de operaciones los reciba

Fig. 2. Responsabilidades por microservicio

La solución construida abarcó el MVP, con las funcionalidades anteriormente descritas, donde dentro de esas funcionalidades se cubrieron varias de las manualidades que tenía el equipo de operación, como lo eran la notificación de los documentos aprobados, que se cargan en el trámite, lo cual le da continuidad al contrato, también, cuando el equipo de operaciones aprueba

los contratos, se logró que se enviara un correo electrónico automático a la empresa que iba a ser responsable de dicho contrato, con una copia del PDF, otra manualidad que se le quitó al equipo de operaciones se encuentra en la propia parametrización del contrato, ya que estos contratos representan más de 250 combinaciones de cláusulas, donde algunas de esas combinaciones anteriormente no eran replicadas al equipo de operaciones, y en esta nueva implementación y migración se logró que todas las combinaciones posibles de un contrato fueran completamente replicadas, esto se logró extendiendo actualizando y extendiendo servicios de integración, también cabe recalcar que dichas combinaciones también se replicaron en el pdf que el usuario podía generar, visualizando la información que iba parametrizando

Se recalcó la importancia de las pruebas unitarias y la cobertura que estas brindan al inicio del proyecto, y como se tenía un mínimo de 70% de cobertura, para el caso de la solución en los 4 microservicios empleados, se obtuvo una cobertura mayor al 90%, lo cual es un buen indicio de la salud del código, aunque para medir esto, no solo se tuvieron en cuenta las coberturas, sino también los *code smells* que son unas detecciones de malas prácticas de código, donde se mantuvieron en 0 para los 4 microservicios, cumpliendo así con los inicialmente propuesto, una solución limpia y que responda de mejor manera a las necesidades del negocio.

Un aspecto sumamente importante que se logró al implementar la solución propuesta, fue la seguridad que se le dio a los microservicios a través de los componentes de la arquitectura de referencia, teniendo presente temas de seguridad como roles y autenticación, ya que se considera de suma importancia limitar los recursos que se están exponiendo en internet, para así prevenir que un usuario externo al negocio, pudiera consumir los servicios que componen la solución. Para esto se usó un mecanismo de autenticación por roles para los distintos servicios construidos, es decir, cada endpoint iba asociado a un rol de negocio para así controlar que solo las personas con el rol encargado de realizar una operación pudieran consumirlo, y los que no estuvieran en los grupos o roles asignados, no tendrían los privilegios para consultar dicha información. Básicamente los componentes de seguridad usados tienen una filosofía que es aplicar filtros a las peticiones web, de esta manera se puede garantizar que existe una autenticación y con base a esto, se aplican criterios de autorización. Este filtro de autenticación busca proteger las peticiones a los recursos expuesto por un aplicativo, en este caso, los 4 micro servicios que componen la solución, de manera en que solo permite consumo de endpoints a usuarios autenticados con el proveedor de identidad del cliente.

VI. ANÁLISIS

Al analizar los resultados obtenidos con la solución presentada, se evidencia que se logra completar los objetivos propuestos, tanto a nivel de funcionalidades, como la salud del código que propone la empresa cliente con la que deben cumplir los proyectos, lo cual es algo sumamente importante el cumplir con los estándares de calidad propuestos, ya que eso indica que se tiene una aplicación que responde ante muchos escenarios de manera correcta, ya que la salud del código también mide temas como la complejidad, la confiabilidad, y demás, y el tener dichos indicadores de manera positiva generan gran confianza al usuario final para el uso del aplicativo en el día a día, y tener plenitud de que se realizarán los procesos de manera adecuada.

Aunque la cobertura mínima para los proyectos de la empresa cliente fuera 70%, se pensó en la manera en que estas métricas avanzan con el pasar del tiempo, en que van evolucionando y pidiéndole más a los aplicativos, por eso se parte de tener esas métrica por encima de lo mínimo, ya que en el momento en que estas suban, los microservicios construidos estarán en toda la facultad de responder de manera adecuada ante esos cambios de porcentaje requerido. Con esta cobertura, y la implementación de la arquitectura de referencia enfocada a eventos, se concluye que la solución cumple con principios inicialmente propuestos, como lo es la escalabilidad, ya que los microservicios estarán construidos de tal manera en que extenderlos sea de manera natural y fluida, y no con una gran complejidad agregada, con esto se logra las metas iniciales las cuales son el resultado del buen uso de los recursos, la arquitectura de referencia, y de principios y patrones de diseño de software.

El aplicativo se vio sometido ante un grupo de expertos de negocio, los cuales tenían experiencia generando contratos día a día en el anterior aplicativo, los cuales expresaron su confianza en el aplicativo y en que les será de utilidad en la continuación de sus labores gracias a la simplificación del mismo.

VII. CONCLUSIONES

Se alcanzaron de manera efectiva los objetivos planteados para el proyecto, migrando con éxito el aplicativo legado con su lógica de negocio, la cual también fue extendida, y en una arquitectura de referencia orientada a eventos, lo cual brindó mayor fluidez a la hora de los consumos de los servicios, así el usuario final puede tener una mejor experiencia al navegar por el aplicativo.

El éxito de este proyecto se debió al buen trabajo de todos los frentes involucrados, como lo son SCRUM, PO, Team members, Negocio, QA, Desarrollo, Arquitectura, ya que desde cada una de las áreas se logró hacer un buen trabajo lo que permitió la fluidez del mismo proyecto. Otro ítem que facilitó mucho esta solución, fue la claridad que tenía la PO y los team members, sobre el objeto de negocio, ya que con la experiencia que poseían, definir flujos y reglas de negocio se dio de manera muy natural, lo cual se veía reflejado en las historias de usuario bien detalladas y definidas, las cuales eran la herramienta para el equipo de desarrollo, entonces se pudo tener un excelente entendimiento a la hora de desarrollar las reglas de negocio que le competen al aplicativo. También cabe recalcar que la arquitectura de referencia, brindó muchas facilidades para definir muchos flujos a nivel técnico, como lo son las conexiones de los servicios, el aprovisionamiento de recursos, los mecanismos de mensajería y demás, todo esto ayudó a construir una aplicación mantenible y escalable.

La alta cobertura de las pruebas unitarias fue otro criterio importante para poder decir que se lograron las expectativas sobre el proyecto, ya que en el mundo del software todos los temas relacionados con la salud del código juegan un papel muy importante, ya sea para mirar los indicadores de los proyectos y definidor mejoras o planes de acción para mejor estos indicadores mencionados anteriormente, ya que dicen demasiado de un proyecto. El tener presente las pruebas unitarias desde el día 0 fue fundamental para garantizar un buen aplicativo, ya que al cubrir la mayor cantidad de escenarios posibles podemos preparar el proyecto para que responda de la mejor manera y no falle por un escenario no esperado.

Con todo esto anteriormente mencionado se tiene la confianza de que la solución entregada es una solución que cumple todos los criterios y estándares propuestos inicialmente.

VIII. RECOMENDACIONES

Un aspecto importante a tener en cuenta, es que la solución se dividió en cuatro microservicios con responsabilidades separadas, por lo cual la documentación juega un papel fundamental para las personas nuevas que entren al proyecto, ya que al tener tantos microservicios enfocados a una aplicación, puede ser algo difícil de comprender o entender en qué microservicio agregar una nueva funcionalidad, o en qué microservicio revisar algún tema puntual, por eso la documentación es algo que se debe seguir escalando a medida que el aplicativo lo hace, haciéndolo constantemente y de manera organizada, ya que será el instructivo del aplicativo para las nuevas personas que lleguen.

Por otra parte, es necesario seguir profundizando en todas las métricas de calidad que componen la salud del código, ya que sin darnos cuenta al añadir una nueva funcionalidad podríamos estar afectando un indicador, sino no se hace de la mejor manera. Por eso es importante seguir complementando esos conocimientos dentro del equipo de desarrollo para que se tenga más presente lo importante que es la salud del código para los proyectos de software.

REFERENCIAS

[1] IBM ‘‘¿Qué es la migración de aplicaciones?’’ IBM, Accedido:17, Dic. 2023. [En línea]. Disponible en: https://www.ibm.com/mx-es/topics/application-migration

[2] O. Filipova and R. Vilão, “Backend Development” en Software Development From A to Z: A Deep Dive Into All the Roles Involved in the Creation of Software. Berkeley, California, USA: Apress, 2018, cap. 5. p. 101-131. [En línea]

[3] R. C. Martin, “The clean architecture” en Clean Architecture: A Craftsman's Guide to Software Structure and Design. Londres, Inglaterra: Prentice Hall, 2018, cap. 22, pp. 195-201. [En línea]

[4] Mozilla ‘‘Introducción a JavaScript asíncrono’’. MDN Web Docs, Accedido:17, Dic. 2023. [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Learn/JavaScript/Asynchronous/Introducing

[5] D. Roldán Martínez, P. J. Valderas Aranda, y V. Torres Bosch, “Acceso a datos en microservicios. Aspectos de diseño” en Microservicios: un enfoque integrado. Madrid, España: RA-MA, 2018, cap 7, pp. 118-119. [En línea]

[6] J. M. Ortega Candel, “Introducción a las arquitecturas basadas en microservicios” en Tecnologías para arquitecturas basadas en microservicios: Patrones y soluciones para aplicaciones desplegadas en contenedores, 1. ed. Saarbrücken, Alemania: Editorial Académica Española, 2020, cap. 1, pp. 7-9 [En línea]

[7] Amazon “Patrón de CQRS” Amazon Web Services, Accedido:17, Dic. 2023. [En línea]. Disponible en: https://docs.aws.amazon.com/es_es/prescriptive-guidance/latest/modernization-data-persistence/cqrs-pattern.html

[8] J. García Fanjul, C. de la Riva Álvarez, J.R. de Diego Rodríguez y J. Tuya González J. “Técnicas y prácticas en las pruebas del software” en Técnicas Cuantitativas para la Gestión en la Ingeniería del Software, J. Tuya, I. R. Román y J. D. Cosín, Eds. La Coruña, España: Netbiblo, 2007, cap. 3, pp. 43-66. [En línea]

[9] Drew, "Ventajas y desventajas de la metodología Scrum," Blog de Drew. Accedido: 3, dic, 2023. [En línea]. Disponible en: https://blog.wearedrew.co/productividad/-ventajas-y-desventajas-de-la-metodologia-scrum

[10] Amazon “¿Qué es DevSecOps?” Amazon Web Services, Accedido:17, Dic. 2023.[En línea]. Disponible en: https://aws.amazon.com/es/what-is/devsecops/