Migración de módulos en un ERP empresarial basado en Java a frameworks modernos

Mateo Baena Chavarriaga

Informe de prácticas académicas para optar al título de Ingeniero de Sistemas

Asesor interno

Deisy Loaiza Berrio, Ingeniería de sistemas con maestría en ingeniería con énfasis en informática

Asesor externo

Jorge Luis Gutiérrez Moncada, Administrador de sistemas informáticos con especialización en business intelligence

Universidad de Antioquia

Facultad de Ingeniería

Ingeniería de Sistemas

Medellín

2023

Cita	Baena Chavarriaga Mateo [1]
Referencia Estilo IEEE (2020)	[1] M. Baena Chavarriaga. “Migración de módulos en un ERP empresarial basado en Java a frameworks modernos”, Trabajo de grado, Ingeniería de Sistemas, Universidad de Antioquia, Medellín, 2023.

UNIVERSIDAD DE ANTIOQUIA
Vicerrectoría de Docencia

Sistema de Bibliotecas

Rector: John Jairo Arboleda Cespedes.

Decano/Director: Julio Cesar Saldarriaga Molina.

Jefe departamento: Diego Jose Luis Botia Valderrama.

Agradecimientos

A mis padres, a mis amigos, a los docentes, a la universidad, a los malos y buenos momentos, durante el proceso, que han permitido mi desarrollo personal y profesional.

TABLA DE CONTENIDO

RESUMEN 7

ABSTRACT 8

I. INTRODUCCIÓN 9

II. OBJETIVOS 12
- A. Objetivo general 12
- B. Objetivos específicos 12

III. PROBLEMA DE INVESTIGACIÓN 13

IV. MARCO TEÓRICO 14

V. METODOLOGÍA 16

VI. RESULTADOS 17
- A. Desarrollos paralelos al estudio de la migración 17
- B. Proceso del estudio e implementación de un prototipo para la migración 18
- C. Infraestructura objetivo 20

VII. DISCUSIÓN 23

VIII. CONCLUSIONES 24

REFERENCIAS 24

LISTA DE FIGURAS

Fig 1. Diagrama integración con plataforma de firma electrónica……**18**
Fig 2. Proceso de migración de un monolito a microservicios……...**19**
Fig 3. Distribución por capas presente en cada servicio del monolito……...**20**
Fig 4. Arquitectura hexagonal generalizada……...**21**
Fig 5. Infraestructura actual del sistema……**21**
Fig 6. Prototipo de la infraestructura objetivo……**22**

SIGLAS, ACRÓNIMOS Y ABREVIATURAS

DevOps	Development Operations (Desarrollo de software y de Operaciones)
CI	Continuous Integration (Integración continua)
CD	Continuous Deployment (Despliegue continuo)
AWS	Amazon Web Services

RESUMEN

La inmobiliaria Acrecer S.A.S. ha desarrollado, bajo el área de tecnología, un sistema de planificación de recursos empresariales (ERP) sobre el cual basa su negocio y operación. Este sistema ha alcanzado los 11 años desde su creación y no ha sido sorpresa su crecimiento a lo largo de este tiempo, debido a que nuevas funcionalidades, integraciones y soluciones a errores han sido implementadas. Sumado a lo anterior, el uso de una arquitectura monolítica, basada en Java, ha acarreado problemas a la hora de realizar nuevas implementaciones debido al alto acoplamiento con el que cuenta el sistema. En este punto, no es sorpresa que el área de tecnología de la empresa haya considerado migrar la aplicación a estilos arquitectónicos y tecnologías modernas, que permitan actualizar el sistema, así como el garantizar una alta disponibilidad, mantenibilidad, escalabilidad, una fácil adopción de prácticas DevOps y uso de servicios en la nube. Gracias a esta necesidad de evolución, surge la propuesta de migrar el monolito a una arquitectura basada en microservicios, por medio del cual se busca desacoplar cada módulo del sistema en un microservicio basado en la arquitectura hexagonal, haciendo uso de un estilo arquitectónico que permite garantizar las cualidades mencionadas previamente. Lo anterior representa un reto importante para el desarrollador, dada la necesidad de enfrentarse a nuevas formas de implementar lo desarrollado, a acoplarse a nuevos flujos de desarrollo, adaptarse al uso de servicios en la nube y prácticas DevOps.

Palabras clave — **Migración sistema, arquitectura monolítica, microservicios, arquitectura hexagonal.**

ABSTRACT

The real estate agency Acrecer S.A.S. has developed, under the IT area, an enterprise resource planning (ERP) system on which it bases its business and operation. This system has reached 11 years since its creation and its growth throughout this time has been significant, due to the fact that functionalities, integrations and new bug fixes have been implemented. The use of a monolithic architecture, based on Java, has caused problems when making new implementations due to the coupling that the system has. At this point, it is not surprising that the IT area has considered migrating the application to modern architectural designs and technologies, which can update the system, as well as guarantee high availability, maintainability, scalability, easy adoption of DevOps practices and use of cloud services. Thanks to this need for evolution, the proposal to migrate the monolith to an architecture based on microservices arises, through which it seeks to decouple each module of the system in a microservice based on the hexagonal architecture, making use of an architectural style that allows guaranteeing the qualities said above. This represents a significant challenge for the developer, given the need to face new ways of implementing what has been developed, to adapt to new development flows, and to adapt to the use of cloud services as well as DevOps practices.

Keywords — **System migration, monolithic architecture, microservices, hexagonal architecture.**

I. INTRODUCCIÓN

Los sistemas de información son objeto de la evolución del mismo, sea porque se añaden nuevas funcionalidades a estos o porque sufren correcciones a errores o problemas que presenten en determinado momento o, también, debido al cambio en alguna librería, framework o lenguaje sobre el cual están basados, todo esto es independiente a la arquitectura bajo la cual fueron diseñados, en un principio.

Sin embargo, el estilo arquitectónico influye en que tan fácil o difícil será aplicar estas evoluciones o correcciones, así como el uso de buenas prácticas de programación, uso de patrones de diseño, etc. dado que es posible encontrar desarrollos altamente acoplados o con baja cohesión que hagan difícil o imposible aplicar estos cambios mencionados, cuando se tienen un sistema bastante avanzado en su desarrollo. Por tanto, no es de esperar que los equipos encargados de estos sistemas tengan en cuenta la palabra "migración" a la hora de querer dar un paso hacia adelante con el sistema, con el fin de facilitar todo lo anteriormente mencionado y lograr una evolución, a nivel tecnológico en el mismo.

No es un misterio que los sistemas empresariales han basado su diseño en arquitecturas monolíticas que permitieran incorporar bajo una misma tecnología una solución a sus necesidades, con el problema de ser soluciones cuyos módulos se encuentran altamente acoplados y que con la aparición de las prácticas orientadas a DevOps, demuestran una difícil integración **[1]**. La evolución y aparición de nuevos diseños y estilos arquitectónicos, así como propuestas brindadas por distintos frameworks, han permitido que se piense en desarrollar sistemas flexibles y escalables que permitan un ciclo de desarrollo rápido, lo cual no ofrece una arquitectura monolítica **[2]**, y que ha llevado a que algunas compañías encuentren como alternativa, para modernizar sus sistemas, el realizar migraciones a estas nuevas arquitecturas, ecosistemas de tecnologías y frameworks, de tal manera que se logre tener sistemas de mantenibles, escalables, con bajo acoplamiento y alta cohesión. Sin embargo, la migración de sistemas no es la única solución para la todos los casos de uso, dado que este proceso es extenso, difícil y costoso, por lo que la mayoría de empresas opta por no migrar sus sistemas **[3**] debido a que resultaría más costoso migrar todo un sistema a mantenerlo tal como está.

Entonces, el presente documento propone dar a conocer el proceso de migración de los módulos del ERP sobre el cual se basa la operación de negocio de la empresa del sector inmobiliario Acrecer S.A.S. La participación en el proyecto consiste en migrar unos módulos específicos del actual sistema (Puntualmente sobre los módulos de personas, inmuebles, contratos, facturación, recibos de caja y egresos) a una implementación actual y moderna basada sobre una arquitectura de microservicios que permita consolidar a este como un producto que pueda ser comercializado a otras empresas del sector. Todo esto sobre una arquitectura ya especificada, un levantamiento de requisitos, un acercamiento al framework que será usado, así como la base sólida del sistema con el que se cuenta.

La migración del sistema actual adquiere sentido para la empresa Acrecer S.A.S. debido a que se cuenta con un sistema con más de 11 años de vida y el cual en su momento no contempló el crecimiento, desarrollo, funcionalidades e integraciones con las que cuenta actualmente. Este sistema está basado en una arquitectura monolítica, con el uso de Java en su versión 11 y que se encuentra desplegado sobre una infraestructura en la nube que intenta implementar prácticas DevOps, por medio del uso de un pipeline de CI/CD en Jenkins. Para la compañía no ha sido misterio que el sistema actual, aunque completo, se encuentra altamente acoplado y que, a pesar de estar modularizado, denota limitaciones a nivel arquitectónico con el continuo crecimiento que ha tenido y la demanda de nuevas funcionalidades. El área de sistemas de la compañía, mediante un estudio de tecnologías, paradigmas, tendencias actuales de desarrollo (orientadas al tipo de sistema con el se cuenta) y hasta de negocio, han encontrado que el camino a la migración, para la evolución del sistema, es factible no solo para la compañía sino para éste como producto.

Como se mencionó antes, esta migración representa un reto considerable debido a las limitantes que se pueden encontrar a nivel de diseño e implementación, por tanto es pertinente realizar un estudio previo de las necesidades y requerimientos que deben cumplirse o superarse por medio de este esfuerzo. Sin embargo, el reto también involucra al desarrollador que debe enfrentarse a la solución de problemas que resulten de la migración, así como la adaptación de funcionalidades ya diseñadas a nuevos entornos con nuevas tecnologías o frameworks que limiten la implementación misma, lo cual hace de esto un reto tentativo a la hora de enfrentarse a nuevas maneras de encontrar soluciones y que además provee un conocimiento amplio de las tecnologías o framework mismo con el cual se está trabajando. Todo lo anterior también supone un esfuerzo económico considerable dada la infraestructura que se requiera y del esfuerzo operativo y de mano

de obra necesaria para el proceso, lo que reafirma el porqué no todas las compañías toman este camino para la evolución o mejora de sus sistemas.

II. OBJETIVOS

A. Objetivo general

Migrar los módulos de una aplicación monolítica basada en Java a una arquitectura de microservicios.

B. Objetivos específicos

- Implementar requerimientos solicitados para la corrección de errores y bugs presentes en el sistema actualmente.
- Desarrollar requerimientos funcionales orientados a la integración del sistema actual con un servicio de firma electrónica.
- Diseñar y desarrollar un prototipo de módulo migrado a microservicio, con el uso de Java 11 y el framework Spring Boot, bajo la arquitectura hexagonal.

III. PROBLEMA DE INVESTIGACIÓN

Tal como se ha mencionado, los sistemas basados en una arquitectura monolítica, en algún momento de su vida, evidenciaron los problemas inherentes a este estilo arquitectónico. La adición de nuevas funcionalidades se hace cada vez más compleja debido al alto acoplamiento que puede lograrse a nivel de código y dependencias. También, se evidencia la dificultad de no poder garantizar una alta disponibilidad debido a lo controlados que deben ser los ciclos de despliegue de los cambios (o arreglos) del sistema mismo. Por tanto, es mediante una migración donde se busca mejorar las características con las que debe contar un sistema para poder garantizar el uso mismo de este, así como su continuo mejoramiento o desarrollo de nuevas funcionalidades. Adicionalmente, una migración a microservicios sería la respuesta natural al problema actual.

Actualmente la migración cuenta con varios retos a abordar. Principalmente el de finalizar integraciones, desarrollos y solución de errores pendientes en algunos módulos con los que cuenta el sistema actualmente. Así mismo el lograr adquirir el conocimiento necesario para trabajar bajo el nuevo estilo arquitectónico planteado, y de la misma manera las tecnologías y frameworks involucradas.

Para lograr el objetivo, se diseñó y desarrolló una de las integraciones pendientes del sistema y de la misma manera se solucionaron algunos errores evidenciados en algunos módulos del mismo. También, de manera paralela, se participó en el diseño y desarrollo de un prototipo de proyecto que evidencie la migración de un módulo de la aplicación, con el uso de Java 11, Spring Boot y el uso de la arquitectura hexagonal en el diseño del microservicio.

IV. MARCO TEÓRICO

Las compañías buscan continuamente que los procesos automatizados que rigen su negocio sean cada vez más rápidos y precisos que los procesos manuales, debido a la facilidad y eficiencia con la que se ejecutan los mismos, de tal manera que estos sean transparentes para el funcionamiento de la empresa y que estos esfuerzos sean aplicados en otros procesos de difícil automatización o en otras áreas del negocio que no requieran una automatización. Bajo este panorama es que, en los años 60, nacen las aplicaciones de negocio computarizadas, las cuales fueron pioneras en automatizar los procesos de negocio de las compañías, por medio del uso de computadoras centrales que recibían solicitudes de operaciones y que retornaban resultados de las mismas en función de los parámetros enviados **[4]**. Sin embargo, a pesar de lograr automatizar parte del funcionamiento de la compañía, estas aplicaciones eran costosas y limitadas, lo que llevó a que este tipo de aplicaciones expandiera su desarrollo y que se pensara en la inclusión de módulos que integran otras funcionalidades que pudiesen satisfacer la operación de las compañías, de tal manera que estos sistemas (para la década de los 80) ya incluian módulos de contabilidad, finanzas, ventas, compras, inventarios, etc. Finalmente, con el aumento de las velocidades de internet, la transformación digital, la aparición de nuevas herramientas de desarrollo y de tecnologías se ha logrado que estos sistemas en la actualidad puedan permitir a una compañía basar su operación en estos mismo (Conocidos actualmente como suites ERP), con la capacidad de operar sus procesos de manera más eficiente **[4]**.

En la actualidad, las compañías usualmente recurren a adquirir alguna suite de ERP que les permita acceder a los beneficios de tener un sistema de este tipo, sistemas como SAP, Microsoft Dynamics, Oracle E-Business Suite y muchos más son ejemplos de estos. Algunos beneficios de usar este tipo de sistemas va desde quitar la responsabilidad de desarrollar un sistema desde cero, cuando ya hay productos desarrollados, hasta el hecho de no pensar en darle mantenibilidad a estos cuando es el mismo proveedor quien ofrece este servicio. Sin embargo, estos sistemas, a pesar de ser completos, no son baratos y muchas compañías quizás no logren explotar en su totalidad la funcionalidad que estos les ofrecen, por lo que algunas optan por desarrollar sistemas que se acercan más al caso de uso y requerimientos que éstas necesiten.

A nivel de diseño de software, no es sorpresa que una compañía, que decide desarrollar su propio sistema empresarial, opte por una arquitectura monolítica. Este estilo arquitectónico,

permite que un sistema sea simple en su desarrollo, testeo, despliegue y escalamiento **[5]** debido a que este plantea que un sistema sea compilado como una unidad unificada, cosa que evidenciamos en las aplicaciones Java donde tanto su frontend como backend es empaquetado (como uno solo) sobre un archivo .EAR. En un principio, para una compañía, este estilo arquitectónico puede ser válido debido a que en una etapa inicial los requerimientos y casos de uso son específicos y con un alcance definido. Sin embargo, con el hipotético crecimiento de una compañía, este sistema podría verse limitado en un futuro, lo que llevaría al diseño e implementación de nuevos módulos o integraciones que permitan expandir la funcionalidad inicial del sistema. Es aquí donde este estilo arquitectónico se ve limitado debido a que la adición de nuevos módulos o funcionalidades va a llevar a que se empiece a evidenciar un alto acoplamiento entre el código base y las dependencias que funcionan con este, lo cual complica el desarrollo del mismo **[4]**. A nivel de despliegue, con el crecimiento del código base y del tamaño del sistema, se evidencia un mayor tiempo de despliegue y también la necesidad de parar la operación del mismo debido a que este debe realizarse sobre todo el sistema, lo cual impacta directamente la disponibilidad del servicio y que lleva a que este proceso sea planeado, controlado y no realizado de manera frecuente **[5]**.

Con el fin de que un sistema de este tipo no quede estancado en su desarrollo y que no se continúe incrementando el código fuente, así como los problemas subsecuentes, se introduce el término de migración. Este proceso permite que un sistema pueda ser implementado bajo un nuevo estilo arquitectónico, versión de lenguaje o que sus dependencias sean actualizadas, lo cual se puede asociar a fin de cuentas a la refactorización del sistema, que no es sencillo, ni se hace sin antes realizar un estudio de factibilidad e investigación que permita conocer los pros y contras de, precisamente, realizarlo **[1]** debido a que no en todos los casos de usos y requerimientos es factible su realización o incluso podría llegar a ser imposible, a tal punto de que un sistema podría ser de tal tamaño, alcance o influencia que este proceso sería inviable **[2]**. En el caso de las aplicaciones monolíticas, este proceso adquiere sentido, luego del estudio ya mencionado, cuando se conoce que otros estilos arquitectónicos podrían otorgar ese desacoplamiento necesario para que nuevas funcionalidades, en un sistema empresarial, no se vuelvan problemas en un monolito. Por ejemplo, usar una arquitectura modular de microservicios podría permitir que, de forma granular, se pueda dividir cada módulo del sistema de manera más efectiva, logrando que se puedan lograr avances de forma independiente sobre cada uno de estos y que los despliegues de los mismos sean menos complejos dada esta independencia **[6]**.

V. METODOLOGÍA

Este proyecto se desarrolla bajo el marco de trabajo SCRUM, donde se busca un desarrollo ágil y participativo con los clientes y agentes del ámbito de negocio, que en conjunto con la ejecución de distintas ceremonias, como dailys, plannings, reviews y retrospectives, busca tener un seguimiento continuo de los objetivos que se van planteando para el proceso de migración o de continua mejora o corrección del sistema.

Cabe destacar que el sistema empresarial con el cual opera Acrecer S.A.S. será la base sobre la cual se logre la migración y también será la base para un producto que pueda comercializarse como suite de ERP para compañías del sector inmobiliario. Por tanto, en un principio, es necesario corregir en el sistema actual algunos errores encontrados en módulos del mismo sistema, así como el desarrollar nuevas integraciones que requiere el sistema por medio de la compañía.

Teniendo ya un mínimo producto viable para Acrecer S.A.S., los esfuerzos pueden ser llevados al proceso de migración sobre la base del proyecto actual, los cuales resultan de un estudio mencionado previamente. Sin embargo, a pesar de tener diseños y requerimientos a cumplir, este proceso (a la fecha de entrega de este documento) aún no ha empezado debido a que otras responsabilidades han adquirido mayor prioridad.

VI. RESULTADOS

A. Desarrollos paralelos al estudio de la migración

Previo a la migración, recordando que el sistema a la fecha de la redacción de este documento, cuenta todavía con desarrollos y correcciones pendientes, debido a que en la búsqueda de la comercialización del mismo se buscaba tener un producto base. Se destaca, particularmente, una implementación mayor que consistia en la integración del sistema con una plataforma de firma electrónica automatizada, desde cero. Esta asignación contemplaba varias semanas de desarrollo, debido al esfuerzo estimado durante las ceremonias de planning, que se llevaron a cabo, y sobre las cuales fue necesario sumar esfuerzos basados en las retroalimentaciones de los actores del negocio, las cuales llevaron a que este fuera terminado a finales del año 2022, como producto mínimo viable.

La integración consistio en consumir unos servicios REST, de una plataforma que permite crear flujos en la firma de documentos, dentro de los cuales se tienen actores que firman estos, en un orden establecido por el negocio, de tal manera que se pudiese automatizar este proceso dentro del sistema en lo casos que se deseara (osea, contratos, que lo permitieran). En términos generales, el funcionamiento de la integración se puede evidenciar en la **Fig. 1**.

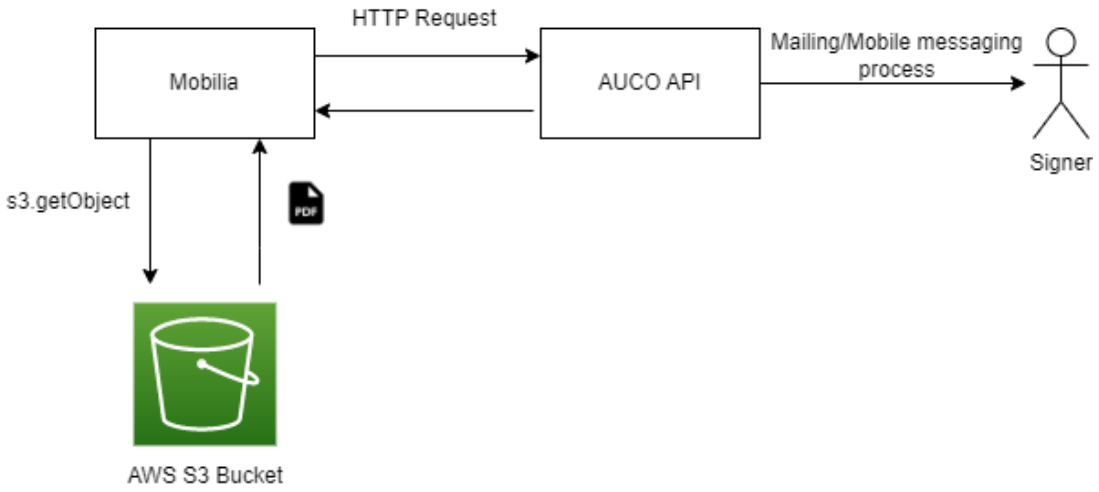

Fig 1. Diagrama integración con plataforma de firma electrónica.

En general, la integración con la plataforma de firma electrónica solicita, por medio del sistema, plantillas de documentos que son albergados en buckets de Amazon S3, el cual es un servicio de AWS para el almacenamiento de archivos. Es necesario tener estos archivos debido a

que posteriormente son codificados en base64 porque así lo solicita la plataforma. Posteriormente, se construye la consulta que permite iniciar el proceso de firma con las personas que deban ser involucradas (bajo unas reglas de negocio), de tal manera que se envíe una solicitud de firma del documento y que se tenga un conocimiento del estado de firma del mismo.

B. Proceso del estudio e implementación de un prototipo para la migración

No existe una receta única para realizar la migración de un sistema. Como se mencionó anteriormente, este proceso puede ser difícil y extenso según la cantidad de código base a migrar, e incluso este puede estar sujeto a alguna refactorización. Por medio de la **Fig. 2** se muestra la arquitectura inicial del sistema y a la que se espera llegar. La arquitectura basada en microservicios que se muestra sigue la analogía propuesta por Lewis y Fowler, donde se muestra que los servicios encapsulados en la capa del negocio son divididos, modulo por modulo, en microservicios **[7]** que permitan el desacople mismo de esta capa. Por medio de esta división se busca tener un microservicio por cada módulo con el que cuenta la aplicación actualmente, lo cual llevaría a tener relaciones o dependencias entre microservicios.

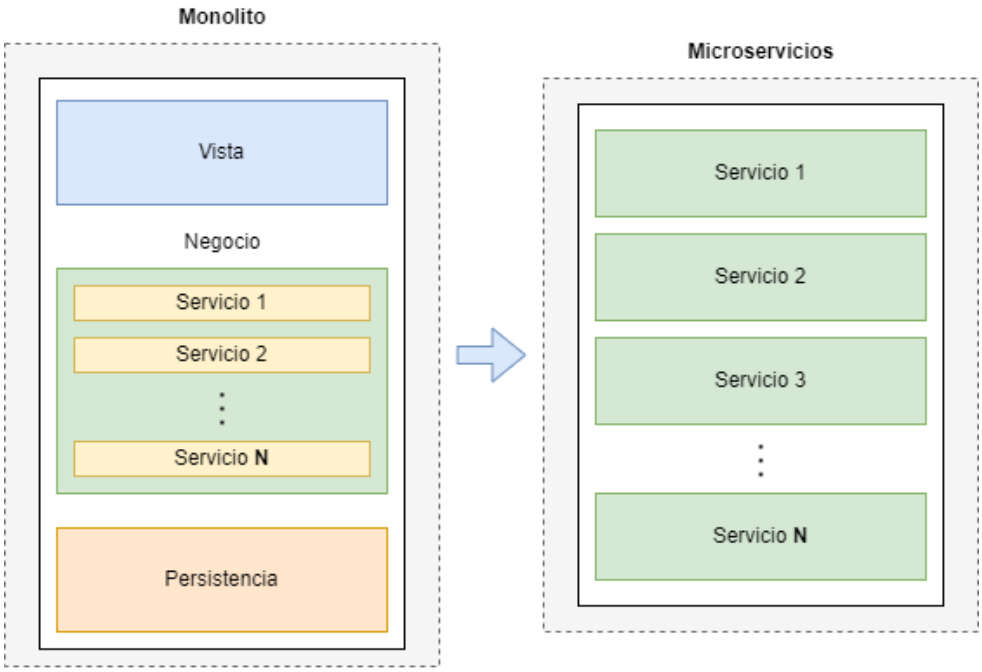

Fig 2. Proceso de migración de un monolito a microservicios.

Actualmente el monolito cuenta con una capa del negocio que divide el flujo de trabajo en tres capas: Servicios, Manejadores y Repositorios. Los cuales logran desacoplar la lógica misma de los servicios hasta cierto punto, esta arquitectura de 3 capas la evidenciamos en la **Fig. 3**.

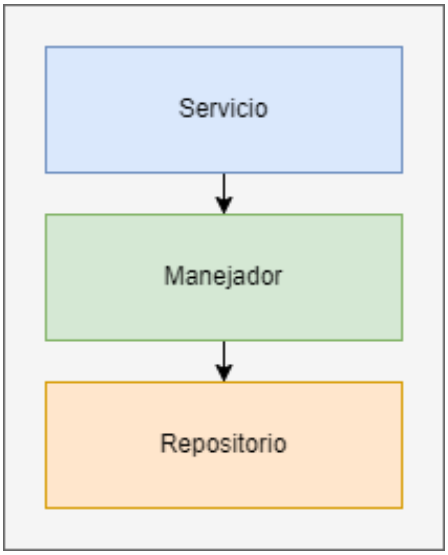

Fig 3. Distribución por capas presente en cada servicio del monolito.

Ahora, para cada microservicio de la aplicación migrada se buscó un estilo arquitectónico moderno, tal como lo propone Lewis y Fowler **[7]**, que también garantizara las cualidades que se buscaban y que adicionalmente permitieran una fácil integración a servicios externos como servicios de mensajería, consumo a servicios REST y conexiones a bases de datos, dado que el sistema actualmente hace uso de estos, por lo que para esto se decidió aplicar la arquitectura hexagonal como estándar para cada microservicio, esto lo podemos evidenciar en la **Fig. 4**. En esta figura encontramos como cada implementación necesaria para una integración con servicios externos es lograda con el uso de adaptadores y puertos, los cuales permiten lograr implementaciones unitarias y de fácil integración al sistema que se tenga, en el caso de la figura, las implementaciones externas sólo son de carácter ilustrativo.

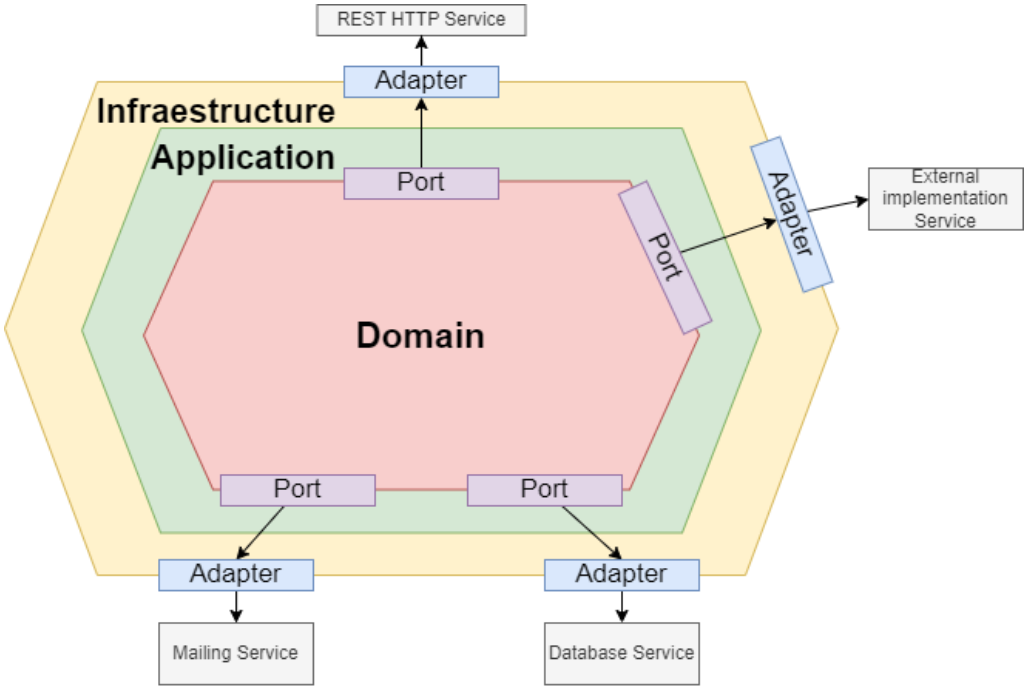

Fig 4. Arquitectura hexagonal generalizada.

C. *Infraestructura objetivo*

La discusión de qué decisión tomar en términos de infraestructura es otro tema que ha sido crucial en el estudio de la migración. Actualmente el sistema se encuentra desplegado en un infraestructura de AWS que, resumidamente, se evidencia en la **Fig. 5**.

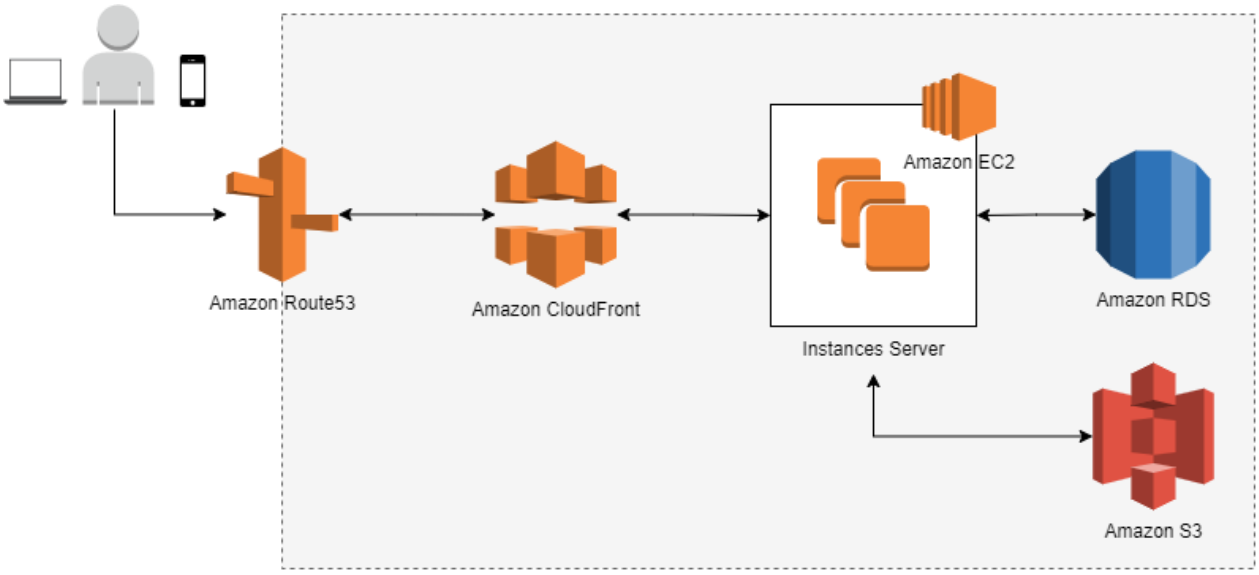

Fig 5. Infraestructura actual del sistema.

En su momento esta infraestructura fue creada de manera empírica por antiguos integrantes del área, sin embargo, hasta el día de hoy esta sigue siendo usada. Como tal, el sistema actualmente no ha presentado falencias asociadas a no garantizar una alta disponibilidad, pero el apropiarse al uso de otros servicios del proveedor de la nube lleva a que se tenga en cuenta un aspecto muy importante del estilo arquitectónico al cual se desea migrar. Este aspecto corresponde a la entrega de un servicio bajo demanda de servicios unitarios, que a diferencia del actual, no necesita permanecer activo incluso cuando no es usado. Lo que llevó a que se pensara en una infraestructura orientada al uso de Lambdas, que en términos generales permiten encapsular servicios (sin importar el lenguaje) de funcionalidades puntuales del sistema y que son consumidos bajo demanda. Esto lo evidenciamos en la **Fig. 6**.

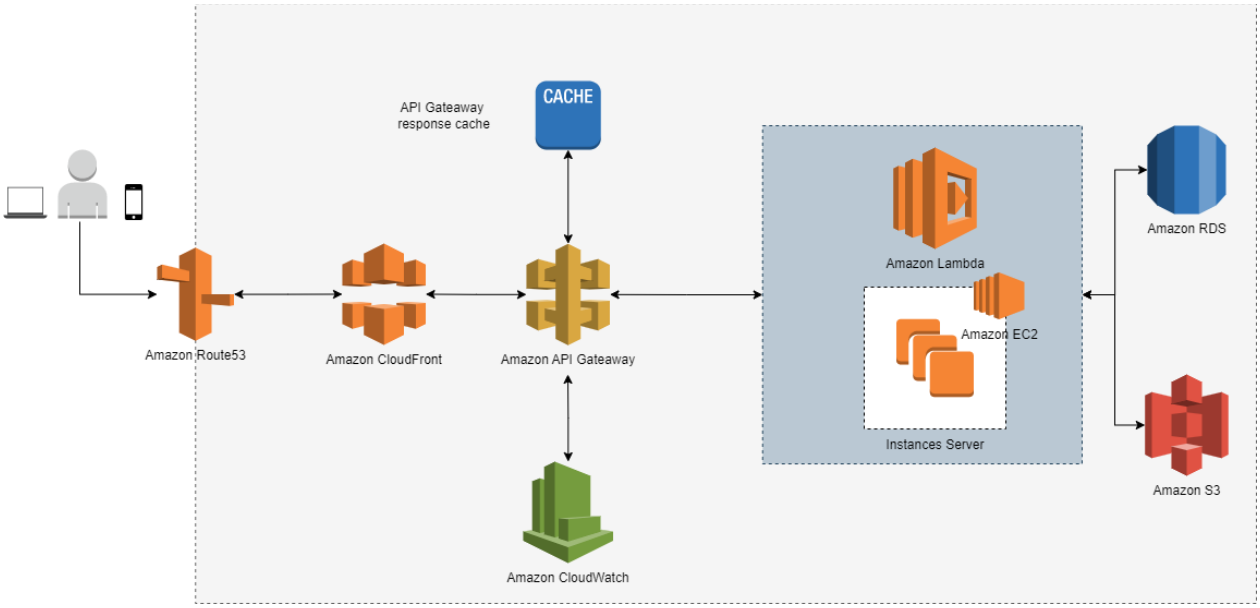

Fig 6. Prototipo de la infraestructura objetivo.

Cabe destacar que no todo el sistema actual se podría migrar a Lambdas debido a que la mayoría de procesos del negocio son funcionalidades extensas que por la lógica de negocio misma no podrían ser compactados. Por tanto, en el uso de estas Lambdas se busca paralelizar a las instancias que se tendrá por microservicio, microservicios que serán contenerizados por lo que se considera su gestión con algún servicio como EKS o ECS. Debido a que el sistema busca ser un producto comercial, lo que se ha pensado por ahora es tener instancias por clientes, debido a que

de la misma manera se tiene planteado el producto actual (sin migración). Por ahora, claramente quedan pendientes otros temas asociados a la escalabilidad misma de la infraestructura así como entrar en detalle en el uso del aprovechamiento del caché que ofrece AWS, sin embargo esto queda para su futuro planteamiento.

VII. DISCUSIÓN

Se diseñó y desarrolló un prototipo de proyecto migrado en base a la aplicación monolítica con la que se cuenta. En este ejercicio se puso a prueba el nivel técnico y teórico necesario para realizar el proceso de migración así como la realización de pruebas funcionales y unitarias que permitieran validar lo realizado, lo cual deja el camino abierto para que se realicen pruebas de integración y end-to-end (cuando se tenga listo el frontend de la aplicación), en este apartado también se contempla la base de datos correspondiente al microservicio, la cual cuenta con información propia de este módulo.

El módulo migrado para la prueba corresponde al módulo de personas, el cual permite crear usuarios, establecer roles y permisos a estos, así como generar reportes de los mismos. Dentro del sistema, este módulo es importante debido a que es usado por parte del negocio para obtener información de los clientes asociados a arriendos, ventas y temas asociados a proyectos inmobiliarios de la compañía.

En cuanto al producto mínimo viable aún no está terminado dado que es necesario exponer este microservicio a un frontend que pueda consumirlo, sin embargo este aún se encuentra en desarrollo, debido a que el equipo de trabajo ha priorizado otras asignaciones asociadas a la consecución del producto actual.

Por otro lado, aún está pendiente la infraestructura sobre la cual se espera desplegar este microservicio del prototipo, dado que esta no fue posible obtenerla de manera gratuita para su prueba en un entorno de AWS.

Actualmente el proyecto se encuentra en un repositorio de Github y se hace uso de un pipeline basado en Github Actions para probar un flujo de CI que permitiese evaluar la pruebas sobre el repositorio, compilar y buildear un artefacto del mismo. Sin embargo, por falta de infraestructura no se puede completar la parte que corresponde al CD, en el conjunto de prácticas DevOps.

Por medio de la prueba de la migración de un módulo del sistema se logró obtener, con el uso de la arquitectura hexagonal, un microservicio con una alta escalabilidad y mantenibilidad, con un código base reducido, gracias al aprovechamiento del framework.

VIII. CONCLUSIONES

El area de tecnologia de Acrecer S.A.S., a pesar de tener 11 años de vida aún tiene muchas oportunidades de crecimiento en términos de la estandarización de procesos, de aseguramiento de la calidad del software y de documentación que permita crecer aún más al área y que permita crear software de mayor calidad.

La idea de migrar el sistema actual tiene todo el sentido del mundo a nivel técnico debido a los problemas que presenta actualmente este, de la misma manera el equipo se ha esforzado por estudiar teórica y técnicamente para dar el paso y de la misma manera ha buscado consolidar un plan para migrar el proyecto consultando a expertos de otras empresas, leyendo documentación técnica y casos de estudio documentos en repositorios de revistas de investigación.

Dada la naturaleza del sistema (Al ser un ERP) se espera que con la migración no se reduzca la complejidad del negocio o los procesos del negocio pero si la base de código del mismo, así como el garantizar la mantenibilidad y escalabilidad del sistema, con el agregado de tener una alta disponibilidad gracias a la integrabilidad que se espera tener con servicios en la nube.

A nivel personal, el desarrollo de la integración permitió conocer los procesos del negocio, así como las motivaciones de los actores involucrados, de tal manera que este se pudiese automatizar correctamente. Procesos como la abstracción de entidades y flujos de trabajo, el diseño de una solución con las herramientas disponibles, la lectura técnica de los servicios que se iban a consumir y la revisión del sistema mismo, fueron fundamentales para la consecución de la integración. El diseño y desarrollo del prototipo de la migración, más allá de que permitió conocer, estudiar e implementar funcionalidades bajo las reglas que impone el uso de un framework, también permitió darle importancia a los artículos científicos asociados a casos de migraciones de sistemas, de tal manera que se pudiera conocer recomendaciones a la hora de realizar una, o casos de uso, limitaciones o consideraciones a tener en cuenta, etc. por lo que no solo se dio importancia a la documentación técnica, la cual fue (y es) igualmente valiosa.

REFERENCIAS

[1] Z. Ren, W. Wang, G. Wu, C. Gao, W. Chen, J. Wei & T. Huang. (2018). Migrating Web Applications from Monolithic Structure to Microservices Architecture. In Proceedings of the 10th Asia-Pacific Symposium on Internetware (Internetware '18). Association for Computing Machinery, New York, NY, USA, Article 7, 1–10. doi: https://doi.org/10.1145/3275219.3275230

[2] F. Ponce, G. Márquez & H. Astudillo.*Migrating from monolithic architecture to microservices: A Rapid Review* [conferencia] 2019 38th International Conference of the Chilean Computer Science Society (SCCC), 2019, pp. 1-7, doi: https://doi.org10.1109/SCCC49216.2019.8966423

[3] Valentina Lenarduzzi, Francesco Lomio, Nyyti Saarimäki, Davide Taibi, Does migrating a monolithic system to microservices decrease the technical debt?, *Journal of Systems and Software*, Volume 169, 2020, 110710, ISSN 0164-1212, doi: https://doi.org/10.1016/j.jss.2020.110710.

[4] P. Patel. (2020). From monoliths to microservices: Modernize your apps now. [Online] Available at: https://developer.ibm.com/blogs/from-monoliths-to-microservices-modernize-your-apps-now/

[5] C. Richardson. (2022). Pattern: Monolithic Architecture. [Online] Available at: https://microservices.io/patterns/monolithic.html

[6] K. Gos & W. Zabierowski. (2020). "The Comparison of Microservice and Monolithic Architecture," 2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH), 2020, pp. 150-153. https://doi.org/10.1109/MEMSTECH49584.2020.9109514

[7] M. Fowler & J. Lewis. (2014). “Microservices a definition of this new architectural term,”[Online] Available at: http://martinfowler.com/articles/microservices.html