RESUMEN

DESARROLLO ÁGIL DEL NUEVO SISTEMA INSTITUCIONAL BASADO EN UNA ARQUITECTURA ORIENTADA A MICROSERVICIOS

por

Abraham Hernández Rivadeneyra

Asesor principal: Alejandro Walterio García Mendoza

RESUMEN DE TESIS DE MAESTRÍA

Universidad de Montemorelos

Facultad de Ingeniería y Tecnología

Título: DESARROLLO ÁGIL DEL NUEVO SISTEMA INSTITUCIONAL BASADO EN UNA NUEVA ARQUITECTURA ORIENTADA EN MICROSERVICIOS.

Investigador: Abraham Hernández Rivadeneyra

Asesor principal: Alejandro Walterio García Mendoza, Maestría en Tecnologías de la Información

Fecha de culminación: Mayo de 2019

Problema

En esta investigacion se pretende dar solucion a la hipotesis planteada, en cuanto a demostrar que es más agil el desarrollo del nuevo sistema institucional basado en una nueva arquitectura de microservicios.

Metodología

En esta investigación, para llevar a cabo la comprobación de que el desarrollo de la nueva arquitectura es mucho más ágil que el anterior, se llevó a cabo la implementación de la metodología de puntos de casos de uso para la obtención del peso que tiene cada módulo. Después del peso obtenido, se necesita saber las horas que

tuvo cada desarrollo, el cual es obtenido por medio de bitácoras donde se describen las horas realizadas.

Resultados

Se llevó la implementación de esta arquitectura en las inscripciones del mes de enero, lo cual comprueba su funcionalidad. A partir de esa fecha, cualquier pago realizado de los servicios de la institución fue realizado por medio del nuevo sistema. En cuanto al desarrollo con una arquitectura en microservicios, se obtuvo que es más ágil que el anterior.

Conclusión

El resultado obtenido en esta investigación comprueba que por medio de la implementación de esta nueva arquitectura se puede agilizar el desarrollo de los módulos faltantes al sistema institucional o de cualquier sistema nuevo.

Universidad de Montemorelos

Facultad de Ingeniería y Tecnología

DESARROLLO ÁGIL DEL NUEVO SISTEMA INSTITUCIONAL BASADO EN UNA ARQUITECTURA ORIENTADA A MICROSERVICIOS

Tesis
presentada en cumplimiento parcial
de los requisitos para el grado de
Maestría en Ciencias Computacionales

por

Abraham Hernández Rivadeneyra

Mayo 2019

DESARROLLO ÁGIL DEL NUEVO SISTEMA INSTITUCIONAL
BASADO EN UNA ARQUITECTURA ORIENTADA
A MICROSERVICIOS

Proyecto

presentado en cumplimiento parcial de
los requisitos para el grado de
Maestría en Ciencias
Computacionales

por

Abraham Hernández Rivadeneyra

APROBADO POR LA COMISIÓN:

M.C. Alejandro Walterio García Mendoza
Asesor principal

Mtro. Carlos Alberto Guzmán Valenzuela
Asesor externo

M.C. Saulo Hernández Osoria
Miembro

Dr. Ramón Andrés Díaz Valladares
Director de Posgrado e Investigación

M.C. Daniel Arturo Gutiérrez Colorado
Miembro

3 de mayo de 2019
Fecha de aprobación

DEDICATORIA

Primero que nada, agradezco a mi Dios por esta oportunidad de aprender cada vez más y llegar hasta donde me encuentro en este momento.

A mis padres Abrahan Hernández Casados y María Rivadeneyra Antele, por su apoyo incondicional en todo tiempo.

A mi hermana que siempre estuvo dándome ánimos para concluir esta nueva etapa.

A cada persona que me ayudó y aconsejó en el transcurso de esta investigación.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RECONOCIMIENTOS

Agradezco primero a Dios, porque Él es el proveedor de todo el conocimiento, y porque me ha guiado en cada etapa de la vida.

Al mi asesor principal, el maestro Alejandro Walterio García Mendoza, por el tiempo dedicado en la revisión y la dirección de mi tesis.

Al maestro Saulo Hernández Longoria, como asesor secundario en esta investigación.

Al maestro Daniel Gutiérrez, como coordinador del postgrado en cuanto a la orientación presentada durante el curso de la maestría.

Al ingeniero Omar Soto, que me apoyó incondicionalmente durante el desarrollo del proyecto.

A cada persona que tomó parte de su tiempo en ayudarme con la revisión del documento.

A cada uno de los maestros que me impartieron clases, una parte fundamental de mi desarrollo en esta nueva etapa.

A cada uno de mis compañeros de la maestría, ya que me acompañaron en esta etapa de mi vida.

A mis padres y mi hermana, que me apoyaron y me motivaron a trabajar arduamente y a concluir este trabajo de la mejor manera.

CAPÍTULO I

ANTECEDENTES DEL PROBLEMA

Planteamiento del problema

Actualmente, el proceso de desarrollo que se lleva a cabo en la Dirección de Tecnología Informática de la Universidad de Montemorelos corresponde a una arquitectura tradicional o también conocida como monolítica, la cual está compuesta por módulos que son adaptados de acuerdo con los requerimientos que van siendo necesarios día con día. Esta arquitectura no facilita la rápida construcción de una aplicación y vuelve cada vez más difícil el mantenimiento de dicho desarrollo.

Se pretende dar solución en cuanto a demostrar que es más ágil el desarrollo del nuevo sistema institucional basado en una nueva arquitectura orientada a los microservicios.

Declaración de problema

El problema a investigar en este estudio fue el siguiente:

La Universidad de Montemorelos cuenta con una arquitectura monolítica dentro de los sistemas actuales. En lo que se ha podido observar, esta proporciona una fácil implementación y una escalabilidad limitada dentro de los márgenes del desarrollo ya existente.

Entre las desventajas de este tipo de implementación se encuentran las siguientes:

1. Arquitecturas que generan aplicaciones grandes y eso las hace complejas al momento de querer realizar algunos cambios o mejoras.

2. La necesidad de implementar toda la aplicación para cada una de las actualizaciones.

3. Al momento de escalar, se pueden presentar algunos conflictos por tratarse de módulos distintos.

4. Es una barrera querer adoptar nuevas tecnologías dentro de las implementaciones actuales.

Por medio del presente estudio se quiere demostrar lo ágil de un nuevo tipo de desarrollo, así como el proceso que se lleva para la creación de una arquitectura orientada a microservicios, mostrando cómo se desarrolla la implementación en un módulo del sistema institucional de la Universidad de Montemorelos, tomando en cuenta las siguientes características para una buena implementación y un mejor funcionamiento:

1. La construcción de los servicios debe ser pequeña.

2. Debe permitir la escalabilidad del sistema de manera independiente, sin afectar el funcionamiento de todo el sistema.

3. Debe hacer que la funcionalidad de cada servicio sea única.

4. Debe poder implementar tecnologías nuevas o de cualquier tipo.

Justificación

La Universidad de Montemorelos requiere de una arquitectura robusta, moderna, que cumpla con estándares internacionales actuales y que agilice la implementación del sistema institucional. El montar el nuevo desarrollo sobre esta arquitectura le permitirá a la dirección de tecnología informática generar software de una manera

más ágil y con una mayor capacidad de adaptación, en comparación con desarrollos actuales. Además, se pretende que la dirección de tecnología informática de la Universidad de Montemorelos llegue a ser un proveedor de soluciones de la Iglesia Adventista del Séptimo Dia (IASD) en México, orientadas bajo este tipo de esquema de desarrollo.

Objetivos

Para llevar a cabo el desarrollo y la implementación de una arquitectura basada en microservicios se plantearon ciertos objetivos a cumplir, estos están divididos en objetivo general y objetivos específicos.

Objetivo general

Para la presente investigación se planteó el siguiente objetivo general:

Dejar establecida de manera global la arquitectura del nuevo sistema institucional en la Universidad de Montemorelos e implementarla en el módulo de caja, demostrando que el desarrollo de la arquitectura de microservicios es más ágil que la arquitectura tradicional con la que se cuenta actualmente en la Universidad de Montemorelos.

Objetivos específicos

A continuación se plantean los siguientes objetivos específicos:

1. Definir las capas con las que cuenta la arquitectura de microservicios del sistema institucional.

2. Determinar las tecnologías con las que se van a desarrollar cada una de las capas.

3. Definir la administración de los microservicios.

4. Diseñar el esquema para la distribución de los microservicios.

5. Demostrar que el nuevo desarrollo basado en una arquitectura de microservicios es más ágil que el desarrollo anterior.

6. Realizar la implementación de la arquitectura de microservicios en el módulo de caja del sistema institucional.

Hipótesis

En esta investigación se plantea la siguiente hipótesis:

Es posible crear una arquitectura orientada a microservicios que implemente estándares internacionales y que agilice el desarrollo del sistema institucional de la Universidad de Montemorelos.

Limitaciones

Algunas limitaciones de esta investigación fueron las siguientes:

1. Las tecnologías utilizadas para el desarrollo corresponden a versiones community.

2. La planeación del tiempo para cumplir con la elaboración de los módulos en las fechas que indica la dirección de sistemas.

3. El control de la decisión que debe tomar la dirección de sistemas para permitir la implementación de la arquitectura como tal.

4. La habilidad que deben tener las personas en el uso de las herramientas que se utilicen.

Delimitaciones

Se presentan algunas delimitaciones en esta investigación:

1. Se trabajo dentro del contexto de la Universidad de Montemorelos.

2. La arquitectura de microservicios fue pensada de acuerdo con los requisitos específicos del módulo de caja de la Universidad de Montemorelos.

3. Es una arquitectura basada en las tecnologías de Mulesoft y Express.

Definición de términos

De acuerdo con la literatura revisada, se definen los siguientes términos:

Arquitectura orientada a servicios (SOA): es la utilización de servicios para dar soporte a los requisitos del negocio. Permite la creación de sistemas altamente escalables que reflejan el negocio de la organización.

Escalabilidad: es la capacidad de adaptación que tiene un software ante el cambio de circunstancias o tecnologías nuevas (Khazaei et al., 2017).

Backend: es la parte de datos del software que no es accesible para el usuario; encargado de contener toda la lógica, el manejo de datos y el acceso a los distintos tipos de servidores (Aerts, Cailliau, de Groote y Noterman, 2016).

Middleware: es una aplicación encargada de proporcionar infraestructura necesaria para respaldar la construcción de aplicaciones sofisticadas e inteligentes, permitiendo el intercambio de información entre aplicaciones (Del Esposte, Kon, Costa y Lago, 2017).

API: siglas correspondientes a Application Programming Interface.

REST: siglas correspondientes a Representational State Transfer.

HTTP: siglas correspondientes a HyperText Transfer Protocol.

JSON: siglas correspondientes a JavaScript Object Notation.

Ejabberd: es un servidor de mensajería en tiempo real (Balalaie, Heydarnoori y Jamshidi, 2016).

IoT: siglas correspondientes a Internet of Things

Seneca: conjunto de tecnología para escribir microservicios y organizar la lógica de negocios de su aplicación (Lv y Wang, 2016).

BPM: siglas correspondientes a Business Process Management.

Contenedor: es una instancia de una imagen Docker. Un contenedor representa la ejecución de una sola aplicación, proceso o servicio (De la Torre, Wagner y Rousos, 2019).

Docker: es una plataforma de código abierto para el despliegue de aplicaciones dentro de contenedores de software, que envuelve una pieza de software en un completo sistema de archivos que contienen todo lo necesario para ejecutar (Vigneshwaran Sudalaikkan, 2016).

CAPÍTULO II

MARCO TEÓRICO

Introducción

La utilización de un sistema puede ser objetivo crítico dentro de una empresa, ya que puede ser de gran beneficio o perder la ventaja competitiva o, incluso, no ser capaz de sobrevivir si no funciona correctamente (Mens, Magee y Rumpe, 2010).

Se puede describir un sistema como un conjunto que está compuesto por una gran variedad de componentes conectados entre sí, que son de vital importancia para el correcto funcionamiento del sistema (Sangwan, 2015).

En el desarrollo de software, los lenguajes de programación son una parte fundamental, ya que están en un cambio constante con el propósito de mejorar la robustez y mejorar la reutilización de código, buscando un enfoque hacia la distribución, modularización y un acoplamiento más flexible (Dragoni et al., 2017).

Arquitectura de software

Bass, Clements y Kazman (2013) definen la arquitectura de software de un programa o sistema informático como la estructura o estructuras del sistema, que comprende elementos de software, las propiedades visibles externamente de esos elementos y las relaciones entre ellos.

Gorton (2011) comenta que una arquitectura debe diseñarse para cumplir con los requisitos y limitaciones específicos de la aplicación para la que está destinada.

La arquitectura de software debe tener definido cada uno de los componentes que existen dentro de su propia estructura, que permita apreciar la interacción con el resto del sistema, así como tener una definición clara del diseño utilizado para mejorar su funcionamiento y así cumplir con el propósito determinado (López Hinojosa, 2017).

En la actualidad se está rodeado por una gran variedad de sistemas, cada uno diseñado con una arquitectura que va de acuerdo con su funcionamiento. Seleccionar la correcta es lo que permite cumplir con el propósito para el cual ha sido creado el sistema (Sangwan, 2015).

Esto ha ocasionado que surja la necesidad de un mejoramiento día con día, no tanto en el ámbito de seguridad o estándares comunes, sino en la reutilización de componentes y el mejoramiento de la robustez dentro del código (Dragoni et al., 2017).

Uno de los ámbitos importantes es la creación rápida de componentes, que permitan y faciliten la interacción de ellos para el cumplimiento de los requisitos funcionales y no funcionales del sistema. Dividir la arquitectura en componentes y separarlos por niveles permite una mejor toma de decisiones de manera jerárquica, ya que cada nivel está compuesto por normas que rigen su nivel (López Hinojosa, 2017).

Es importante tener en cuenta que la existencia de dependencias entre componentes obliga a que, al realizar un cambio, se generen otras modificaciones en los demás. Al eliminar esas dependencias innecesarias, los cambios se localizan y no se propagan a través de los demás componentes de la arquitectura (Gorton, 2011).

El Lenguaje descriptivo arquitectónico es una herramienta que ayuda para tener una buena documentación y facilitar la comunicación que debe existir entre el arquitecto y las personas interesadas del proyecto (Mens et al., 2010).

Dentro de la arquitectura de software, otro elemento fundamental es que el arquitecto cuente con ciertas habilidades. Gorton (2011) señala las siguientes:

1. La comunicación, donde el arquitecto debe tener la facilidad de comunicarse con los clientes para permitirse entender bien qué es lo que el cliente requiere y espera de un sistema, ya que él será el intermediario entre el cliente y su equipo de desarrollo.

2. El conocimiento de la ingeniería de software, porque el tenerlo le dará al arquitecto una mayor credibilidad de su trabajo.

3. Un conocimiento amplio en el uso de tecnologías, en especial de las que él maneja. Así como la utilización de herramientas que sean necesarias para llevar a cabo este tipo de desarrollo y el estar actualizado dentro de los estándares de tecnologías, diseños y arquitecturas de software.

4. La gestión de riesgos es un factor que el arquitecto debe de tomar en cuenta y hacerlo saber a los clientes o al personal con el que trabaja para estar atentos a cualquiera de esas circunstancias.

Es necesario tener en cuenta que dentro de esta área se presentarán desafíos en cuanto a una arquitectura con alta complejidad, al cambio de tecnologías en el lapso de desarrollo y a la necesidad de funcionar con otro tipo de sistemas (Strimbei, Dospinescu, Strainu y Nistor, 2015).

Arquitectura de microservicios

¿Que son los microservicios? Es el desarrollo de una aplicación compuesta por una gran cantidad de componentes pequeños, independientes, ligeros y con una funcionalidad única (López Hinojosa, 2017). Para la invocación de estos microservicios es necesario una API HTTP REST (MuleSoft, 2019).

En la Figura 1 se puede observar el claro crecimiento en los últimos años que ha tenido el interés de arquitectura basada en microservicios. Todo eso de acuerdo con el aumento de la tecnología, ya que entre más rápido sea, los consumidores quieren sistemas más rápidos (Avidan y Otharsson, 2018).

En el desarrollo actual se ve involucrado el uso de procesos orientados a un desarrollo ágil, haciendo que las aplicaciones deban ser más rápidas (Rahman y Gao, 2015). La utilización de microservicios cada vez va ganando gran popularidad dentro del área de desarrollo de software, realizando un cambio desde la forma en que se percibe y se diseña la aplicación (Dragoni et al., 2017)

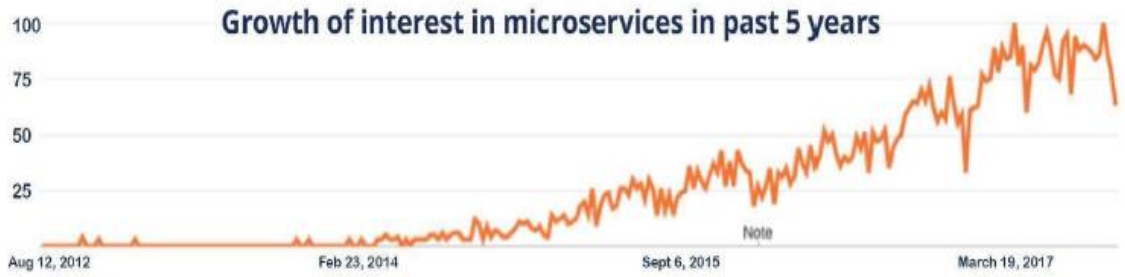


Figura 1. Interest in microservices architecture has grown rapidly since 2012 as a way to solve common problems with application monoliths (Avidan y Otharsson, 2018).

En este tipo de arquitectura es necesario dividir la lógica comercial en pequeños componentes o servicios que contengan estructuras ligeras, permitiendo una integración de los procesos existentes (Oberhauser, 2016). Se busca con esto una creación de microservicios que funcionen de manera independiente a los otros y sean organizados de acuerdo con las funciones que realiza cada uno

(Khazaei et al., 2017). Esta arquitectura es una herramienta que permite simplificar el desarrollo de las aplicaciones, facilitando el desarrollo de una implementación rápida y un mejor mantenimiento de ella. De esta forma, pasa a ser una mejor opción para la gente que desarrolla dichos sistemas (Lv y Wang, 2016). Con la descomposición de una aplicación grande en un conjunto de componentes más pequeños, se permite desarrollar, desplegar, escalar, manejar y visualizar cada uno de ellos de forma independiente (López Hinojosa, 2017). Los microservicios apuntan a la mejora de un óptimo desarrollo, una implementación como tal y a tener la mejor estructura interna dentro del software (Heinrich et al., 2017).

De acuerdo con Mazzara et al. (2018), los principios básicos que rigen el uso de los microservicios son los siguientes:

1. Un contexto delimitado, que consiste en la combinación de funcionalidades que sean iguales.

2. El tamaño, ya que es una característica que distingue a los microservicios donde se pretende crearlos lo más pequeño posible, permitiendo su modificación en cualquier momento y que no afecte la funcionalidad de otros.

3. La independencia, encargada de que el acoplamiento que exista entre cada uno de ellos sea más flexible, donde ellos puedan operar de manera individual.

López Hinojosa (2017) propone cuatro criterios, que son utilizados para realizar la descomposición de microservicios en pequeños componentes; entre ellos se encuentran los siguientes: descomposición por funcionalidades, de acuerdo con su identificación para que encaje y se puedan juntar; descomposición por madurez, encargado de agruparlos dependiendo de sus requerimientos

funcionales y no funcionales; descomposición por el patrón de acceso a datos, basado en el acceso de la recuperación de los datos y descomposición por contexto, realizada a través de servicios que van a la misma entidad.

Nielsen (2015) engloba las características más comunes que deben tener los microservicios, que son las siguientes:

1. Un componente a través de servicios, el cual consiste en la integración de una funcionalidad dentro de los servicios que funciona de forma autónoma.

2. Organización de equipos en torno a las capacidades empresariales, que permite la creación de servicios por equipos diferentes.

3. Productos no proyectos. Este microservicio no se entrega como producto, ya que es utilizado por el equipo mientras sea útil para el desarrollo.

4. Gestión descentralizada de datos, en la que los datos contenidos en cada microservicio son únicos, ya que no permiten el acceso a su información por algún otro medio que no sea el mismo.

5. Capacidad multilenguaje, en que el uso de los microservicios permite la implementación de cualquier tipo de lenguaje de programación, aceptando la integración de distintas tecnologías. En la Figura 2 se observa cómo puede estar compuesto.

6. Despliegues continuos, que pueden ser elaborados y probados por separado.

7. Autocontenido, pueden ser de creación de microservicios autónomos.

8. Comunicación fuera del servicio, que es por medio de protocolos de solicitud y respuestas.

9. Responsabilidad única. En esta parte el microservicio debe contener una única función.

Los autores Avidan y Otharsson (2018) dicen que la integración de un microservicio consta de la utilización de una API, una unidad lógica dentro de la aplicación y tener un almacén de datos. En la Figura 3 se puede observar a detalle cada una de estas partes que componen un microservicio.

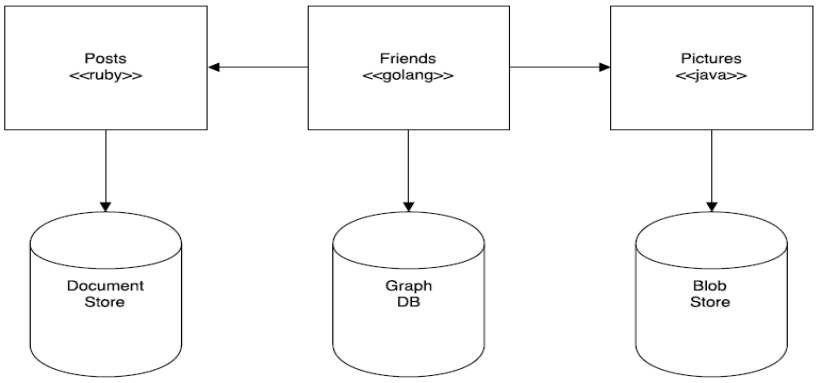


Figura 2. An example of a microservice architecture. Each of the applications' components is independent and written in a different programming language. Standard protocols are used to facilitate communication (Aerts et al., 2016).

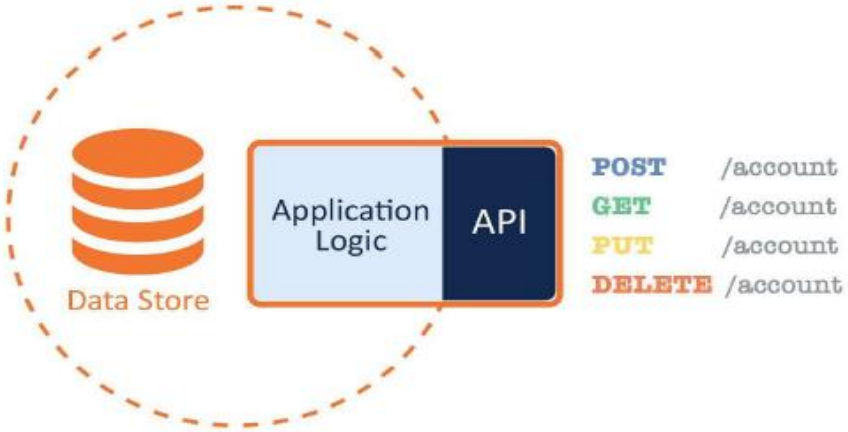


Figura 3. A microservice includes three parts: a data store, application logic, and an API (Avidan y Otharsson, 2018).

Otra característica fundamental de los microservicios es que cuentan con dominios HTTP y REST, lo cual contiene una comunicación ligera. De esta manera, con el paso de tiempo, esta comunicación va sustituyendo el uso de XML por un formato de tipo JSON (Schermann, Cito y Leitner, 2015). La Figura 4 corresponde a una arquitectura de microservicios básica.

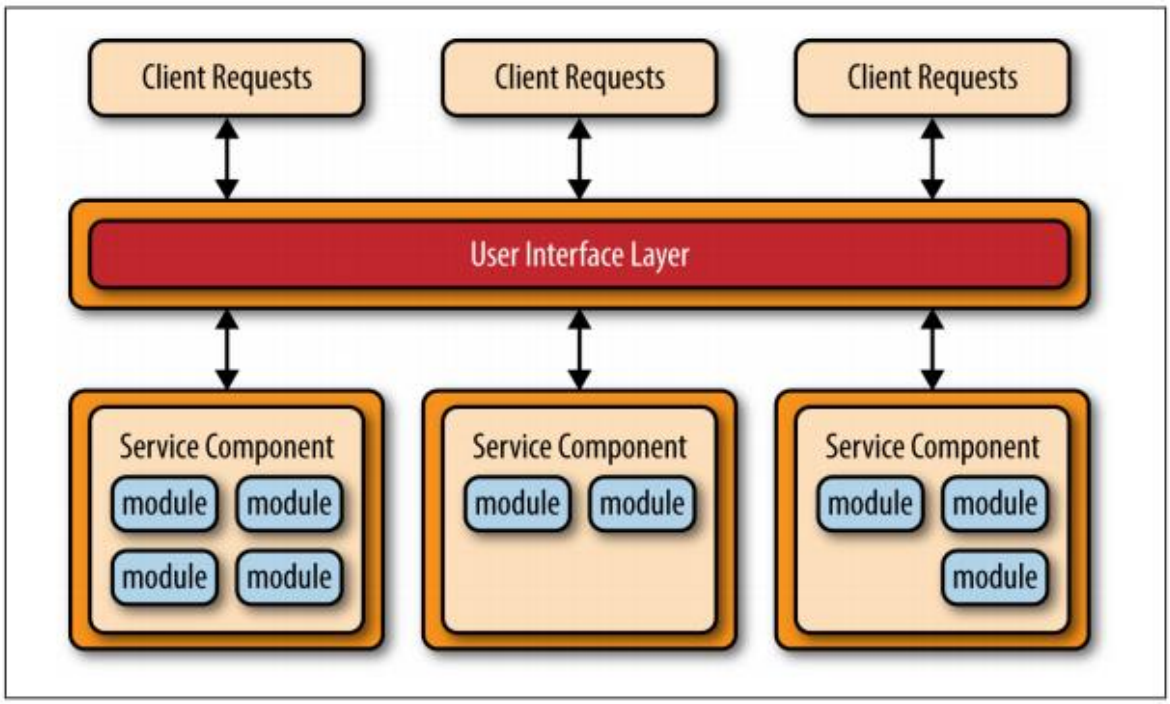


Figura 4. Basic microservices architecture pattern (Richards, 2015).

Escalabilidad

La escalabilidad es una de las características principales que se le atribuyen a este nuevo modelo de arquitectura orientada a microservicios, comparándolo con una arquitectura monolítica, la cual se ve afectada en el momento de llevar a cabo la

implementación. El utilizar este tipo de arquitectura permite escalar de mejor manera, sin presentar tantos inconvenientes. Incluso el uso de algunos principios y técnicas utilizados durante la implementación proporcionan características que la benefician. La automatización, orquestación, el descubrimiento de servicios, el equilibrio de la carga y un claro agrupamiento permiten que el sistema sea escalable (Dragoni et al., 2017).

El escalamiento dentro de una arquitectura monolítica se vuelve complicado ya que se tiene que escalar todo y si se trabaja con servicios más pequeños como lo son los microservicios, se pueden escalar solo los necesarios sin ejecutar toda la aplicación (Newman, 2015). En la Figura 5 se puede observar un ejemplo de ello.

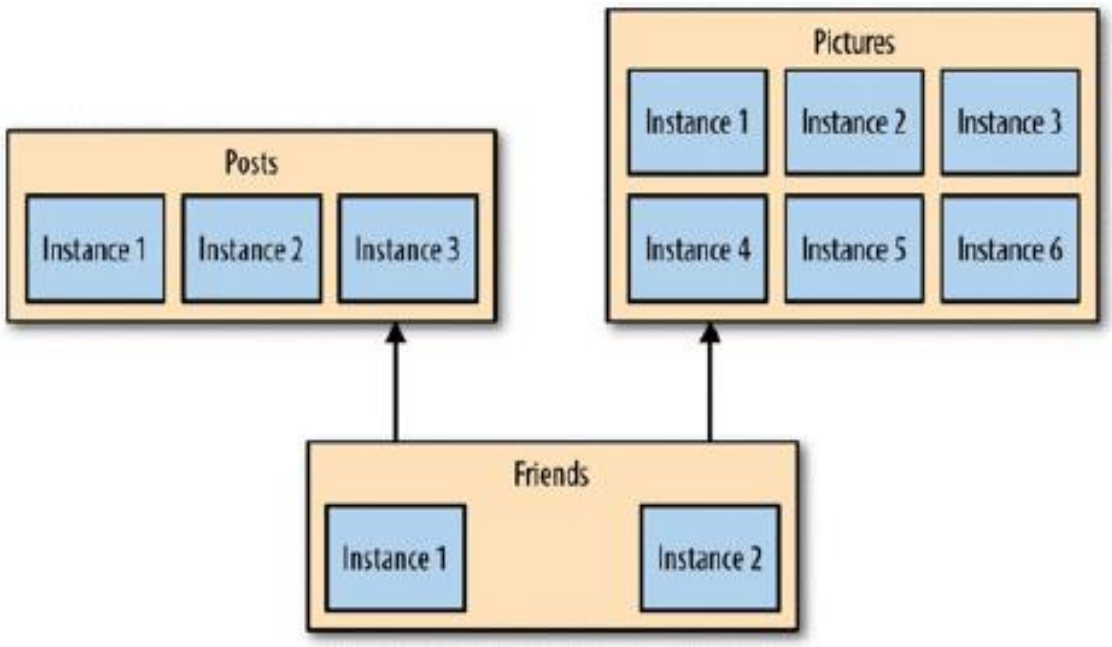


Figura 5. You can target scaling at just those microservices that need it (Newman, 2015).

Contenedores

El uso de la tecnología de contenedores se ha vuelto importante en el lapso de la implementación de microservicios, ya que la industria ha mostrado interés en el uso de ellos en los últimos años. Los contenedores ofrecen un tipo de virtualización más ligero, concediendo mejoras en el rendimiento. Entre las ventajas de este enfoque esta la eliminación de la necesidad al momento de emular o compilar las instrucciones consecutivamente, eliminando cualquier tipo de sobrecarga al sistema y administrando el uso de las aplicaciones o servicios (Aerts et al., 2016), así como el aislamiento y el control de los recursos para ciertos procesos (Amaral et al., 2015).

Dentro del área de los contenedores, otro aspecto fundamental son los docker, compuestos de código abierto que permiten la optimización de proyectos para la implementación de aplicaciones que puedan ejecutarse independientemente, tanto en la nube como en cualquier otro tipo de plataforma (De la Torre et al., 2019).

De la arquitectura monolítica a microservicios

Realizar una migración se ve mayormente recurrida al momento de presentar problemas durante la implementación de nuevos requisitos. Llevar una migración de arquitecturas monolíticas a microservicios ofrece una gran cantidad de beneficios. Entre ellos, se proporciona una mejor adaptabilidad a los cambios tecnológicos, se reduce el tiempo de comercialización y se obtiene una mejor estructuración del equipo de desarrollo en cuanto a los servicios. Entre las necesidades a las que se ve sometido el llevar una migración, están la reutilización, la necesidad de que

los datos sean únicos, la realización de un despliegue automatizado y la incorporación de una escalabilidad ágil (Balalaie et al., 2016). Para obtener un mejor enfoque de la diferencia que existe entre la arquitectura monolítica y una arquitectura de microservicios se puede observar en la Figura 6 ese contraste que existe.

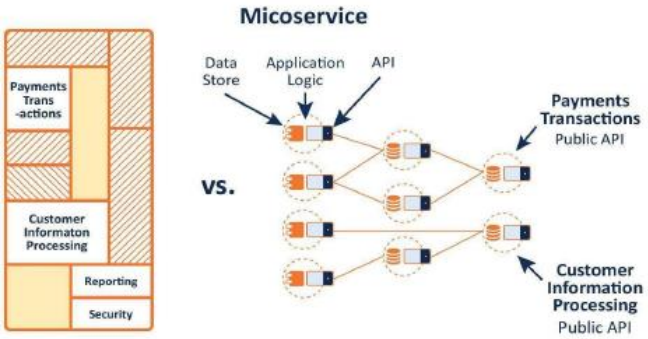


Figura 6. Whereas a typical application monolith is tightly coupled, microservices decouple independent business functions into separate services so that changes to any one function will not interfere with the other functions (Avidan & Otharsson, 2018).

Arquitectura monolítica

El desarrollo de software está vinculado cada vez más a satisfacer las necesidades de automatización de procesos internos dentro del sector empresarial. Por lo que se ve obligado a seguir las tendencias impuestas por el desarrollo, así como del lenguaje de programación y una dependencia en la experiencia que existe implementando una arquitectura tradicional o monolítica (López Hinojosa, 2017).

Strimbei et al. (2015) hacen referencia a que el nombre monolítico se refiere a la organización de los elementos fundamentales de la aplicación en un solo componente o unidad, lo cual va orientado a un enfoque tradicional.

Este tipo de arquitectura tradicional se descompone en pequeñas piezas que proporcionan servicios únicos, que son los encargados de tener un funcionamiento y de la interacción de estas piezas pequeñas (Khazaei et al., 2017). La lógica se ve ejecutada en un servidor de aplicaciones o en un programa que se encarga de realizar todo lo necesario para el funcionamiento. Son grandes y son desarrolladas por múltiples equipos, requiriendo una estructura cuidadosa para cada uno de los cambios. También es requerida la existencia de múltiples servicios API que se encargan de proporcionar la lógica o los procesos del negocio (Stephens, 2015).

Los sistemas monolíticos al momento de escalar, requieren de un equilibrador de carga, ya que es un sistema completo que se encuentra implementado en varios servidores. Lo cual hace imposible escalar una parte de este tipo de arquitectura; siempre es todo o nada (MuleSoft, 2019).

A continuación, Nielsen (2015) dice de manera clara cuáles son las características con las que debe contar un sistema monolítico.

1. Base de código único. El código de un sistema normalmente es grande; cuando ese sistema crece también existe un crecimiento en su código.

2. Lenguaje de código especifico. La construcción del sistema está construida generalmente en un lenguaje único de programación.

3. Escala horizontal. El escalar ciertos componentes dentro del sistema es complicado, ya que para eso es necesaria la ejecución de otras instancias.

4. Arquitectura de tres niveles. Consta de la interfaz de usuario, de una base de datos y de un sistema por el lado del server.

5. Solo proceso. La ejecución del sistema monolítico es ejecutada de una sola vez.

6. Componentes enredados. Los componentes dentro del sistema no cuentan con una única propiedad.

7. Largos ciclos de despliegue. El realizar algún cambio en la funcionalidad de un componente, por más pequeño que sea, genera la compilación de todo el sistema.

Mediante la implementación de la arquitectura monolítica se pueden realizar y tener grandes alcances de lo que es un buen software, tomando en cuenta que el líder de proyecto tiene el suficiente conocimiento de esa área.

MuleSoft (2019) señala que existen algunos problemas a los que se enfrentan, tales como los siguientes: realizar la evolución de una aplicación monolítica es difícil ya que, conforme pasa el tiempo, cada vez son más grandes; la falta de reutilización de elementos, ya que solo puede llevarse a cabo por medio de aplicaciones que permitan la implementación monolítica; y la agilidad operacional al no poder implementarse de la manera correcta.

Una manera para resumir la comparación entre una arquitectura tradicional y monolítica y una arquitectura orienta a microservicio está representada en la Figura 7.

Category	Monolithic architecture	Microservices architecture
Code	A single code base for the entire application.	Multiple code bases. Each microservice has its own code base.
Understandability	Often confusing and hard to maintain.	Much better readability and much easier to maintain.
Deployment	Complex deployments with maintenance windows and scheduled downtimes.	Simple deployment as each microservice can be deployed individually, with minimal if not zero downtime.
Language	Typically entirely developed in one programming language.	Each microservice can be developed in a different programming language.
Scaling	Requires you to scale the entire application even though bottlenecks are localized.	Enables you to scale bottle-necked services without scaling the entire application.

Figura 7. Comparing monolithic and microservices architectures (Daya et al., 2015).

Trabajo relacionado

En la utilización de microservicios se puede apreciar que no solo los proveedores de software como IBM y Microsoft los utilizan, sino también empresas encargadas de ofrecer cierto contenido, como Netflix, los han adoptado y los usan en sus avances tecnológicos (Balalaie et al., 2016).

El realizar un cambio hacia el uso de los microservicios es un asunto delicado en estos días, ya que las compañías deben verse involucradas en la restructuración de sus sistemas de backend para la implementación de un nuevo modelo (Dragoni et al., 2017)

A continuación, se mostrarán algunas investigaciones en las cuales se ve el uso e implementación de arquitecturas basadas en microservicios, así como algunos casos donde se realizaron migraciones de una arquitectura tradicional o monolítica a una de microservicios.

Primer caso

La creación del nuevo sistema de Seneca está basada en esta nueva arquitectura de microservicios y fue implementado en la gestión de una estación de gas. La aplicación consta de monitoreo de los tanques y máquinas de bombeo, manejo de datos, cálculo de tarjetas, compras, ventas e inventarios. El desarrollo y la implementación de esta nueva arquitectura simplifica la comunicación que consiste en una API REST, cola de mensajes, envío y suscripción, el flujo de datos con un tipo de formato JSON y la adaptación de patrones en el tipo de respuesta (Lv y Wang, 2016).

Segundo caso

Intercity es una plataforma que explora el uso de la arquitectura de microservicios, una plataforma de código abierto que apunta al desarrollo de nuevas iniciativas en la implementación de ciudades inteligentes. Esta plataforma ofrece una arquitectura flexible en la adaptación de nuevas tecnologías, permitiendo la integración de nuevos requisitos funcionales y no funcionales y ofreciendo varias formas de escalabilidad.

Intercity ofrece la función de middleware, permitiendo mejor escalabilidad y capacidad de desarrollo, mantenimiento y modularidad. La plataforma aporta el código abierto para el análisis e investigación, así como la implementación de ella en otras secciones. Todo eso enfocado a un desarrollo de software ágil. La adopción de esta arquitectura para la implementación fue de acuerdo con la escalabilidad que debía tener y la capacidad de evolucionar a nuevas tendencias. En la Figura 8 se puede apreciar cómo está distribuida la arquitectura de microservicios con la que cuenta intercity (Del Esposte et al., 2017).

Tercer caso

MiCado, un orquestador dinámico a nivel de aplicación en la nube basado en microservicios, donde se analiza cuáles son las ventajas y desafíos que propone este tipo de arquitectura. Se investiga la manera de reemplazar la forma en que se suministran los servicios en la nube, haciéndolos automáticamente e independientes de la nube. A los desarrolladores les permiten crear gran variedad de aplicaciones mediante la optimización de costos y la reutilización de su código por medio de APIs de alto nivel. La nube es una plataforma natural para realizar la implementación de este tipo de arquitectura, ya que permite la ejecución y la asignación de recursos independientes de

acuerdo con las funcionalidades que se requieren. Otra característica es que los microservicios ofrecen APIs para que exista comunicación entre ellos, ya que es su medio para acceder a las características de los demás. Entre los desafíos que deben tomar en cuenta son el direccionamiento IP y los puertos de microservicios, además de la capacidad de ejecutar varios microservicios en la misma instancia, la asignación de microservicios a instancias de acuerdo con sus características, la manera de garantizar que los microservicios están asignados en su instancia correspondiente y la manera de escalar en cualquier dirección. En la Figura 9 se representa la arquitectura con la que se busca solucionar los desafíos que se mencionan (Visti et al., 2016).

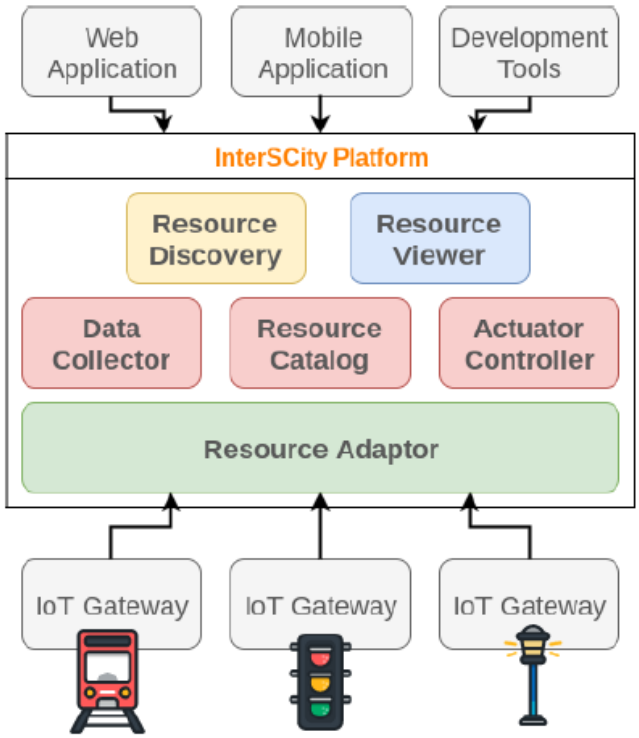


Figura 8. The InterSCity Platform Architecture (Del Esposte et al., 2017).

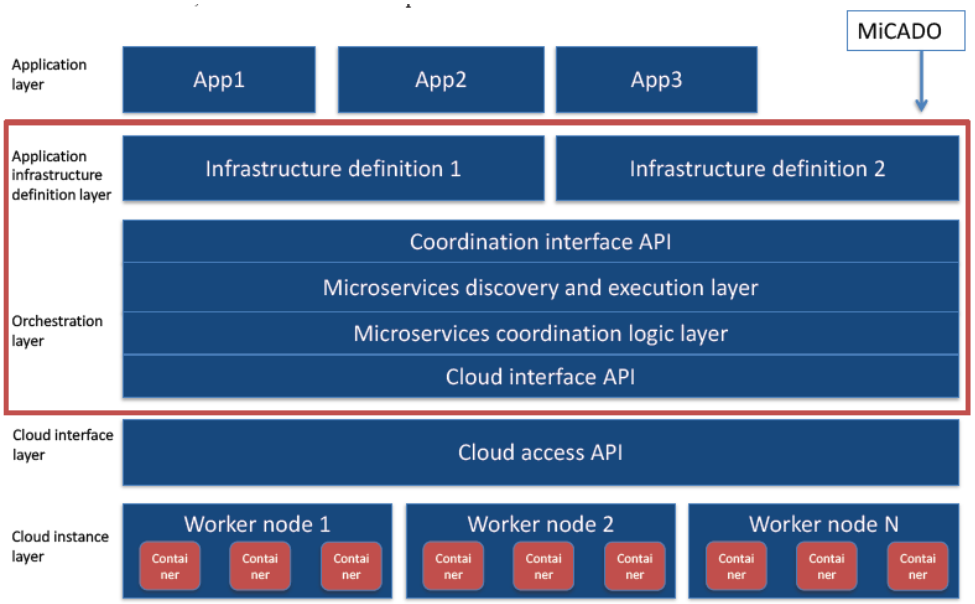


Figura 9. Generic MiCADO architecture (Visti, Kiss, Terstyanszky, Gesmier y Winter, 2016).

Cuarto caso

La realización de un portal de rayos X transitorios es otro proyecto que está basado en una arquitectura de software liviana hecha con microservicios, dándole un mejor ciclo de vida y una mejor resiliencia. La arquitectura que propone consta de tres capas; la primera es la capa de presentación, la segunda es la capa de aplicación, donde contienen los microservicios y la tercera es una capa encargada de la interacción con cualquier tipo de infraestructura. Los módulos están creados para permitir la reutilización y la implementación de características estándar para un desarrollo más ágil, así como la asignación de una funcionalidad única a cada uno de ellos. La utilización de microservicios dentro de portal permite la buena administración de conexiones y actividades, mediante el balanceo de cargas para que no exista un desperdicio de recursos. También

facilita la actualización, la mejora y la corrección de ciertos errores de manera independiente a los demás. Esto permite mejorares ciclos de desarrollo, fases de análisis, depuración y un despliegue más sencillo, permitiendo la agregación de nuevas funcionalidades (D'Agostino et al., 2016).

Quinto caso

Esta investigación presenta el enfoque de un marco escalable llamado ScarR, con una arquitectura basada en microservicios, que se encarga de proporcionar recomendaciones en tiempo real. Este marco le permite soportar flujos de datos, actualizaciones sin recálculos costosos en tiempo real, así como una arquitectura escalable con una adaptación de diferentes tipos de algoritmos y el soporte de gran cantidad de evaluaciones. Este es un marco de código abierto desarrollado con un lenguaje de programación en Java y ya se ha utilizado en varios proyectos comerciales. La comunicación de los servicios es a través de HTTP / REST, lo que facilita el intercambio de datos. En la Figura 10 se aprecia la arquitectura con la cual se está implementado este tipo de enfoque, donde se aprecian cuáles son servicios que describen la comunicación que existe entre ellos (Lacic, Traub, Kowald y Lex, 2015).

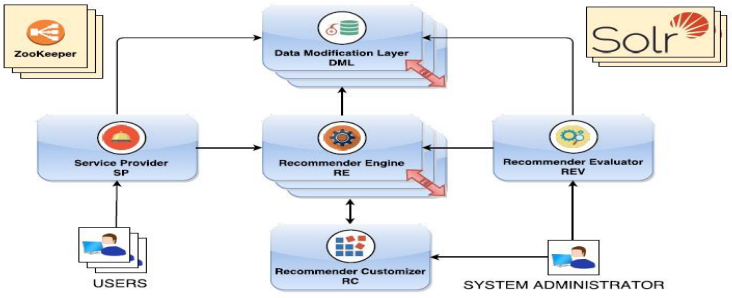


Figura 10. The system architecture of ScaR for a distributed environment. Each service is a standalone HTTP server which knows the locations of its communicating partners with the help of ZooKeeper (Lacic et al., 2015).

Sexto caso

Otto es una de las tiendas en línea más grandes de Europa, la cual comenzó con una reestructuración completa del software del comercio electrónico, basado en una arquitectura de microservicios, ya que le facilitaría ciertas propiedades como la escalabilidad, la agilidad y la confiabilidad. Se buscó dar solución a una gran cantidad de desarrolladores que crean este software, permitiéndoles un mayor crecimiento al momento de escalar.

Los microservicios implementan una forma conocida como verticales; esa forma de almacenamiento ayudó en la identificación de lo que compone esta arquitectura. La comunicación entre las verticales y el acceso a los microservicios era por medio de las API REST. En la Figura 11 se muestra la descomposición vertical existente en Otto.

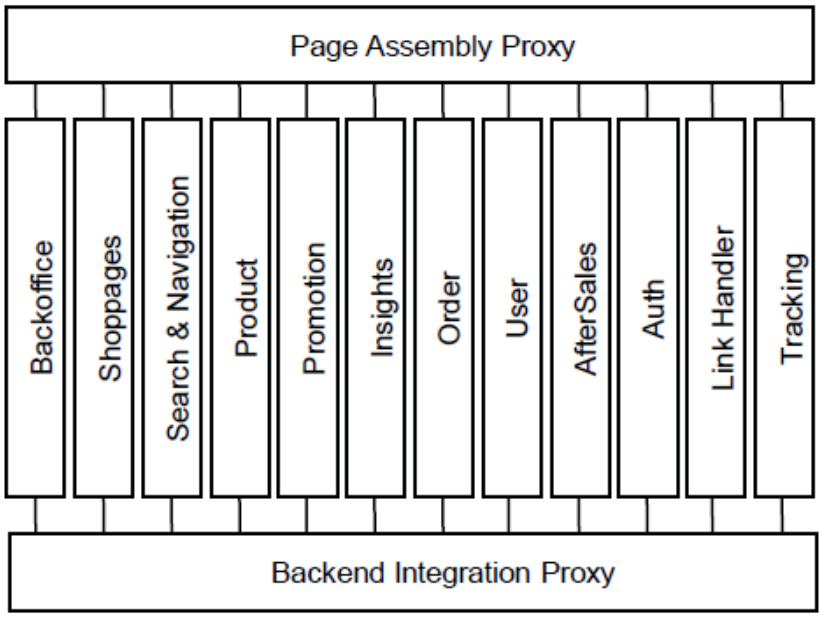


Figura 11. Current vertical decomposition at otto.de (Hasselbring y Steinacker, 2017).

Para cumplir con la implementación de esta arquitectura, Otto menciona que las características que la componen son la integración de verticales, una comunicación entre verticales, las verticales y microservicios, un escalamiento al momento de entrega, la agilidad y confiabilidad, el monitoreo de microservicios, un escalado de microservicios y el uso compartido de códigos, así como la reutilización. En la Figura 12 se observa el cambio que hubo durante la implementación de microservicios a determinados años (Hasselbring y Steinacker, 2017).

Séptimo caso

Backtory es desarrollado por PegahTech, que es encargado de ofrecer servicios de backend a desarrolladores móviles, sin importar el lenguaje de programación. Backtory se vio en la necesidad de migrar la arquitectura de sus sistemas debido a problemas presentados al momento de agregar nuevas funcionalidades, como lo fue el proporcionar un chat como servicio. Entre las necesidades a las que se enfrentaba estaba la reutilización de código, la necesidad descentralizada de los datos, el despliegue automatizado y la existencia de una escalabilidad. En la Figura 13 se observa a detalle cómo estaba compuesta la arquitectura de Backtory antes de la implementación de microservicios y cuál fue el resultado después de ella (Balalaie et al., 2016).

Octavo caso

La elaboración de estructuras de microservicios es basada con herramientas de modelado de procesos de negocios; es otra manera con la cual se puede iniciar una arquitectura. El utilizar el modelado de procesos ayuda al entendimiento de los procesos de cada microservicio, así como la descomposición según las capacidades y casos

de uso del negocio. En la actualidad existen herramientas que ayudan a las organizaciones describiendo sus estructuras, procesos, datos y algunas otras perspectivas; entre esas herramientas, se encuentran los BPM. La utilización de estas herramientas ayuda al desarrollo de los microservicios, ya que el desarrollador no se preocupa por la lógica del sistema. En la Figura 14 se muestra una arquitectura de microservicios desarrollada para la utilización de herramientas BPM en torno a la creación de un editor de Petri Net (Alpers, Becker, Oberweis y Schuster, 2015).

De acuerdo con los trabajos e investigaciones que fueron mostradas, se observa una fuerte participación del uso de esta arquitectura de microservicios. Teniendo como propósito general acelerar la forma en que se desarrolla el software, así como enfocarse en una mayor innovación, también se busca una mejor integración de nuevas tecnologías que les permitan permanecer dentro de las nuevas tendencias y que le permitan una mayor escalabilidad; estos son los factores por los que se recurre a la implementación de esta arquitectura de microservicios.

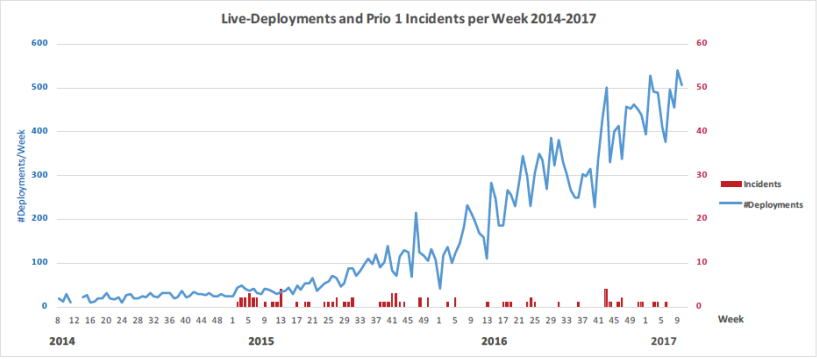


Figura 12. Number of life deployments per week at otto.de over the last two years. Despite the significant increase of deployments, the number of live incidents remains on a very low level (Hasselbring y Steinacker, 2017).

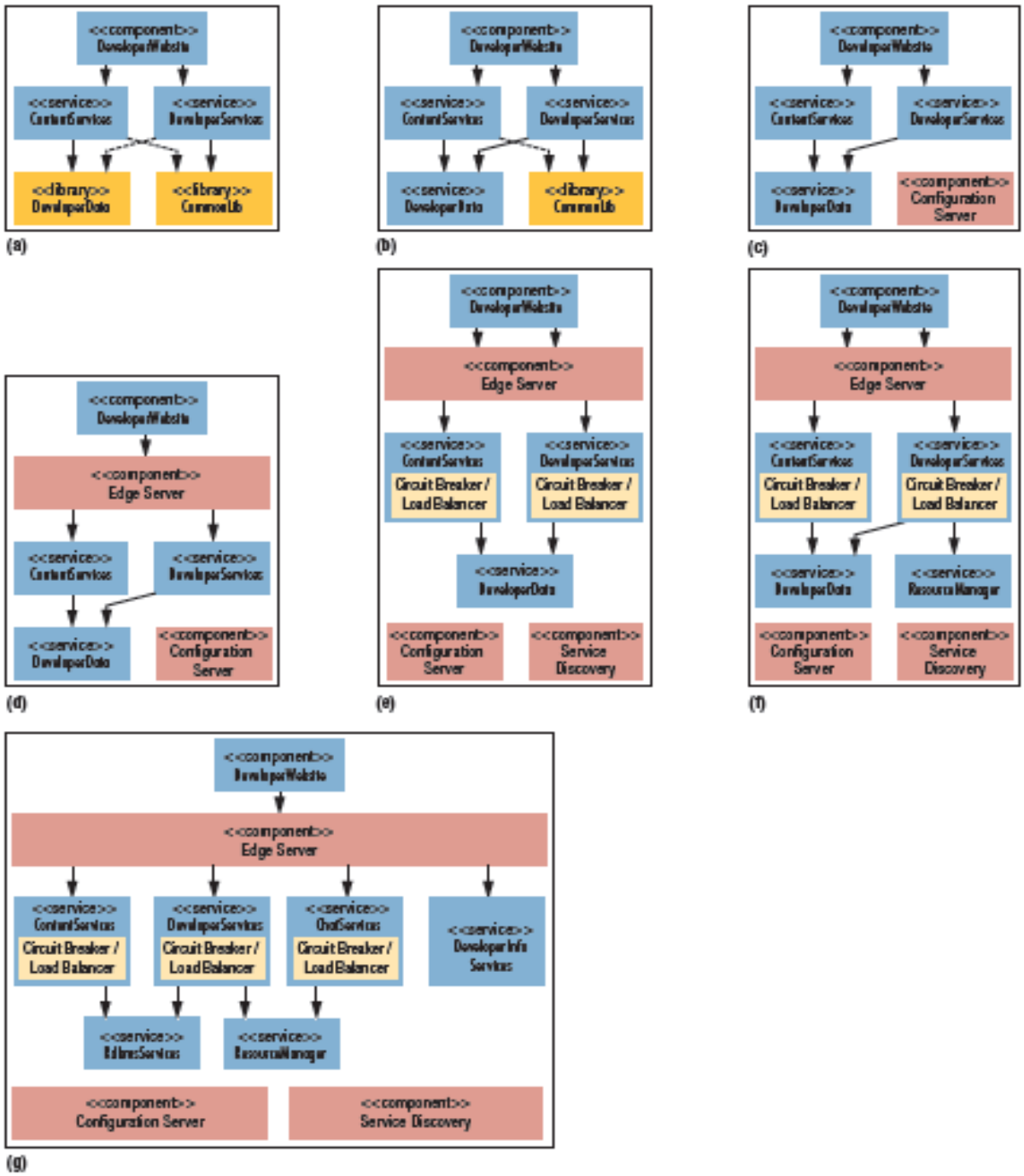


Figura 13. Migrating backory to microservices. Solid arrows indicate service calls; dashed arrows indicate library dependencies: (a) backtory's architecture before the migration, (b) transforming developerdata to a service, (c) introducing the configuration server, (d) introducing the edge server, (e) introducing dynamic service collaboration. (f)introducing resourcemanager y (g) backtory's target architecture after the migration (Balalaie et al., 2016).

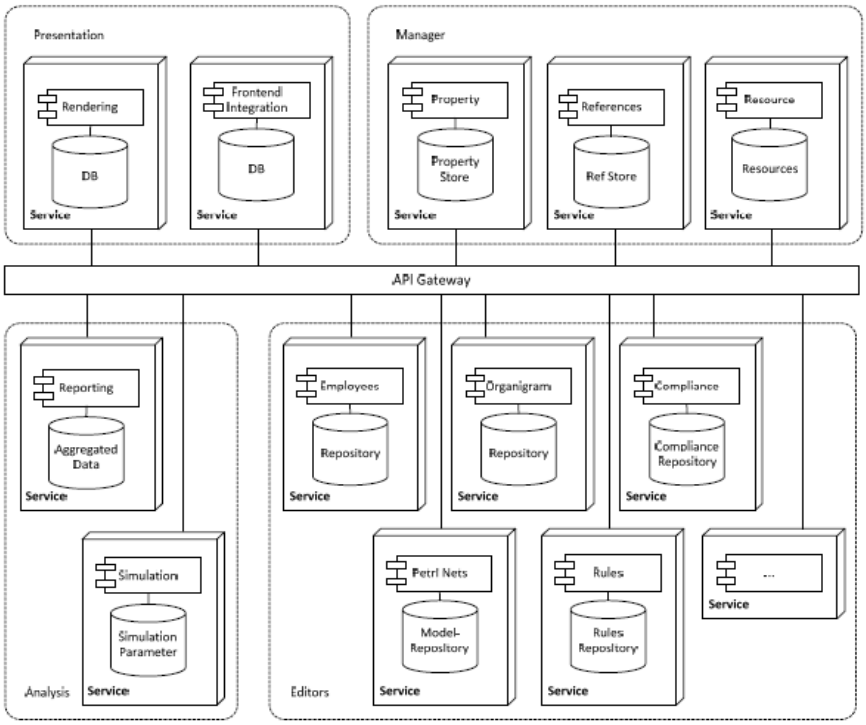


Figura 14. Desarrollo de sistemas usando el estilo de arquitectura de microservicio (Alpers et al., 2015).

CAPÍTULO III

ARQUITECTURA DE MICROSERVICIOS: PROPUESTA

Introducción

En esta sección se pretende documentar el proceso que se llevó a cabo para realizar la implementación de la arquitectura de microservicios en el nuevo sistema institucional para la Universidad de Montemorelos. En la actualidad no existe un estándar con el que se indique cómo documentar este tipo de arquitecturas; al no existir algo ya definido se ha tomado como guía para la elaboración del análisis y el diseño ciertos elementos del libro escrito por Dennis, Wixom y Tegarden (2015). Este es un libro orientado a objetos; por lo tanto, se analizó cada uno de los procesos y diseños que son mencionados, para que se aplicaran a la nueva arquitectura de microservicios.

A continuación, se presentan los análisis y diseños realizados para crear la documentación. Los procesos son los siguientes: diagrama de casos de uso, descripción de casos de uso, diagrama de paquetes, diagrama de colaboración y diagrama de microservicios.

Diagrama de casos de uso

El uso del diagrama de casos de uso permite el modelaje de los procesos que son realizados dentro del módulo de caja para así comprender mejor su funcionamiento.

Este diagrama del módulo de caja del sistema institucional cuenta con cinco actores: el alumno, el empleado, el cliente, la auditoría y el cajero. Cuenta, además, con cinco casos de uso: el pago a otros servicios, el pago de enseñanza, la cancelación de recibos, la cancelación de recibos por auditoría y el cierre de caja. En este diagrama se observa una integración con otras interfaces con las que tiene contacto este módulo: cálculo de cobro, pago en línea y solicitud de pago. En la Figura 15 se puede apreciar este caso de uso.

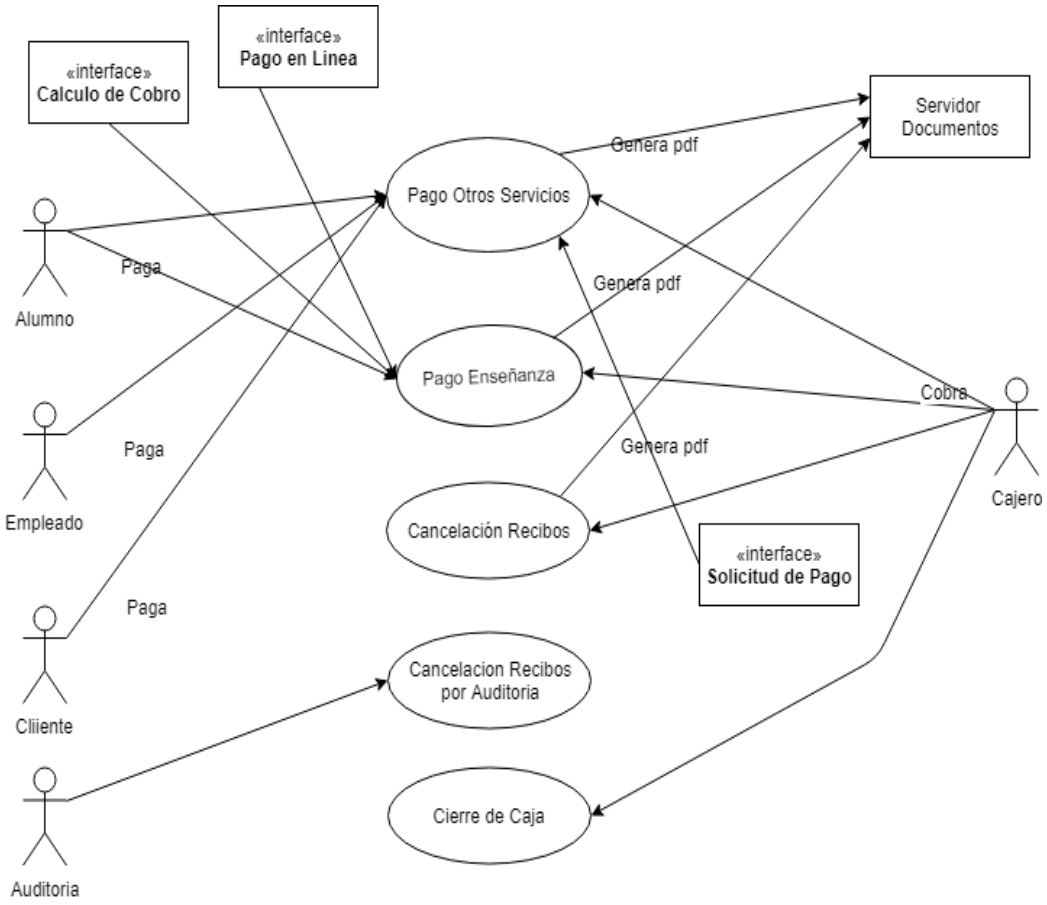


Figura 15. Diagrama de caso de uso del módulo de caja.

Descripción de casos de uso

El agregar descripciones de casos de uso permite documentar los aspectos de manera individual y proporcionar una información detallada de acuerdo con cada funcionalidad. En las Figuras 16, 17, 18, 19 y 20 se menciona la descripción de cada uno de los casos de uso.

Caso de uso: Pago otros servicios	Id: 1	Nivel importancia: Medio
Actor principal: Empleado, cliente, cajero	Tipo de caso de uso: Detalles, esencial	
Partes interesadas e intereses: Empleado / Cliente: Tiene en claro a qué servicio se realizará el pago Cajera: Es el encargado de realizar el cobro de dicho pago		
Breve Descripción: En este caso de uso se describe el proceso cuando un cliente o empleado llega a la caja para realizar el pago de algún servicio externo al académico.		
Trigger: El cliente o empleado debe llegar a caja para realizar un pago		
Tipo: Externo	Relaciones: Asociación – Empleado, Cajera	
Flujo normal de evento: 1- Selección de fondo 2- Selección de captura genérica 3- Agregar cuenta de servicio 4- Agregar nombre del cliente 5- Agregar descripción 6- Agregar importe 7- Agregar acc code 8- Marcar el tipo de pago Efectivo, T. Crédito, T. Debito 9- Agregar Dimensiones correspondientes 10- Seleccionar el tipo de pago que corresponda a efectivo, T. Crédito, T. Debito 11- Poner importe a pagar 12- Impresión de recibo		
Sub flujo: 1- Selección de fondo 2- Selección de captura genérica 3- Agregar cuenta de servicio 4- Agregar nombre del cliente 5- Agregar descripción 6- Agregar importe 7- Agregar acc code 8- Marcar tipo de pago cheque o transferencia 9- Agregar Dimensiones correspondientes 10- Seleccionar el tipo de pago que corresponda a cheque o transferencia. 11- Agregar si corresponde a dólar 12- Poner importe a pagar 13- Imprimir recibo		
Flujos alternativos / excepciones: 1- Cuando se agrega una cuenta que no existe 2- Al agregar dimensiones erróneas		

Figura 16. Descripción del caso de uso pago otros servicios.

Caso de uso: Pago enseñanza	Id: 2	Nivel importancia: Medio
Actor principal: Alumno, cajera	Tipo de caso de uso: Detalles, esencial	
Partes interesadas e intereses: Alumno: Debe conocer su matrícula y su monto a pagar Cajera: Es la encargada de realizar el cobro de dicho pago de la enseñanza		
Breve Descripción: En este caso de uso se describe el proceso de cuando un alumno se presenta a caja para pagar el importe de enseñanza		
Trigger: El alumno debe realizar el pago de su enseñanza		
Tipo: Externo	Relaciones: Asociación – Alumno, Cajera Incluir – Interfaz de cálculo de cobro y pago en línea	
Flujo normal de evento: 1- Cajera selecciona fondo 2- Selección de enseñanza 3- Cajera pide matricula 4- Se ingresa la matrícula de alumno 5- Verifica datos del alumno académicos 6- Se muestran los saldos correspondientes. 7- Se selecciona la carrera. 8- Se selecciona el origen de pago normal 9- Se selecciona el tipo de pago efectivo, T. Crédito, T. Debito. 10- Se pregunta el monto a pagar 11- Se ingresa el monto a pagar 12- Se realiza el pago 13- La cajera imprime recibo 14- La cajera entrega recibo al alumno		
Sub flujo: 1- Selección de fondo 2- Selección de enseñanza 3- Se ingresa la matrícula de alumno 4- Verifica datos del alumno académicos 5- Se muestran los saldos correspondientes. 6- Se selecciona la carrera. 7- Se selecciona el origen de colportaje, bonificación o donación 8- Se selecciona el tipo de pago efectivo, T. Crédito o T. Debito. 9- Se pregunta el monto a pagar 10- Se ingresa el monto a pagar 11- La cajera imprime recibo		
Flujos alternativos / excepciones: 1. Ingresar una matrícula no existente 2. Se selecciona alguno origen incorrecto 3. Se selecciona un tipo de pago incorrecto		

Figura 17. Descripción del caso de uso pago enseñanza.

Caso de uso: Cancelación de recibos	Id: 3	Nivel importancia: Medio
Actor principal: Cajera	Tipo de caso de uso: Detalles, esencial	
Partes interesadas e intereses: Cajera: Es la encargada de revisar el recibo que se mandara a cancelación		
Breve Descripción: En este caso de uso se pretende describir el proceso que se realiza cuando la cajera manda a cancelar un recibo que presentó algún error o simplemente ocupa remover de la contabilidad.		
Trigger: Que se haya generado o cobrado mal un recibo.		
Tipo: Externo	Relaciones: Asociación – Cajero	
Flujo normal de evento: 1- Selecciona cancelar de recibo 2- Selecciona fondo 3- Ubicación de recibo a cancelar 4- Cancelar recibo 5- Cambia estatus a espera de cancelación		
Sub flujo: 1- No existe por el momento.		
Flujos alternativos / excepciones: 1- No existe hasta el momento		

Figura 18. Descripción del caso de uso cancelación de recibos.

Caso de uso: Cancelación de recibos por auditoria	Id: 4	Nivel importancia: Medio
Actor principal: Auditoria	Tipo de caso de uso: Detalles, esencial	
Partes interesadas e intereses: Auditoria: Es la encargada de aprobar la cancelación de cualquier tipo de recibo.		
Breve Descripción: En este caso de uso se pretende describir el proceso que se realiza para la cancelación completa de un recibo, después que la cajera mande a cancelar un recibo auditoria es la encargada de aprobar si es necesario la cancelación de dicho recibo.		
Trigger: Que con anticipación algún cajero haya mandado a cancelar recibos		
Tipo: Externo	Relaciones: Asociación – Auditoria	
Flujo normal de evento: 1- Selecciona cancelación de recibo auditoria 2- Selección de fondo al cual pertenece el recibo 3- Selección del recibo en cancelación 4- Cancelar recibo 5- Elimina recibo 6- Recibo cancelado		
Sub flujo: 1- No existe hasta el momento		
Flujos alternativos / excepciones: 1- No existe hasta el momento		

Figura 19. Descripción del caso de uso cancelación de recibo por auditoría.

Caso de uso: Cierre de caja	Id: 5	Nivel importancia: Medio
Actor principal: Alumno, Empleado, Cajera	Tipo de caso de uso: Detalles, esencial	
Partes interesadas e intereses: Cajera: Interesada en que al finalizar el día las cuentas cuadren		
Breve Descripción: En este caso de uso se pretende describir el proceso que se realiza para que una cajera al finalizar el día realice el cierre y compruebe que la cantidad que tiene en recibos cuadre con el total que tiene en dinero.		
Trigger: No se generen más recibos y se totalice todo el dinero.		
Tipo: Externo	Relaciones: Asociación – Cajera	
Flujo normal de evento: 1- Selección de cierre de caja 2- Selección de fondo 3- Muestra separación de dinero por forma de pago 4- Muestra el total 5- Muestra todos los recibos 6- Se despliega una calculadora para contar el dinero 7- Sale el total por denominación 8- Cuenta que cuadre 9- Cierre de caja		
Sub flujo: 1- No existe hasta el momento		
Flujos alternativos / excepciones: 1- Cuando los totales no coinciden		

Figura 20. Descripción del caso de uso cierre de caja.

Diagrama de paquete de los microservicios

La realización del diagrama de paquete de los microservicios fue inspirada en el uso de los diagramas de paquetes que son utilizados en una arquitectura orientada a objetos, realizando un ajuste enfocado a la nueva arquitectura de microservicios. Esto permite mostrar cómo se encuentran distribuidos los microservicios que se utilizan en este sistema, ayudando a la identificación de manera clara y rápida en su ubicación, así como saber la distribución que llevan los módulos con los que cuenta el sistema. De acuerdo con las tecnologías ocupadas en esta arquitectura tenemos Mulesoft y Express. En la Figura 21 se aprecia cómo está compuesto este módulo de caja.

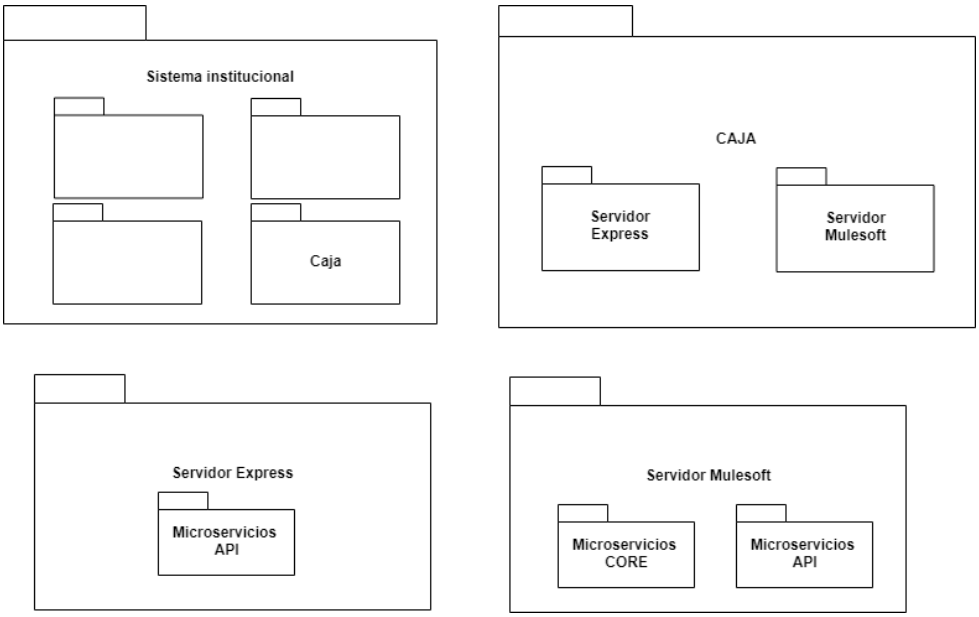


Figura 21. Diagrama de paquete de los microservicios.

Diagrama de colaboración

El diagrama de colaboración permite mostrar cuál es el funcionamiento en cuanto a las actividades que se realizan en cada uno de los casos de uso, permitiendo dar un aspecto dinámico al uso que se le da a cada microservicio. También ayuda a ver cómo se lleva la distribución de los microservicios y en qué momento del proceso se están utilizando.

En cada uno de los diagramas es fácil la identificación de un proceso que debe realizarse. Para llevar a cabo la identificación de qué microservicio es utilizado dentro de un diagrama, se optó por que el nombre esté escrito en subrayado; esto permite diferenciarlo de otro proceso que se realice. En el caso de pago a enseñanza, solo se agregaron unos microservicios globales que están compuestos por microservicios más pequeños; esta distribución se podrá apreciar en las Figuras 22, 23, 24, 25 y 26.

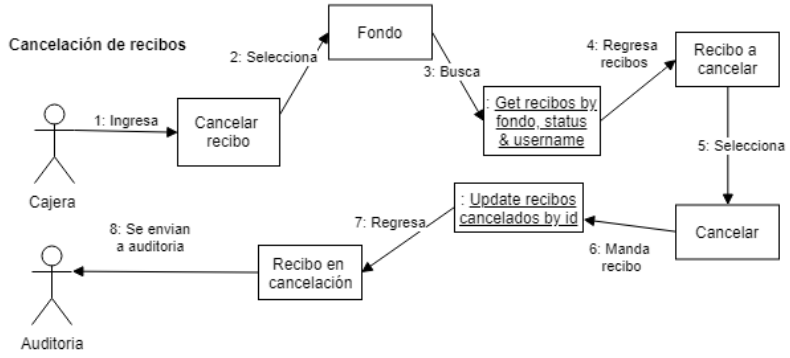


Figura 22. Cancelación de recibos.

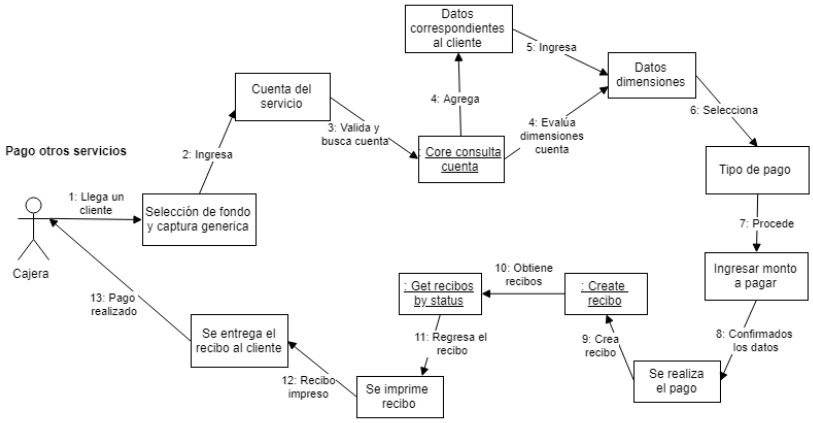


Figura 23. Pago otros servicios.

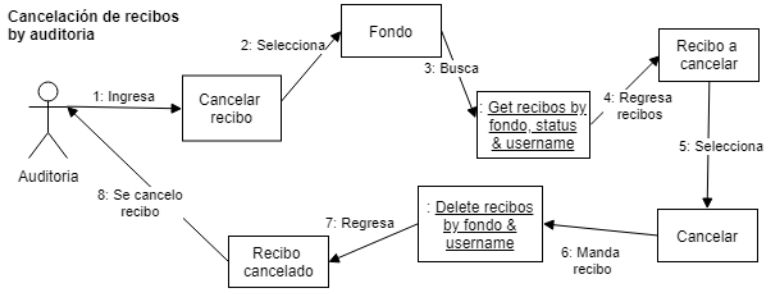


Figura 24. Cancelación de recibos por auditoría.

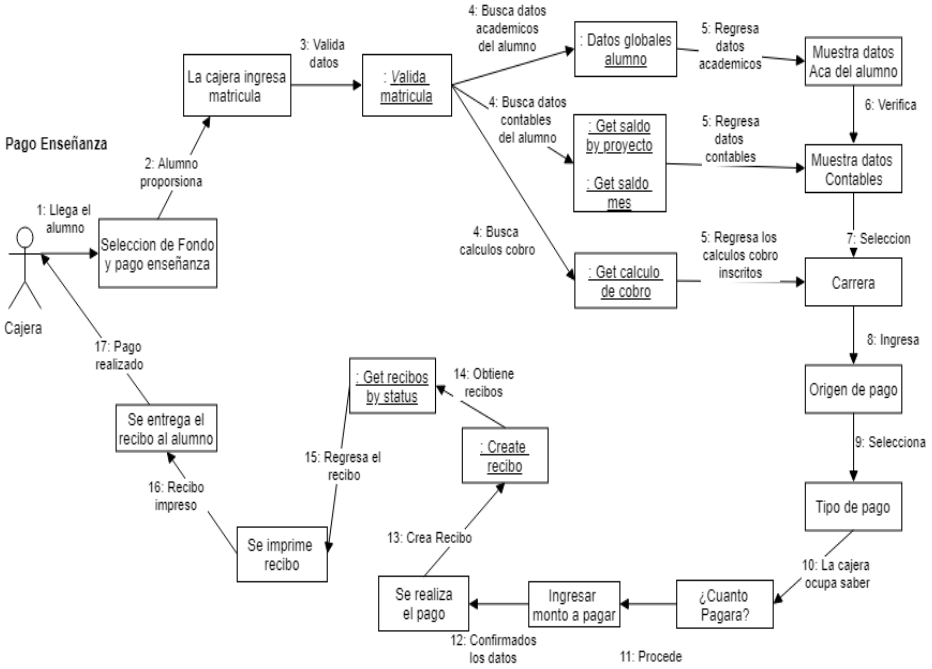


Figura 25. Pago enseñanza.

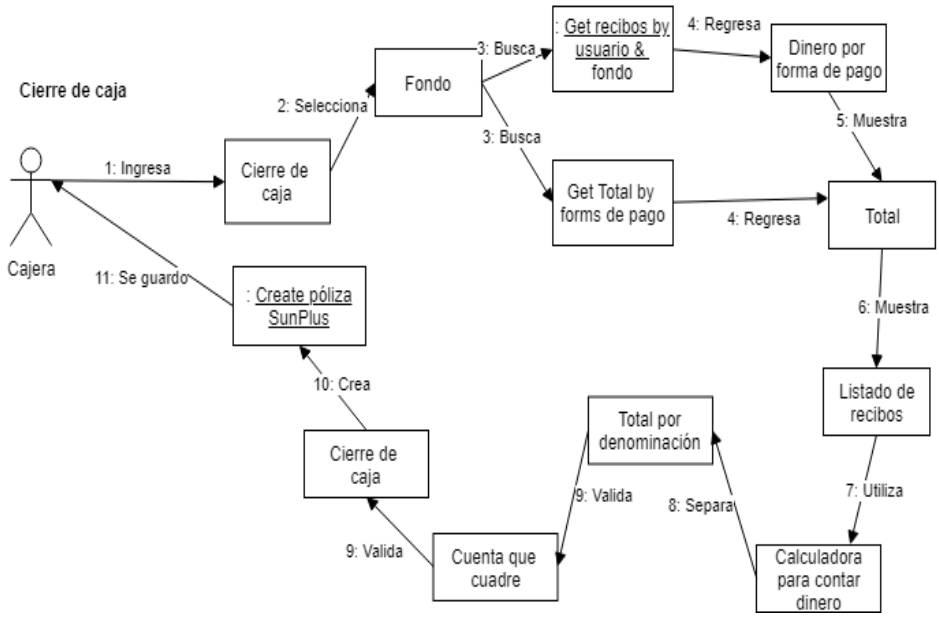


Figura 26. Cierre de caja.

Diagrama de capas

En la Figura 27 se muestra el diagrama de capas, en el cual se desarrolló esta arquitectura, donde se aprecian los componentes en los cuales está distribuida esta arquitectura. Este diseño está basado en el uso que se les da a los microservicios de acuerdo con la empresa Mulesoft (MuleSoft, 2019).

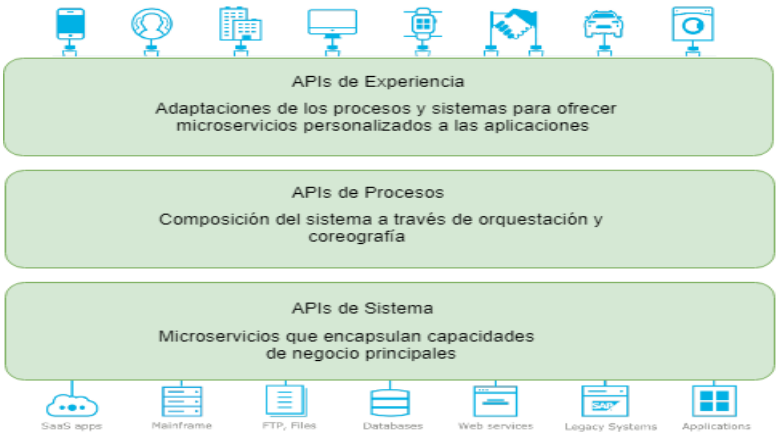


Figura 27. Diagrama de propuesto por MuleSoft (MuleSoft, 2019).

Diagrama de microservicios

El diagrama de microservicios va de acuerdo con el de diagrama de capas presentado anteriormente, donde se muestran cuáles son los microservicios y cómo están clasificados dentro de esa arquitectura. En la sección de APIs de experiencia solo se cuenta con un microservicio que es el encargado de englobar a todos los que están dentro de la capa de APIs del sistema. En el Apéndice C se muestra cada uno de los microservicios que se observan en la Figura 28; se plasma una descripción, el método al que responde, los parámetros necesarios para la invocación y el retorno de cada uno de ellos.

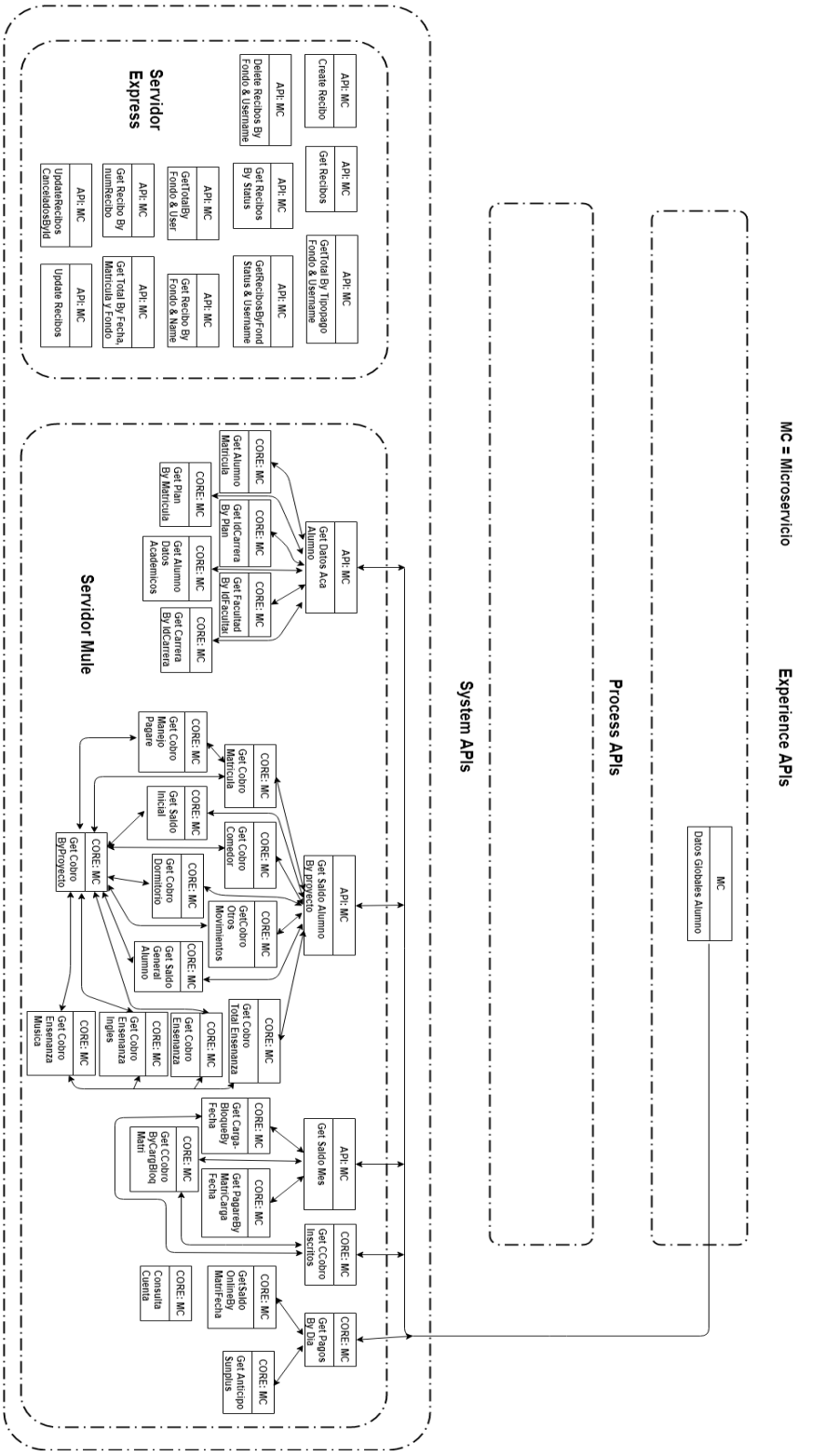


Figura 28. Diagrama de los microservicios utilizados en el módulo de caja.

CAPITULO IV

METODOLOGÍA

Introducción

Una vez implementada la arquitectura propuesta, en este capítulo se mostrará la descripción de la metodología que se utilizó para demostrar que el desarrollo de una arquitectura basada en microservicios del nuevo sistema institucional de la Universidad de Montemorelos es más ágil que el desarrollo de software tradicional existente en la institución.

Descripción del proyecto

Para llevar a cabo esta comparación entre la arquitectura tradicional o monolítica y una basada en microservicios, es necesario contar con ambas partes a evaluar. En este caso, los módulos con los cuales se trabajó para dar una solución a esta problemática fueron los siguientes:

1. Caja. Módulo del nuevo desarrollo basado en una arquitectura orientada a microservicios.

2. Portal de facturas. Módulo con un antiguo desarrollo basado en una arquitectura tradicional o monolítica.

Para llegar a un resultado, es necesario comprender que, dentro de la gestión de proyectos, el tiempo que se lleva en desarrollarlo es un punto vital para que sea llevado a la realización. Existen varias maneras en las que se puede medir el tiempo

que se lleva en la elaboración de un proyecto. Entre esas, la que más se ajustó al desarrollo es la de puntos de casos de uso.

Puntos de casos de uso

El libro escrito por Wazlawick (2014) señala que la metodología de puntos de casos de uso consiste en analizar detalladamente; identificar cuáles son los actores que interactúan dentro del sistema, los respectivos casos de uso, así como la complejidad de cada uno de los sistemas; para así poder estimar el esfuerzo necesario en horas que se emplean en desarrollar cada sistema.

La utilización de una metodología como esta permite dar ciertas prioridades dentro del desarrollo de un nuevo proyecto.

Para poder aplicar esta metodología en cada uno de los proyectos, es necesario cumplir con los siguientes pasos: darle un tamaño a cada módulo y obtener el esfuerzo de cada tamaño obtenido.

Para llevar a cabo la implementación del punto número dos, que es el encargado de obtención del esfuerzo, es necesario aplicar una fórmula con la que se calcula cuántas horas llevará terminar ese desarrollo. Pero en estos casos se hizo un ajuste al no considerar necesaria dicha implementación de la fórmula, ya que para poder comprobar cuántas horas se llevó el desarrollo de cada uno de los módulos se realizó la implementación de bitácoras. En el Apéndice A se adjuntan las correspondientes a las horas que se llevó el desarrollo del nuevo del sistema institucional del módulo de caja y en el Apéndice B se adjuntan las horas que llevó el desarrollo del módulo de portal de facturas.

Para la obtención del punto número uno es necesaria la realización de los siguientes procesos planteados por esta metodología de puntos de caso de uso.

UAW - Peso del actor no ajustado

En UAW se agregan los actores que participan en cada uno de los módulos correspondientes y se le asigna un valor a cada uno de ellos, dependiendo su complejidad.

UUCW - Peso del caso de uso no ajustado

En UUCW se agregan los casos de uso que corresponden a cada uno de los módulos, dividiéndolos en los siguientes: (a) caso de uso simple, que consta de cero a tres transacciones, (b) casos de uso medio, compuesto de cuatro a siete transacciones y (c) casos de uso complejo, que se componen de más de siete transacciones.

UUCP - Puntos de caso de uso no ajustados

Para la obtención de los puntos de caso de uso no ajustados, se requiere la suma del peso del actor no ajustado (UAW) y el peso del caso de uso no ajustado (UUCW). Se ve representado en la siguiente fórmula:

$$UUCP = UAW + UUCW$$

Después de la obtención de los puntos de caso de uso no ajustados, es necesario realizar un ajuste de ellos, por lo cual se requiere la obtención de los factores de complejidad técnicos y factores ambientales.

TCF - Factor de complejidad técnica

Como el nombre lo dice, los factores de complejidad técnica van de acuerdo con la dificultad que tiene el proyecto; cada uno de ellos cuenta con un peso que va de cero a cinco como máximo, donde cero significa que no existe nada de ese factor y cinco significa que hay gran participación en el proyecto. Los factores de complejidad técnicos que se utilizan para sacar los puntos se encuentran en la Tabla 1, donde se indica su respectivo peso.

Tabla 1

Factores técnicos para ajustar los puntos de caso de uso

Código	Factor	Peso
T1	Sistema distribuido	2
T2	Tiempo de respuesta / objetivos de desempeño	1
T3	Eficiencia del usuario final	1
T4	Complejidad de procesamiento interno	1
T5	Diseño orientado a la reutilización del código	1
T6	Fácil de instalar	0.5
T7	Fácil de operar	0.5
T8	Portabilidad	2
T9	Diseño orientado a un fácil mantenimiento	1
T10	Procesamiento concurrente / paralelo	1
T11	Seguridad	1
T12	Acceso para / a código de terceros	1
T13	Necesidades de entrenamiento del usuario	1

Al asignarle una calificación a cada factor se debe multiplicar por el peso que contiene cada uno. La suma correspondiente del resultado de la multiplicación corresponde a TFactor. Esto se ve representado en la formula siguiente, donde se obtiene los valores de cada factor técnico.

$$TCF = 0.6 + (0.01 * TFactor)$$

Para la asignación de cada una de las calificaciones a cada factor técnico, se utilizó la guía mencionada en el libro y se encuentran en el Apéndice D.

EF – Factores ambientales

Los factores ambientales van de acuerdo con el desempeño del equipo de desarrollo. Este cuenta con ocho factores que permiten la evaluación del ambiente en el que se trabaja y se califican de la misma manera de cero a cinco. Cero significa que *el equipo no se encuentra motivado*, tres corresponde a que *la motivación del equipo es promedio* y cinco, *que el equipo está muy motivado*. Los factores que intervienen se pueden observar en la Tabla 2.

Tabla 2

Factores ambientales para el ajuste de casos de uso

Código	Factor	Peso
E1	Familiaridad con el proceso de desarrollo.	1.5
E2	Experiencia en la aplicación.	0.5
E3	Experiencia orientada a objetos	1
E4	Experiencia analista líder	0.5
E5	Motivación	1
E6	Requisitos estables	2
E7	Trabajadores a tiempo parcial	-1
E8	Dificultad con el lenguaje de programación.	-1

De la misma manera, se debe multiplicar la calificación asignada al factor por el peso correspondiente; luego se debe realizar la suma de los resultados obtenidos para obtener el EFactor. Esto se ve representado en la siguiente formula.

$$EF = 1.4 - (0.03 * EFactor)$$

Para la asignación de cada una de las calificaciones a cada factor ambiental, se utilizó la guía mencionada en el libro y se encuentran en el Apéndice D.

UCP – Puntos de caso de uso ajustados

Para la obtención de los puntos de casos de uso ajustados, es necesario contar con el total de puntos de casos de uso no ajustados (UUCP); así también con los datos de los factores técnicos (TCF) y de los factores ambientales (TF). Con esos valores obtenidos, es necesario aplicarlos en la siguiente fórmula:

$$UCP = UUCP * TCF * EF$$

De esa manera se obtienen los puntos de casos de uso ajustados correspondientes a cada uno de los módulos que se necesitan medir. Se llevó la implementación de este procedimiento empezando por el módulo de caja, seguido por el portal de facturas.

Módulo de caja

El módulo de caja es el encargado de procesar los pagos realizados por estudiantes, clientes y empleados para cubrir cobros en la enseñanza o algún otro pago de servicios que ofrece la institución. Este módulo corresponde al nuevo desarrollo en el que se está implementado una arquitectura orientada a microservicios.

A continuación, se muestra el caso de uso correspondiente a este desarrollo y se describe cómo se llegó a la resolución en cada proceso aplicado a caja.

1. UAW - Peso del actor no ajustado. Los actores con los que cuenta el módulo de caja son los corresponden a los que están en la Tabla 3.

La suma de los puntos que obtiene cada actor corresponde a la siguiente operación:

UAW = 3 + 3 + 3 + 3 + 3 + 2 + 2 + 2, UAW = 21

Tabla 3

Descripción de los actores del módulo de caja

Actor	Calificación
Alumno	3
Empleado	3
Cliente	3
Auditoria	3
Cajero	3
Cálculo de cobro	2
Solicitudes de pago	2
Pago en línea	2

2. UUCW - Peso del caso de uso no ajustado. Para sacar el peso de los casos de uso, tenemos que clasificarlos en una de las secciones, dependiendo en el número de transacciones que realiza. En la Tabla 4 se pueden apreciar.

La suma correspondiente de los puntos obtenidos en los casos de uso sin ajustar corresponde a la siguiente operación:

UUCW = 5 + 5 + 5 + 15 + 15, UUCW = 45

3. UUCP – Puntos de caso de uso no ajustados. Para obtener los UUCP es necesario realizar una suma del peso del actor no ajustado (UAW) y la de los casos de uso sin ajustar (UUCW), como se ve representado en la siguiente operación:

UUCP = UAW + UUCW, UUCP = 21 + 45, UUCP = 66

4. TCF - Factor de complejidad técnica. Para observar de manera detallada el análisis de los factores de complejidad técnicos del módulo correspondiente a caja, se pueden observar en la Tabla 5, cada uno con su valor obtenido. De acuerdo con los datos obtenidos en cada uno de los factores técnicos y la obtención de TFactor se lleva la implementación de la formula correspondiente.

Tabla 4

Clasificación de los casos de uso del módulo de caja

Casos de uso	Calificación
Casos de uso simple	
Cancelación de recibos	5
Cancelación de recibos por auditoria	5
Cierre de caja	5
Casos de uso medio	
Casos de uso complejos	
Pago otros servicios	15
Pago enseñanza	15

Tabla 5

Evaluación de los factores técnicos de complejidad del módulo de caja

Cod	Factor	P	T	P * T
T1	Sistema distribuido	2	3	6
T2	Tiempo de respuesta / objetivos de rendimiento	1	1	1
T3	Eficiencia del usuario final	1	2	2
T4	Complejidad de procesamiento interno	1	1	1
T5	Diseño con el objetivo de reutilizar el código	1	5	5
T6	Fácil de instalar	0.5	5	2.5
T7	Fácil de operar	0.5	0	0
T8	Portabilidad	2	0	0
T9	Diseño con el objetivo de facilitar el mantenimiento	1	4	4
T10	Procesamiento concurrente / paralelo	1	1	1
T11	Seguridad	1	3	3
T12	Acceso para / código de terceros	1	0	0
T13	Necesidades de formación del usuario	1	1	1
TFactor				26.5

Para la resolución de los factores de complejidad técnica, se representa la siguiente operación:

TCF = 0.6 + (0.01* TFactor), TCF = 0.6 + (0.01 * 26.5)

TCF = 0.6 + 0.265, TCF = 0.865

5. EF – Factores ambientales. El análisis de los factores ambientales correspondientes al módulo de caja se puede apreciar aplicado en la Tabla 6. De acuerdo con los datos obtenidos en cada uno de los factores técnicos y la obtención de EFactor, se lleva la implementación de la fórmula y así se obtiene el factor ambiental.

Tabla 6

Evaluación de factores ambientales del módulo de caja

Cod	Factor	P	T	P * T
E1	Familiaridad con el proceso de desarrollo.	1.5	4	6
E2	Experiencia en la aplicación.	0.5	4	2
E3	Experiencia orientada a objetos	1	3	3
E4	Experiencia analista líder	0.5	5	2.5
E5	Motivación	1	3	3
E6	Requisitos estables	2	2	4
E7	Trabajadores a tiempo parcial	-1	3	-3
E8	Dificultad con el lenguaje de programación.	-1	1	-1
EFactor				16.5

Para la resolución de los factores de complejidad técnica, se representa la siguiente operación:

EF = 1.4 – (0.03 * Efactor)

EF = 1.4 – (0.03 * 16.5)

EF = 1.4 – (0.495), EF = 0.905

6. UCP – Puntos de casos de uso ajustados. En este caso se realiza el ajuste de los puntos de casos de uso, donde se multiplican los valores obtenidos en UUCP, TCF y EF. Esto da como resultado el peso correspondiente al módulo de caja en puntos de casos de uso.

Para la resolución de los puntos de casos de uso ajustados, se presenta la siguiente operación:

UCP = UUCP * TCF * EF

UCP = 66 * 0.865 * 0.905 UCP = 51.666

Módulo portal de facturas

El módulo portal de facturas que está en comparación cuenta con una arquitectura tradicional conocida como monolítica. La parte de este desarrollo del sistema corresponde al módulo del portal de facturas, en el cual se integra una nueva versión de facturas electrónicas y donde se utilizan los complementos de pagos pedidos por el SAT.

En la Figura 29 se muestra el diagrama de casos de uso que corresponde a dicho módulo, permitiendo realizar un análisis y entender cómo están distribuidos cada uno de los componentes que se hicieron para evaluar el desarrollo.

De acuerdo con esta figura, se presenta la realización de los procesos de esta metodología para determinar un peso correspondiente a este módulo.

1. UAW - Peso del actor no ajustado. Los actores que participan en el módulo de portal de facturas están en la Tabla 7.

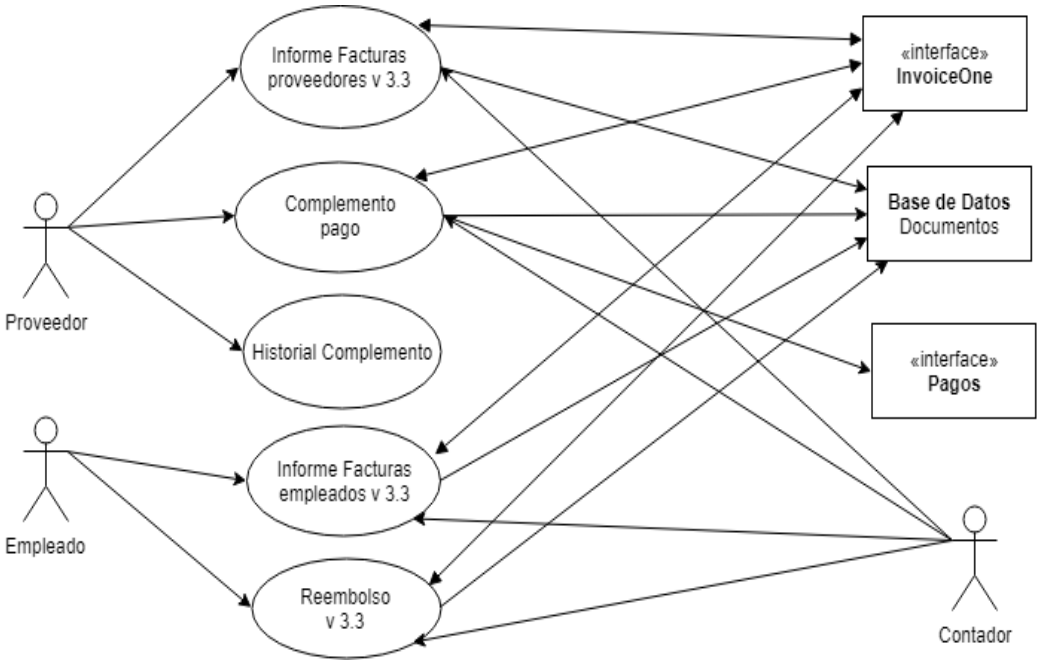


Figura 29. Diagrama de casos de uso del portal de facturas.

Tabla 7

Descripción de los actores del módulo del portal de facturas

Actor	Calificación
Proveedor	3
Contador	3
Empleado	3
InvoiceOne	2
Pagos	2

La suma de los puntos de los actores corresponde a la siguiente operación:

UAW = 3 + 3 + 3 + 2 + 2

UAW = 13

2. UUCW - Peso del caso de uso no ajustado. Para sacar el peso de los casos de uso, hay que clasificarlos en una de las secciones, dependiendo el número de

transacciones que se realizan. En la Tabla 8 se muestra la clasificación de los casos de uso. La suma referente a los puntos obtenidos en los casos de uso sin ajustar corresponde a la siguiente operación:

UUCW = 5 + 10 + 10 + 15 + 15

UUCW = 55

Tabla 8

Clasificación de los casos de uso del módulo de portal de facturas

Casos de uso	Calificación
Casos de uso simple	
Historial complemento	5
Casos de uso medio	
Informe facturas emp. V. 3.3	10
Reembolso	10
Casos de uso complejos	
Complemento de pago	15
Informe facturas emp. V. 3.3	15

3. UUCP – Puntos de caso de uso no ajustados. Para la obtención de UUCP es necesaria la suma del peso del actor no ajustado (UAW) y la del peso de casos de uso no ajustados (UUCW), como se ve representado en la siguiente operación:

UUCP = UAW + UUCW

UUCP = 13 + 55

UUCP = 68

4. TCF - Factor de complejidad. Para observar de manera detallada el análisis de los factores de complejidad técnicos del módulo correspondiente al portal de facturas, se puede visualizar en la Tabla 9, cada uno con su valor obtenido.

De acuerdo con los datos obtenidos en cada uno de los factores de complejidad técnicos y el resultado de TFactor, se lleva la implementación de la fórmula correspondiente, que queda de la siguiente manera:

TCF = 0.6 + (0.01* TFactor),

TCF = 0.6 + (0.01 * 13)

TCF = 0.6 + (0.13)

TCF = 0.73

5. EF – Factores ambientales. El análisis de los factores ambientales correspondientes al módulo del portal de facturas se puede apreciar aplicado en la Tabla 10.

De acuerdo con los valores obtenidos en cada uno de los factores ambientales y el resultado de EFactor, se lleva la implementación de la fórmula correspondiente, que queda de la siguiente manera:

EF = 1.4 – (0.03 * Efactor)

EF = 1.4 – (0.03 * 12.5),

EF = 1.4 – (0.375),

EF = 1.025.

6. UCP – Puntos de caso de uso ajustados. En este punto se realiza el ajuste de los puntos de casos de uso, donde se multiplican los valores obtenidos en UUCP, TCF y EF.

Esto da como resultado el peso correspondiente al módulo de portal de facturas en puntos de casos de uso, como se ve representado en la siguiente operación:

UCP = UUCP * TCF * EF,

UCP = 68 * 0.73 * 1.025,

UCP = 50.881

Tabla 9

Evaluación de los factores técnicos de complejidad del módulo de portal de facturas

Cod	Factor	P	T	P * T
T1	Sistema distribuido	2	0	0
T2	Tiempo de respuesta / objetivos de rendimiento	1	1	1
T3	Eficiencia del usuario final	1	1	1
T4	Complejidad de procesamiento interno	1	0	0
T5	Diseño con el objetivo de reutilizar el código	1	0	0
T6	Fácil de instalar	0.5	4	2
T7	Fácil de operar	0.5	0	0
T8	Portabilidad	2	0	0
T9	Diseño con el objetivo de facilitar el mantenimiento	1	2	2
T10	Procesamiento concurrente / paralelo	1	3	3
T11	Seguridad	1	3	3
T12	Acceso para / código de terceros	1	0	0
T13	Necesidades de formación del usuario	1	1	1
TFactor				13

Tabla 10

Evaluación de factores ambientales del módulo de portal de facturas

Cod	Factor	P	T	P * T
E1	Familiaridad con el proceso de desarrollo.	1.5	3	4.5
E2	Experiencia en la aplicación.	0.5	3	1.5
E3	Experiencia orientada a objetos.	1	3	3
E4	Experiencia analista líder.	0.5	5	2.5
E5	Motivación.	1	2	2
E6	Requisitos estables.	2	2	4
E7	Trabajadores a tiempo parcial.	-1	3	-3
E8	Dificultad con el lenguaje de programación.	-1	2	-2
EFactor				12.5

Análisis de los tamaños

Después de analizar ambos módulos por medio de la metodología de puntos de casos de uso y de haber obtenido el peso de cada uno, se podrá deducir el tiempo en horas que llevó el desarrollo. En la Figura 30 se puede observar que el módulo de caja realizado con el nuevo desarrollo de microservicios consta de un tamaño de 51.666 puntos, por encima del módulo del portal de facturas que cuenta con 50.881 puntos. Al tener el tamaño de ambos módulos, es necesario sacar las horas que se llevó en cada desarrollo. Para realizar el cálculo de horas, en los Apéndice A y B se agregaron las bitácoras de los módulos. En esas bitácoras se encuentra plasmada la realización de ciertas actividades correspondientes al desarrollo de los casos de uso mencionados y también se encuentran indicadas las horas que se invirtieron en cada una de las actividades.

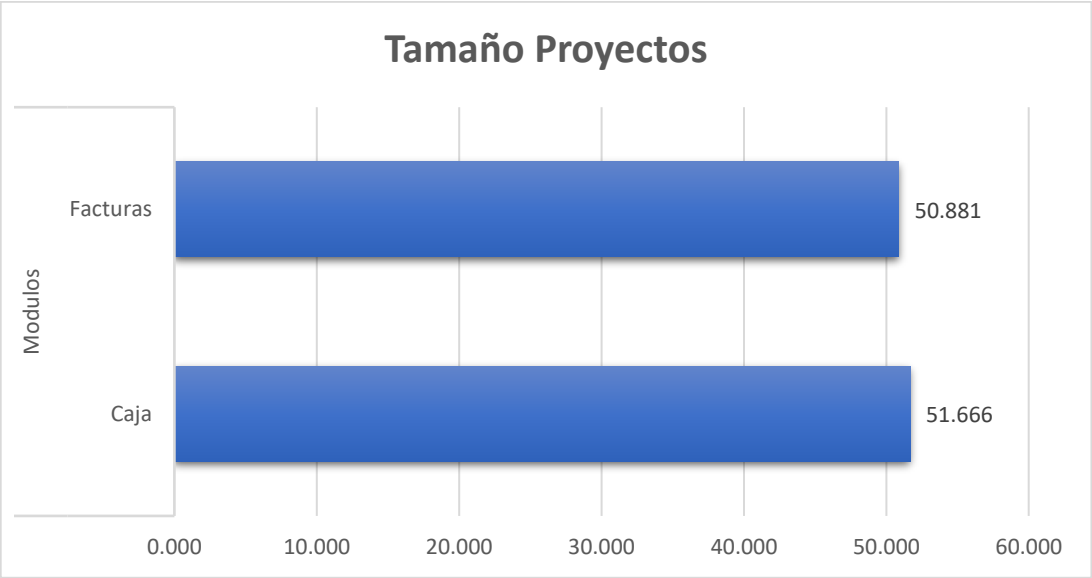


Figura 30. Tamaños del módulo de portal de facturas y caja.

CAPÍTULO V

RESUMEN, DISCUSIÓN, CONCLUSIONES Y RECOMENDACIONES

Resumen

Este capitulo contiene un resumen de la investigación realizada con respecto al desarrollo de un nuevo sistema institucional basado en una arquitectura orientada a microservicios que sea más ágil que el actual empleado en la Universidad de Montemorelos.

Discusión

Dentro del desarrollo ágil cabe destacar que, para la obtención del peso correspondiente a cada uno de los módulos, se encuentra todo documentado en las bitácoras mencionadas en los apéndices, que corresponden a la parte del backend de cada sistema; así como los procesos y las llamadas realizadas a los microservicios necesarios para su funcionamiento.

No se debe pasar por alto el hecho de que el equipo de desarrollo está compuesto por la misma persona en la elaboración de ambos sistemas, facilitando el poder responder a los cuestionamientos que se mencionan en la resolución de la metodología, comprobando que los factores ambientales están orientados hacia una misma persona.

Como se mencionó con anterioridad, al no existir formas de medir o metodologías enfocadas a una arquitectura de microservicios, se hizo el análisis basado en los

puntos de casos de uso, el cual ayudaría a resolver esta problemática. Durante el desarrollo, se aplicó el uso correcto de los primeros pasos de esta metodología hasta el momento de analizar los factores técnicos y ambientales donde se hicieron ajustes dependiendo de su funcionamiento, para que fueran de acuerdo con el uso de micro-servicios.

De la misma manera, los diagramas que fueron utilizados para documentar se ajustaron permitiendo que el cliente pueda definir de manera clara qué es lo que quiere y que, de la misma manera, el desarrollador entienda qué es lo que tiene que realizar.

Conclusiones

En cuanto a la implementación de esta arquitectura de microservicios en el sistema institucional del módulo de caja, se puede comprobar que ya está en uso. A partir del 7 de enero de 2019 se empezó a utilizar dicho sistema con esta nueva arquitectura; desde esa fecha, cualquier pago o anticipo que se abona a la enseñanza de algún estudiante o un pago que se realiza a algún otro servicio de los que ofrece la institución es realizado por este nuevo sistema.

Con respecto a las horas que fueron plasmadas en las bitácoras y el resultado obtenido durante la implementación de los puntos de casos de uso a cada uno de los módulos, se encontró lo siguiente: el módulo de portal de facturas tiene un peso menor al que tuvo el nuevo desarrollo del módulo de caja. En la Figura 31 se observa la diferencia notable entre las horas de desarrollo que tuvo cada uno. El módulo de caja cuenta con 111 horas de desarrollo, menos que el portal de facturas que cuenta con 202 horas.

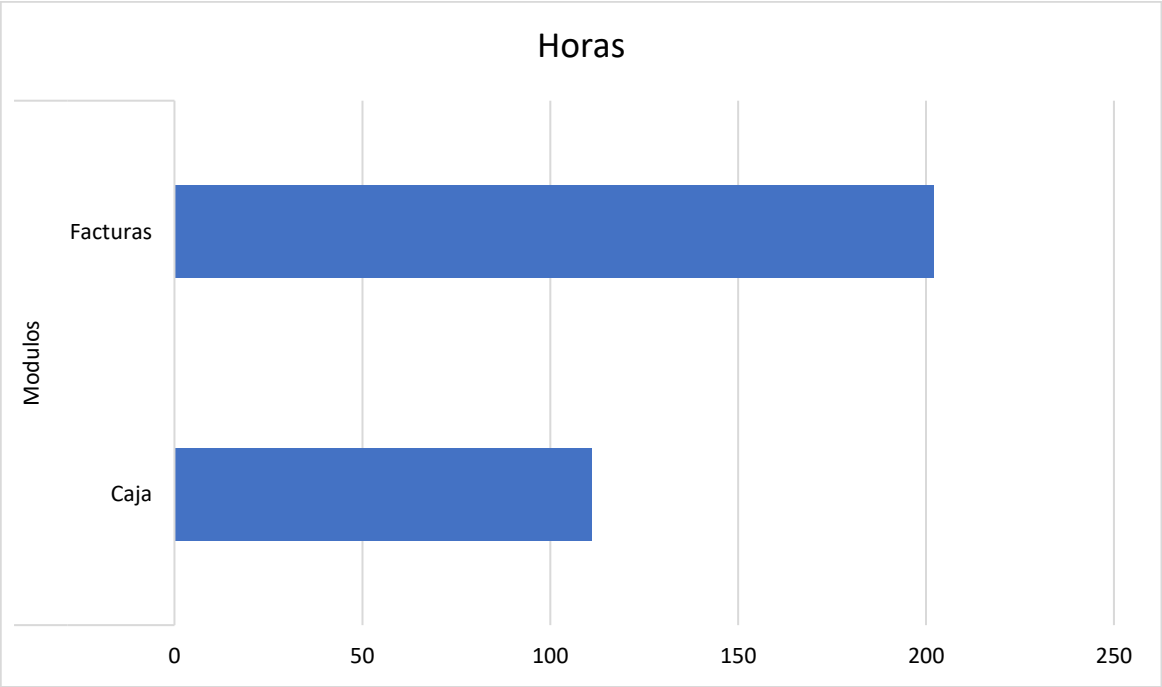


Figura 31. Horas de desarrollo de ambos módulos.

Entre las contribuciones que ofrece esta investigación están las siguientes:

1. La comprobación de que el desarrollo del nuevo sistema institucional de la Universidad de Montemorelos basado en una arquitectura de microservicios es mucho más ágil que un desarrollo convencional de una arquitectura monolítica.

2. El funcionamiento del módulo de caja en el sistema institucional de la Universidad de Montemorelos es más ágil.

3. Se potenció la integración de diagramas para la documentación de microservicios, al no existir un estándar en la actualidad.

4. Se implementaron nuevas tecnologías en la creación de microservicios de la Universidad de Montemorelos, tales como los servidores de Mulesoft y Express.

Recomendaciones

Se dan las siguientes recomendaciones para la elaboración de trabajos futuros:

1. El desarrollo de los demás módulos del sistema institucional de la Universidad de Montemorelos.

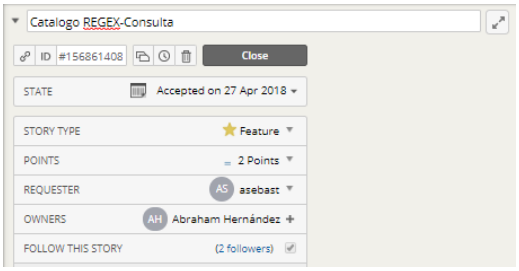
2. La implementación de nuevas tecnologías para la administración de los microservicios.

3. El perfeccionamiento de los diagramas utilizados para documentar los microservicios, así como el desarrollo de nuevos diagramas que permitan un mejor entendimiento de su funcionamiento.

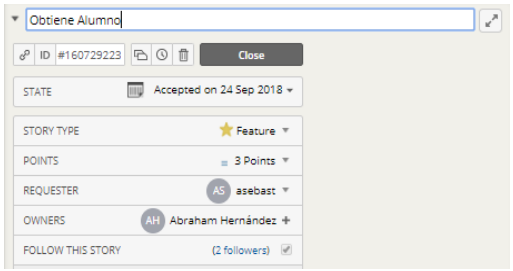
4. Estandarizar los microservicios.

APÉNDICE A

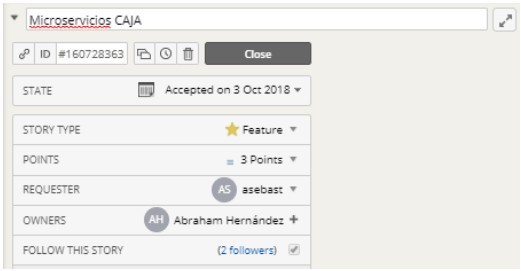
BITÁCORA DE HORAS DEL DESARROLLO DEL MÓDULO DE CAJA


Catalogo REGEX-Consulta
ID #156861408
Close
STATE Accepted on 27 Apr 2018
STORY TYPE Feature
POINTS 2 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

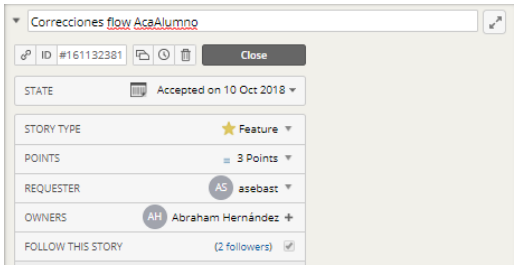
2


Obtiene Alumno
ID #160729223
Close
STATE Accepted on 24 Sep 2018
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

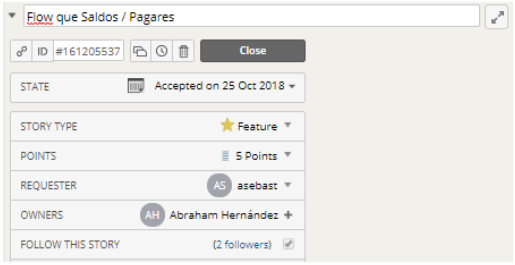
3


Microservicios CAJA
ID #160728363
Close
STATE Accepted on 3 Oct 2018
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

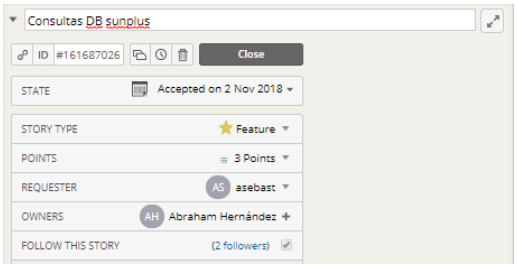
3


Correcciones flow AcaAlumno
ID #161132381
Close
STATE Accepted on 10 Oct 2018
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

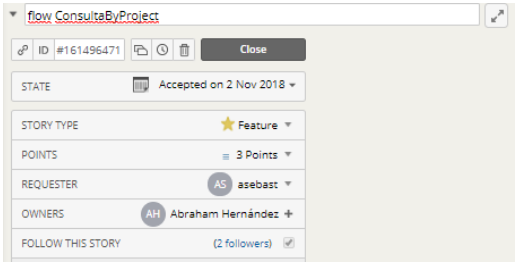
3


Flow que Saldos / Pagares
ID #161205537
Close
STATE Accepted on 25 Oct 2018
STORY TYPE Feature
POINTS 5 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

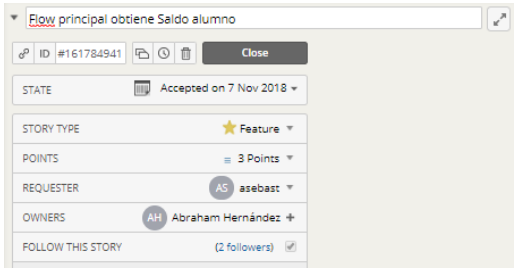
5


Consultas DB sunplus
ID #161687026
Close
STATE Accepted on 2 Nov 2018
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

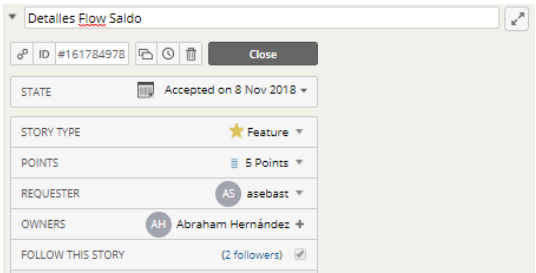
3


flow ConsultaByProject
ID #161496471
Close
STATE Accepted on 2 Nov 2018
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

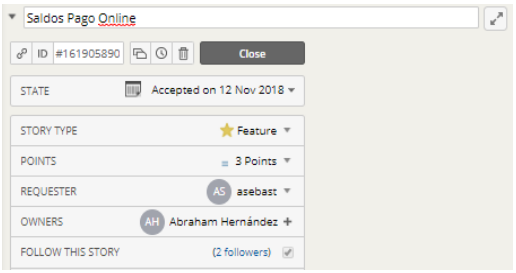
3


Flow principal obtiene Saldo alumno
ID #161784941
Close
STATE Accepted on 7 Nov 2018
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

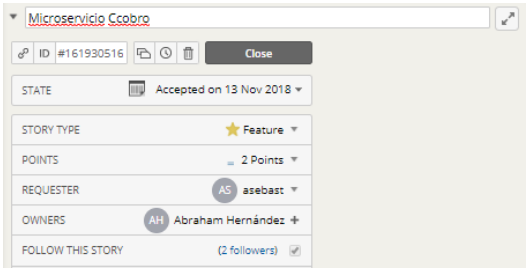
3


Detalles Flow Saldo
ID #161784978
Close
STATE Accepted on 8 Nov 2018
STORY TYPE Feature
POINTS 5 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

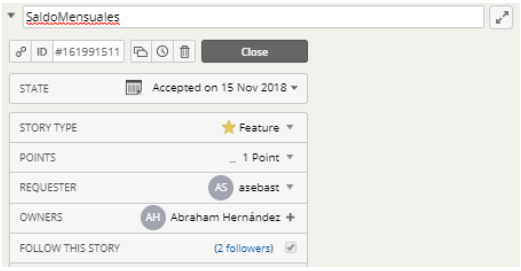
5


Saldos Pago Online
ID #161905890
Close
STATE Accepted on 12 Nov 2018
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

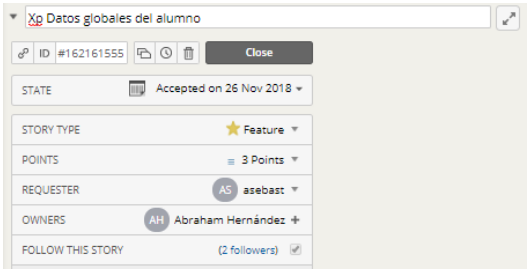
3


Microservicio Ccobro
ID #161930516
Close
STATE Accepted on 13 Nov 2018
STORY TYPE Feature
POINTS 2 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

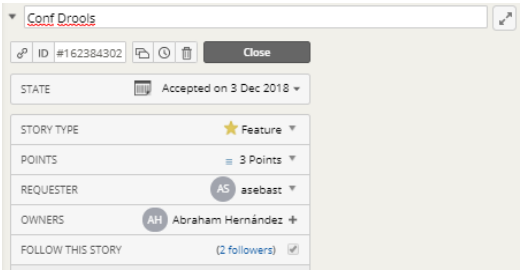
2


SaldoMensuales
ID #161991511
Close
STATE Accepted on 15 Nov 2018
STORY TYPE Feature
POINTS 1 Point
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

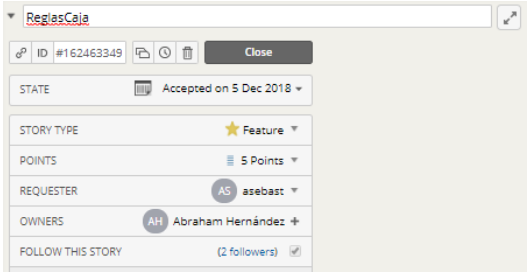
1


Xp Datos globales del alumno
ID #162161555
Close
STATE Accepted on 26 Nov 2018
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

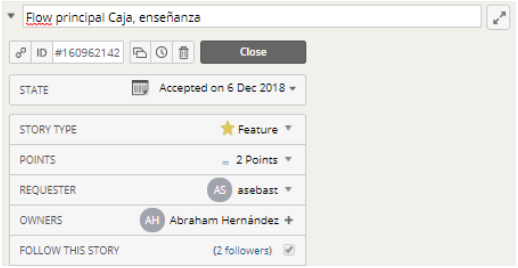
3


Conf Drools
ID #162384302
Close
STATE Accepted on 3 Dec 2018
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

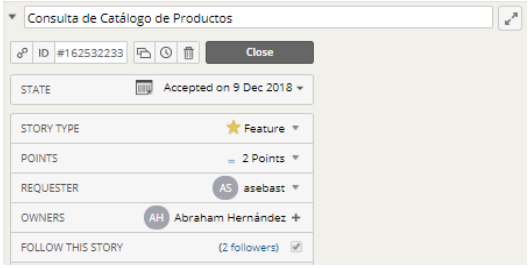
3


ReglasCaja
ID #162463349
Close
STATE Accepted on 5 Dec 2018
STORY TYPE Feature
POINTS 5 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

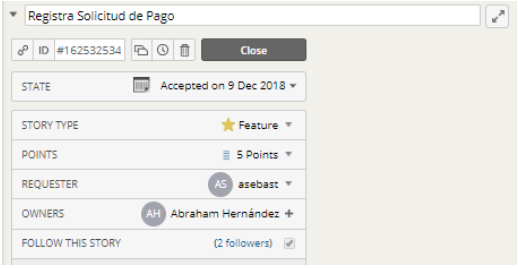
5


Flow principal Caja, enseñanza
ID #160962142
Close
STATE Accepted on 6 Dec 2018
STORY TYPE Feature
POINTS 2 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

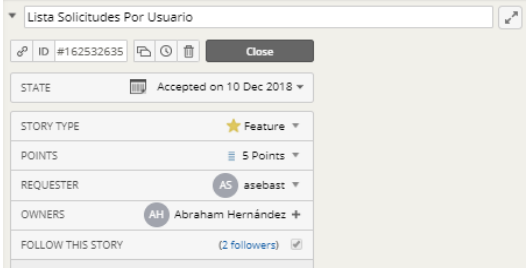
2


Consulta de Catálogo de Productos
ID #162532233
Close
STATE Accepted on 9 Dec 2018
STORY TYPE Feature
POINTS 2 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

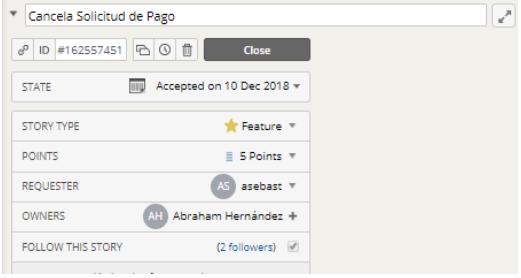
2


Registra Solicitud de Pago
ID #162532534
Close
STATE Accepted on 9 Dec 2018
STORY TYPE Feature
POINTS 5 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

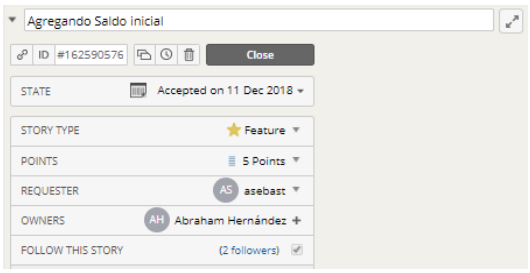
5


Lista Solicitudes Por Usuario
ID #162532635
Close
STATE Accepted on 10 Dec 2018
STORY TYPE Feature
POINTS 5 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

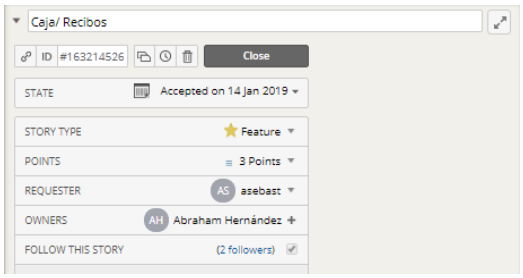
5


Cancela Solicitud de Pago
ID #162557451
Close
STATE Accepted on 10 Dec 2018
STORY TYPE Feature
POINTS 5 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

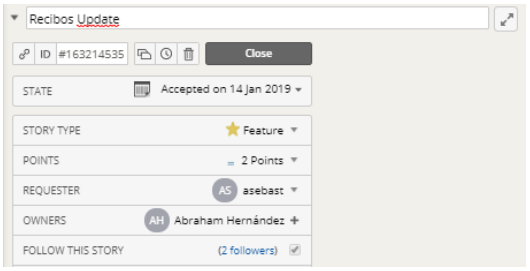
5


Agregando Saldo inicial
ID #162590576
Close
STATE Accepted on 11 Dec 2018
STORY TYPE Feature
POINTS 5 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

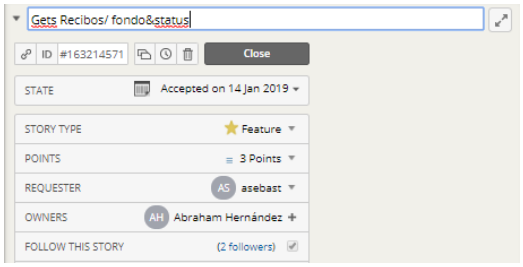
5


Caja/ Recibos
ID #163214526
Close
STATE Accepted on 14 Jan 2019
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

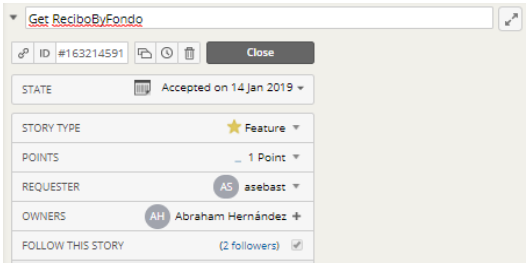
3


Recibos Update
ID #163214535
Close
STATE Accepted on 14 Jan 2019
STORY TYPE Feature
POINTS 2 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

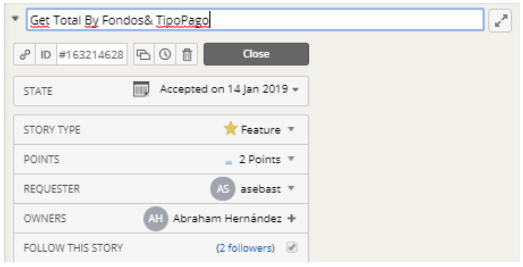
2


Gets Recibos/ fondo&status
ID #163214571
Close
STATE Accepted on 14 Jan 2019
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

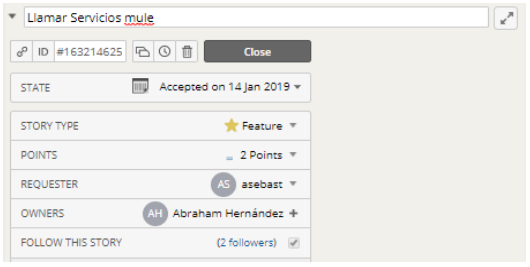
3


Get ReciboByFondo
ID #163214591
Close
STATE Accepted on 14 Jan 2019
STORY TYPE Feature
POINTS 1 Point
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

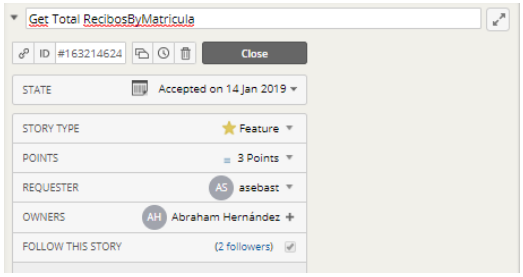
1


Get Total By Fondos& TipoPago
ID #163214628
Close
STATE Accepted on 14 Jan 2019
STORY TYPE Feature
POINTS 2 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

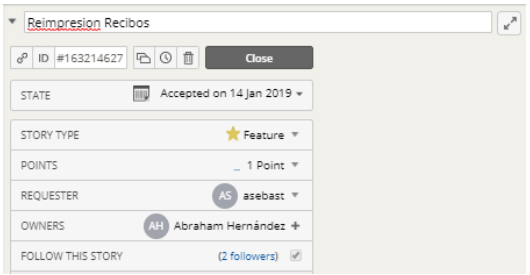
2


Llamar Servicios mule
ID #163214625
Close
STATE Accepted on 14 Jan 2019
STORY TYPE Feature
POINTS 2 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

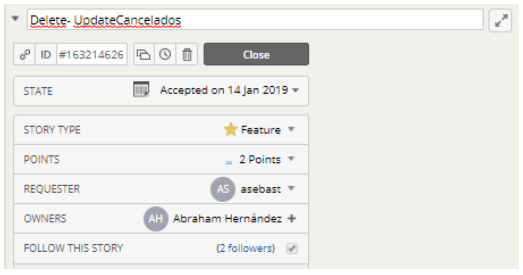
2


Get Total RecibosByMatricula
ID #163214624
Close
STATE Accepted on 14 Jan 2019
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

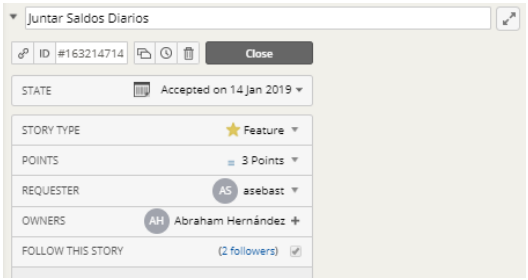
3


Reimpresion Recibos
ID #163214627
Close
STATE Accepted on 14 Jan 2019
STORY TYPE Feature
POINTS 1 Point
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

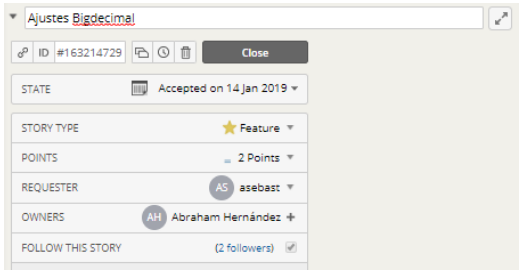
1


Delete- UpdateCancelados
ID #163214626
Close
STATE Accepted on 14 Jan 2019
STORY TYPE Feature
POINTS 2 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

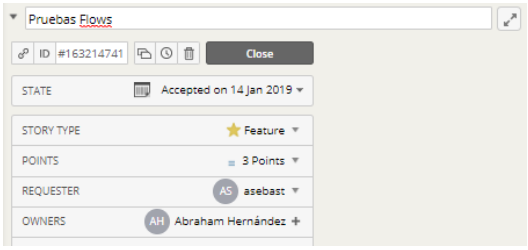
2


Juntar Saldos Diarios
ID #163214714
Close
STATE Accepted on 14 Jan 2019
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

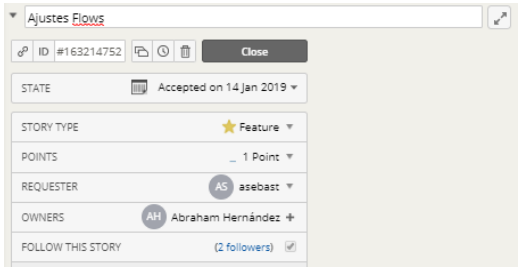
3


Ajustes Bigdecimal
ID #163214729
Close
STATE Accepted on 14 Jan 2019
STORY TYPE Feature
POINTS 2 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

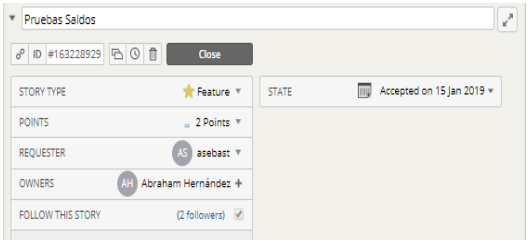
2


Pruebas Flows
ID #163214741
Close
STATE Accepted on 14 Jan 2019
STORY TYPE Feature
POINTS 3 Points
REQUESTER asebast
OWNERS Abraham Hernández

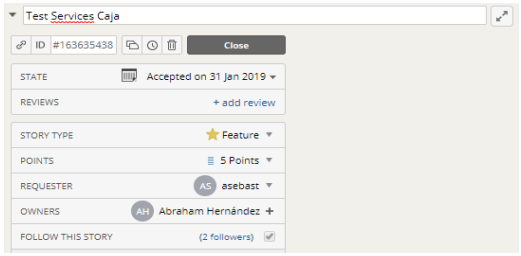
3


Ajustes Flows
ID #163214752
Close
STATE Accepted on 14 Jan 2019
STORY TYPE Feature
POINTS 1 Point
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

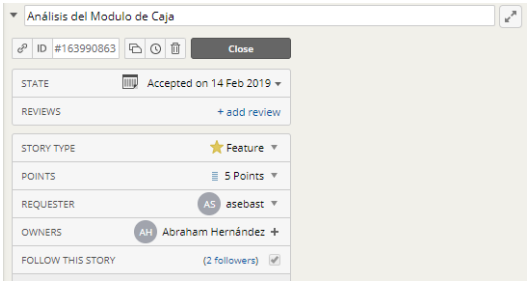
1


Pruebas Saldos
ID #163228929
Close
STORY TYPE Feature
STATE Accepted on 15 Jan 2019
POINTS 2 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

2


Test Services Caja
ID #163635438
Close
STATE Accepted on 31 Jan 2019
REVIEWS + add review
STORY TYPE Feature
POINTS 5 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

5


Análisis del Modulo de Caja
ID #163990863
Close
STATE Accepted on 14 Feb 2019
REVIEWS + add review
STORY TYPE Feature
POINTS 5 Points
REQUESTER asebast
OWNERS Abraham Hernández
FOLLOW THIS STORY (2 followers)

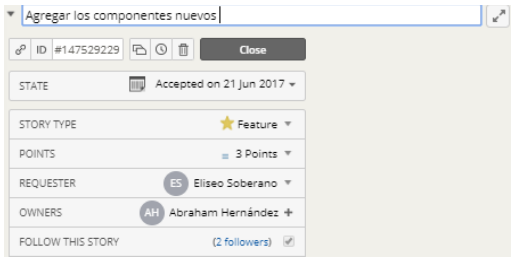
5

APÉNDICE B

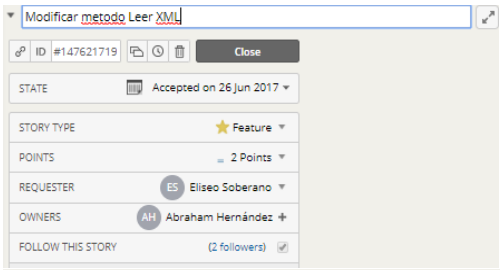
BITÁCORA DE HORAS DE DESARROLLO DEL MÓDULO DE PORTAL DE FACTURAS

Se dividen estas bitácoras en dos secciones (a) primera sección y (b) segunda sección. Se hizo de esta manera ya que en los detalles de las bitácoras mostradas aparece points, los cuales en la primera sección se había tomado la decisión que un punto correspondería a 3 horas de desarrollo. Después de cierta fecha se cambió ese entandar en que un punto correspondería a una hora de desarrollo, que son los que corresponden a la segunda sección.

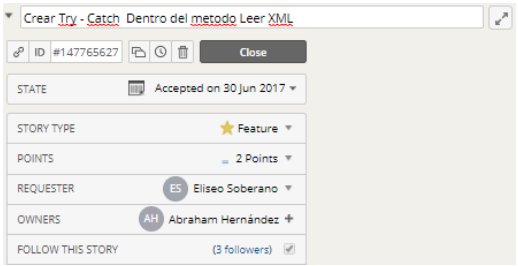
(a) Primera sección.



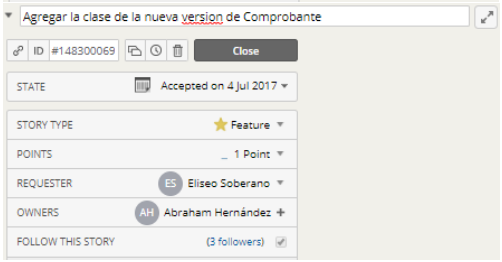
9



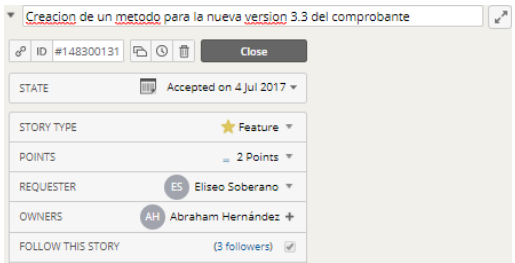
6



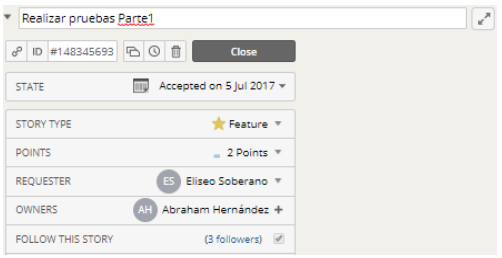
6



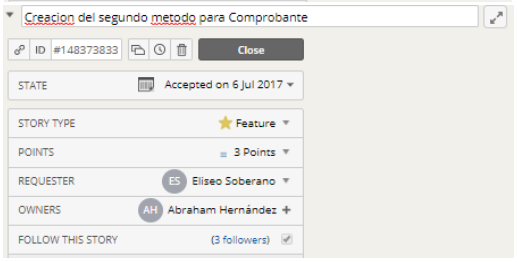
3



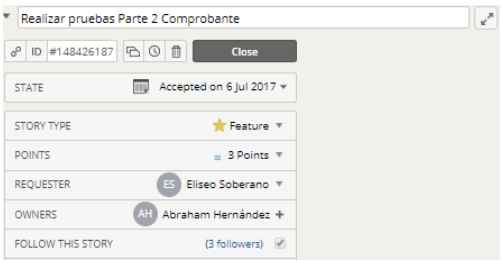
6



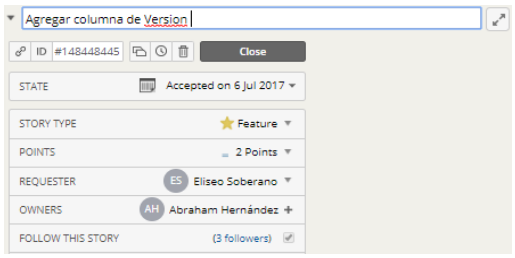
6



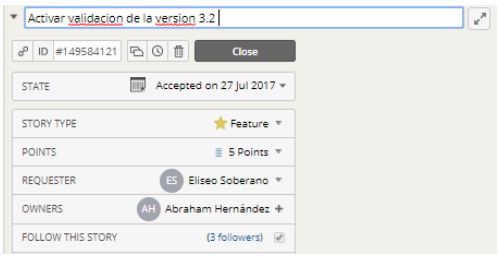
9



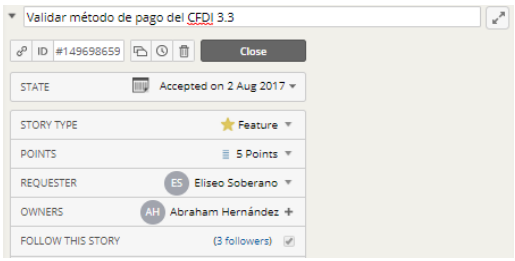
9



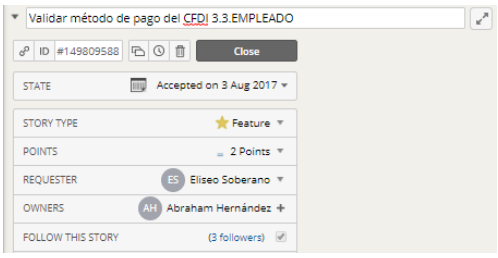
6



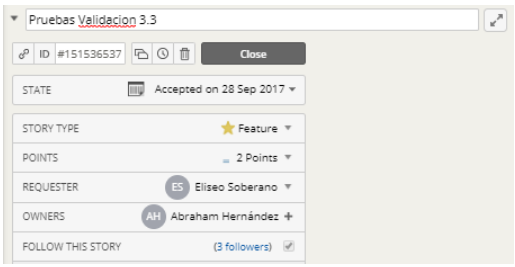
15



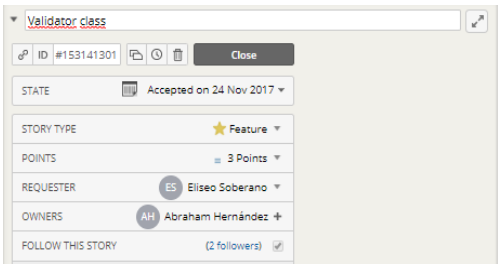
15



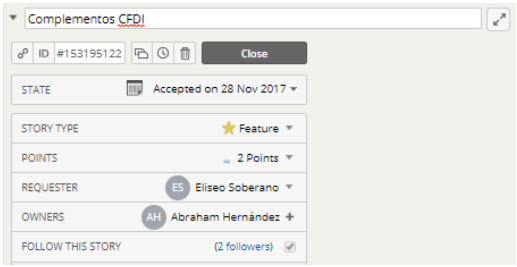
6



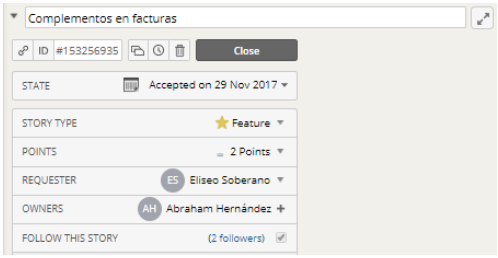
6



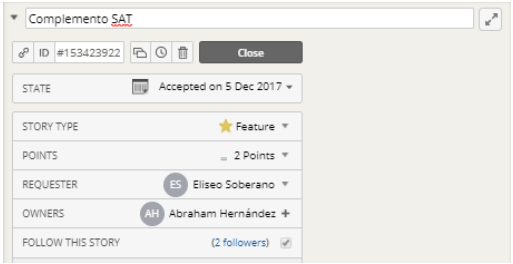
9



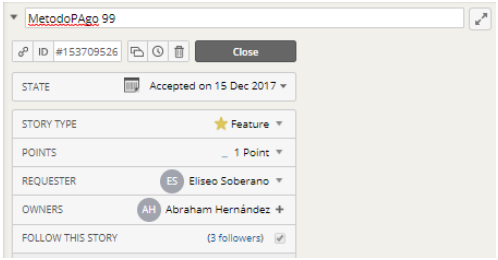
6



6

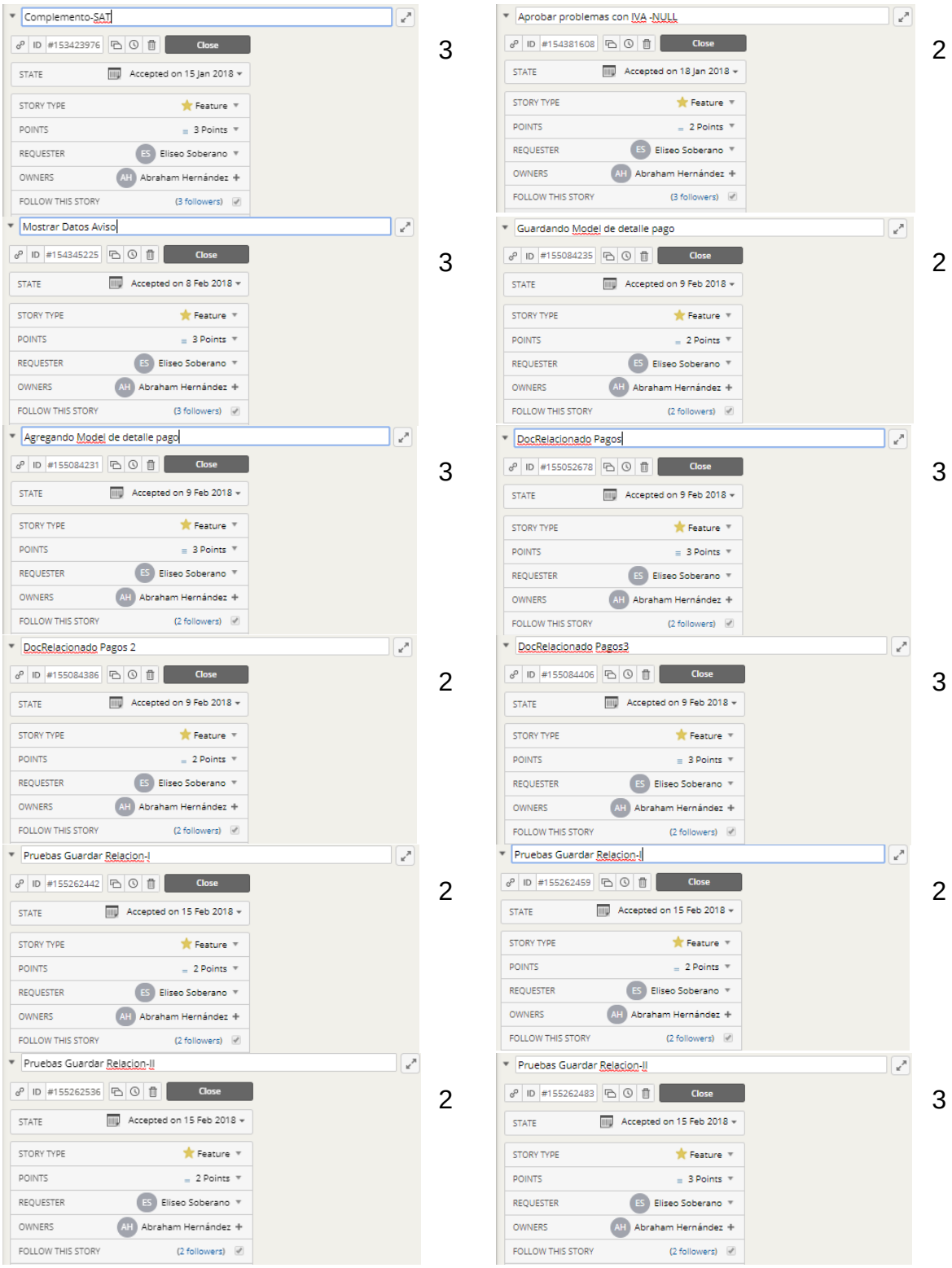


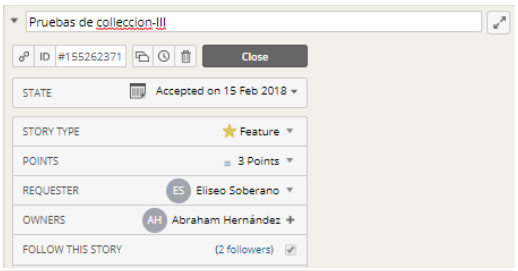
6



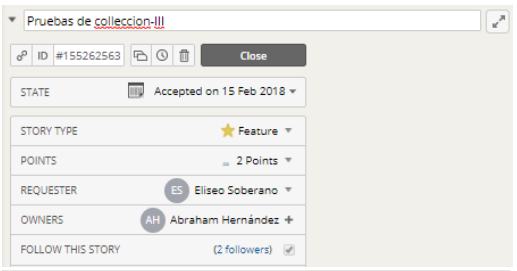
3

(b) Segunda sección.

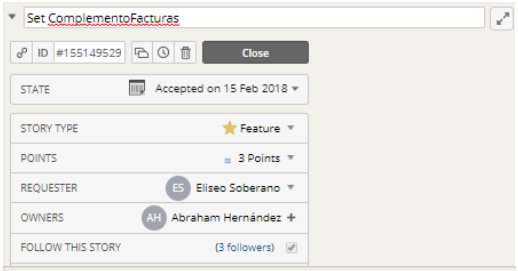




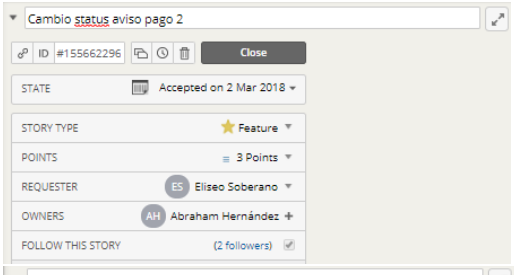
3



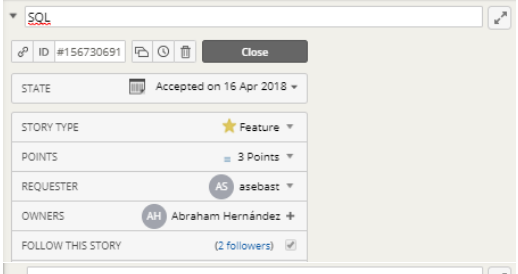
2



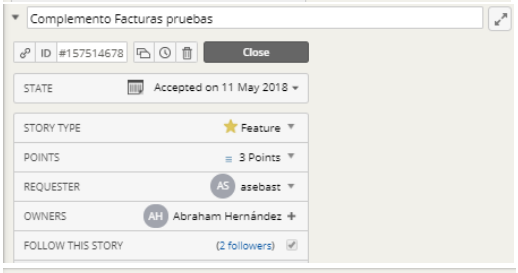
3



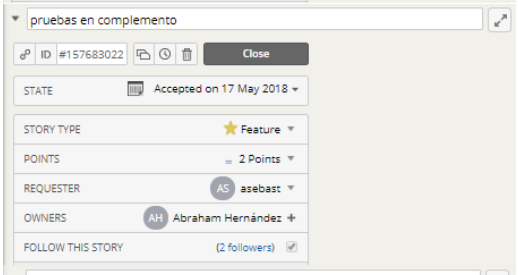
3



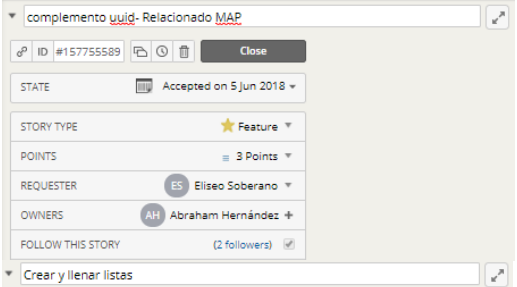
3



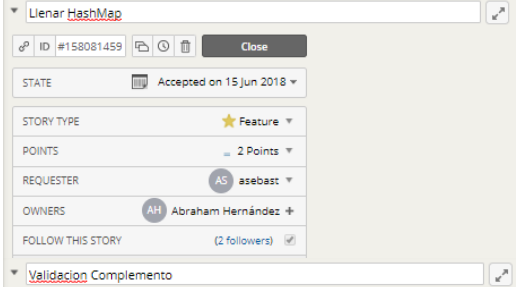
3



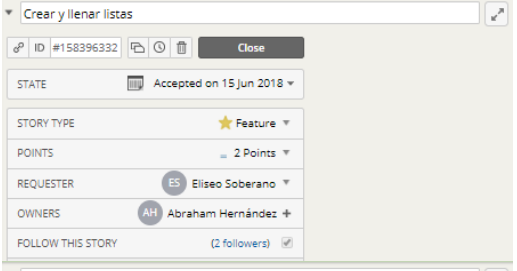
2



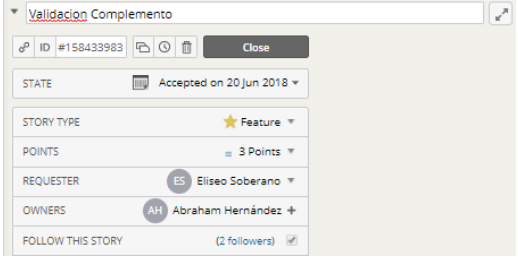
3



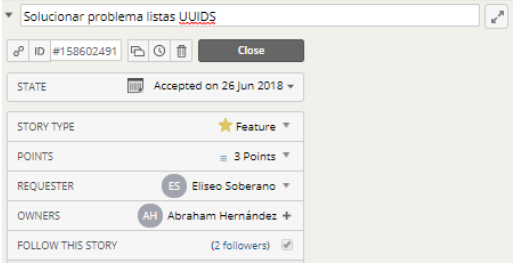
2



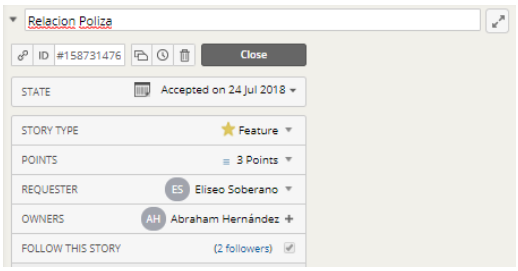
2



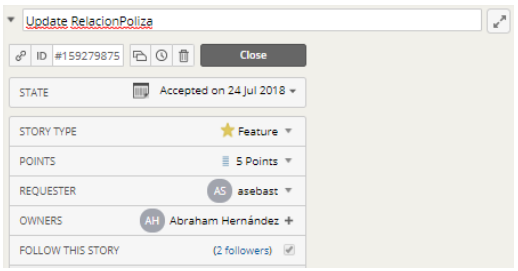
3



3


Relacion Poliza
ID #158731476
Close
STATE
Accepted on 24 Jul 2018
STORY TYPE
Feature
POINTS
3 Points
REQUESTER
ES
Eliseo Soberano
OWNERS
AH
Abraham Hernández
FOLLOW THIS STORY
(2 followers)

3


Update RelacionPoliza
ID #159279875
Close
STATE
Accepted on 24 Jul 2018
STORY TYPE
Feature
POINTS
5 Points
REQUESTER
AS
asebast
OWNERS
AH
Abraham Hernández
FOLLOW THIS STORY
(2 followers)

5

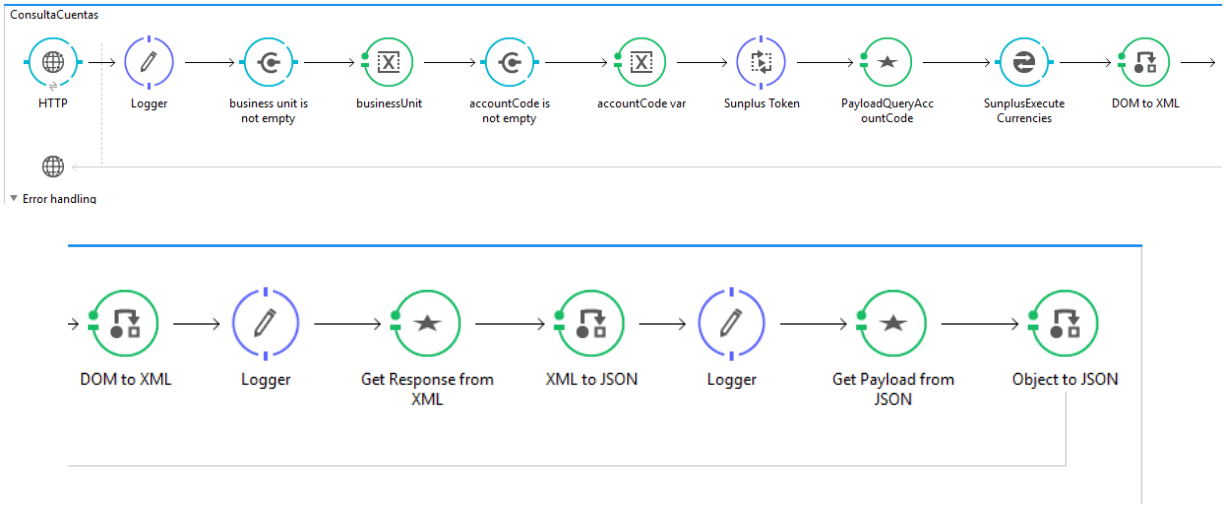
APÉNDICE C

MICROSERVICIOS DEL MÓDULO DE CAJA Y PORTAL DE FACTURAS

Se muestran los microservicios que fueron utilizados en el desarrollo del nuevo módulo de caja, se cuenta con dos servidores (a) Microservicios de MuleSoft y (b) Microservicios de Express. En la primera sección se dividen los microservicios en CORE, API y EXP. En la segunda sección solo se cuenta con microservicios API. En cada uno de ellos se agrega el nombre, una breve descripción de lo que hace, el método que utiliza, los parámetros que son necesarios para el funcionamiento y que es lo que retorna ese microservicio. De esta manera el desarrollador o alguna persona nueva que ocupe saber el funcionamiento de ellos pueda entenderlos.

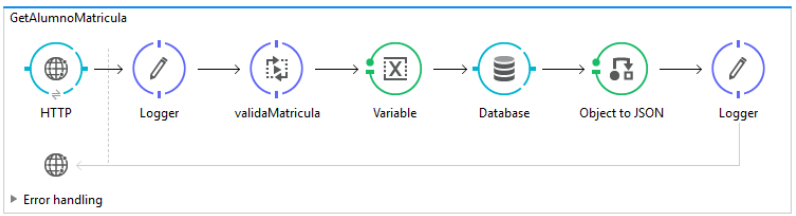
(a) Microservicios de MuleSoft - CORE

ConsultaCuentas



Este microservicio es el encargado de validar las cuentas y regresar cuales son las dimensiones que contiene dicha cuenta.	POST	Query businessUnit accountCode	{ "AccountCode": "133110", "AccountType": "4", "Description": "ANTICIPO PARA GASTOS", "EnterAnalysis1": "2", "EnterAnalysis10": "2", "EnterAnalysis2": "1", "EnterAnalysis3": "1", "EnterAnalysis4": "2", "EnterAnalysis5": "2", "EnterAnalysis6": "2", "EnterAnalysis7": "1", "EnterAnalysis8": "2", "EnterAnalysis9": "2"}

GetAlumnoMatricula



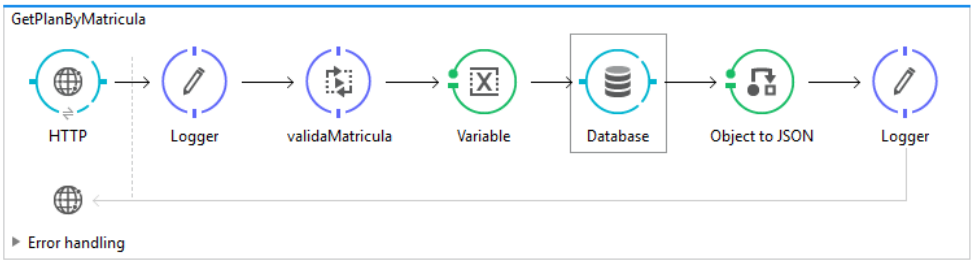
Este microservicio es el encargado de traer el alumno por medio de la matrícula.

GET

Matricula


```
[
  {
    "NOMBRE": "HERNANDEZ RIVADENEYRA ABRAHAM",
    "CODIGO_PERSONAL": "1090442"
  }
]
```


GetPlanByMatricula



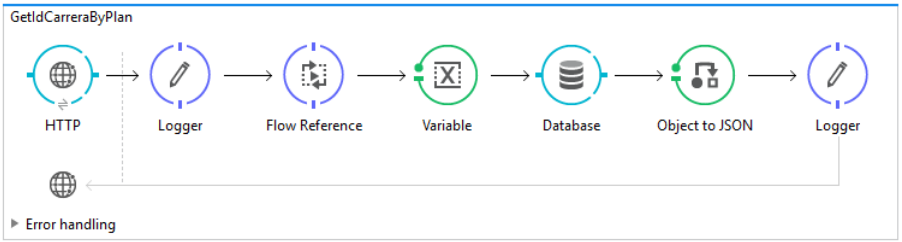
Este microservicio es el encargado de traer a que plan este inscrito cierto alumno por medio de su matrícula.

GET

Matricula


```
[
  {
    "CODIGO_PERSONAL": "1090442",
    "PLAN_ID": "MCCS2010"
  }
]
```


GetIdCarreraByPlan



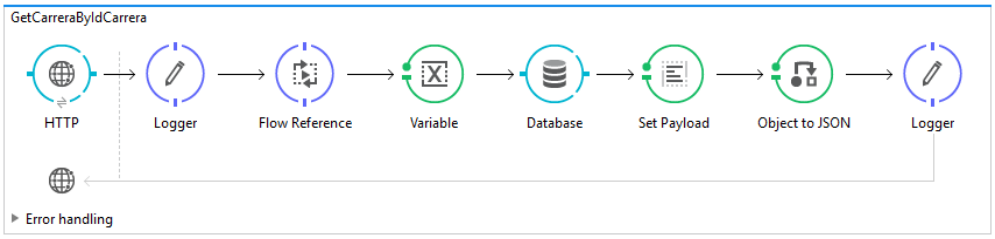
Este microservicio es el encargado de traer los id de carrera a los cuales está inscrito el alumno.

GET

PlanId


```
[ {
  "CAR": "10405",
  "NOMBRE_CARRERA": "Maestría en Ciencias Computacionales Área de Ingeniería de Software",
  "FACULTAD_ID": "104",
  "CARRERA_ID": "INMCCOM01",
  "PLAN_ID": "MCCS2010"
}]
```

GetCarreraByIdCarrera

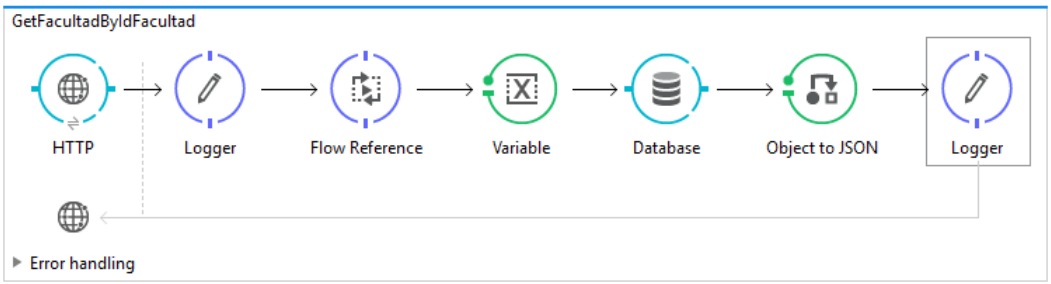


Este microservicio es el encargado de traer la carrera a la cual pertenece un estudiante por medio de la idCarrera.

GET CarreraId


```
{
  "CARRERA": "Maestría en Ciencias Computacionales con Acentuación en Tecnología Multimedia",
  "CARRERA_ID": "INMCCOM01",
  "FACULTAD_ID": "104",
  "CCOSTO_ID": "1.01.2.04.06"
}
```


GetFacultadByIdFacultad

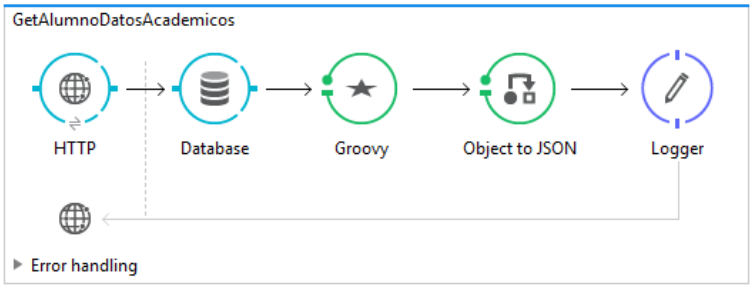


Este microservicio es el encargado de traer la facultad a la cual pertenece un estudiante por medio del idfacultad.

GET FacultadId


```
[
  {
    "NOMBRE_FACULTAD": "Ingeniería y Tecnología"
  }
]
```


GetAlumnoDatosAcademicos

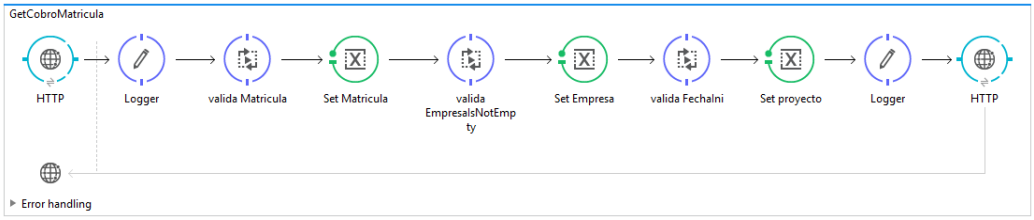


Este microservicio se encarga de traer el dato académico del dormitorio en el cual este dicho estudiante.

GET Matricula


```
[
  {
    "dormitorio": "1"
  }
]
```

GetCobroMatricula



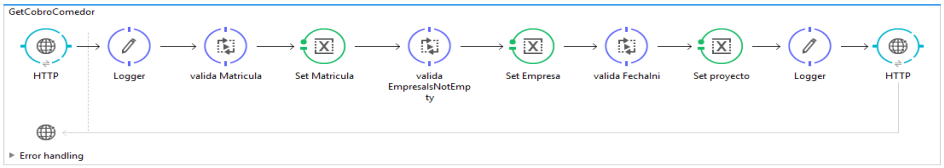
Este microservicio es el encargado de traer el saldo correspondiente de solo la matricula.

GET

Matricula
Empresa
Fechainicial

```
[
  {
    "importe": 0
  }
]
```

GetCobroComedor



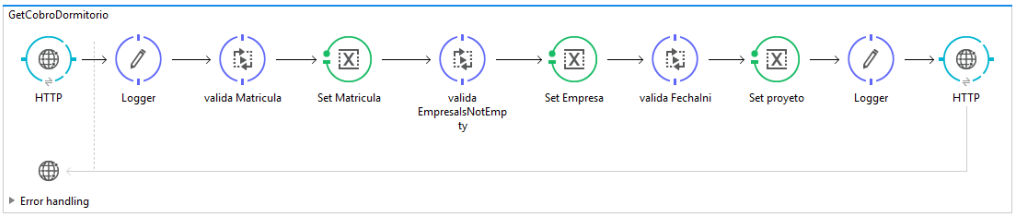
Este microservicio es el encargado de traer el saldo correspondiente de solo el comedor.

GET

Matricula
Empresa
Fechainicial

```
[
  {
    "importe": 0
  }
]
```

GetCobroDormitorio



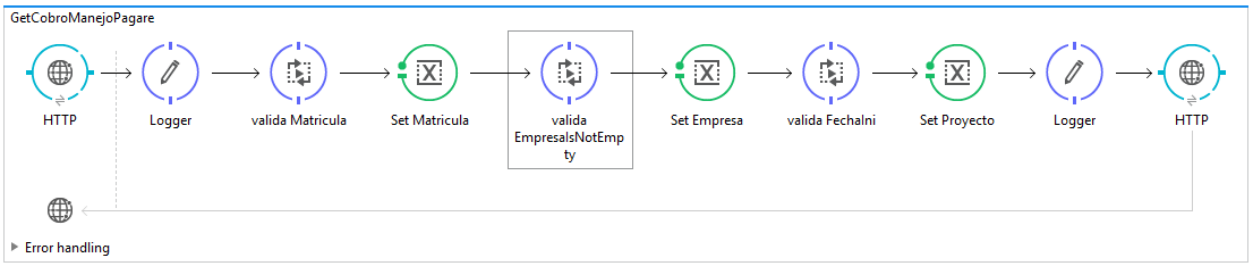
Este microservicio es el encargado de traer el saldo correspondiente de solo el dormitorio.

GET

Matricula
Empresa
Fechainicial

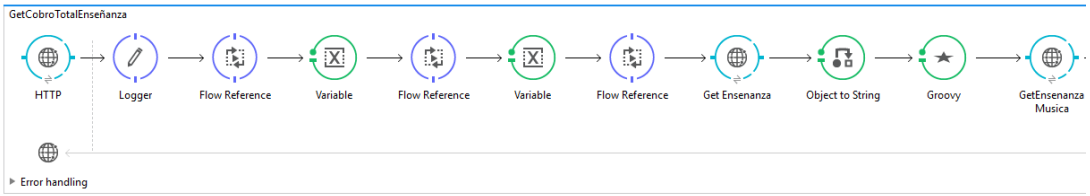
```
[
  {
    "importe": 0
  }
]
```

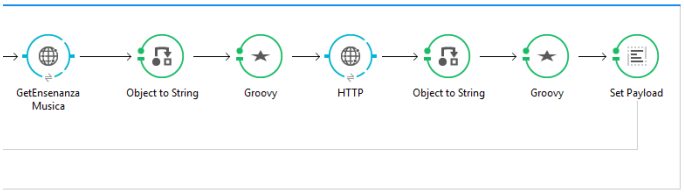
GetCobroManejoPagare



Este microservicio es el encargado de traer el saldo correspondiente de solo manejo de pagare	GET	Matricula
		Empresa
		Fechainicial
			[
			{
			"importe": 0
			}
			]

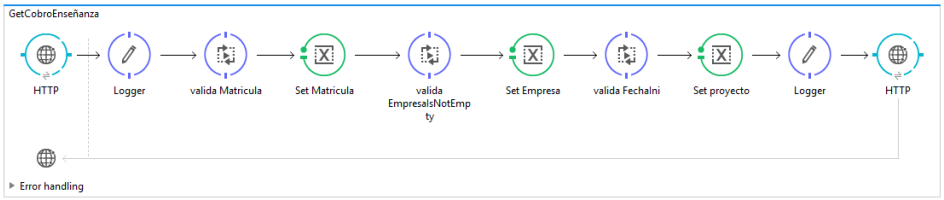
GetCobroTotalEnseñanza





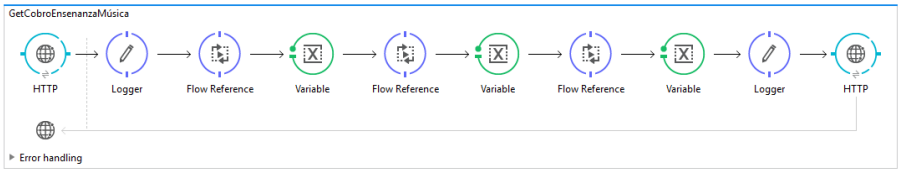
Este microservicio es el encargado de juntar el total de todos los saldos que tenga el alumno en cuanto a la enseñanza.	GET	Matricula
		Empresa
		Fechainicial
			{
			"ensenanza": 0.0
			}

GetCobroEnseñanza



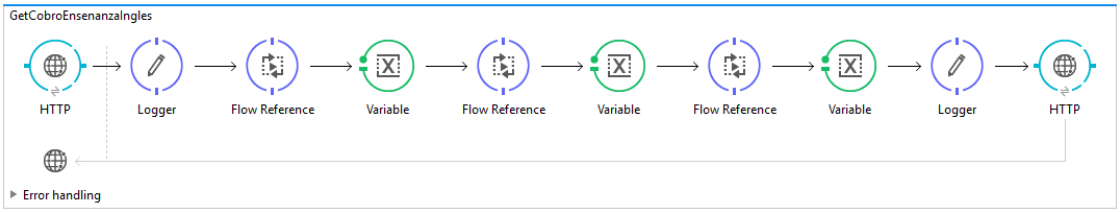
Este microservicio es el encargado de traer el saldo correspondiente a la pura enseñanza académica de alguna carrera.	GET	Matricula
		Empresa
		Fechainicial
			[
			{
			"importe": 0
			}
			]

GetCobroEnsenanzaMúsica



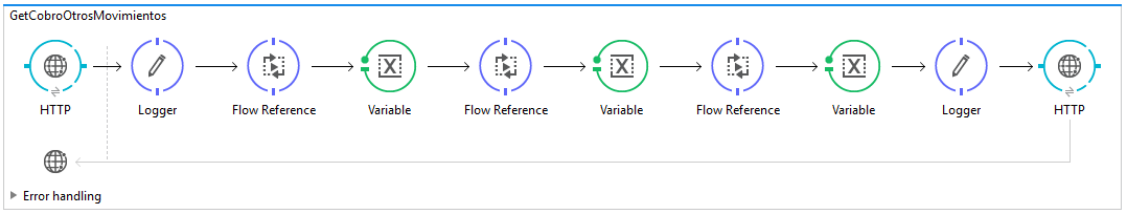
Este microservicio es el encargado de traer el saldo correspondiente a la enseñanza del conservatorio de música.	GET	Matricula
		Empresa
		Fechainicial
			[
			{
			"importe": 0
			}
			]

GetCobroEnsenanzaIngles



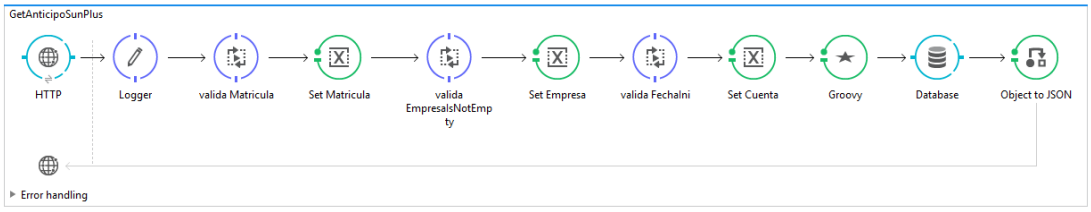
Este microservicio es el encargado de traer el saldo correspondiente a la enseñanza de idiomas.	GET	Matricula
		Empresa
		Fechainicial
			[
			{
			"importe": 0
			}
			]

GetCobroOtrosMovimientos



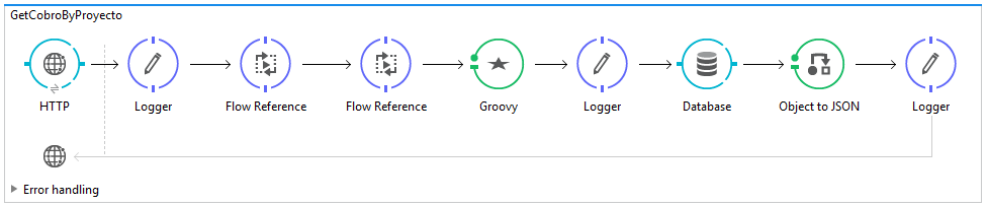
Este microservicio es el encargado de traer el saldo correspondiente a cuál movimiento o cargo que haya realizado el estudiante.	GET	Matricula
		Empresa
		Fechainicial
			[
			{
			"importe": 0
			}
			]

GetAnticipoSunPlus



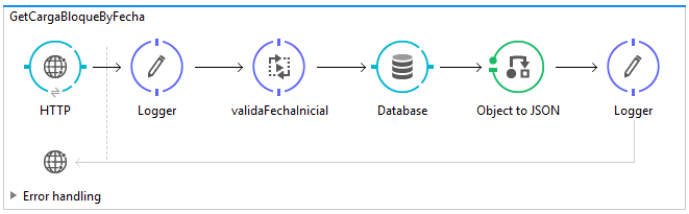
Este microservicio es el encargado de traer el saldo que el alumno tenga en su cuenta de anticipo.	GET	Matricula
		Empresa
		Fechainicial
			[
			{
			"importe": 0
			}
			]

GetCobroByProyecto



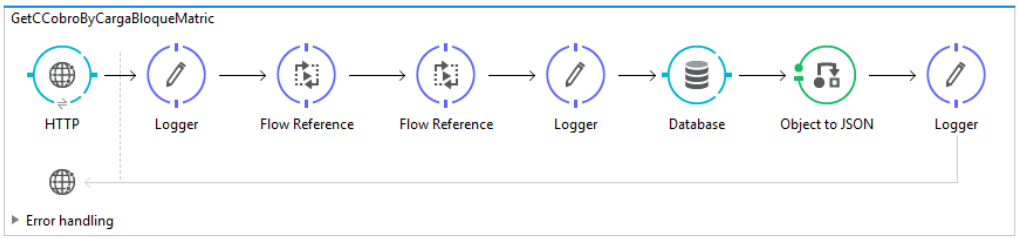
Este microservicio es el encargado de traer cualquier tipo de cobro dependiendo el proyecto que se le pida.	GET	Matricula Proyecto Empresa Fechainicial	[{ "importe": 0 }]

etCargaBloqueByFecha



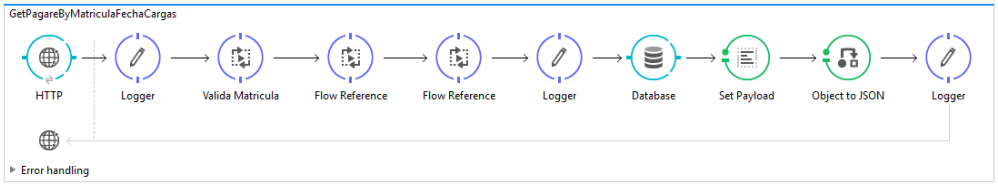
Encargado de traer todas las cargas que tenga activa el estudiante	GET	Fechainicial	[{
			"BLOQUE_ID": 3,
			"CARGA_ID": "17181B"
			},
			{
			"BLOQUE_ID": 1,
			"CARGA_ID": "18192B"
			}, {
			"BLOQUE_ID": 7,
			"CARGA_ID": "18191A"
			},
			{
			"BLOQUE_ID": 3,
			"CARGA_ID": "18191B"
			},
			{
			"BLOQUE_ID": 1,
			"CARGA_ID": "18195B"
			},
			{
			"BLOQUE_ID": 1,
			"CARGA_ID": "18191B"
			},
			{
			"BLOQUE_ID": 4,
			"CARGA_ID": "18191B"
			}, {
			"BLOQUE_ID": 1,
			"CARGA_ID": "18194B"
			},
			{
			"BLOQUE_ID": 1,
			"CARGA_ID": "18197B"
			},
			{
			"BLOQUE_ID": 1,
			"CARGA_ID": "18197A"
			},
			{
			"BLOQUE_ID": 2,
			"CARGA_ID": "18191B"
			}, {
			"BLOQUE_ID": 5,
			"CARGA_ID": "18191B"
			},
			{
			"BLOQUE_ID": 6,
			"CARGA_ID": "18191B"
			},
			{
			"BLOQUE_ID": 2,
			"CARGA_ID": "18194B"
			},
			{
			"BLOQUE_ID": 1,
			"CARGA_ID": "18193C"
			}]

GetCCobroByCargaBloqueMatric



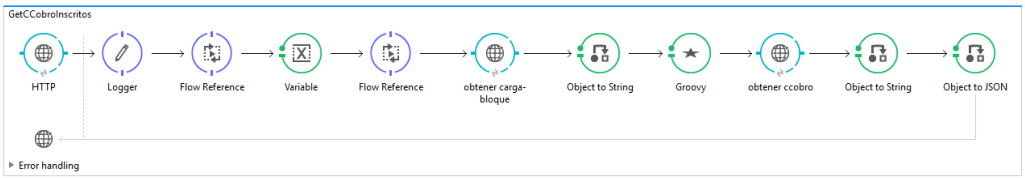
Este microservicio es el encargado de regresar los cálculos de cobros del estudiante.	GET	Matricula Cargabloque	[{ "NOMBRE_PLAN": "MAESTRÍA EN CIENCIAS COMPUTACIONALES CO", "FOLIO": "18191B10153", "CARGA_ID": "18191B", "ID": 289755, "PLAN_ID": "MCCS2010" }]

GetPagareByMatriculaFechaCargas



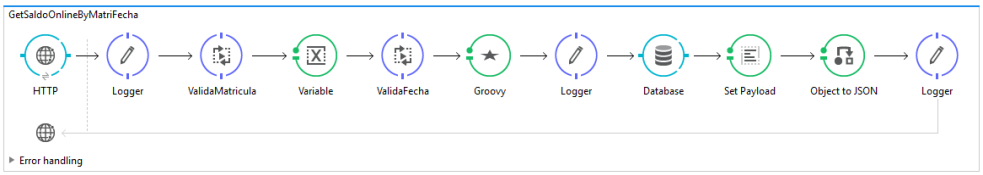
Este microservicio es el encargado de traer el saldo del pagare correspondiente al mes en que se solicite.	GET	Matricula
		Carga
		Fechainicial
		Fechafinal
			{
			"IMPORTE": 0
			}

GetCCobroInscritos



Este microservicio es el encargado de traer los cálculos de cobro a los cuales está inscrito el estudiante.	GET	Matricula
		Fechainicial
			[
			{
			"NOMBRE_PLAN": "MAESTRÍA EN CIENCIAS COMPUTACIONALES CO",
			"FOLIO": "18191B10153",
			"CARGA_ID": "18191B",
			"ID": 289755,
			"PLAN_ID": "MCCS2010"
			}
			]

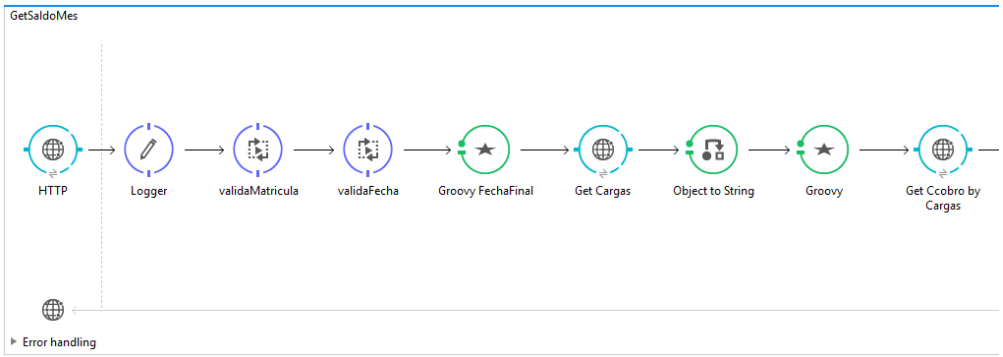
GetSaldoOnlineByMatriFecha

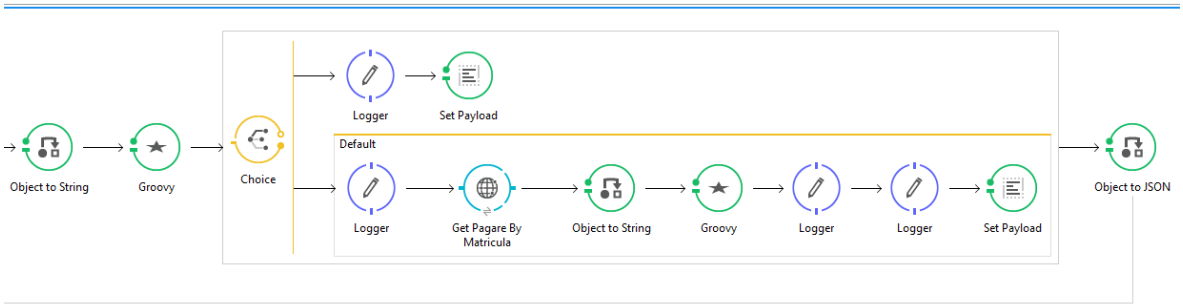


Este microservicio es el encargado de traer el saldo del estudiante si realizo algún deposito en línea.	GET	Matricula
		Fechainicial
			{
			"importesaldoonline": 0
			}

(a) Microservicios de MuleSoft - API

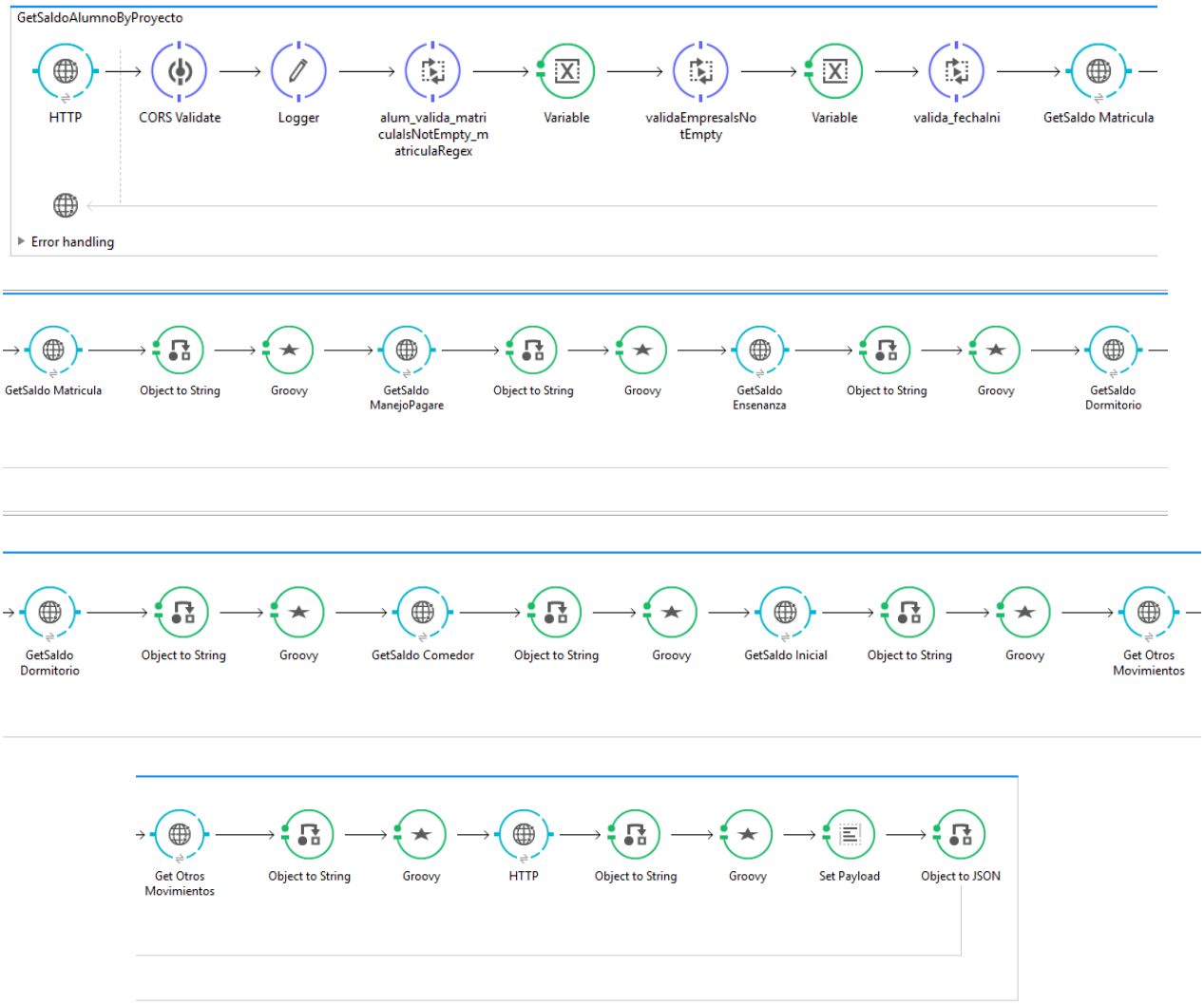
GetSaldoMes





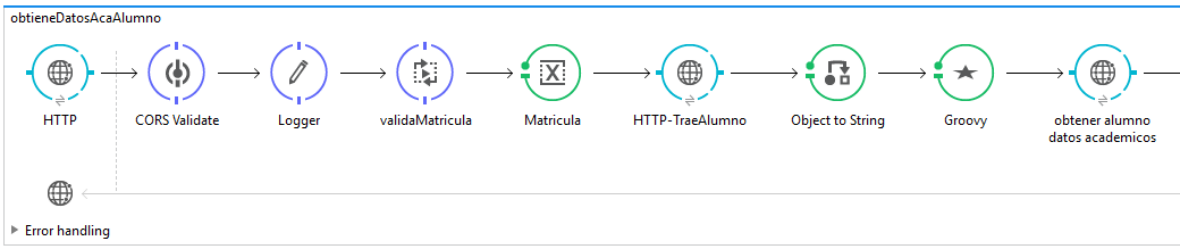
Este microservicio es el encargado de traer el saldo que tiene que pagar el alumno en el mes correspondiente.	GET	Matricula
		Fechainicial
			[
			{
			"saldoMes": 0
			}
			]

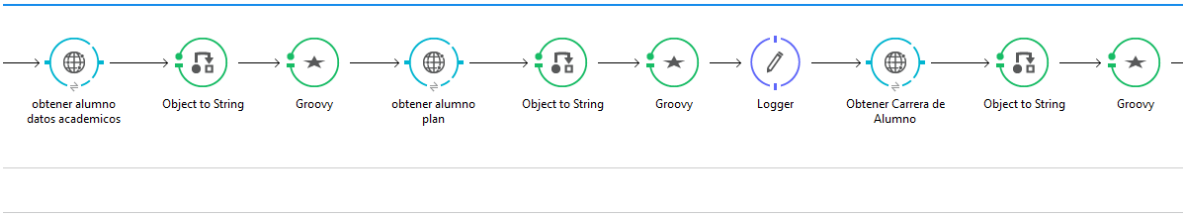
GetSaldoAlumnoByProyecto

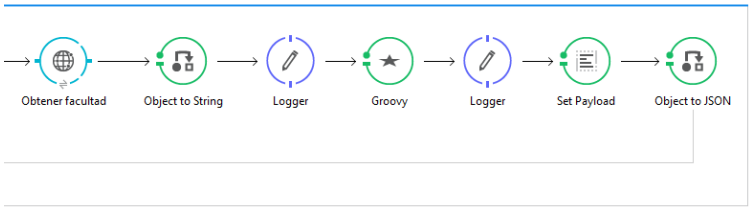


Este microservicio es el encargado de regresar un JSON con todos los saldos de proyectos que tenga el alumno.	GET	Matricula Fechainicial Fondo	{ "SaldoInicial": -1639.94, "Matricula": 0, "Ensenanza": 0, "Dormitorio": 0, "Comedor": 0, "OtrosMovimientos": 0, "saldoGlobalAlumno": 0 }

Get Datos Acá Alumno





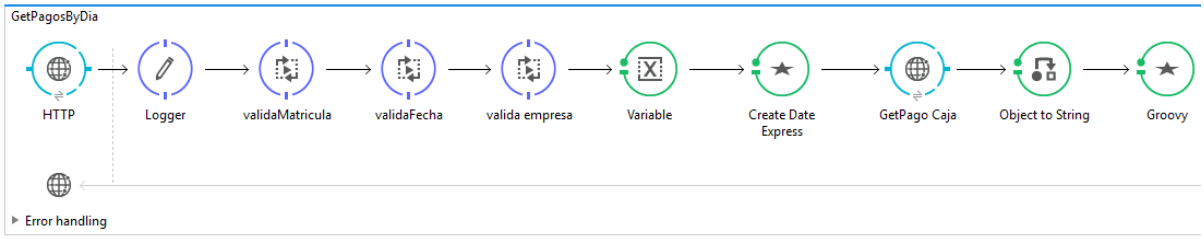


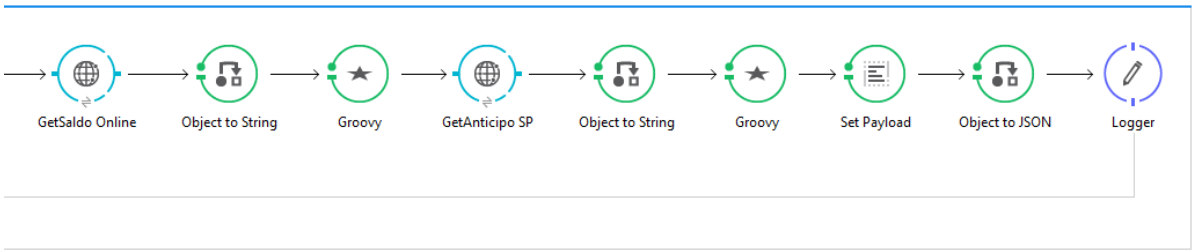
Este microservicio es el encargado de regresar un JSON con todos los datos académicos correspondiente a un alumno.

GET Matricula


```
[
  {
    "nombreCompleto": "HERNANDEZ RIVADENEYRA ABRAHAM",
    "matricula": "1090442",
    "planId": "MCCS2010",
    "carreraId": "INMCCOM01",
    "carrera": "Maestría en Ciencias Computacionales Área de Ingeniería de Software",
    "facultadId": "104",
    "facultad": "Ingeniería y Tecnología",
    "dormitorio": "1"
  }
]
```

GetPagosByDia



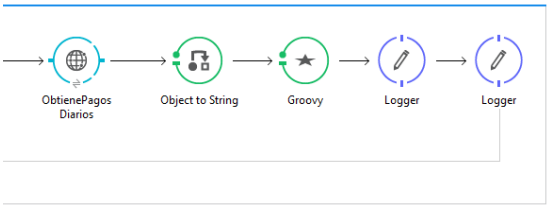


Este microservicio es el encargado de traer los saldos de que hayan realizado en el transcurso de un día.	GET	Matricula
		Fechainicial
		Empresa
			{
			"importeOnline": 0,
			"ensenanzaCaja": 0,
			"anticipoSp": 0
			}

(a) Microservicios de MuleSoft - EXP

Este Microservicio corresponde a la capa de APIs de experiencia donde son adaptados los microservicios para el uso de las aplicaciones (MuleSoft, 2019)

Datos Globales Alumno



Este microservicio es el encargado de juntar y llamar todos los microservicios que sean necesarios para mostrar los datos correspondientes de un estudiante.

GET Matricula

{ "cCobro": [{
"CARGA_ID": "18191B",
"FOLIO": "18191B10153",
"ID": 289755,
"NOMBRE_PLAN": "MAESTRÍA EN CIENCIAS COMPUTACIONALES CO",
"PLAN_ID": "MCCS2010"
}],
"datosAlumno": [{
"carrera": "Maestría en Ciencias Computacionales Área de Ingeniería de Software",
"carreraId": "INMCCOM01",
"dormitorio": "1",
"facultad": "Ingeniería y Tecnología",
"facultadId": "104",
"matricula": "1090442",
"nombreCompleto": "HERNANDEZ RIVADENEYRA ABRAHAM",
"planId": "MCCS2010"
}],
"pagoByProject": {
"Comedor": 0,
"Dormitorio": 0,
"Ensenanza": 0,
"Matricula": 0,
"OtrosMovimientos": 0,
"SaldoInicial": -1639.94,
"saldoGlobalAlumno": 0
},
"saldoDiario": {
"anticipoSp": 0,
"ensenanzaCaja": 0,
"importeOnline": 0
},
"saldoMensual": [{
"saldoMes": 0
}
],
}

(b) Microservicios de Expres – API

Create Recibo

```
router.post(
  '/recibo',
  (req, res, next) => {
    const caja = new Caja({
      numeroRecibo: req.body.numeroRecibo,
      contabilidad: req.body.contabilidad,
      pagoA: req.body.pagoA,
      tipoRecibo: req.body.tipoRecibo,
      fecha: req.body.fecha,
      estatus: req.body.estatus,
      totalPago: req.body.totalPago,
      username: req.body.username,
      alumnoProyectos: req.body.alumnoProyectos,
      movimientos: req.body.movimientos
    });
    caja.save();
    res.status(201)
      .json({
        recibo: {
          id: caja._id,
          numeroRecibo: caja.numeroRecibo,
          contabilidad: caja.contabilidad,
          pagoA: caja.pagoA,
          fecha: caja.fecha,
          estatus: caja.estatus,
          totalPago: caja.totalPago,
          username: caja.username,
          alumnoProyectos: caja.alumnoProyectos,
          movimientos: caja.movimientos
        },
        message: 'El recibo se grabo de manera exitosa'
      });
  });
```

Este microservicio es el encargado de crear el recibo cuando se realiza algún pago en cualquiera de las cajas.	POST	-	{ “recibo”: { "_id": "5c9105a9eaa20e2fb8dd6bba", "numeroRecibo": "217223", "contabilidad": "10", "pagoA": "Enseñanza", "fecha": "19032019", "estatus": "Activo", "totalPago": "1090.78", "username": "kiry", "alumnoProyectos": { "Matricula": "0", "Ensenanza": "-2410.1", "Dormitorio": "0", "Comedor": "0" }, "movimientos": [{ "alumno": { "matricula": "1180779", "nombreCompleto": "TAPIA GARCIA REBECCA SARAI", "facultadId": "107", "facultad": "Preparatoria", "carrera": "BACHILLERATO GENERAL EN HUMANIDADES Y COMUNICACIÓN", "carreraId": "INBACHI01", "planId": "PEHC2011", "dormitorio": "0" }, "tipoPago": [{ "_id": "5c9105a9eaa20e2fb8dd6bbc", "tipoPago": "TC", "importe": { "$numberDecimal": "1090.78" }, "fecha": "19032019", "numeroTarjeta": "", "banco": "", "numeroCheque": "", "cuentaCheque": "", "numeroReferencia": "" }], "capturaGenerica": [], "_id": "5c9105a9eaa20e2fb8dd6bbb", "cuentaContable": "S1180779", "descripcion": "Ensenanza", "origen": "SCOLPO01", "folioCC": "-1", "nombre": "TAPIA GARCIA REBECCA SARAI" }],}, “message”: 'El recibo se grabó de manera exitosa'}

DeleteReciboByFondo&Username

```
router.post(
  '/borrar/:fund/:username',
  (req, res, next) => {

    Caja.deleteMany({
      contabilidad: req.params.fund, username: req.params.username
    }).exec(function (err, resp) {
      if (err) {
        return res.status(402)
          .json({
            message: 'Error al intentar borrar los recibos'
          });
      } else {
        return res.status(201)
          .json({
            message: 'Los recibos se borraron de manera exitosa'
          });
      }
    });
  });
```

Este microservicio es el encargado de eliminar los recibos de alguno usuario y fondo.	POST	Fondo
		Username
			{
			“message”: 'Los recibos se borraron de manera exitosa'
			}

Update Recibos Cancelados By Id

```
router.post('/actualiza/recibos/cancelados/:id',
  (req, res, next) => {
    Caja.updateOne({
      _id: req.params.id
    },
      {
        $set: { estatus: "Cancelado", totalPago: "0", "movimientos.$[].tipoPago.$[].importe": "0" }
      })
      .then(result => {
        res.status(200)
          .json({
            message: 'Se actualizo con exito el recibo',
          })
      });
  });
```

Este microservicio es el encargado de actualizar recibos de estatus cancelados.	POST	Id	{
			"message": 'Se actualizo con exito el recibo',
			}

GetRecibos

```
router.get('/recibos',
  (req, res, next) => {
    Caja.find()
      .then(documents => {
        res.status(200)
          .json({
            message: 'Posts fetched succesfully',
            recibos: documents
          });
      });
  });
```

Este microservicio es el encargado mostrar el listado de todos los recibos existentes.

Nota: Se muestra solo un recibo de todo el listado.

GET -


```
{
  "alumnoProyectos": {
  "Matricula": "0",
  "Ensenanza": "0",
  "Dormitorio": "0",
  "Comedor": "0"
 },
  "_id": "5c92705aeaa20e2fb8dd718f",
  "numeroRecibo": "217309",
  "contabilidad": "10",
  "pagoA": "Enseñanza",
  "fecha": "20032019",
  "estatus": "Activo",
  "totalPago": "20000",
  "username": "pedroi",
  "movimientos": [
{
 "alumno": {
     "matricula": "1120121",
     "nombreCompleto": "CABRERA MARTINEZ STELL GE-
RALDINE",
     "facultadId": "103",
     "facultad": "Ciencias de la Salud",
     "carrera": "Licenciatura en Terapia Física y Rehabilitación",
     "carreraId": "INTERAP01",
     "planId": "TEFR2010",
     "dormitorio": "0"
  },
  "tipoPago": [
    {
     "_id": "5c92705aeaa20e2fb8dd7191",
     "tipoPago": "E",
  "importe": {
     "$numberDecimal": "20000"
     },
     "fecha": "20032019",
     "numeroTarjeta": "",
     "banco": "",
     "numeroCheque": "",
     "cuentaCheque": "",
     "numeroReferencia": ""
    }
  ],
  "capturaGenerica": [],
   "_id": "5c92705aeaa20e2fb8dd7190",
   "cuentaContable": "S1120121",
   "descripcion": "Ensenanza",
   "origen": "SPAGON01",
   "folioCC": "-1",
   "nombre": "CABRERA MARTINEZ STELL GERALDINE"
}
],
"__v": 0
},
…
```

GetRecibosByStatus

```
router.get('/recibos/by/estatus/:estatus', (req, res, next) => {
  Caja.find({
    estatus: req.params.estatus
  })
    .then(documents => {
      console.log(documents);
      res.status(200)
        .json({
          message: 'Posts fetched succesfully',
          recibos: documents.map(mov => {
            return {
              id: mov._id,
              numeroRecibo: mov.numeroRecibo,
              contabilidad: mov.contabilidad,
              pagoA: mov.pagoA,
              fecha: mov.fecha,
              estatus: mov.estatus,
              totalPago: mov.totalPago,
              username: mov.username,
              alumnoProyectos: mov.alumnoProyectos,
              movimientos: mov.movimientos.map(m => {
                return ({
                  alumno: m.alumno,
                  cuentaContable: m.cuentaContable,
                  descripcion: m.descripcion,
                  tipoPago: m.tipoPago.map(tp => {
                    return ({
                      tipoPago: tp.tipoPago,
                      importe: tp.importe,
                      fecha: tp.fecha,
                      numeroTarjeta: tp.numeroTarjeta,
                      banco: tp.banco,
                      numeroCheque: tp.numeroCheque,
                      cuentaCheque: tp.cuentaCheque,
                      numeroReferencia: tp.numeroReferencia
                    })
                  }),
                  origen: m.origen,
                  folioCC: m.folioCC,
                })
              })
            }
          })
        });
    });
});
```

Este microservicio es el encargado de mostrar los recibos de acuerdo con el estatus solicitado, muestra todos los datos, en este caso se mostró un recibo con el estatus de Activo.

Nota: Se muestra solo un recibo de todo el listado.

GET Estatus


```
{  "alumnoProyectos": {
  "Matricula": "0",
  "Ensenanza": "0",
  "Dormitorio": "0",
  "Comedor": "0"
 },  "_id": "5c92705aeaa20e2fb8dd718f",
  "numeroRecibo": "217309",
  "contabilidad": "10",
  "pagoA": "Enseñanza",
  "fecha": "20032019",
  "estatus": "Activo",
  "totalPago": "20000",
  "username": "pedroi",
  "movimientos": [
{ "alumno": {
     "matricula": "1120121",
     "nombreCompleto": "CABRERA MARTINEZ STELL GERALDINE",
     "facultadId": "103",
     "facultad": "Ciencias de la Salud",
     "carrera": "Licenciatura en Terapia Física y Rehabilitación",
     "carreraId": "INTERAP01",
     "planId": "TEFR2010",
     "dormitorio": "0"
   }, "tipoPago": [
    {     "_id": "5c92705aeaa20e2fb8dd7191",
     "tipoPago": "E",
    "importe": {
     "$numberDecimal": "20000"
     },  "fecha": "20032019",
     "numeroTarjeta": "",
     "banco": "",
     "numeroCheque": "",
     "cuentaCheque": "",
     "numeroReferencia": ""
    } ],
   "capturaGenerica": [],
    "_id": "5c92705aeaa20e2fb8dd7190",
    "cuentaContable": "S1120121",
    "descripcion": "Ensenanza",
    "origen": "SPAGON01",
    "folioCC": "-1",
    "nombre": "CABRERA MARTINEZ STELL GERALDINE"
}],"__v": 0},…
```

GetReciboByFondo&Name

```
router.get('/recibos/by/contabilidad/:contabilidad/:username', (req, res, next) => {
  Caja.find({
    contabilidad: req.params.contabilidad,
    username: req.params.username
  })
    .then(documents => {
      res.status(200)
        .json({
          message: 'Posts fetched succesfully',
          recibos: documents.map(mov => {
            return {
              id: mov._id,
              numeroRecibo: mov.numeroRecibo,
              contabilidad: mov.contabilidad,
              pagoA: mov.pagoA,
              fecha: mov.fecha,
              estatus: mov.estatus,
              totalPago: mov.totalPago,
              username: mov.username,
              alumnoProyectos: mov.alumnoProyectos,
              movimientos: mov.movimientos
            }
          })
        })
    })
```

Este microservicio es el encargado de mostrar todo el listado de recibos existentes por fondo, estatus y username

Nota: Solo se muestra un recibo de todo el listado.

GET

Fondo
Username


```
{
  "id": "5c92705aeaa20e2fb8dd718f",
  "numeroRecibo": "217309",
  "contabilidad": "10",
  "pagoA": "Enseñanza",
  "fecha": "20032019",
  "estatus": "Activo",
  "totalPago": "20000",
  "username": "pedroi",
  "alumnoProyectos": {
     "Matricula": "0",
     "Ensenanza": "0",
     "Dormitorio": "0",
     "Comedor": "0"
  },
  "movimientos": [
   {
  "alumno": {
  "matricula": "1120121",
  "nombreCompleto": "CABRERA MARTINEZ STELL GERALDINE",
  "facultadId": "103",
  "facultad": "Ciencias de la Salud",
  "carrera": "Licenciatura en Terapia Física y Rehabilitación",
  "carreraId": "INTERAP01",
  "planId": "TEFR2010",
  "dormitorio": "0"
  },
  "tipoPago": [
   {
   "_id": "5c92705aeaa20e2fb8dd7191",
   "tipoPago": "E",
   "importe": {
     "$numberDecimal": "20000"
     },
   "fecha": "20032019",
   "numeroTarjeta": "",
   "banco": "",
   "numeroCheque": "",
   "cuentaCheque": "",
   "numeroReferencia": ""
   }
  ],
  "capturaGenerica": [],
  "_id": "5c92705aeaa20e2fb8dd7190",
  "cuentaContable": "S1120121",
  "descripcion": "Ensenanza",
  "origen": "SPAGON01",
  "folioCC": "-1",
  "nombre": "CABRERA MARTINEZ STELL GERALDINE"
  }
  ],
},
```

GetTotalByFondo&User

```
router.get('/total/by/contabilidad/sumatotal/:fondo/:user', (req, res, next) => {
  Caja.aggregate([{
    $project: {
      _id: 1,
      numeroRecibo: 1,
      contabilidad: 1,
      "movimientos.tipoPago.tipoPago": 1,
      "movimientos.tipoPago.importe": 1,
      username: 1
    },
  },
  {
    $unwind: "$movimientos"
  },
  {
    $unwind: "$movimientos.tipoPago"
  },
  {
    $match: {
      $and: [{
        contabilidad: req.params.fondo
      }, {
        username: req.params.user
      }]
    }
  },
  {
    $group: {
      _id: {},
      total: {
        $sum: "$movimientos.tipoPago.importe"
      }
    }
  }
  ])
    .then(documents => {

      res.status(200)
        .json({
          message: 'Posts fetched succesfully',
          recibos: documents.map(t => {
            return {
              total: t.total.toString()
            }
          }),
        });
    });
});
```

Este microservicio es el encargado de mostrar el total que hay por fondo y usuario, un fondo de académico y user.	GET	Fondo
		User
			{
			"message": "Posts fetched succesfully",
			"recibos": [
			{
			"total": "357058.68"
			}
			],
			}

GetReciboBynumRecibo

```
router.get('/recibo/:numRecibo',
  (req, res, next) => {
    Caja.findById(req.params.numRecibo)
      .then(recibo => {
        if (recibo) {
          res.status(200)
            .json({
              message: 'Recibo encontrado con exito!',
              recibo: recibo,
            });
        } else {
          res.status(404)
            .json({
              message: 'Recibo no encontrado!'
            });
        }
      })
  })
```

Este microservicio es el encargado de traer el recibo por medio de numero de recibo que asignado.

GET numRecibo


```
{
"id": "5c92751ceaa20e2fb8dd7208",
 "numeroRecibo": "217313",
 "contabilidad": "10",
 "pagoA": "Enseñanza",
 "fecha": "20032019",
 "estatus": "Activo",
 "totalPago": "7500",
 "username": "pedroi",
 "alumnoProyectos": {
  "Matricula": "50",
  "Ensenanza": "-1648.42",
  "Dormitorio": "0",
  "Comedor": "0"
  }, "movimientos": [
   {   "alumno": {
  "matricula": "1180591",
  "nombreCompleto": "SABIDO CARRILLO MARIEL ESPERANZA",
  "facultadId": "107",
  "facultad": "Preparatoria",
  "carrera": "BACHILLERATO GENERAL BILINGÜE EN HUMANIDADES
Y COMUNICACIÓN",
  "carreraId": "INBACHI01",
  "planId": "PBHC2011",
  "dormitorio": "0"
  }, "tipoPago": [
 {   "_id": "5c92751ceaa20e2fb8dd720a",
  "tipoPago": "T",
  "importe": {
    "$numberDecimal": "7500"
   },
  "fecha": "20032019",
  "numeroTarjeta": "",
  "banco": "",
  "numeroCheque": "",
  "cuentaCheque": "",
  "numeroReferencia": ""
  } ],
 "capturaGenerica": [],
   "_id": "5c92751ceaa20e2fb8dd7209",
   "cuentaContable": "S1180591",
   "descripcion": "Ensenanza",
   "origen": "SPAGON01",
   "folioCC": "-1",
   "nombre": "SABIDO CARRILLO MARIEL ESPERANZA"
  }
 ],
},
```

GetTotalByTipoPagoFondo&Username

```
router.get('/total/by/tipo-pago/:tipoPago/:fondo/:username', (req, res, next) => {
  Caja.aggregate([{
    $project: {
      _id: 1,
      numeroRecibo: 1,
      contabilidad: 1,
      "movimientos.tipoPago.tipoPago": 1,
      "movimientos.tipoPago.importe": 1,
      username: 1,
    },
  },  {
    $unwind: "$movimientos"
  },  {
    $unwind: "$movimientos.tipoPago"
  },  {
    $match: {
      $and: [{
        contabilidad: req.params.fondo
      }, {
        username: req.params.username
      }, {
        "movimientos.tipoPago.tipoPago": req.params.tipoPago
      }]
    }
  },
  {
    $group: {
      _id: {},
      total: {
        $sum: "$movimientos.tipoPago.importe"
      }
    }
  }
  ])
    .then(documents => {

      res.status(200)
        .json({
          message: 'Posts fetched succesfully',
          recibos: documents.map(t => {
            return {
              total: t.total.toString()
            }
          }),
        });
    });
});
```

Este microservicio es el encargado de traer el total que hay por tipo de pago, fondo y username.	GET	Tipopago
		Fondo
		Username
			{
			"message": "Posts fetched succesfully",
			"recibos": [
			{
			"total": "40602"
			}
			],
			}

GetTotalByFechaFondo&Matricula

```
router.get('/total/by/fecha-matricula/:fecha/:matricula/:fondo', (req, res, next) => {
  console.log('Obteniendo total por fecha y matricula', req.params.matricula);
  Caja.aggregate([{
    $project: {
      _id: 1,
      numeroRecibo: 1,
      contabilidad: 1,
      fecha: 1,
      "movimientos.alumno.matricula": 1,
      "movimientos.tipoPago.importe": 1
    },
  },
  {
    $unwind: "$movimientos"
  },
  {
    $unwind: "$movimientos.alumno"
  },
  {
    $unwind: "$movimientos.tipoPago"
  },
  {
    $match: {
      contabilidad: req.params.fondo,
      fecha: req.params.fecha,
      "movimientos.alumno.matricula": req.params.matricula
    }
  },
  {
    $group: {
      _id: {},
      total: {
        $sum: "$movimientos.tipoPago.importe"
      }
    }
  }]
  )
    .then(
      documents => {
        res.status(200).json({
          total: documents.map(t => {
            return {
              total: t.total.toString()
            }
          }),
        });
      }
    );
});
```

Este microservicio es el encargado de regresar el total que tiene un alumno por fecha, fondo y su matrícula.	GET	Fecha
		Matricula
		Fondo
			{
			"total": [
			{
			"total": "19000"
			}
			],
			}

UpdateRecibosById

```
router.post('/actualiza/recibos/:id',
  (req, res, next) => {
    Caja.updateOne({
      _id: req.params.id
    },
      {
        $set: { estatus: "En Cancelacion" }
      })
      .then(result => {
        res.status(200)
          .json({
            message: 'Se actualizo con exito el recibo',
          })
      });
  });
```

Este microservicio es el encargado de actualizar recibos a un estatus en cancelación	POST	Id	{
			"message": "Se actualizo con exito el recibo"
			}

APÉNDICE D

REFERENCIAS PARA EL CÁLCULO DE FACTORES DE COMPLEJIDAD TÉCNICOS Y FACTORES AMBIENTALES

Referencia para el cálculo de factores de complejidad técnicos

Referencias para el cálculo de la calificación asignada al ajuste de los puntos de caso de uso, respecto a los factores técnicos (Wazlawick, 2014).

T1 – Distributed system: Is the system architecture centralized or distributed?

0. The application ignores any aspect related to distributed processing.
1. The application generates data that will be processed by other computers with human intervention (for example, spreadsheets or preformatted files sent by media or email).
2. Application data are prepared and transferred automatically for processing in other computers.
3. Application processing is distributed, and data are transferred in just one direction.
4. Application processing is distributed and data are transferred in both directions.
5. Application processes must be executed in the most appropriate processing core or computer, which is dynamically determined.

T2 – Response time/performance objectives: What is the importance of the application response time to its users?

0. No special performance requirement was defined by the client.
1. Performance requirements were established and revised, but no special action must be taken.
2. Response time and transfer rates are critical during peak hours. No special design for processor core use is necessary. The deadline for most processes is the next day.
3. Response time and transfer rates are critical during commercial hours. No special design for processor core use is necessary. Requirements regarding deadlines for communication with interfaced systems are restrictive.
4. In addition to 3, performance requirements are sufficiently restrictive for requiring performance analysis tasks during design.
5. In addition to 4, performance analysis tools must be used during design, development, and/or implementation, in order to meet the client's performance requirements.

T3 – End-user efficiency: Is the application designed to allow final users just to do their job or is it designed to improve their efficiency?

0. The application does not need any of the items below.
1. The application needs one to three of the items below.
2. The application needs four to five of the items below.
3. The application needs six or more of the items below, but there is no requirement related to user efficiency.
4. The application needs six or more of the items below, and the user efficiency requirements are so strong that the design must include features to minimize typing, maximize defaults, use templates, etc.
5. The application needs six or more of the items below, and the user efficiency requirements are so strong that the design activities must include tools and special processes to demonstrate that the performance goals are obtained.

The following items must be considered for the assessment of the end-user efficiency item:[9]

- Navigational help (for example, dynamically generated menus, adaptive hypermedia, etc.).
- Online help and documentation.
- Automated cursor movement.
- Predefined function keys.
- Batch tasks submitted from online transactions.
- High use of colors and visual highlights in screens.
- Minimizing the number of screens to reach the business goals.
- Bilingual support (counts as four items).
- Multilingual support (counts as six items).

T4 – Internal processing complexity: Does the application need complex algorithms?

0. None of the options below.
1. One of the options below.
2. Two of the options below.
3. Three of the options below.
4. Four of the options below.
5. All five options below.

The following options need to be considered for assessing internal processing complexity:

- Careful control (for example, special audit processing) and/or secure processing specific to the application.
- Extensive logical processing.
- Extensive mathematical processing.
- Lots of exception processing resulting from incomplete transactions that needs to be reprocessed, such as incomplete automated teller machine transactions caused by interruption of communication, missing data values, or failed data change.
- Complex processing to manage multiple inputs and output possibilities, such as multimedia or device independency.

T5 – Design aiming for code reusability: Is the application designed so that its code and artifacts will be highly reusable?

0. There is no concern about producing reusable code.
1. Reusable code is generated for use inside the same project.
2. Less than 10% of the application must consider more than the user needs.
3. 10% or more of the application must consider more than the user needs.
4. The application must be specifically packaged and/or documented for facilitating reuse, and the application must be customizable by the user at the level of source code.
5. The application must be specifically packaged and/or documented for facilitating reuse, and the application must be customizable by the user with the use of parameters.

T6 – Easy to install: Will the application be designed so that its installation is automatic (for example, in the case of users with low or unknown technical capacity), or is there no special concern about it?

0. The client established no special consideration, and no special setup is necessary for installation.
1. The client established no special consideration, but a special setup is required for installation.
2. The client established requirements for data conversion and installation, and conversion and installation guides must be provided and tested. The impact of conversion in the project is not considered important.
3. The client established requirements for data conversion and installation, and conversion and installation guides must be provided and tested. The impact of conversion on the project is considerable.
4. In addition to 2, tools for automatic conversion and installation must be provided and tested.
5. In addition to 3, tools for automatic conversion and installation must be provided and tested.

T7 – Easy to operate:[10] Are there special requirements regarding the operation of the system?

0. No special considerations about the operation of the system besides normal backup procedures were established by the user.
1. One of the items below applies to the system.
2. Two of the items below apply to the system.
3. Three of the items below apply to the system.
4. Four of the items below apply to the system.
5. The application is designed to operate in nonsupervised manner. "Nonsupervised" means that human intervention is not necessary for keeping the system operational, even if crashes occur, except maybe for the first startup and final turn off. One of the application features is automatic error recovery.

For the evaluation of the easy to operate factor, the following items must be considered:

- Effective processes for initialization, backup, and recovery must be provided, but operator intervention is still necessary.
- Effective processes for initialization, backup, and recovery must be provided, and no operator intervention is necessary (counts as two items).
- The application must minimize the need for data store in offline media (for example, tapes).
- The application must minimize the need for dealing with paper.

T8 – Portability: Is the application or parts of it designed to work on more than one platform?

0. There is no user requirement to consider the need for installing the application on more than one platform.
1. The design must consider the need for the system to operate in different platforms, but the application must be designed to operate only in *identical* hardware and software environments.
2. The design must consider the need for the system to operate in different platforms, but the application must be designed to operate only in *similar* hardware and software environments.
3. The design must consider the need for the system to operate in different platforms, but the application must be designed to operate in *heterogeneous* hardware and software environments.
4. In addition to 1 or 2, a documentation and maintenance plan must be elaborated and tested to support operation in multiple platforms.
5. In addition to 3, a documentation and maintenance plan must be elaborated and tested to support operation in multiple platforms.

T9 – Design aiming for easy maintenance: Does the client require that the application must be easy to change in future?

0. None of the items below.
1. One of the items below.
2. Two of the items below.
3. Three of the items below.
4. Four of the items below.
5. Five or more of the items below.

To evaluate this factor, the following items are considered:

- A flexible report structure must be provided to deal with simple queries such as logical binary operators applied to just one logical archive (count as one item).
- A flexible report structure must be provided to deal with medium complexity queries such as logical binary operators applied to more than one logical archive (count as two items).
- A flexible report structure must be provided to deal with high complexity queries such as combinations of logical binary operators applied to one or more logical archives (count as three items).
- Business control data are kept in tables managed by the user with interactive online access, but changes must only be effective on the next day (count as one item).
- Business control data are kept in tables managed by the user with interactive online access, and changes are effective immediately (count as two items).

T10 – Concurrent/parallel processing: Must the application be designed in order to deal with problems related to concurrency such as, for example, data and resource sharing?

0. No concurrent access to data is expected.
1. Concurrent access to data is expected sometimes.
2. Concurrent access to data is expected frequently.
3. Concurrent access to data is expected all the time.
4. In addition to 3, the user indicates that a lot of multiple accesses are going to happen, forcing performance analysis tasks and deadlock resolution during design.
5. In addition to 4, the design requires the use of special tools to control access.

T11 – Security: Are the security needs just nominal or are a special design and additional specifications required?

0. There are no special requirements regarding security.
1. The need for security must be taken into account in design.
2. In addition to 1, the application must be designed so that it can be accessed only by authorized users.
3. In addition to 2, access to the system will be controlled and audited.
4. In addition to 3, a security plan must be elaborated and tested to support access control to the application.
5. In addition to 4, a security plan must be elaborated and tested to support auditory.

T12 – Access for/to third-party code: Is the application going to use code already developed, such as commercial off-the-shelf (COTS) components, frameworks or libraries? High reuse of good quality software reduces the value of this item as it implies less development effort.

0. Highly reliable preexistent code will be used extensively for developing the application.
1. Highly reliable preexistent code will be used in small parts of the application.
2. Preexistent code that eventually needs to be adjusted will be used extensively for developing the application.
3. Preexistent code that eventually needs to be adjusted will be used in small parts of the application.
4. Preexistent code that needs to be fixed or is hard to understand will be used in the application.

5. No preexistent code will be used in the application or questionable quality code will be used in the application.

T13 – User training needs: Will the application be easy to use, or must extensive training be given to future users?

0. There are no specific requirements for user training.
1. Specific user training requirements have been mentioned.
2. There are formal specific user training requirements, and the application must be designed to facilitate training.
3. There are formal specific user training requirements, and the application must be designed to support users with different levels of training.
4. A detailed training plan must be elaborated for the transition phase and executed.
5. In addition to 4, the users are geographically distributed.

Referencia para el cálculo de factores ambientales

Referencias para el cálculo de la calificación asignada al ajuste de los puntos de caso de uso, respecto a los factores ambientales (Wazlawick, 2014).

E1 – Familiarity with the development process: This factor assesses the team's experience with the development process they are using. Originally the factor mentioned the Unified Process, and some authors even mention UML here. But this has been adjusted to reflect the fact that other processes and modeling languages could be used instead.

0. The team does not have experience with the development process or does not use any process.
1. The team has theoretical knowledge about the development process, but no experience.
2. A few members of the team have already used the process in one project.
3. A few members of the team have used the process in more than one project.
4. Up to half the team has used the process in many projects.
5. More than half the team has used the process in many projects.

E2 – Experience in the application: This factor assesses the familiarity of the team with the application area or domain. For example, if the application is about e-commerce, have the team already worked with systems in the same area?

0. No team member has any experience in projects in the same area.
1. Some team members have 6 to 12 months of experience in projects in the same area.
2. Some team members have 12 to 18 months of experience in projects in the same area.
3. Most team members have 18 to 24 months of experience in projects in the same area.
4. Most team members have greater than two years of experience in the same area.
5. All team members have greater than two years of experience in the same area.

E3 – Object-oriented experience: This factor must not be confused with either *familiarity with the development process* (E1) or *experience in the application* (E2): this is about the experience of the team in doing object-oriented analysis, modeling, design, and programming, independently of the application area and the development process used.

0. The team has no experience in object-oriented techniques.
1. All team members have some experience in object-oriented techniques (up to one year).
2. Most team members have 12 to 18 months of experience in object-oriented techniques.
3. All team members have 12 to 18 months of experience, or most team members have 18 to 24 months of experience in object-oriented techniques.
4. Most team members have more than two years of experience in object-oriented techniques.
5. All team members have more than two years of experience in object-oriented techniques.

E4 – Lead analyst experience: This factor measures the experience of the lead analyst with requirement analysis and object-oriented modeling.

0. The lead analyst has no experience.
1. The lead analyst has previous experience in a single similar project.
2. The lead analyst has about one year of experience in more than one similar project.
3. The lead analyst has about two years of experience in similar projects.
4. The lead analyst has more than two years of experience in similar projects.
5. The lead analyst has more than three years of experience in a variety of projects.

E5 – Motivation: This factor describes the team's motivation.[11]

0. The team has no motivation at all. Without constant supervision the team becomes unproductive. The team only does what is strictly asked.
1. The team has very little motivation. Constant supervision is necessary to keep productivity at acceptable levels.
2. The team has little motivation. Management interventions are necessary from time to time to maintain productivity.

3. The team has some motivation. Usually the team has initiative, but management intervention is still necessary sporadically to keep productivity.
4. The team is well motivated. The team is self-managed usually, but the existence of supervision is still necessary because productivity can be lost without it.
5. The team is highly motivated. Even without supervision everyone knows what has to be done, and pace is maintained indefinitely.

E6 – Stable requirements: This factor evaluates if the team has been able to keep requirements stable in past projects, minimizing their change during the project. This factor may vary from project to project, because there are domains where requirements change often independently of the capacity of the analysts. This factor must assess only software changes that were caused by the team's inability to discover the right requirements.

0. There is no historic data about requirement stability or, in the past, poor analysis caused big changes in requirements after the project started.
1. Requirements were predominantly unstable in the past. Clients asked for many changes caused mainly by incomplete or incorrect requirements.
2. Requirements were unstable in the past. Clients asked for some changes caused by incomplete or incorrect requirements.
3. Requirements were relatively stable in the past. Clients asked for changes in secondary functionalities with some regularity. Changes to main functionalities were asked for seldom.
4. Requirements were mostly stable in the past. Users asked for little changes, especially cosmetic ones. Changes in main or secondary functionalities were unusual.
5. Requirements were totally stable in the past. Little changes, if any, had no impact on projects.

E7 – Part-time workers: This is a negative factor, that is, contrary to the first six environmental factors, higher values here represent more development time, not less. A team with many members involved in other projects or activities is less productive than a dedicated team.

0. No team member is part time on the project.
1. Up to 10% of the team is part time.
2. Up to 20% of the team is part time.
3. Up to 40% of the team is part time.
4. Up to 60% of the team is part time.
5. More than 60% of the team is part time.

E8 – Difficulty with programming language: This is another negative factor, meaning that high values here are bad for the development time.

0. All team members are very experienced programmers.
1. Most team members have at least two years of experience in programming.
2. All team members have at least 18 months of experience in programming.
3. Most team members have at least 18 months of experience in programming.
4. A few team members have some experience in programming (not more than one year).
5. All team members are inexperienced programmers.

REFERENCIAS

Aerts , N., Cailliau, E., de Groote, L. y Noterman, L. (2016). *A comparative study on containers and related technologies*. Recuperado de https://www.researchgate.net/publication/320961475_A_comparative_study_on_containers_and_related_technologies

Alpers, S., Becker, C., Oberweis, A. y Schuster, T. (2015, septiembre). *Microservice based tool support for business process modelling*. Conferencia presentada en 19th International Enterprise Distributed Object Computing Workshop, Adelaide, Australia. doi:10.1109/EDOCW.2015.32

Amaral, M., Polo, J., Carrera, D., Mohomed, I., Unuvar, M. y Steinder, M. (2015, septiembre). *Performance evaluation of microservices architectures using containers*. Conferencia presentada en 14th International Symposium on Network Computing and Applications, Cambridge, USA. doi:10.1109/NCA.2015.49

Avidan, Z. y Otharsson, H. (2018). *Accelerating the digital journey from legacy systems to modern microservices.* Reston, VA: OpenLegacy.

Balalaie, A., Heydarnoori, A. y Jamshidi, P. (2016). Microservices architecture enables DevOps: Migration to a cloud-native architecture. *IEEE Software, 33*(3), 2-12. doi:10.1109/MS.2016.64

Bass, L., Clements, P. y Kazman, R. (2013). Software architecture in practice. Boston, MA: Addison-Wesley

D'Agostino, D., Roverelli, L., Zereik, G., De Luca, A., Salvaterra, R., Belfiore, A., . . . Tiengo, A. (2016, junio). *A Microservice-based portal for X-ray transient and variable sources*. Conferencia presentada en 8th International Workshop on Science Gateways, Pavia, Italy. doi:10.7287/peerj. preprints.2519v2

Daya, S., Van Duy, N., Eati, K., Ferreira, C. M., Glozic, D., Gucer, V., . . . Vennam, R. (2015). *Microservices from theory to practice: Creating applications in IBM Bluemix using the microservices approach.* International Business Machines Corporation. Recuperado de https://www.redbooks.ibm.com/redbooks/pdfs/sg248275.pdf

De la Torre, C., Wagner, B. y Rousos, M. (2019). *.NET microservices: Architecture for containerized .NET applications.* Redmond, WA: Microsoft Corporation.

Del Esposte, A. M., Kon, F., Costa, F. M. y Lago, N. (2017, abril). *InterSCity: A scalable microservice-based open source platform for smart cities*. Conferencia presentada en 6th International Conference on Smart Cities and Green ICT Systems, Porto, Portugal. doi:10.5220/ 0006306200350046

Dennis, A., Wixom, B. H. y Tegarden, D. (2015). *Systems analysis & design: An object-oriented approach with UML.* Danvers, MA: Wiley.

Dragoni, N., Lanese, I., Larsen, S., Mazzara, M., Mustafin, R. y Safina, L. (2017). *Microservices: How to make your application scale*. doi:10.1007/978-3-319-74313-4_8

Gorton, I. (2011). *Essential software architecture.* London: Springer.

Hasselbring, W. y Steinacker, G. (2017, abril). *Microservice architectures for scalability, agility and reliability in E-Commerce*. Conferencia presentada en International Conference on Software Architecture Workshops, Gothenburg, Suecia. doi:10. 1109/ICSAW. 2017.11

Heinrich, R., Van Hoorn, A., Knoche, H., Li, F., Lwakatare, L. E., Pahl, C., . . . Wettinger, J. (2017, abril). *Performance Engineering for Microservices: Research Challenges and Directions.* Conferencia presentada en 8th ACM/SPEC International Conference on Performance, L'Aquila, Italia. doi:10.1145/3053600. 3053653

Khazaei, H., Ravichandiran, R., Park, B., Bannazadeh, H., Tizghadam, A. y Leon-Garcia, A. (2017). E*lascale: Autoscaling and monitoring as a service*. Recuperado de https://www.researchgate.net/publication/320975600_Elascale_Autoscaling_and_Monitoring_as_a_Service

Lacic, E., Traub, M., Kowald, D. y Lex, E. (2015, septiembre). *Scar: Towards a real-time recommender framework following the microservices architecture.* Conferencia presentada en 9th ACM Conference on Recommender Systems, Viena, Australia. Recuperado de https://www.researchgate .net/publition/281931941 _ScaR_Towards_a_Real-Time_Recommender_Framework_Following_the_ Microservices_Architecture

López Hinojosa, J. D. (2017). *Arquitectura de software basada en microservicios para el desarrollo de aplicaciones web de la asamblea nacional* (Tesis de maestria). Universidad Técnica del Norte, Ibarra, Ecuador.

Lv, H. y Wang, S. (2016, agosto). *Design and application of IoT microservices based on Seneca*. Conferencia presentada en 3rd International Conference on Information and Communication Technology for Education, Toronto, Canada. doi:10.12783/dtcse/ icte2016/4814

Mazzara, M., Dragoni, N., Bucchiarone, A., Giaretta, A., Larsen, S. T. y Dustdar, S. (2018). Microservices: Migration of a mission critical system. *IEEE Transactions on Services Computing, 1*(1), 1-14. doi:10.1109/TSC.2018.2889087

Mens, T., Magee, J. y Rumpe, B. (2010). Evolving software architecture descriptions of critical systems. *Computer, 43*(5), 42 - 48. doi:10.1109/MC.2010.136

MuleSoft, I. (2019). *Best practices for microservices*. Recuperado de https://www.mulesoft.com/lp/whitepaper/api/microservices-best-practices

Newman, S. (2015). *Building microservices.* Sebastopol, CA: O'Reilly Media.

Nielsen, C. D. (2015). *Investigate availability and maintainability within a microservice architecture* (Tesis de maestría). Aarhus University, Aarhus, Dinamarca.

Oberhauser, R. (2016, enero). *Microflows: Lightweight automated planning and enactment of workflows comprising semantically-annotated microservices.* Conferencia presentada en Sixth International Symposium on Business Modeling and Software Design, Aalen, Alemania. doi:10.1007/978-3-319-57222-2_9

Rahman, M. y Gao, J. (2015, marzo). *A reusable automated acceptance testing architecture for microservices in behavior-driven development*. Conferencia presentada en The First International Workshop on Mobile Cloud TaaS in conjunction with IEEE SOSE2015, San Francisco, USA. doi:10.1109/ SOSE.2015.55

Richards, M. (2015). *Software architecture patterns.* Sebastopol, CA: O'Reilly Media.

Sangwan, R. S. (2015). *Software and systems architecture in action.* Boca Raton, FL: CRC Press.

Schermann, G., Cito, J. y Leitner, P. (2015). *All the services large and micro: Revisiting industrial practice in services computing*. *1*, 1-13. doi:10.7287/PEERJ.PREPRINTS.1291

Stephens, R. (2015). *Beginning software engineering*. Indianapolis, IN: John Wiley & Sons.

Strimbei, C., Dospinescu, O., Strainu, R. M. y Nistor, A. (2015). Software architectures – present and visions. *Informatica Economică, 19*(4), 13-27. doi:10.12948/issn14531305/19.4.2015.02

Vigneshwaran Sudalaikkan, L. (2016). *Preservation of low latency service request processing in dockerized microservice architectures* (Tesis de maestría). Colorado State University, Fort Collins, USA.

Visti, H., Kiss, T., Terstyanszky, G., Gesmier, G. y Winter, S. (2016). MiCADO – *Towards a microservice-based cloud application - level dynamic orchestrator*. DOI: 10.7287/PEERJ.PREPRINTS.2536

Wazlawick, R. S. (2014). *Object-oriented analysis and design for information systems: Modeling with UML, OCL, and IFML.* Waltham, MA: Elsevier.