Desarrollo de software backend para el aplicativo Rinn Customer App de la empresa VICA Technology

Informe Final
Modalidad: Práctica Profesional

Carlos Albeiro Rendon Garcia
104612020523

Asesor de la empresa: Ing. Fabián Yesid Vidal
Directora: PhD. Sandra Milena Roa Martínez

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Programa Ingeniería de Sistemas
Popayán, septiembre de 2022

TABLA DE CONTENIDO

Tabla de Figuras

Listado de Tablas

1 INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

El servicio de *delivery* gestionado desde aplicaciones móviles, por las ventajas que ofrece en materia generación de nuevas oportunidades de ventas incrementales, retroalimentación en tiempo real y reducción de costos, se presenta como uno de los favoritos tanto por las empresas de todos los sectores, como para los clientes, por la comodidad que ofrece. De ahí, que la arquitectura de software y aplicaciones móviles para procesos de *delivery* se convierte en un nicho de mercado muy interesante para el campo disciplinar de los ingenieros de sistemas y para las empresas especializadas en soportes de tecnología TI.

Rinn Customer App, es una aplicación móvil del servicio *delivery* que incluye la gestión de pedidos y pagos de restaurantes, licores y otros bienes. Vale destacar que, esta aplicación fue seleccionada por iNNpulsa como parte de una de las 51 empresas con mayor potencial en Colombia, dentro del componente de economía naranja, reconocimiento que le permitirá obtener fuentes de financiación e inversión para acelerar su crecimiento [1].

Adicionalmente, para aprovechar el potencial del aplicativo, es decir, para cumplir con las expectativas de escalabilidad y, contribuir al crecimiento de la empresa VICA Technology, es necesario brindar una experiencia de compra rápida y segura; para lo cual se requiere que la plataforma sea robusta, de tal manera que la experiencia del cliente se garantice con el aumento del volumen de visitas y compras, lo que será posible mediante el empleo de una tecnología flexible y escalable[1].

Ahora bien, dentro de los aspectos priorizados por VICA Technology que deben ser atendidos, se encuentran el aumento de funcionalidades, como la construcción de una *Application Programming Interface* (API) para el soporte de la nueva pasarela de pagos Wompi, así como el diseño e implementación de una API para proveer el soporte a la nueva billetera de Rinn App.

En lo que se refiere a la necesidad del API para el soporte de la nueva pasarela de pagos Wompi, es un aspecto de urgente atención por parte de la empresa, pues el aplicativo Rinn App actualmente cuenta con una disponibilidad reducida de medios de pagos a saber: billetera Rinn App, tarjetas de crédito y débito con Valor de Verificación de Tarjeta (CVV), situación que no resulta acorde para poder satisfacer las expectativas que tiene la empresa VICA Technology en lo que se refiere al crecimiento del número de clientes.

---

[1] Plataformas robustas, el ‘must have’ del e-commerce - Consumotic https://www.consumotic.mx/ecommerce/plataformas-robustas-el-must-have-del-e-commerce/ (accessed Nov. 08, 2021).

En esa dirección, la pasarela de pagos Wompi, de Bancolombia, resulta atractiva para la empresa, principalmente porque: presenta un funcionamiento considerablemente simple, con una interfaz muy amigable que permite a cualquier persona sin experiencia poder generar enlaces de pago con diferentes valores y compartirlos directamente en redes sociales, WhatsApp o su sitio web; generación de códigos QR con mucha facilidad desde su plataforma y principalmente porque amplía las opciones de pago para el aplicativo, incluyéndose todas las tarjetas por medio de Pagos Seguros en Línea (PSE), Nequi, tarjetas débito y crédito con CVV y botón Bancolombia[2]. Así, la integración de ésta a Rinn APP mejora la funcionalidad del aplicativo en lo que corresponde a los métodos de pago, ofreciéndose mayores opciones al cliente al momento de pagar logrando contribuir al buen funcionamiento y crecimiento de servicio de pagos en la aplicación, de tal manera que se pueda responder en debida forma ante el aumento de la demanda por el servicio.

Asociado a la integración de la nueva billetera a Rinn App, la organización manifiesta que su intención es poner a disposición del cliente una manera sencilla, segura y rápida de realizar los pagos, por medio de la recarga de la billetera ya sea, vinculando una cuenta bancaria y/o tarjeta de crédito, usando Nequi, PSE o Botón Bancolombia facilitando realizar pagos de forma simplificada, en especial de servicios públicos y SOAT. En ese orden, para la empresa el diseño e implementación de una API para proveer el soporte a la nueva billetera Rinn App por medio de la cual se habilitan pagos de SOAT y determinados servicios públicos se torna prioritaria para ser desarrollada en el contexto de esta práctica profesional.

Siendo un objetivo del desarrollo de la ingeniería de software “producir un sistema, aplicación o producto de alta calidad, para lograrlo, los ingenieros de software deben emplear métodos efectivos junto con herramientas modernas dentro del contexto de un proceso maduro de desarrollo del software” [2]. Lo cual, sin lugar a duda, incluye la realización de pruebas del sistema.

En este contexto y con relación a la práctica profesional que pretende desarrollarse, cabe señalar que actualmente la empresa VICA Technology para el aplicativo que administra, Rinn App, no cuenta con un ambiente de pruebas que le permita evaluar la calidad de dicho aplicativo, teniendo como posibilidad latente que se puedan presentar fallos, bien sea de diseño o de funcionalidad, los cuales le pueden generar dificultades en la calidad del producto y afectar la credibilidad de la empresa.

Por lo anterior, la empresa VICA Technology requiere incorporar a su proceso actual de negocio, específicamente, en lo que se refiere al ciclo de vida del aplicativo Rinn App, una herramienta o sistema de pruebas, que le permita realizar evaluaciones de la calidad del software a partir de la constatación de métricas

[2] “Tecnología, la verdadera ganadora de la era Post-Covid - Softtek.” https://softtek.eu/tech-magazine/digital-transformation/tecnologia-la-verdadera-ganadora-de-la-era-post-covid/ (accessed Nov. 08, 2021)

tales como: corrección, facilidad de mantenimiento, integridad y facilidad de uso [2].

Para tal efecto, se observa que se ajusta a las necesidades que tiene la empresa VICA Technology realizar la implementación de un ambiente o entorno de pruebas para el aplicativo Rinn App, en el cual se despliegue el aplicativo, se ejecuten y se realicen las pruebas de las funcionalidades, incluyéndose test de integración continua, depuración de errores, etc.

Es así como, la realización de la práctica profesional en la empresa, VICA Technology, pretende contribuir en la robustez y confiablidad de la aplicación Rinn Customer APP, lo cual comprende, entre otros aspectos, al buen funcionamiento y crecimiento de servicio de pagos en la aplicación de tal manera que pueda responder en debida forma ante el aumento de la demanda de servicios, diseño de una API para dar soporte de billetera virtual de la API y la construcción de un ambiente de pruebas para el aplicativo Rinn Customer App.

Finalmente, la realización de esta práctica profesional como modalidad de trabajo de grado permitirá al estudiante del programa de ingeniería de sistemas de la Universidad del Cauca desarrollar y fortalecer habilidades y conocimientos relacionados a un sistema de alta demanda, a un modelo de negocio muy empleado, el delivery y dar solución a problemas reales mediante el apoyo y desarrollo de funcionalidades y actividades específicas para el software backend en el aplicativo Rinn Customer APP para la empresa VICA Technology.

1.1 OBJETIVOS

1.1.1 Objetivo General

Apoyar el desarrollo de software backend de la empresa VICA Technology en la ejecución de actividades y construcción de funcionalidades para el aplicativo Rinn App.

1.1.2 Objetivos Específicos

- Implementar un ambiente de pruebas para el despliegue de nuevas funcionalidades del aplicativo Rinn App.
- Construir una API para soportar la funcionalidad de la nueva pasarela de pagos Wompi.
- Diseñar e implementar una API para proveer el soporte de la nueva billetera virtual de Rinn App que habilita pagos de SOAT y Servicios públicos.

1.2 METODOLOGÍA

El desarrollo de la práctica profesional adopta una metodología que es brindada por la empresa en donde se promueve aplicar prácticas basadas en su

experiencia y trabajos anteriores, las cuales han permitido alcanzar y dar cumplimiento de los objetivos que se ha propuesto. Esta metodología de trabajo se divide en tres fases que permiten desarrollar el trabajo y cumplir con los objetivos:

1. Adaptación y reconocimiento del sistema actual
2. Desarrollo de software
3. Documentación

- FASE 1. Reconocimiento del sistema actual

Rinn App es un aplicativo que emplea varias tecnologías que se integran para su funcionamiento, lo cual hace necesario y pertinente iniciar conociendo el funcionamiento básico y fundamental de su sistema, luego comprender la relación entre sus componentes y partir de esto, determinar de forma específica las áreas en que se involucran materialmente los objetivos de la práctica. A través del acompañamiento de la empresa, el practicante podrá ir explorando, reconociendo y familiarizarse con el ambiente en que se desempeñará.

  - Principales actividades:
    - Conocer el funcionamiento del sistema
    - Comprender la relación entre sus componentes
    - Identificar las áreas específicas de trabajo
    - Caracterizar el código legado

- FASE 2. Construcción de ambiente de pruebas y desarrollo de software

Para implementar el ambiente de pruebas se requiere conocer los componentes que integran el sistema para lograr replicar su configuración. Por ende, se logrará tener una experiencia plena de lo que significa exponer como servicio una aplicación para los usuarios. Luego se diseñará el API de pagos que utilizará el aplicativo y en el que se integrará la nueva pasarela, siendo este un aspecto fundamental para definir aspectos de arquitectura y a partir de ahí se pasar a construirla. Finalmente se diseñará e implementará un API para habilitar los pagos de SOAT y servicios públicos que proveerán un acercamiento al practicante de los servicios que se están utilizando por los usuarios para no realizar los pagos presencialmente.

  - Principales actividades:
    - Implementar un ambiente de pruebas para desplegar en este, las nuevas funcionalidades que se desarrollarán.
    - Diseñar y construir el API de pagos que utilizara la nueva pasarela de pagos
    - Diseñar una API para habilitar pagos de SOAT y servicios públicos
    - Implementar la API mencionada en el punto anterior

- Documentación:

Durante el periodo de práctica se llevarán a cabo reuniones programadas y periódicas con el tutor de la universidad y con el asesor de la empresa para mantener un acompañamiento, asesoría y además hacer seguimiento para el cumplimiento de los objetivos. También de forma mensual se entregarán informes a la universidad que permitirán evidenciar las actividades y el desarrollo de las metas durante la práctica. Finalmente, se elaborará un documento final tipo monografía de la práctica profesional desarrollada en donde se detallará todo el proceso.

- Principales actividades:
  - Efectuar reuniones periódicas con el asesor y tutor
  - Elaboración de informes mensuales
  - Elaboración de la monografía de la práctica profesional.

1.3 APORTES

El desarrollo de la práctica profesional en la empresa VICA Technology generó aportes significativos desde los puntos de vistas social, económico, empresarial, personal, institucional y académico. Con relación al aporte social, este trabajo contribuyó al fortalecimiento del servicio *delivery*, el cual permitió responder a las necesidades de aprovisionamiento de víveres y bienes básicos en la crisis generada en el marco de la emergencia sanitaria provocada por el COVID 19, por las restricciones que el Gobierno Nacional impuso para controlar la pandemia. Concretamente, el soporte *backend* para el aplicativo Rinn Customer App permitió que la empresa pudiese soportar el aumento de la demanda generada por el cambio en las condiciones sociales.

Desde el punto de vista económico, igualmente el trabajo realizado en materia de soporte *backend* para el aplicativo Rinn Customer App y el fortalecimiento del sistema de pagos en la aplicación, significa un aporte a la economía digital, pues se contribuye al desarrollo de nuevos sectores en Tecnologías de la información y la comunicación (TIC), consistente en el empleo de entornos móviles digitales para comercializar bienes y servicios, lo cual responde a las tendencias de evolución del comercio electrónico en Colombia y en el Cauca.

En el ámbito empresarial, las actividades desarrolladas en el marco de la práctica profesional contribuyen al fortalecimiento del sector empresarial con mayor potencial en Colombia, esto es, el comercio electrónico de bienes y servicios básicos en un entorno móvil. Concretamente de la aplicación Rinn Customer App, aplicación seleccionada por iNNpulsa como parte de una de las 51 empresas con mayor potencial en Colombia[3]. Vica Technology se propone a mejorar la aplicación implementado un ambiente de pruebas para desplegar el software e identificar errores, así mismo la inclusión de arquitectura de desarrollo basada en

[3] Infobae.” https://www.infobae.com/america/colombia/2021/05/14/conozca-los-emprendimientos-con-mayor-potencial-en-colombia/ (Disponible Nov. 08, 2021)

microservicios y por capas en la construcción de una API para soportar la funcionalidad de la nueva pasarela de pagos Wompi, ello para minimizar el impacto de los errores, facilitar los procesos de actualización de la aplicación y aumentar la escalabilidad para responder al aumento del tráfico en la aplicación. En otras palabras, el trabajo propende por dar solución a problemáticas registradas en la empresa a intervenir, las cuales se identificaron a partir del análisis de la situación actual y las necesidades de la organización.

En el terreno personal, el trabajo realizado fortalece las capacidades laborales del estudiante, toda vez que permitió aplicar los conocimientos adquiridos en la Universidad del Cauca, así mismo, trabajar en temas no abordados en la etapa de formación. Aspectos, que, sin lugar a duda, enriquecen mi perfil profesional y me permitirá desenvolverme correctamente en el ámbito laboral.

Desde lo académico, el desarrollo de la practica enriquece el campo de conocimiento de la ingeniería de sistemas, consistente en la aplicación de arquitecturas de vanguardia los microservicios en el ámbito de la economía digital, experiencia que podrá dar cuenta de la efectividad de tales modelos en las tecnologías que se centran en el comercio electrónico.

Por último, la práctica implica una contribución desde la perspectiva institucional, dado que se presenta como una expresión de la proyección social de la universidad en la comunidad empresarial, poniendo a disposición de esta comunidad, el conocimiento adquirido para la solución de problemáticas relevantes.

1.4 ORGANIZACIÓN DEL DOCUMENTO

A continuación, se describe de manera general el contenido y organización del presente informe final:

CAPITULO 1: INTRODUCCIÓN: Hace referencia al presente capitulo donde se describe el planteamiento del problema, los aportes generados y los objetivos a cumplir.

CAPITULO 2: MARCO TEÓRICO: En este capítulo se describen los conceptos teóricos más relevantes que se emplearon para la realización del trabajo de grado.

CAPITULO 3: CARACTERIZACIÓN DEL SISTEMA Y EL API: En este capítulo se realiza una caracterización del sistema identificando las arquitecturas aplicadas.

CAPITULO 4: REQUERIMIENTOS E ITERACIONES: En este capítulo los requerimientos solicitados por la empresa y el desarrollo de las iteraciones junto con sus actividades y el resultado en cuanto a tiempo de su elaboración.

CAPITULO 5: IMPLEMENTACIÓN DEL AMBIENTE DE PRUEBAS Y CONSTRUCCIÓN DE SOFTWARE: En este capítulo se describen el proceso para llevar a cabo los objetivos planteados y las funcionalidades creadas.

CAPITULO 6: LECCIONES APRENDIDAS: En este capítulo se describen las lecciones aprendidas generadas a partir del desarrollo de la práctica profesional.

CAPITULO 7: CONCLUSIONES: En este capítulo se describen las conclusiones generadas a partir del desarrollo de práctica profesional.

CAPITULO 8: BIBLIOGRAFÍA: En el último capítulo del presente documento se listan las referencias bibliográficas de los artículos, libros y demás recursos utilizados para el desarrollo del trabajo de grado.

2 MARCO TEÓRICO

2.1 Ambiente de Pruebas

El desarrollo de software, al ser una obra humana, puede involucrar defectos, errores y fallas. Aunque se han construido diferentes protocolos para disminuir la probabilidad que se presenten estos problemas en el software no se ha llegado a eliminar la posibilidad de un mal funcionamiento en su ejecución; es por esta razón que las pruebas de software son una parte fundamental y siempre harán parte de su desarrollo. Las pruebas en la industria de software se definen como el proceso que consiste en que todas las actividades de ciclo de vida, tanto estáticas como dinámicas relacionadas con la planeación, preparación y evaluación de productos de software y productos relacionados con el trabajo para determinar que cumplen con los requisitos especificados, para detectar defectos y para demostrar que son actos alineados a los objetivos [3].

En este sentido, las pruebas de software están encaminadas a la detención de errores y fallos para eliminarlos. Ahora bien, la preponderancia de dichas pruebas en la actualidad está dada por el hecho de que la supervivencia de muchas empresas está ligada al buen funcionamiento del software. En ese orden, el principal objetivo de las pruebas de software es “aportar calidad al producto que se está desarrollando”[4], lo cual involucra examinar características como: adecuación funcional, eficiencia de desempeño, compatibilidad, usabilidad, fiabilidad, seguridad, mantenibilidad y portabilidad.

Respecto a la importancia de un ambiente de pruebas para desplegar las aplicaciones se mencionan las siguientes razones [5]:

- Encuentra y remedia errores que se desencadenan en la operación de las aplicaciones, los cuales s resultan de difícil identificación.
- Posibilita la identificación de errores en un lapso pertinente, suministrando un tiempo suficiente para subsanarlos ante de la puesta en funcionamiento del producto y su puesta disposición del cliente, con lo cual además se mejora la experiencia de este.
- Permiten que el programa sea puesto en funcionamiento con los niveles más altos de calidad posible.
- Disminuye la probabilidad de que sucedan fallos que puedan comprometer la operación de la plataforma, lo cual repercute en una mejor experiencia de usuario.

Por estas razones, las pruebas de software son parte importante dentro del proceso de desarrollo y puesta en funcionamiento de las aplicaciones, siendo un aspecto relevante para las empresas que pertenecen al sector.

Entonces es importante incluir las pruebas en el proceso de desarrollo, para lo cual es necesario contar con un ambiente o entorno en el cual se pueda desplegar el software. Desde dicha perspectiva, resulta pertinente definir el alcance de tal concepto. El cual puede entenderse como cualquier espacio en el que el software

se somete a diferentes usos experimentales. Se puede decir, entonces, que es un espacio en el cual se despliega una copia del sistema que se encuentra en producción, es decir, un clon, con el propósito de identificar y corregir errores y fallos, empleando máquinas virtuales. Dicho procedimiento, hace referencia a un ambiente de pruebas [6].

Con relación a la importancia del ambiente de pruebas cabe decir que es significativa, tanto para el desarrollador como para la organización que se beneficia del software. Desde la perspectiva del desarrollo, disponer de un ambiente de pruebas es trascendente porque se constituye en un entorno seguro en el cual se pueden realizar cambios, implementar otras funcionalidades, verificar funcionamiento y realizar otras pruebas sin que los errores o fallos que se presenten puedan tener una incidencia negativa en el software en producción [7].

De otro lado, desde la perspectiva de la organización los ambientes de prueba pueden reducir significativamente los costos anuales de control de calidad y evitar problemas relacionados a aspectos legales por incumplimientos, multas o sanciones, a más gastos por necesitar recursos adicionales al extender el tiempo de los proyectos, a impactos en la calidad del servicio y satisfacción de los clientes, disminución en la participación en el mercado y disminución en los ingresos por la afectación de las relaciones con los accionistas o la perdida de la imagen con el cliente [10].

Por último, es relevante indicar que no es la finalidad última de las pruebas de software determinar si un programa funciona o no; el propósito el generar un valor añadido al programa, consistente en mejorar la calidad y la confiabilidad, incluyéndose la eliminación o minimización de errores. Desde dicha lógica la definición de prueba en el ámbito del desarrollo del software puede ser entendida “como el proceso de ejecución de un programa con la intención de encontrar errores” [11].

2.2 Pasarela de pagos

La pasarela de pago o Terminal de punto de Venta Virtual (TPVV) hace referencia al dispositivo y tecnología que ayudan en las tareas de gestión de un establecimiento comercial de venta al público. Este dispositivo proporciona el servicio de pago telemático al ciudadano a través de Internet. Una pasarela de pago es un servicio de un proveedor de servicios de aplicación de comercio electrónico con el que se autorizan pagos a negocios electrónicos/online, ventas online al detalle, negocios con presencia física y online simultánea (modelo de negocio *brick and clics* o en su traducción literal "ladrillo y cliqueo"), o a negocios tradicionales (modelo de negocio *brick and mortar*, o en su traducción literal "ladrillo y mortero") [12].

Desde otra perspectiva, se define como una plataforma que presta un servicio intermediario entre una página o aplicación de comercio electrónico y una entidad

financiera cuando se realizan transacciones bancarias online. En otras palabras, se trata de una herramienta que se integra a la tienda virtual, encargándose de almacenar la información de la institución financiera que maneja las cuentas de compradores y vendedores. Así mismo, valida la veracidad del medio de pago empleado por el comprador y gestiona la transferencia del dinero desde la cuenta del comprador a la cuenta del vendedor [13].

Igualmente, son conocidas como “Gateway” de pago. Su funcionamiento gira en torno a la autorización de pagos o negocios electrónicos. En dicha labor cifran información sensible como son números de tarjetas de crédito, posibilitando que la transferencia de datos entre el cliente y la plataforma sea realizada de manera segura. En estas condiciones, la pasarela de pago cumple el papel de intermediario de pago entre la entidad financiera que recibe el pago y la tienda electrónica [14].

El funcionamiento de una pasarela de pago se produce en el momento aprueba un pago en una tienda electrónica que tenga habilitado el servicio de pago. En primer lugar, el cliente la compra dando clic en el botón de pago o similar, procediendo a introducir los datos de la tarjeta de crédito o su equivalente, por ejemplo, nombre completo titular, número de la tarjeta, fecha de vencimiento y código CSC. En segundo lugar, el navegador web o la aplicación cifra a través de SSL la información hacia la plataforma del vendedor. En tercer lugar, el servidor del vendedor reenvía la información de la transacción al servidor de pago ubicado en la pasarela de pago y procede a reenviar la información a la entidad bancaria del vendedor, En cuarto lugar, el banco emisor de la tarjeta recibe el pedido de autorización y envía la respuesta a la pasarela de pago, mediante la entidad bancaria del vendedor, incluyendo un código de respuesta, señalando si acepta o rechaza la operación. Por último, la pasarela de pago recibe la respuesta, procesa el pago y le informa al cliente el estado de la transacción [14].

Las Principales Pasarelas de pago en Colombia según la Asociación Colombiana de Empresas de Tecnología e Innovación Financiera (Colombia Fintech) estimaron que en Colombia operan al menos una decena de pasarelas de pago entre las que destacan PayU Latam, ePayco, Mercado Pago, PayPal y Wompi[9] Así mismo la revista online marketing4ecommerce autodenominada como “…el primer medio de comunicación online especializado en Marketing Digital, eCommerce y el ecosistema de startups” publica un top 10 de las mejores pasarelas de pago en Colombia donde se encuentran EPayco, Mercado Pago, Oyster, Pagos Inteligentes, PayPal, PayU, PayZen, Stripe, Wompi, Zona Pagos [15].

2.2.1. Wompi

Es una pasarela de pagos creada por Bancolombia que posibilita que las empresa y comercios inscritos en la plataforma puedan aceptar y procesar pagos desde cualquier lugar del mundo. Se caracteriza por aceptar diversos medios de pago como PSE, Nequi y las tarjetas de Nequi, tarjetas de crédito y débito, botón

Bancolombia y en efectivo por medio de las sucursales Bancolombia, así mismo funciona con los modelos Gateway y agregador. La afiliación a la plataforma es gratuita y dispone de tres planes para crear una cuenta: Básico, con tarifa gratuita por transacción aprobada, medio de pago aceptado las transferencias bancarias entre cuentas Bancolombia, incluye el servicio de generación de link de pagos para ser enviado al cliente; intermedio, con una tarifa de $ 649 COP + IVA, medio de pago aceptado las transferencias bancarias entre cuentas Bancolombia y la soluciones para el cliente Link de pagos, Widget / Web checkout y API; avanzado, dentro del cual se cobra una tarifa de $ 700 + 2.65% + IVA, en el cual se aceptan todos los medios de pago e incluye soluciones para el cliente del plan intermedio [16].

2.2.2. EPayco

Se trata de una pasarela de pagos especializada que permite a los clientes o comercios electrónicos recibir pagos, cobros y recargas de una manera sencilla con alto grado de seguridad. La principal característica de esta pasarela está representada en que, funciona como un modelo agregador y Gateway, lo cual le permite a las tiendas electrónicas recaudar el dinero de las operaciones comerciales a través de diferentes medios y formas de pago [17].

El enfoque de esta pasarela se expresa en tres conceptos: IU/UX, Seguridad, Tecnología. Lo primero, se refiere a un diseño y entorno de trabajo estructurado para satisfacer las necesidades de las empresas y los clientes; lo segundo hace alusión, a la garantía de la protección de todas las transacciones durante toda la operación; lo último, alude a una plataforma robusta que cumpla con los más altos estándares de desarrollo [18].

En relación con el modelo de negocio en que se sustenta la plataforma, cabe señalar que funciona de dos maneras: Gateway y Agregador. El primero, ofrece la posibilitad un procesamiento transaccional y la compensación directa en las cuentas bancarias de la empresa vendedora; la afiliación en este modelo para el año 2022 tiene un costo $ 490.000 IVA incluido y un valor por transacción exitosa que oscila entre $101 y $269, de acuerdo con los montos de la transacción. El segundo, incluye el procesamiento transaccional, compensación en cuentas de ePayco y el abono posterior a las cuentas de la empresa en un tiempo estimado de entre 24 y 72 horas siguientes a la operación; en este caso la afiliación es de carácter gratuito y una tarifa por transacción exitosa 2.68% + $900 + IVA para comercios con cuentas Davivienda o Daviplata, y de 2.99% + $900 + IVA cuando la cuenta del comerciante es de otro banco vigentes al 11 de julio de 2022 [19].

Así mismo, la pasarela admite una amplia gama de medios de pago: tarjetas de crédito Visa, Diners Club, Mastercard, American Express, Diners Club, Banco de Occidente, Codensa, Daviplata; Tarjetas débito y PSE; pagos internacionales, SAFETY PAY y PayPal; efectivo, Efecty, Red Servi, punto red, Su Red, entre otros [19].

2.3 *E-wallet* o Billetera Electrónica

En la actualidad se puede acceder a cualquier servicio o productos de forma digital utilizando nuestros smartphones, computadores o *tablets*, etc., y ha logrado un gran impacto en el sector financiero y sus servicios. Esto implica que desde estos dispositivos una persona pueda realizar las operaciones básicas para el manejo de su cuenta como consultar la cuenta, hacer consignaciones o transferir fondos haciendo que uno de sus beneficios sea el ahorro de tiempo al no tener que desplazarse a sus instalaciones físicas como cajeros u oficinas [20].

Las Electronic-wallet (e-wallet) conocidas en nuestro idioma como billetera electrónica, monedero electrónico o cartera digital es un nuevo medio de pago a través de internet el cual se basa en el concepto de una billetera de la forma tradicional o física de la cual se puede sacar dinero y hacer un pago.[21] Estos pagos pueden estar relacionados a bienes y servicios o cuando se necesita comprar algo. Las billeteras electrónicas se crean en un teléfono móvil y también se puede utilizar para los pagos electrónicos, ya que se puede vincular a sus cuentas electrónicas para realizar pagos en varios sitios web [22]. Ha obtenido gran relevancia por los beneficios que ofrece como la seguridad al permitir guardar por ejemplo los datos de las tarjetas de crédito y débito para acreditar las compras o pagos que se desean efectuar, comodidad al permitir gestionar el saldo y su cuenta desde una aplicación y rapidez al tratarse de transacciones en línea [21].

Desde el punto de vista de la su funcionalidad, la wallet o billetera digital se entiende que se trata de aquella herramienta que permite a sus usuarios almacenar, enviar y recibir tokens generados. El proceso por el cual se transfieren los tokens entre dos usuarios se trata de proceso que se conoce dentro de los sistemas de Blockchain regularmente como firma digital. Cada usuario cuenta con una clave privada y una clave pública que le permite interactuar con los otros usuarios del sistema. En este caso de uso, la interface de usuario será una interface de usuario simple, similar a la interface que se utiliza en las billeteras electrónicas más populares (mercado libre, Glovo, nubi, etc.) y como las billeteras electrónicas que se utilizan en los bancos comerciales [16].

Las billeteras electrónicas han adquirido una importancia inusitada en la actualidad, potencialidad en vigencia de la emergencia sanitaria del COVID 19, así se trata de un sistema de pago online cuya principal ventaja está representada en que simplifica las transacciones económicas que se realizan de manera electrónica. De esa manera, a los consumidores se le ha puesto a su disposición una herramienta para administrar los recursos económicos de forma virtual, permitiéndole realizar operaciones de manera sencilla y segura [17].

2.4 Mobile Wallet

La Mobile wallet o billetera móvil, es la forma más reciente de pago por medio del móvil o smartphone siendo estas una aplicación versátil y más avanzada que incluye una inscripción, el almacenamiento de la información de identidad del cliente, la información de pago y los detalles de la dirección de envío de forma segura, así como otros elementos que uno puede encontrar en una billetera, como tarjetas de membresía, tarjetas de fidelidad y tarjetas de viaje. En otras palabras, las billeteras móviles reemplazan una billetera por un smartphone equipado con las funciones de una tarjeta bancaria o una tarjeta de crédito logrando así convertirse en el motor del comercio móvil [23].

2.5 Servicio Delivery

El *delivery* es propio de la lengua inglesa, cuyo significado es entrega, vale decir, está referido al proceso logístico que por lo general antes era realizado por el vendedor directamente al cliente, el cual le correspondía trasportarlo por sus propios medios hasta su casa. Posteriormente, surgió la entrega a domicilio en los países de corte anglosajón, subsiguientemente traído a América Latina. Luego, este tipo de actividades es ejecutado a través de llamadas telefónicas que realiza el cliente y entregados directamente por la empresa. Ahora, por las condiciones requeridas para llevar a cabo la entrega, no todas las empresas tenían la capacidad para ofrecer la posibilidad de la entrega a domicilio, quedando muchas organizaciones sin poder comercializar sus productos a través de este mecanismo [18].

Con el paso del tiempo, la manera habitual en que se relaciona vendedor y el cliente se fue transformando, con una tendencia a la desaparición de compra directa en la tienda, remplazada por los avances tecnológicos, teniendo que las empresas acudir a las mismas para poder mantenerse en el mercado. Así, es en dicho contexto que es surge el servicio de delivery, en el cual el proceso logístico de entrega ya no lo realiza el vendedor, sino que el mismo es realizado por una empresa que presta el servicio por un costo adicional al producto o sin el mismo [18].

En Colombia, cuando se referiere a los domicilios o el servicio de domicilios prácticamente estamos hablando de *delivery*, una palabra que cada vez más rápido se entiende en otros idiomas sin necesidad de traducción y esto debido a la adopción de modelos de negocios basados en esta [24]. Básicamente su definición consiste, cito textualmente “…en el servicio que ofrece un comercio para entregar sus productos en el domicilio del comprador” [25]. El servicio cubre la entrega de comida preparada, de productos de supermercados como despensa y farmacia, y por lo general se pueden utilizar vehículos tipo motocicleta y bicicleta en su mayoría, pero el delivery no excluye automóviles y entregas a pie.

A principios de la década de 1990 en Polonia se inició la entrega a domicilio, limitándolo a un tipo de comida que es la entrega de pizza, quienes prestaban este servicio eran pertenecientes a grandes empresas cadenas globales. Otro tipo de restaurantes ocasionalmente hacían entregas a domicilio de almuerzos o cenas a sus clientes, situación que en los últimos 10 años ha cambiado, en los inicios de la entrega a domicilio los restaurantes prestaban el servicio directamente, tiempo después ya entraron empresas especializadas que interactúan como intermediarios entre restaurantes y clientes inicialmente se recibían los pedidos vía telefónica, pero actualmente se reciben estos pedidos vía internet, a través de una aplicación en un dispositivo móvil [24].

Más recientemente, las grandes empresas de alimentación han empezado a ofrecer a los clientes la entrega a domicilio de prácticamente todos los productos alimenticios ya que en el pasado se entregaban alimentos crudos o preprocesados. Esta tendencia ha sido aprovechada y las nuevas tecnologías relacionadas con la preparación de las comidas, el aumento de los conocimientos informáticos y el acceso a los teléfonos inteligentes han permitido a los consumidores encargar comidas preparadas para su casa u oficina. *“Esto es importante en el contexto de la gastronomía en el espacio urbano, porque significa que toda la ciudad se convierte efectivamente en el espacio gastronómico, en lugar de sólo los restaurantes, bares y cafés* [25].

2.6 Tecnologías utilizadas por Rinn App

Para el desarrollo del objeto central de la práctica, esto es, llevar a cabo el desarrollo de software backend de la empresa VICA Technology en la ejecución de actividades y la construir de funcionalidades para el aplicativo Rinn App será necesario, en mayor o menor medida, acudir a una serie de tecnología adicionales, las que se describen a continuación.

2.6.1 MongoDB

Se trata de una base de datos NoSQL de tipo no relacional que se centra en documentos. Es un sistema de base de datos multiplataforma que se orienta a documentos de código abierto y licenciado como GNU AGPL 3.0.[26]. En la actualidad, es considerado uno de los sistemas **NoSQL** que ha adquirido mayor renombre desde la creación de estos. Lo cual se ve reflejado en que muchas grandes empresas usan su sistema para su aplicación principal u otras aplicaciones internas, por ejemplo, MailBox de Dropbbox, Expedida. Linkedin, SAP, o MTV, ETC. Siendo aliado de grandes empresas del ambiente tecnológico como IBM o Red Hat [27].

Su principal rasgo es que reemplaza el concepto de "fila" por el "documento", permite que los documentos incrustados y *arrays*, al ser orientado a documentos permiten representar relaciones jerárquicas complejas con un único registro.[28]

Así mismo, para su funcionamiento emplea el estándar JavaScript Object Notation, o JSON para el manejo de los datos [29].

Entre las principales características de MongoDB se destacan:

1) Se trata de una base de datos *schemaless*, o sin esquemas, lo cual permite almacenar documentos sin que sea requerido que los mismos cumplan un esquema preestablecido. Además de estar dotada de una flexibilidad, la cual se desprende del empleo de JSON y BSON [27].
2) Alta velocidad y con facilidad para el aumento de la escalabilidad, lo cual se desprende que este sistema no dispone de carga transaccional. La escalabilidad horizontal, se expresa en que MongoDB está estructurado de manera ilimitada, mediante su replicación y su *sharding*, pudiendo almacenar información en varios equipos vinculados por red [30].
3) La base de datos que se analiza se caracteriza por una alta velocidad de consulta, lo cual es producto de que la información se almacena en archivos en formato BSON, versiones modificadas de JSON, lográndose la flexibilidad de datos, lo cual posibilidad mejorar la búsqueda y localización de la información [30].

Adicionalmente, se debe mencionar que la base de datos MongoDB por las características técnicas de: escalabilidad horizontal, permite búsquedas de campo, indexación para aumentar la eficiencia, replicación, indexación, puede ejecutarse en distintos servidores y enorme capacidad para almacenar información; elementos que hacen que pueda trabajar con grandes cantidades de información [30].

MongoDB en el desarrollo de API’s

MongoDB permite a los desarrolladores aprovechar las funcionalidades tales como: documentos JSON enriquecidos, lenguaje de consulta potente, transacciones de varios documentos y API auténticas [31]. Igualmente, a la automatización de tareas comunes de administración de bases de datos, como alta disponibilidad, copias de seguridad, cifrado y planificación de la infraestructura. Igualmente, MongoDB facilita la creación de aplicaciones innovadoras, pudiéndose escalar las mismas dependiendo de las necesidades de los clientes [32].

2.6.2 Elasticsearch

Elasticsearch es un motor distribuido de búsqueda y análisis, el cual se utiliza en aplicaciones con características de búsquedas complejas. En relación con su estructura se debe decir lo siguiente: está formado por clústeres, los cuales se tratan de nodos que almacenan los datos y brindan capacidades de búsqueda e

indexación de estos. Los clústeres están conformados por nodos, cada uno de los cuales son un servidor único que almacena datos y participa. Dentro del clúster se encuentran los nodos. Un nodo es un servidor único, el cual almacena los datos y participa en las búsquedas de indexación del clúster. A su turno, los nodos están formados por índices, los cuales se tratan de una serie de documentos con características semejantes. En los índices se realizan las operaciones de indexación, búsqueda, actualización y eliminación de los documentos que contiene [33].

En relación con el funcionamiento, cabe señalar que Elasticsearch está compuesto por dos capas. Por un lado, un sistema encargado de la coordinación de nodos de un clúster y mantenimiento de sus datos y, de otro lado, un motor de búsqueda con las funcionalidades de indexación y búsqueda de documentos [33].

El subsistema de coordinación funciona al elegir un nodo particular para que sea el maestro del clúster, es decir, crea un nodo que mediante el módulo "*Discovery*" busca un nodo maestro con el mismo nombre del clúster, si lo encuentra se une a él y si no, lo crea y se asigna el nodo como maestro [34].

En la capa del motor de búsqueda, al llegar los datos desde *Logstash*, estos se deben almacenar en el índice especificado en una lista de direcciones o en el indicado mediante el mapeo. En primer lugar, el cliente genera una petición a un nodo, este determina en qué fragmento primario debe indexarse y una vez hecho este paso, el documento en el fragmento primario se propaga a las réplicas. Pero si se realiza una consulta para recuperar información, el cliente realiza una petición a un nodo llamado coordinador, este determina en qué fragmento se encuentra el documento, envía la solicitud a los nodos de datos y estos realizan la solicitud de forma local, que consiste en realizar la consulta en cada fragmento, pasando por todos ellos mediante el método *round-robin* y finalmente, devuelve [35].

Dentro de los beneficios de Elasticsearch para una tienda virtual esta facilitar el desarrollo de motor de búsqueda propio, en razón a que el mismo puede personalizarse por completo. Así, cuando se opera una plataforma o tienda online, se puede configurar la función de búsqueda que incluya la exploración de perfiles registrados GCS [35].

2.6.3 Kibana

Se trata de un complemento de visualización de datos para Elasticsearch, de código abierto, el cual ofrece facilidades sobre el contenido indexado de un clúster de Elasticsearch. Gracias a esta herramienta los usuarios podrán crear gráficos estadísticos y mapas de grandes volúmenes de datos [36].

Kibana ofrece sus visualizaciones a través de una aplicación web. El punto en el que escuda el servidor web que soporta esta aplicación es fácilmente configurable.

De esta forma el usuario solo tiene que utilizar su navegador web para acceder a estas visualizaciones [36].

2.6.4 Redis

Se trata de un motor de base de datos, el cual puede entenderse como un almacén de estructura de datos en memoria de código abierto (licencia BSD), que se emplea como agente de base de datos, caché y mensaje. En su funcionamiento, está integrado por estructuras de datos como cadenas, hashes, listas, conjuntos, conjuntos ordenados con consultas de rango, mapas de bits, hiper blogs, índices geoespaciales con consultas de radio y flujos. Redis tiene replicación incorporada, secuencias de comandos *Lua*, desalojo de LRU, transacciones y diferentes niveles de persistencia en el disco, y proporciona alta disponibilidad a través de Redis *Sentinel* y particionamiento automático con Redis Clúster [37].

Como principales características del motor de base de datos se pueden mencionar las siguientes:

1) Velocidad en su desempeño, lo cual es consecuencia de que todos los datos de Redis se ubican en la memoria principal del servidor, a diferencia de la gran mayoría de base de datos en los que la información se hace en el disco o SSD. Así las cosas, al no ser necesario acceder a discos, Redis permite acceder a los datos a partir de algoritmos más sencillos, en los que se requieren un menor número de instrucciones en la CPU; de esa manera, las operaciones se ejecutan en menos de un milisegundo [37].
2) Estructura de datos en memoria, expresada en que Redis posibilita a los usuarios almacenar claves correspondientes a diferentes tipos de datos: Estructuras de datos en memoria: Redis permite a los usuarios almacenar claves que se corresponden con diversos tipos de datos. El tipo de datos fundamental es una cadena, que puede componerse de texto o datos binarios y tener un tamaño de hasta 512 MB. Redis también admite listas de cadenas en el orden en el que se han agregado, conjuntos de cadenas sin ordenar, conjuntos clasificados ordenados por puntuación, hashes que almacenan una lista de campos y valores, e *HyperLogLogs* que cuentan los elementos únicos de un conjunto de datos. Con Redis, se puede almacenar en la memoria prácticamente cualquier tipo de datos [38].

2.6.5 Rabbit MQ

Se trata de un bróker de mensajería de código libre, el cual está orientado a la distribución de mensajería entre aplicaciones. Su principal finalidad es servir de intermediario de mensajes o administración de colas para intercambiar datos entre diferentes aplicaciones [39].

Su funcionamiento se da a partir de la implementación del *Advance Message Queuing Protocol* (AMQP), cuya marcha posibilita la comunicación asincrónica entre trasmisor y receptor de los mensajes. Así, a través de AMQP se estructura una arquitectura modular, a partir de la cual son divididas las tareas de intercambios de mensajes [39].

La principal característica de Rabbit MQ es su capacidad de multidifusión. Así este bróker pueda enviar una información determinadas a múltiples destinatarios. Ello puede hacerse porque Rabbit MQ emplea una cola individual para cada usuario que se encuentra suscrito a la mensajería, de esa manera el bróker permite observar los consumidores que ya recibieron la información y los que están pendientes de recibirla. Siendo posible eliminar el mensaje una vez todos los destinatarios recepciones el mensaje [39]. Así mismo, se rescata que se trata de un bróker robusto y escalable, el cual está integrado con diversos plugin que posibilitan el monitoreo de la mensajería en tiempo real, así mismo establecer configuraciones recomendables para lograr un mejor desempeño en el movimiento de los mensajes y las colas de mensajería [40].

2.6.6 Nginx

Se trata de un software de servidor web de código abierto, open source, sustentado en la licencia BSD, de alto rendimiento, el cual puede fungir como servidor proxy que maneja eficientemente los protocolos IMAP/POP3, pudiendo igualmente actuar como proxy inverso y balanceador de carga [41].

La principal ventaja de este software es que dispone de una arquitectura asincrónica, controlada por eventos. Elementos que hace que Nginx sea considerado como uno de los servidores más confiables en la actualidad, principalmente porque está dotado de una alta velocidad y escalabilidad. Características que le permiten soportar muchas conexiones; siendo uno de los servidores más utilizados por sitios de alto tráfico, como por ejemplo gigantes de la internet como: Google, Netflix, Adobe, Cloudflare, WordPress.com y muchos más [42].

Otra característica, de Nginx que a la vez se convierte en una ventaja, es que el mismo tiene facilidad de configuración. En esa dirección, los ficheros de configuración son simples archivos de texto divididos en bloques de directivas, con campos de valores que pueden ser cambiados por el administrador, y que serán los que definan el comportamiento del servidor. Los bloques de directivas vienen determinados por los módulos del programa, es decir, si se activa un nuevo módulo en NGINX se dispondrá de un juego específico de directivas para él, pudiéndose indicar en la configuración. Además, es posible anidar bloques de directivas para indicar una configuración específica en caso de ser necesario. Dicha situación ofrece un alto grado de flexibilidad que ofrece el servidor con respecto a sus capacidades de configuración [43].

2.6.7 Verne MQ

VerneMQ se trata de un bróker MQTT de alto rendimiento. El cual goza de una alta escalabilidad, tanto desde el punto de vista horizontal como vertical. Gracias a ello, el bróker tiene el potencial para manejar una enorme cantidad y editores de forma simultánea. Ahora bien, dicha capacidad de manejo de un tráfico alto permita mantener una baja latencia y alta tolerancia a las fallas. Atributos que lo convierten en un centro de mensajes confiable para una plataforma de alto tráfico [44].

Para poder atender un alto tráfico, VerneMQ se fundamenta en Erlang OTP, la mejor tecnología actualmente disponible para construir sistemas de mensajería altamente escalables. Lo cual permite que VerneMQ se escale horizontal y verticalmente mediante la utilización completa de arquitecturas multinúcleo. Por tales razones el bróker actualmente tiene la capacidad de soportar más de un millón de conexiones simultáneas [45].

En esas condiciones, el bróker resulta muy atractivo para plataformas web de alto tráfico, primero porque es de fácil configuración, permitiendo que múltiples clientes se conecten con un mismo id, además porque ofrece la opción de suscripción compartida, elementos que explican su gran escalabilidad [45].

2.6.8 HA PROXY

Se trata de un software de distribución libre, cuya principal función es el balanceo de carga de tráfico, esto es distribuir las peticiones en los diversos servidores disponibles para evitar su saturación [45].

HA Proxy es un proxy inverso gratuito, muy rápido y fiable que ofrece alta disponibilidad, equilibrio de carga y proxy para aplicaciones basadas en TCP y HTTP. Es particularmente adecuado para sitios web de muy alto tráfico y alimenta una parte significativa de los más visitados del mundo. A lo largo de los años, se ha convertido en el equilibrador de carga de código abierto estándar de facto, ahora se envía con la mayoría de las distribuciones principales de Linux y, a menudo, se implementa de forma predeterminada en plataformas en la nube. Dado que no se anuncia a sí mismo, solo sabemos que se usa cuando los administradores lo informan. El equipo central de HA Proxy mantiene múltiples versiones en paralelo. Desde la versión 1.8, se emiten dos versiones principales cada año. El primer dígito generalmente indica un cambio importante (formato de configuración, etc.) pero en la práctica rara vez cambia. El segundo dígito indica nuevas características. Ambos constituyen una rama. Aparece un número adicional después de estos dígitos para indicar la versión de corrección de errores. El equipo central despliega muchos esfuerzos para respaldar las correcciones de las versiones anteriores y tiene mucho cuidado de no romper nada. Por esta razón, es muy importante mantenerse actualizado dentro de una sucursal, es decir, tener el número más alto posible en los últimos dígitos [46].

2.6.9 Postman

Se trata de un entorno de desarrollo API, el cual posee múltiples funciones para tal efecto: lo cual va desde las pruebas de las consultas hasta la documentación automática, lo cual involucra el diseño y la monitorización [47].

Por tales condiciones, se puede decir que está dirigido a desarrolladores web, permitiendo realizar peticiones HTTP a cualquier API. Su principal fortaleza es la utilidad que esta herramienta ofrece en el momento de programar y hacer pruebas, permitiendo comprobar el correcto funcionamiento del software desarrollado [48].

Además de ser una herramienta que permite realizar peticiones a API desarrolladas por terceros, así como probar las creadas por el propio desarrollador. Adicionalmente, esta herramienta brinda una interfaz de usuario amigable para la realización de solicitudes HTML, pues permite que se puede probar una funcionalidad de una API, evitando tener que escribir un código extenso solo para tal efecto [49].

Igualmente, brinda la posibilidad de guardar y agrupar un amplio conjunto de solicitudes que se conocen como "*Collections*", presentada en formas de sencillas carpetas en distintos niveles en los que se organizan las peticiones HTTP. Además de que la herramienta a la necesidad fundamental que demanda la creación y el funcionamiento de una API, consistente en la disposición de una documentación completa y lo suficiente estructurada. En esa dirección, Postman crea tal documentación de manera automática, utilizando la información de las peticiones y las descripciones introducidas al crearlas. La documentación puede ser configurada pública o privada, ofreciendo un límite de 1000 visualizaciones al mes en su versión básica. Así mismo, junto a la documentación que se genera, exhibe la manera como deben implementarse las peticiones a la API en diferentes lenguajes de programación, lo cual resulta muy útil a la hora de desarrollar.

3 CARACTERIZACIÓN DEL SISTEMA Y EL API

3.1 Caracterización del Sistema

Rinn App hace parte de las nuevas aplicaciones denominadas Super Apps debido a que ofrece diferentes servicios dentro de ella como la compra de productos y comida, un sistema de mensajería, un sistema de control para la gestión de los pedidos por geolocalización, una aplicación definida como panel administrativo; que sirve para llevar un control de las finanzas, para establecer parámetros de configuración y para gestionar las cuentas de los usuarios; una aplicación para que los domiciliarios, que dentro del contexto de Rinn se definen como rinneros, puedan gestionar los pedidos y una aplicación dedicada a los comercios con la

funcionalidad de que puedan tomar al instante los pedidos de los usuarios y además llevar un control del historial y los detalles del pedido.

3.1.1 La arquitectura de Microservicios

Rinn es un sistema robusto y escalable que encaja dentro de una arquitectura orientada a microservicios y para validarlo hay que definir y resaltar las características que presenta dicha arquitectura. Para ofrecer una visión de lo que se entiende por microservicios, es necesario señalar lo que se concibe por servicios, en ese sentido pueden entenderse como un conjunto de aplicaciones o tecnologías que tienen la capacidad para intercambiar datos entre ellos [50]. En esas condiciones, cuando se hace referencia a microservicio dentro del ámbito del desarrollo, se hace referencia a un componente elemental de software que se encarga de realizar una determinada función, haciéndola bien, por ejemplo: aprovisionar bajo demanda, escalar de forma elástica, proporcionar tolerancia a fallas y conmutación por error, etc. [51]. En el contexto de Rinn estos servicios se encuentran distribuidos en seis servidores que permanecen activos las 24 horas del día.

La arquitectura basada en microservicios hace referencia a un estilo de arquitectura de software, desde la cual se concibe que este elemento puede ser dividido en pequeños servicios de tal manera que sean los más independientes posibles entre ellos. En consecuencia, desde dicha perspectiva los microservicios pueden considerarse como pequeños servicios autónomos que trabajan de manera coordinada [52].

Entonces, el modelo de arquitectura basado en microservicios parte de la idea central de realizar la división el software de la aplicación de tal manera que funcionen de la forma más independientemente posible, pudiendo ser remplazables y actualizables sin mayores complicaciones, como por ejemplo el servicio de colas mqtt de Rinn App, que está en un servidor independiente y su configuración esta aislade de los demás servicios [51]. En otras palabras, “su idea fundamental es la separación de los diversos servicios que compondrían una aplicación, para así poder desplegarlos, modificarlos o ampliarlos cuando se desee, sin interferir con el resto. Por esta razón, la arquitectura de microservicios tiene una escalabilidad muy sencilla.” [53]. En Rinn se identifican los siguientes servicios: la base de datos, Redis, Elasticsearch, el manejador de colas, el balanceador de carga, el proxy inverso, las notificaciones *push*, las ofertas, el manejador de gráficos para la carga de imágenes, el API, el despachador, la aplicación de los clientes, los rinneros y los aliados.

El modelo fundamentado en microservicios rompe con las arquitecturas monolíticas y así mismo entra a solucionar dificultades que presentaban las mismas. A continuación, en la Figura 1 se compara de forma gráfica la arquitectura monolítica frente a la de microservicios:

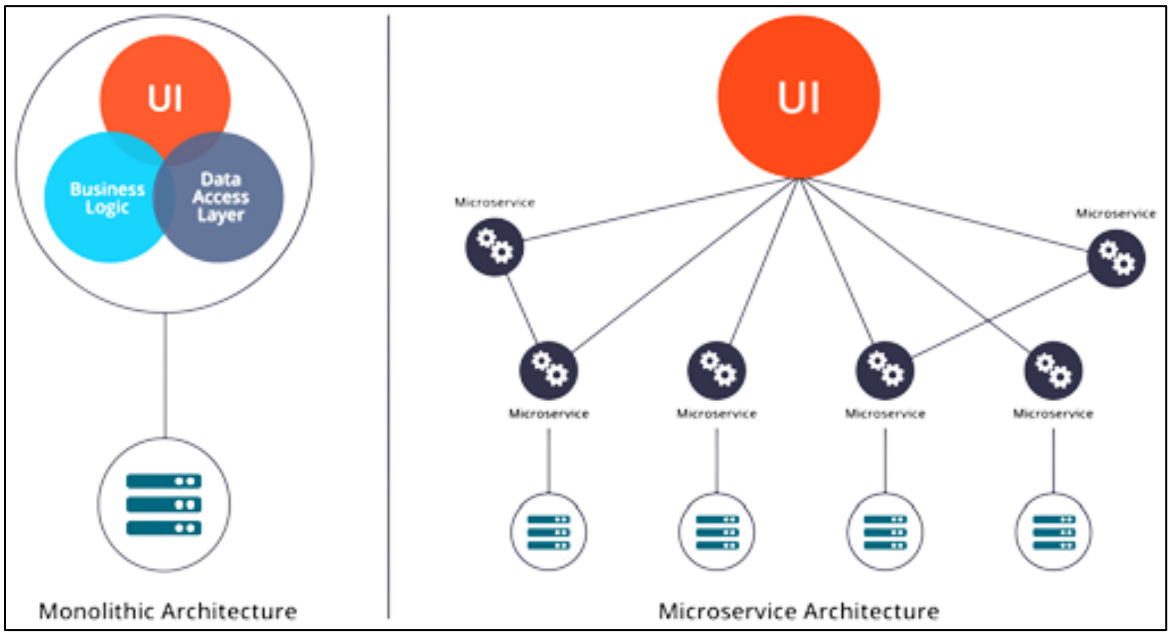


Figura 1. Comparación arquitecturas monolíticas y multiservicios [53].

En aquellas todos los procesos se encuentran estrechamente asociados, ejecutándose como un solo servicio, por lo cual cuando algún proceso de la aplicación presenta un pico de demanda, es necesario escalar toda la arquitectura. Igualmente, dentro de este modelo cuando se desea agregar o mejorar las características de una aplicación, ello resulta mucho más compleja a medida que se va extendiendo la base de código, aumenta la dificultad de las pruebas y la implementación de nuevas funcionalidades. Lo cual, a su vez, incrementa el riesgo de indisponibilidad de la aplicación porque el estrecho vínculo entre los procesos aumenta el impacto de un error en toda la aplicación.

Por el contrario, desde la arquitectura de enfoque en microservicios cada proceso de la aplicación es creado como componente independiente (seis servidores dedicados a las aplicaciones y tecnologías), cada uno de los cuales se ejecutan como un servicio y ellos a su vez se comunican a través de una interfaz bien definida por intermedio de API's ligeras y protocolos de comunicación específicos de los servicios. En esas condiciones, cada microservicio se puede actualizar, implementar y escalar para satisfacer la exigencia de picos de demanda de funciones específicas de la aplicación [54]. estos contrastes entre los modelos se ilustran en la siguiente manera:

Cabe mencionar la independencia, la cual se considera la característica más importante ya que gracias a eso es posible aprovechar las ventajas que este modelo de arquitectura, así como también la escalabilidad, flexibilidad y protección

frente a fallos. Por ello, los microservicios se despliegan de manera independiente [55].

Ventajas de la arquitectura basada en microservicios

- La disminución del impacto provocado por fallos. En una aplicación se pueden presentar fallos y su origen puede ser diverso, al hacer que los servicios sean independientes entre sí el impacto puede ser aislado a unos cuentos o solo un servicio [55]. Como por ejemplo al haber un problema con el servicio de Elasticsearch para las búsquedas las aplicaciones como Rinn App. Rinn driver o Rinn Aliado continúan funcionando, pero se inhabilita esta característica en su funcionamiento, pero las demás funcionalidades continúan en ejecución.
- La heterogeneidad del sistema, expresado en que a la aplicación sea capaz de permitir que los usuarios accedan a servicios y ejecuten aplicaciones bajo cualquier plataforma (esto es diferentes redes, hardware, sistemas operativos y lenguajes de programación) evidenciado en el contexto de Rinn como el acceso web a la aplicación del superadmin y del despachador y desde los dispositivos móviles para las aplicaciones del cliente, los rinneros y el aliado [56].
- La escalabilidad eficiente, que es la capacidad de respuesta al crecimiento del número de usuarios y a la posibilidad para replicar los recursos, para que un número mayor de usuarios pueda tener acceso a la aplicación, sin tener que replicar la totalidad de la aplicación, por ejemplo, cuando se quiere agregar un nuevo servidor de Redis para el manejo de eventos en Rinn lo que se hace es configurarlo de forma independiente y luego añadir su dirección en el balanceador de carga sin afectar ningún otro servicio [55].
- La flexibilidad y facilidad en la actualización, consiste en que la actualización se realiza solamente sobre el microservicio que se necesite con el fin de que no se vea afectada la toda de la aplicación. Lo cual, a su vez, les permite a las aplicaciones estar a la vanguardia de la tecnología, por cuanto pueden incorporar nuevos avances a través de actualizaciones individuales de microservicios específicos [56].

En la Figura 2 se representa la arquitectura de microservicios de alto nivel que se ha implementado en Rinn App. Los clientes de las aplicaciones móbiles estan en la parte superior, en el lado izquierdo el balanceador de carga que controla el trafico de los servicios de Redis, Elasticsearch, vernemq y RabbiMQ y el api junto con su respaldo. Y en el lado derecho esta el proxy reverso Nginx, las aplicaciones web del despachador y el panel administrativo y los servicios como las notificaciones, las ofertas, la búsqueda y la carga de imagenes.

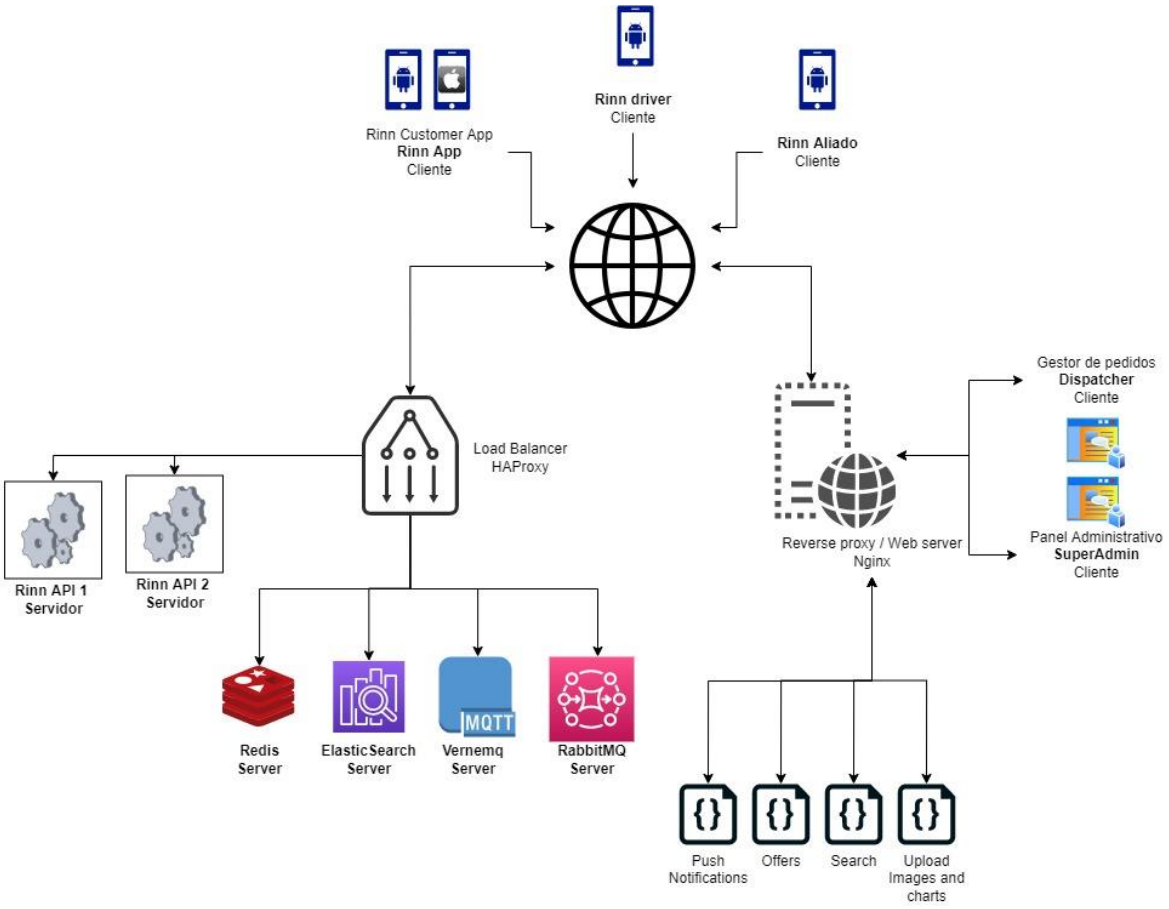


Figura 2. Arquitectura de microservicios de Rinn App [57].

Finalmente, las características anteriores implican un costo más elevado del que sería una arquitectura monolítica debido a que los servicios están corriendo en seis servidores diferentes y esto implica que cada microservicio consume sus propios recursos de memoria y CPU.

3.2 Caracterización del API de Rinn App

El API de Rinn App presenta una arquitectura cliente-servidor para establecer la comunicación entre los aplicativos Rinn Customer App para iOS y Android, siendo estos: Rinn driver, Rinn Aliado, el Superadmin y el despachador con el API y los demás servicios, y además, un sistema basado en capas compuesto por: 1) Capa de interacción, que es la encargada de exponer los datos a los usuarios y los servicios que provee, 2) Capa de aplicación que se encarga de contener las aplicaciones que dan la funcionalidad a la API y 3) Capa de gestión de la información encargada de almacenar y gestionar todos los datos relacionados a la configuración y lógica de negocio del sistema [58].

3.2.1 Arquitectura cliente-servidor

Desde el punto de vista tradicional, la arquitectura cliente-servidor se concibe como un modelo de sistemas distribuidos, en los cuales múltiples clientes realizan múltiples peticiones a múltiples servidores, que procesan la información y devuelven los resultados a quien corresponda. Escenario en que tanto los clientes como los servidores pueden ser modelados como componentes o conjunto de componentes, y por lo tanto podrían estar relacionados de alguna forma. Cada uno de ellos podría contener cierta lógica de negocios, propiedades, estados y reglas que definan su comportamiento.[59] En la estructura, intervienes dos actores, a saber:

Cliente: Programa ejecutable que participa activamente en el establecimiento de las conexiones. Envía una petición al servidor y se queda esperando por una respuesta. Su tiempo de vida es finito una vez que son servidas sus solicitudes, termina el trabajo.[57] Los clientes en son: Rinn Customer App para iOS y Android, Rinn driver, Rinn Aliado, el Superadmin y el despachador

Servidor: Es un programa que ofrece un servicio que se puede obtener en una red. Acepta la petición desde la red, realiza el servicio y devuelve el resultado al solicitante. Al ser posible implantarlo como aplicaciones de programas, puede ejecutarse en cualquier sistema donde exista TCP/IP y junto con otros programas de aplicación. El servidor comienza su ejecución antes de comenzar la interacción con el cliente. [57]En este caso el API de Rinn App.

En la Figura 3 se expresa en un alto nivel la arquitectura cliente-servidor empleada en Rinn App:

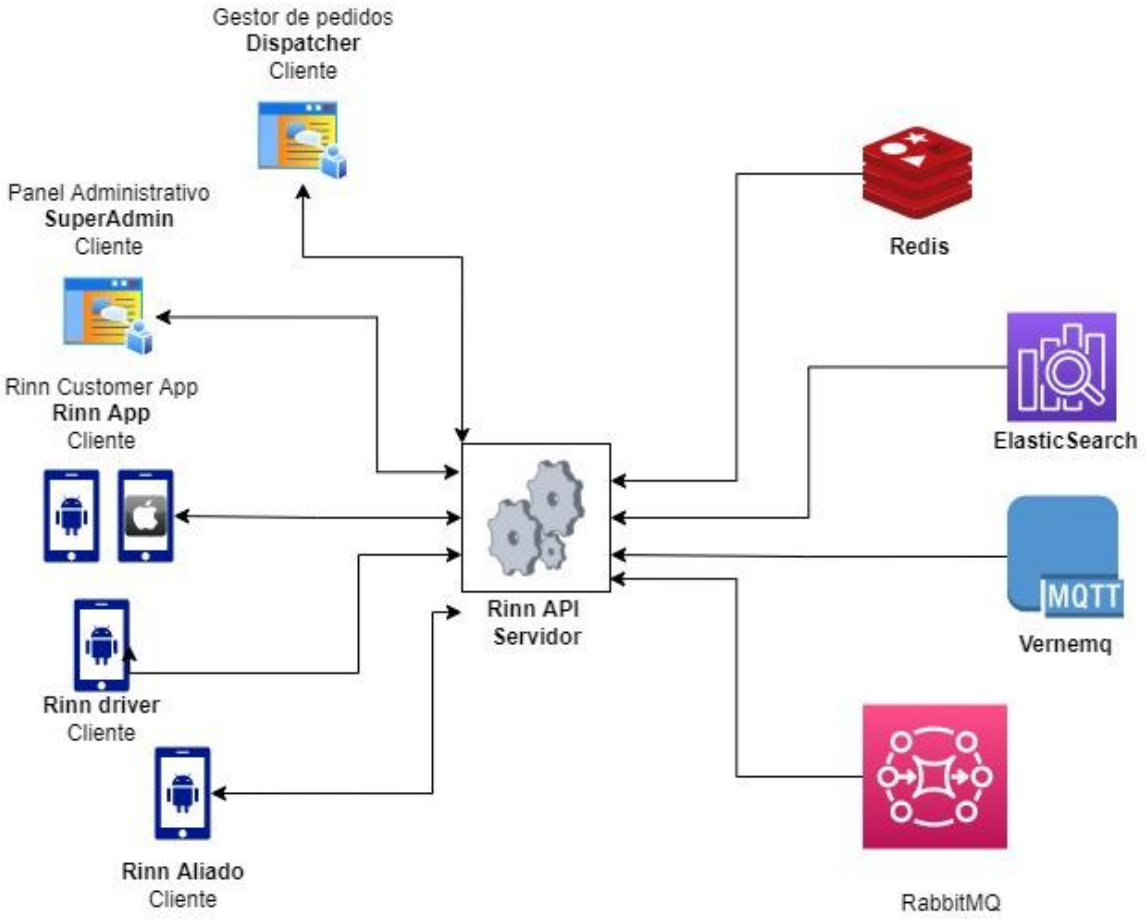


Figura 3. Arquitectura cliente servidor en Rinn App [57].

En donde los clientes (indicado bajo el nombre en el lado izquierdo) establecen una comunicación con el servidor en este caso el API para realizar peticiones.

3.2.2 Sistema de arquitectura en capas.

Se trata de un modelo de programación en el cual el objetivo preponderante es llevar a cabo la separación de la lógica de negocio de la lógica del diseño. El ejemplo más clásico de este esquema es la separación entre la capa de datos y la capa de presentación al usuario [60].

Concretamente, el modelo en capas se fundamenta en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. En esas condiciones, los roles se refieren al tipo y la forma de interacción con otras capas, en tanto que las responsabilidades exhiben las funcionalidades que implementas. Estructura que responde a la problemática que se refleja por el hecho de que cuando más aumenta el proceso operativo de la empresa, las necesidades crecen hasta desbordar los procesos, la estructura en varias campas absorbe el aumento del proceso operativo [61].

En la figura 4, se presenta el funcionamiento del modelo típico de capas

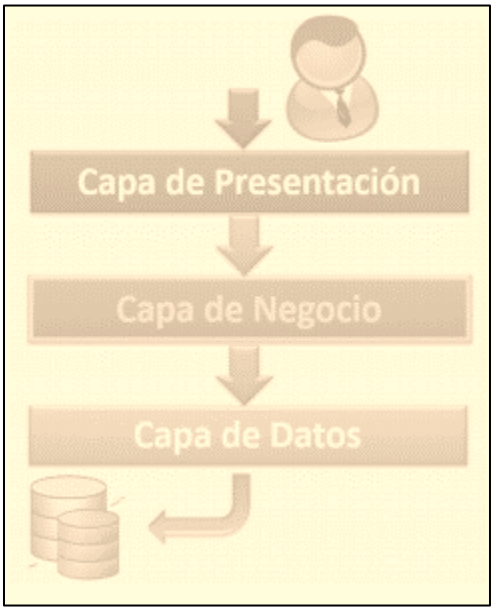


Figura 4. Modelo típico de capas [61].

La capa de presentación (interacción) es la responsable de la presentación visual de la aplicación. En este caso el API de Rinn provee una capa de presentación al exponer las rutas como la interfaz que da acceso a sus servicios. La capa de presentación enviará mensajes a los objetos de la capa de negocio, la cual responderá directamente o mantendrá un diálogo con la capa de datos, la cual proporcionará los datos que se enviarán como respuesta a la capa de presentación [61]. La definición de esta capa se encuentra contenida en el directorio *routes* contenida dentro de la carpeta *web*, como se puede observar en la Figura 5.

La capa de negocio es la responsable del procesamiento que tiene lugar en la capa de aplicación. Esta capa intermedia contendría los objetos que corresponden con las entidades de la aplicación, la capacidad de mantenimiento y de reutilización [61]. En la Figura 5 se puede observar la definición de esta capa se encuentra contenida en el directorio *commonModels* contenida dentro de la carpeta *web*.

La capa de datos se encarga de acceder a los datos, se debe usar la capa de datos para almacenar y recuperar toda la información de sincronización del Sistema. Es aquí donde se implementa las conexiones al servidor y la base de datos propiamente dicha, se invoca a los procedimientos almacenados los cuales reciben solicitudes de almacenamiento o recuperación de información desde la

capa de negocio. Conexión a la base de datos, en la cual se estable la comunicación con el gestor de base de datos a través de dos funciones, una que permite la conexión y un más que me permite desconectarme del gestor [61]. La definición de esta capa se encuentra contenida en el directorio *models*, como se observa en la Figura 5.

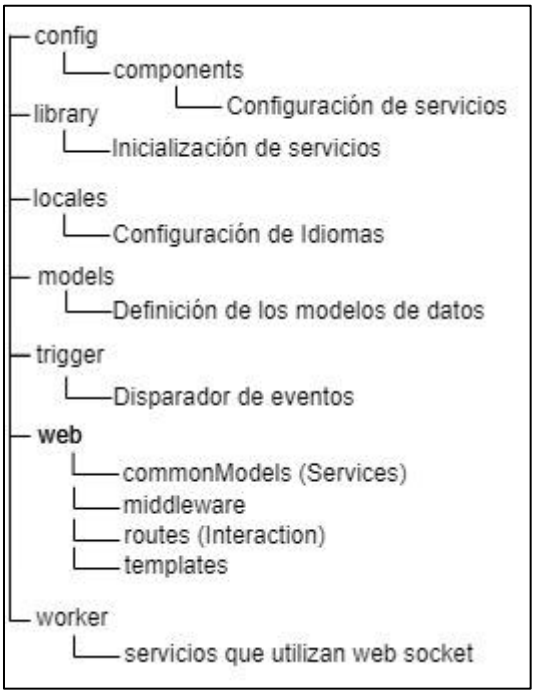


Figura 5. Organización lógica de directorios del código del API [61].

Por lo anterior, fueron identificados los lugares específicos en que se agregarían las nuevas funcionalidades, que se ubican en la carpeta de enrutadores (*routes*), la cual hace parte de la capa de interacción, en los controladores de la capa de aplicación y en los modelos de la capa de gestión de la información. Todo esto conlleva a la comprensión de aspectos relacionados con la programación y adaptación que permitan continuar usando el patrón y prácticas de desarrollo ya implementadas conservando así la integridad del código y su arquitectura.

4 REQUERIMIENTOS E ITERACIONES PARA LA IMPLEMENTACIÓN DEL AMBIENTE DE PRUEBAS Y LA CONSTRUCCIÓN DEL SOFTWARE

La ejecución de esta práctica profesional fue dividida en dos partes importantes para llevar a cabo los objetivos propuestos. La primera parte está enfocada en la implementación de un ambiente de pruebas ya que es inexistente en el entorno de desarrollo de Rinn App, la cual se realizó durante la segunda iteración, y la

segunda parte es la construcción del software en donde se realizará el desarrollo de las funcionalidades deseadas y comprende la tercera, cuarta y quinta iteración.

Es importante mencionar que la empresa VICA Technology utiliza un enfoque de desarrollo ágil, SCRUM, y para cada iteración realizó una planeación y estimación de actividades especificada en su respectivo *sprint backlog*. Las iteraciones fueron realizadas por mes con una intensidad horaria de 8 horas diarias, una duración máxima de 196 horas por mes y distribuidas en 6 días a la semana de lunes a sábado.

Las actividades estimadas tienen una prioridad que fue establecida por la empresa dependiendo de relevancia de los objetivos de la práctica, es decir, las actividades entre más sea su aporte y complementen los objetivos específicos mayor prioridad tendrá dicha actividad. También tienen un tiempo estimado de realización en horas.

4.1 REQUERIMIENTOS

A continuación, se presenta el consolidado de los requerimientos solicitados por la empresa VICA Technology, los cuales fueron organizados de la siguiente forma:

1. Para la implementación del ambiente de pruebas los requerimientos son asignados a cada servidor que se configurará, en este caso son dos servidores uno para la persistencia de datos y otro para el balanceador de carga, el proxy inverso y las aplicaciones.

2. Para las funcionalidades los requerimientos se agrupan por componentes lógicos para facilitar el diseño y el desarrollo de las iteraciones o *sprints*.

4.1.1 Requerimientos para el Ambiente de pruebas

El servidor de Gestión de Información o servidor de persistencia de datos debe tener instalado y configurado el sistema de base de datos, el motor de búsqueda, el panel de visualización de datos para el motor de búsqueda y el sistema de base de datos en memoria. Los requerimientos y versiones se especifican en la Tabla 1.

Tabla 1 - Requerimientos referentes a implementar el ambiente de pruebas en el servidor de Gestión de Información

Servidor	Requerimientos
Gestión De Información	Instalar la versión empleada de MongoDB por Rinn App (4.0.9) y configurar el respaldo existente

	Restringir el acceso remoto a la base de datos y establecer protocolos de seguridad
	Instalar la versión empleada de Elasticsearch por Rinn App (6.x) y establecer los parámetros de configuración adecuados al servidor
	Restringir el acceso remoto a Elasticsearch y establecer protocolos de seguridad
	Instalar la versión empleada de Kibana por Rinn App y configurar el respaldo de índices existente
	Instalar la versión empleada de Redis por Rinn App (3.0.6)
	Restringir el acceso remoto a Redis y establecer protocolos de seguridad

El servidor del Balanceador de carga, Proxy inverso y aplicaciones contiene la parte central del sistema o lo que podría decirse lo que le da vida a Rinn App. Debe tener instalado y configurado el balanceador de carga HAProxy, el proxy inverso y servidor web Nginx, el bróker de mensajería RabbitMQ y el bróker de mensajería Vernemq, además se debe configurar el certificado SSL para el tráfico HTTPS y desplegar las aplicaciones web del superadmin, el despachador de pedidos, las aplicaciones que gestionan las notificaciones *push*, las ofertas, las búsquedas, la carga de imágenes y gráficos y el API de Rinn App. Las versiones y el detalle de los requerimientos se especifican en la Tabla 2.

Tabla 2 - Requerimientos referentes a implementar el ambiente de pruebas en el servidor del balanceador de carga, proxy inverso y aplicaciones

Servidor	Requerimientos
Balanceador de carga, Proxy inverso y aplicaciones	Instalar la versión empleada de HA Proxy por Rinn App (1.6.3) y establecer parámetros de configuración
	Registrar los servicios del servidor de Gestión de Información para redirigir el tráfico
	Instalar la versión empleada de Rabbit MQ por Rinn App (3.7.15) y configurar el respaldo de colas existente
	Instalar la versión empleada de Nginx por Rinn App (1.10.3)
	Establecer la configuración de Nginx para redirigir el tráfico hacia el panel administrativo, el despachador de pedidos, la API y los servicios de notificaciones, ofertas, carga de gráficos y búsqueda
	Configuración del certificado SSL para la redirección a los servicios en Nginx
	Instalar la versión empleada de Vernemq por Rinn App (1.9.1) y agregar el certificado SSL a su configuración

	Desplegar el código del panel Administrativo (Superadmin)
	Desplegar el código del API de Rinn App
	Desplegar el código del despachador de pedidos
	Desplegar el código de notificaciones *push*
	Desplegar el código para la gestión de las ofertas
	Desplegar el código para la gestión de las búsquedas
	Desplegar el código para la carga de imágenes y gestión de gráficos

Requerimientos Construcción del Software

El método de pago con tarjetas requiere ser procesado por una nueva pasarela de pagos, almacenar las fuentes de pago que genera en una nueva estructura de información, actualizar el flujo y la lógica de cómo se procesan las transacciones y, además, se debe definir un CRUD para gestionar este método de pago. A continuación, se detallan los requerimientos en la Tabla 3.

Tabla 3 - Requerimientos método de pago con tarjeta

FUNCIONALIDAD 1: MÉTODO DE PAGO TARJETAS	
Componente	**Requerimientos**
Lista los métodos de recarga	Proveer un endpoint que permita listar este método de pago
	Incluir un código que lo identifica, el valor máximo y mínimo de recarga, la URL de la imagen e indicar si está habilitado para realizar recargas.
Adición de una fuente de pago con Tarjeta	Proveer un endpoint que permita adicionar una fuente de pago con tarjeta que reciba el número, el CVC, mes y año de expiración y el propietario de la tarjeta
	Los valores recibidos deben estar encriptados
	Se deben validar todos los valores recibidos
	Se procesa la información, se almacena la fuente de pago recibida y los últimos 4 dígitos del número de la tarjeta
Lista de fuentes de pago	Proveer un endpoint que permita listar las fuentes de pago registradas por un usuario
Eliminar una fuente de pago	Proveer un endpoint que permita eliminar una fuente de pago registrada por un usuario
Establecer una fuente de pago por defecto	Proveer un endpoint que permita establecer una fuente de pago por defecto para realizar pagos.
	Se debe indicar en el detalle de todas las fuentes de pago si es por defecto
	Sólo puede existir una fuente de pago por defecto

El método de pago PSE requiere ser procesado por una nueva pasarela de pagos y actualizar el flujo y la lógica de cómo se procesan las transacciones. A continuación, se detallan los requerimientos en la Tabla 4.

Tabla 4 - Requerimientos método de pago PSE

FUNCIONALIDAD 2: MÉTODO DE PAGO PSE	
Componente	**Requerimientos**
Lista los métodos de pago (recarga)	Proveer un endpoint que permita listar este método de pago
	Incluir un código que lo identifica, el valor máximo y mínimo de recarga, URL de la imagen e indicar si está habilitado para realizar recargas.
Procesar un pago con PSE	Proveer un endpoint que permita procesar un pago con PSE que reciba el tipo de persona, el banco, el valor, la cedula y la dirección IP del dispositivo.
	Se deben validar todos los valores recibidos
	El endpoint entregará en la respuesta el link de redirección para continuar el proceso de pago con el respectivo banco

El método de pago Botón Bancolombia requiere ser implementado usando la nueva pasarela de pagos y crear el flujo y la lógica de cómo se procesan las transacciones. A continuación, se detallan los requerimientos en la Tabla 5.

Tabla 5 - Requerimientos método de pago Botón Bancolombia

FUNCIONALIDAD 2: MÉTODO DE PAGO BOTÓN BANCOLOMBIA	
Componente	**Requerimientos**
Lista los métodos de pago (recarga)	Proveer un endpoint que permita listar este método de pago
	Incluir un código que lo identifica, el valor máximo y mínimo de recarga, URL de la imagen e indicar si está habilitado para realizar recargas.
Procesar un pago con Botón Bancolombia	Proveer un endpoint que permita procesar un pago con Botón Bancolombia que reciba el valor del pago.
	Se debe validar el valor
	El endpoint entregará en la respuesta el link de redirección para continuar el proceso de pago con el respectivo banco

El método de pago Nequi requiere ser implementado usando la nueva pasarela de pagos, almacenar las fuentes de pago que genera en una nueva estructura de información, crear el flujo y la lógica de cómo se procesan las transacciones y, además, se debe definir un CRUD para gestionar este método de pago. A continuación, se detallan los requerimientos en la Tabla 6.

Tabla 6 - Requerimientos método de pago Nequi

FUNCIONALIDAD 4: MÉTODO DE PAGO NEQUI

Componente	Requerimientos
Lista los métodos de pago (recarga)	Proveer un endpoint que permita listar este método de pago
	Incluir un código que lo identifica, el valor máximo y mínimo de recarga, URL de la imagen e indicar si está habilitada para realizar recargas.
Adición de una fuente de pago con Nequi	Proveer un endpoint que permita adicionar una fuente de pago con Nequi que reciba el nombre, la cédula y el número de teléfono del usuario Nequi.
	Se deben validar todos los valores recibidos
	Se procesa la información, se almacena la fuente de pago recibida y el número de teléfono
Lista de fuentes de pago	Proveer un endpoint que permita listar las fuentes de pago con Nequi registradas por un usuario
Eliminar una fuente de pago Nequi	Proveer un endpoint que permita eliminar una fuente de pago con Nequi registrada por un usuario
Establecer una fuente de pago por defecto	Proveer un endpoint que permita establecer una fuente de pago por defecto para realizar pagos.
	Se debe indicar en el detalle de todas las fuentes de pago si es por defecto
	Sólo puede existir una fuente de pago por defecto

Las fuentes de pago generadas con tarjetas y que son almacenadas deben pasar por una validación de seguridad que ayude a garantizar que quien autoriza la adición de esta fuente para pagos es el propietario. Los requerimientos relacionados a la validación se especifican en la tabla 7.

Tabla 7 - Requerimientos validación fuente de pago con tarjeta

FUNCIONALIDAD 5: VALIDACIÓN DE FUENTE DE PAGO CON TARJETA	
Componente	**Requerimientos**
Adición de una fuente de pago con Tarjeta	Cuando se adiciona una fuente de pago con tarjeta se debe generar un cobro aleatorio entre 100 y 500 pesos y almacenarlo en los detalles de la fuente de pago
	Se debe guardar la fuente de pago con estado de pendiente de validación
Validar una fuente de pago con tarjeta	Proveer un endpoint que permita validar una fuente de pago con tarjeta que reciba el código de la fuente de pago y el valor del cobro aleatorio.
	Si la validación es exitosa se actualiza el estado a fuente de pago validada
	Se deben permitir únicamente 3 intentos para validar la fuente de pago. Si se completan quedará con un estado de no validada.
Lista de fuentes de pago	El endpoint que permitirá listar las fuentes de pago registradas por un usuario validadas y pendientes de validación

Las fuentes de pago generadas con Nequi y que son almacenadas deben ser validadas debido a que la autorización la realiza el usuario por medio de la aplicación de Nequi. Los requerimientos relacionados a la validación se especifican en la tabla 7.

Tabla 8 - Requerimientos validación fuente de pago con Nequi

FUNCIONALIDAD 6: VALIDACIÓN DE FUENTE DE PAGO NEQUI	
Componente	**Requerimientos**
Adición de una fuente de pago con Nequi	Cuando se adiciona una fuente de pago con nequi se debe almacenar con un estado de pendiente de validación
Validar una fuente de pago con Nequi	Proveer un endpoint que permita validar una fuente de pago con Nequi que verifica si el usuario ya aprobó la solicitud en su aplicación de Nequi.
	Si el usuario la aprobó se actualiza el estado a fuente de pago a validada
Lista de fuentes de pago	El endpoint que permitirá listar las fuentes de pago registradas por un usuario validadas y pendientes de validación

Las fuentes de pago con tarjeta agregadas y validadas deben permitir realizar recargas a la billetera de Rinn App. A continuación, en la tabla 9 se listan los requerimientos relacionados:

Tabla 9 - Requerimientos recarga de billetera con fuente de pago con tarjeta

FUNCIONALIDAD 7: RECARGA DE BILLETERA CON FUENTE DE PAGO CON TARJETA	
Componente	**Requerimientos**
Creación de un pago usando fuente de pago con tarjeta	Proveer un endpoint que permita crear un pago para recarga de billetera usando una fuente de pago con tarjeta
	Debe recibir el código de la fuente de pago, la cantidad y la dirección IP del dispositivo.
	Se deben validar todos los valores recibidos
	La respuesta debe incluir los datos del resultado del pago
Billetera	Actualizar el saldo de la billetera después de procesar exitosamente el pago

Las fuentes de pago con Nequi agregadas y validadas deben permitir realizar recargas a la billetera de Rinn App. A continuación, en la tabla 10 se listan los requerimientos relacionados:

Tabla 10 - Requerimientos recarga de billetera con fuente de pago con Nequi

FUNCIONALIDAD 8: RECARGA DE BILLETERA CON FUENTE DE PAGO CON NEQUI	
Componente	**Requerimientos**
Creación de un pago usando fuente de pago con tarjeta	Proveer un endpoint que permita crear un pago para recarga de billetera usando una fuente de pago con Nequi
	Debe recibir el código de la fuente de pago, la cantidad y la dirección IP del dispositivo.
	Se deben validar todos los valores recibidos
	La respuesta debe incluir los datos del resultado del pago
Billetera	Actualizar el saldo de la billetera después de procesar exitosamente el pago

El método de pago PSE debe permitir realizar recargas a la billetera de Rinn App. A continuación, en la tabla 11 se listan los requerimientos relacionados:

Tabla 11 - Requerimientos recarga de billetera con PSE

FUNCIONALIDAD 9: RECARGA DE BILLETERA CON PSE	
Componente	**Requerimientos**
Validación del pago de PSE	Proveer un endpoint que permita validar un pago PSE para recarga de billetera
	Debe recibir un identificador del pago
	Se deben validar identificador el identificador recibido
	La respuesta debe incluir los datos del resultado del pago
Billetera	Actualizar el saldo de la billetera después de procesar exitosamente el pago

El método de pago Botón Bancolombia debe permitir realizar recargas a la billetera de Rinn App. A continuación, en la tabla 12 se listan los requerimientos relacionados:

Tabla 12 - Requerimientos recarga de billetera con Botón Bancolombia

FUNCIONALIDAD 10: RECARGA DE BILLETERA CON BOTÓN BANCOLOMBIA	
Componente	**Requerimientos**
Validación del pago de PSE	Proveer un endpoint que permita validar un pago con Botón Bancolombia para recarga de billetera
	Debe recibir un identificador del pago
	Se deben validar identificador el identificador recibido
	La respuesta debe incluir los datos del resultado del pago
Billetera	Actualizar el saldo de la billetera después de procesar exitosamente el pago

La nueva billetera de Rinn app requiere implementar productos de compra de SOAT y pago de Servicios Públicos. A continuación, en la tabla 13 se listan los requerimientos relacionados:

Tabla 13 - Requerimientos billetera móvil de Rinn App

FUNCIONALIDAD 11: MÓDULO DE PAGOS Y RECARGAS	
Componente	**Requerimientos**
Billetera	Proveer un endpoint que permite obtener la lista de los tipos de productos de billetera habilitados
SOAT	Proveer un endpoint que permite crear una cotización de SOAT el cual recibe la placa del vehículo, el tipo de vehículo y el tipo y número de identificación del propietario
	Se deben validar todos los valores recibidos
	Debe retornar la información de 1 o más pólizas si aplica el vehículo
	Para vehículos de más de una póliza se debe proporcionar un endpoint que permita actualizar la cotización del SOAT
	Proporcionar un endpoint que permita realizar la compra del SOAT el cual recibe los datos del comprador como nombre, genero identificación, celular, correo electrónico, ubicación y el identificador de la cotización
	Al finalizar la compra se debe retornar la información respectiva al resultado de la transacción
Servicios Públicos y facturas	Proveer un endpoint que permite consultar el costo de una factura de servicios públicos el cual el proveedor del servicio y la referencia de la factura
	Se deben validar todos los valores recibidos
	Debe retornar la información del costo de la factura asociado a un identificador
	Proporcionar un endpoint que permita realizar el pago de la factura el cual recibe el identificador anterior
	Al finalizar la compra se debe retornar la información respectiva al resultado de la transacción
Recarga de Celular	Proveer un endpoint que permite listar los operadores móviles en Colombia disponibles para recarga de saldo
	Proporcionar un endpoint que permita realizar la compra de una recarga de celular el cual recibe el número de teléfono y el monto deseado
	Se debe validar que el número de celular sea válido en Colombia y que el monto esté dentro de un rango especifico
	Al finalizar la compra se debe retornar la información respectiva al resultado de la transacción

Para desarrollar los requerimientos que la empresa VICA Technology solicita utilizando SCRUM, se dividen los requerimientos en actividades y cada sprint o

iteración tiene una duración de 1 mes, donde cada actividad dentro de cada sprint tiene una prioridad (baja, media o alta, según lo indique la empresa) y dos duraciones: una prevista y otra real (ambas en horas). A continuación, se documenta la primera iteración para la cual se especifican las actividades relacionadas a su finalidad y las iteraciones siguientes en el capítulo 5 realizadas a lo largo de la práctica profesional basándose en un proceso enmarcado en 3 pasos: agrupar las actividades de manera lógica, analizar el sprint con respecto a las estimaciones y prioridades, y mostrar el producto generado. Cabe resaltar que, para el caso de las estimaciones, en VICA Technology se tiene las siguientes rúbricas:

- Una estimación es buena si existe un desfase con respecto a la duración real menor o igual al 10%
- Una estimación es regular si existe un desfase con respecto a la duración real entre 10% y 20%
- Una estimación es mala si existe un desfase con respecto a la duración real mayor al 30%

4.2 ITERACIONES

Durante la práctica se desarrollaron cinco (5) iteraciones con una duración de un mes por iteración. Cada iteración corresponde a un sprint de desarrollo iniciando desde el sprint 0 hasta el sprint 4. A continuación se detallan las actividades desarrolladas durante cada una de ellas, se da un resumen de las actividades y un análisis del tiempo respecto a la estimación. La primera iteración correspondiente al sprint 0 se detalla a continuación y las iteraciones restantes serán detalladas en el capítulo 5.

4.2.1 PRIMERA ITERACIÓN

Se definió como sprint cero está primer iteración debido a los objetivos que se definieron. Como tal fue una preparación de manera intensiva para poder abordar de forma adecuada los requerimientos de las iteraciones siguientes. En esta primera iteración se recibió una capacitación de temas relacionados al negocio, a las razones que llevaron a la creación de Rinn y las proyecciones que tienen con su crecimiento y evolución y principalmente se entran a tocar los temas relacionados al sistema y las tecnologías del ecosistema de Rinn.

A continuación, se muestra una especificación de las actividades que comprenden la primera iteración:

Tabla 14 - Primera iteración: actividades referentes a conocer el funcionamiento del sistema

FINALIDAD	ACTIVIDADES
Conocer el funcionamiento del sistema	Recibir una capacitación sobre la misión visión y objetivos de la Empresa VICA Technology y su aplicativo Rinn App
	Primer acercamiento al sistema Rinn App, arquitectura, componentes y tecnologías

La empresa VICA Technology al iniciar la práctica presentó su aplicativo Rinn App de forma detallada, se expusieron el objetivo principal de la aplicación, la población objetivo, la visión que tienen de expansión a futuro, los lineamientos con que se rigen y además el estado actual de la empresa en cuanto a sus valores estadísticos de crecimiento económico y pronósticos. Adicionalmente fue presentado el equipo de trabajo, entre ellos, los transportadores que son denominados rinneros, despachadores y directivos. Posteriormente, fueron abordadas cuestiones técnicas en donde se enseñó la infraestructura de hardware y software que utiliza la aplicación tal como la distribución de los servidores y la capacidad de estos, y aspectos relacionados con la arquitectura en que se desarrolló.

A partir de lo anterior, se identificaron los componentes que conforman Rinn App, primero las aplicaciones individuales, siendo estas: la aplicación del despachador de pedidos, la aplicación del rinnero, la aplicación del cliente, la aplicación del administrador y segundo la relación entre los servidores y las tecnologías que emplean para su comunicación y funcionamiento. Cabe destacar que fue necesario repetir algunas explicaciones para comprender la arquitectura dada la complejidad y el tamaño de la aplicación, y la falta de experiencia en este tipo de sistemas en un ambiente de producción.

Al conocer la API que brinda el funcionamiento a la aplicación se evidencia que está compuesta de muchos servicios, dependencias y paquetes o librerías, lo cual permitió inferir que, aunque es un sistema complejo, dichos elementos son necesarios para que funcione adecuadamente y en un ambiente de producción. Como tal la API emplea un framework de desarrollo denominado Hapi para Node.js, el cual fue desarrollado originalmente para manejar la alta demanda del Black Friday de Walmart y continúa siendo una opción para las necesidades de backend de nivel empresarial [62]. Esta API tiene una organización de directorios para el código, entre ellos, uno para las rutas, otro para los modelos de base de datos (los cuales emplean Mongo un gestor no relacional), otro para los controladores y otro para los middlewares, entre los más importantes por mencionar.

Tabla 15 - Primera iteración: actividades referentes a comprender la relación entre sus componentes

FINALIDAD	ACTIVIDADES
Comprender la	Investigar, conocer y aprender a utilizar la tecnología PM2

relación entre sus componentes	Investigar, conocer, comprender y aprender a utilizar la tecnología REDIS
	Investigar, conocer, comprender y aprender a utilizar la tecnología RABBIT MQ
	Investigar, conocer, comprender y aprender a utilizar la tecnología NGINX
	Investigar, conocer, comprender y aprender a utilizar la tecnología HAPROXY
	Investigar, conocer, comprender y aprender a utilizar la tecnología ELASTIC SEARCH
	Investigar, conocer, comprender y aprender a utilizar la tecnología VERNE MQ

Tabla 16 - Primera iteración: actividades referentes a caracterizar el código legado

FINALIDAD	ACTIVIDADES
Caracterizar el código legado	Exposición de la API que utiliza Rinn App
	Identificación de las arquitecturas de software aplicadas
	Identificación de las áreas específicas de trabajo dentro de la API

Tabla 17 - Primera iteración: actividades referentes a Desarrollo de una API con HapiJs y MongoDB

FINALIDAD	ACTIVIDADES
Desarrollo de una API con HapiJs y MongoDB	Crear un servidor usando HapiJs
	Autenticación de usuario
	Creación de rutas para la gestión de usuarios
	Manejo de archivos estáticos
	Almacenamientos de usuarios en base de datos

Durante este proceso se tomó como base fundamental la documentación oficial del framework HapiJs [63]. y de Mongo DB [64]. y se usaron paquetes que se conocían con experiencia previa como *handlebars, inert, Mongoose y Nodemon*. El API consistía en proveer rutas para la autenticación de usuarios con un nombre de usuario (*username*) y una contraseña (*password*) y también para almacenar nuevos usuarios registrando mediante un formulario web el nombre de un usuario lo en base de datos e iba mostrando por pantalla los registros guardados. Para ello fue necesario crear un CRUD de usuarios, cada usuario tenía los atributos *username, password* y la fecha de creación. Usando mongoose se creó un esquema y se exporto como un modelo para ser utilizado dentro del API. Para renderizar las vistas se emplearon las funcionalidades del framework junto con el paquete handlebars el cual se configura para que el directorio donde estarán

almacenados los archivos HTML puedan ser accedidos y renderizados en el navegador. Esta forma de implementar la aplicación web fue más rápida ya que no involucró una tecnología adicional como un framework de desarrollo *frontend* y todo funcionaba desde el servidor en NodeJs. Cabe destacar que actualmente las aplicaciones están tendiendo a no cargar las vistas desde el servidor, lo que hacen es aislar las vistas del código del servidor empleando *frameworks* como React, Angular o Vue para su creación.

Tabla 18 – Primera Iteración: resumen comparativo entre la duración prevista de las actividades y su duración real

Actividad	Prioridad	Duración Prevista	Duración Real
Recibir una capacitación sobre la misión visión y objetivos de la Empresa Vica Technology y su aplicativo Rinn App	Alta	2	2
Primer acercamiento al sistema Rinn App, arquitectura, componentes y tecnologías	Alta	4	5
Exposición de la API que utiliza Rinn App	Alta	3	2
Identificación de las áreas específicas de trabajo dentro de la API	Alta	4	5
Investigar, conocer y aprender a utilizar la tecnología PM2	Alta	8	5
Correr bajo el servicio de PM2 un aplicativo de contabilidad desarrollado en NodeJs	media	2	1
Adicionar cambios al aplicativo de contabilidad para que se presente nueva información en los informes que genera	Alta	8	7
Investigar, conocer, comprender y aprender a utilizar la tecnología REDIS	Media	24	25
Investigar, conocer, comprender y aprender a utilizar la tecnología RABBIT MQ	Media	24	21
Investigar, conocer, comprender y aprender a utilizar la tecnología NGINX	Media	16	16
Investigar, conocer, comprender y aprender a utilizar la tecnología HAPROXY	Media	16	17
Investigar, conocer, comprender y aprender a utilizar la tecnología ELASTIC SEARCH	Media	16	22
Investigar, conocer, comprender y aprender a utilizar la tecnología VERNE MQ	Media	16	18
Investigar, conocer, comprender y aprender a utilizar la tecnología Mokoon	Baja	10	10
Utilizar mokoon para generar *endpoints* que simulen respuestas de una pasarela especifica	Baja	14	10
Desarrollo de una API Rest sencilla en el framework HapiJs para NodeJs para conocer sus fundamentos	Alta	24	32
		191	**198**
		Más de lo	104%

previsto	

Conocer el funcionamiento del sistema
Comprender la relación entre sus componentes
Aplicación práctica
Caracterizar el código legado
Otros

Finalizado el periodo de la primera iteración se hace un análisis del desarrollo de esta, se realizaron 16 actividades y su duración total (real), es decir, el tiempo (en horas) para completarlas fue de 198 horas. La estimación realizada, fue de 191 horas para finalizar con todas las actividades, por lo tanto, se realizó una buena estimación, aunque haya un desfase de 7 horas adicionales, lo que equivale a un 4% más de lo estimado. Además, el 12% de las actividades fueron de prioridad baja, 44% de prioridad media y 44% de prioridad alta.

4.2.2 SEGUNDA ITERACIÓN

En la segunda iteración las actividades realizadas se enfocaron en aspectos como la instalación y la configuración de las tecnologías, el despliegue de las aplicaciones y la comunicación entre los servidores. Todo esto para cumplir con el objetivo de implementar el ambiente de pruebas para Rinn App. En este se desplegaron las nuevas funcionalidades desarrolladas y se probaron para poder aplicar esos cambios en el ambiente de producción. Finalizado lo anterior se entra a diseñar una solución para que el API soporte el nuevo sistema de pasarela de pagos Wompi.

A continuación, se muestra una especificación de las actividades que comprenden la primera iteración:

Para implementar el ambiente de pruebas es necesario conocer el sistema que actualmente está corriendo en producción, identificar las arquitecturas de software que está empleando y sus componentes para poder replicarlo y establecer dicho ambiente. La tabla 19 especifica las actividades que permitieron conocer el sistema:

Tabla 19 - Segunda iteración: actividades referentes a la definición del ambiente de pruebas

FINALIDAD	ACTIVIDADES
Definición del ambiente de pruebas	Caracterizar el sistema de Rinn App
	Identificas los componentes fundamentales para el funcionamiento del sistema
	Establecer las características del ambiente de pruebas

La caracterización fue descrita a detalle en el capítulo 3 y las otras actividades en dentro de la sección “5.1 IMPLEMENTACIÓN DEL AMBIENTE DE PRUEBAS”.

Tabla 19 - Segunda iteración: actividades referentes a la configuración del servidor de gestión de Información

FINALIDAD	ACTIVIDADES
Configurar el Servidor de gestión de Información	Instalación y configuración de la base de datos en MongoDB
	Instalación y configuración de Elasticsearch
	Instalación y configuración de Kibana
	Instalación y configuración de Redis

La primera aplicación que se despliega en el servidor Balanceador de carga, Proxy inverso y aplicaciones es el API. Esta requiere que se configure el su entorno con las tecnologías que utiliza y cambiar algunos parámetros de su configuración para que consuma los servicios del ambiente de pruebas. Ver Tabla 20.

Tabla 20 - Segunda iteración: actividades referentes al despliegue del API

FINALIDAD	ACTIVIDADES
Despliegue del API	Instalación y configuración de NodeJs
	Restaurar las variables de entorno
	Modificar las variables de entorno para que utilicen los servicios del ambiente de pruebas
	Clonar el código del repositorio e instalas los paquetes y dependencias necesarios

Las tecnologías necesarias para establecer la comunicación entre los servidores, el flujo del tráfico y las solicitudes se establecen con las actividades de la Tabla 21.

Tabla 21 - Segunda iteración: actividades referentes a la configuración del servidor del balanceador de carga, proxy inverso y aplicaciones

FINALIDAD	ACTIVIDADES
Configurar el servidor del balanceador de carga, Proxy inverso y aplicaciones	Instalación y configuración de HAProxy
	Instalación y configuración de Rabbit MQ
	Instalación y configuración de Nginx
	Instalación y configuración de Vernemq
	Configuración del certificado SSL

Despliegue de las aplicaciones que complementan todo el ecosistema de Rinn App. No se toman en cuenta las aplicaciones que corren en los dispositivos móviles debido a que son configuradas por cada desarrollador móvil respectivamente. Ver Tabla 21.

Tabla 22 - Segunda iteración: actividades referentes al despliegue de las aplicaciones

FINALIDAD	ACTIVIDADES
Despliegue de las aplicaciones secundarias	Desplegar el código del panel Administrativo (Superadmin)
	Desplegar el código del despachador de pedidos
	Desplegar el código de notificaciones push
	Desplegar el código de ofertas
	Desplegar el código de búsqueda
	Desplegar el código de gráficos

Tabla 23 - Segunda iteración: actividades referentes al diseño de una solución para soportar el nuevo sistema de pagos Wompi

FINALIDAD	ACTIVIDADES
Diseño una solución para soportar el nuevo sistema de pagos Wompi	Analizar la solución del arquitecto de software
	Verificar la viabilidad de la implementación respecto a su aplicación en el código
	Proponer posibles mejoras a la solución inicial
	Crear el modelo ER de la solución
	Crear un diagrama de secuencia

Tabla 24 – Segunda Iteración: resumen comparativo entre la duración prevista de las actividades y su duración real

Finalidad	Prioridad	Duración prevista	Duración Real	Completado %
Definición del ambiente de pruebas	Alta	20	25	100
Configurar el Servidor de gestión de Información	Alta	20	38	100
Despliegue del API	Alta	26	35	100
Configurar el servidor del balanceador de carga	Alta	61	66	100
Despliegue de las aplicaciones secundarias	media	24	31	100
Diseño de la solución para soportar el nuevo sistema de pagos Wompi	Alta	40	-	0
		191	**195**	**85.7%**

Implementar un ambiente de pruebas para Rinn App
Diseñar el sistema de pagos
Otros

Al terminar el tiempo destinado a completar la segunda iteración se hace un análisis del desarrollo de esta, se realizaron 22 actividades de 27 y su duración total (real), es decir, el tiempo (en horas) para completarlas fue de 195 horas. La estimación para las 22 actividades fue de 151 horas por tanto respecto a la duración real se invirtió aproximadamente 29% más del tiempo previsto. La estimación realizada, fue de 191 horas para finalizar con todas las actividades, pero no alcanzaron a completar todas las actividades, por lo tanto, se no se realizó una buena estimación, y las actividades que no se completaron se transfieren a la siguiente iteración. Además, el 17% de las actividades fueron de prioridad media y el 83% de prioridad alta.

4.2.3 TERCERA ITERACIÓN

La tercera iteración gira entorno a la implementación de la nueva pasarela de pagos. Para ello se hace un estudio de dicha pasarela, se documenta y se resaltan aspectos importantes. Luego se analiza y valida la solución propuesta y posteriormente se construye el API. A raíz de que en la segunda iteración se completó el tiempo y el objetivo relacionado al diseño de la pasarela no se realizó, se decide que sea transferido a la actual iteración y se establece de forma prioritaria para su realización.

Para iniciar el proceso de implementación de la pasarela de pagos Wompi es necesario leer su documentación y así mismo extraer de ella la información que se va a utilizar. La documentación tiende a ser amplia y el acceso recurrente puede ser demorado entonces Al hacer la extracción de la información relevante se tiene acceso más rápido a la información crucial o relevante. A continuación, en la Tabla 25 se especifican las actividades:

Tabla 25 – Tercera Iteración: actividades referentes a la documentación de la pasarela de pagos Wompi

FINALIDAD	ACTIVIDADES
Documentación de la pasarela de pagos Wompi	Registrar las rutas al API de Wompi que se van a utilizar en la implementación
	Identificar el tipo de autenticación que utilizan las rutas al API de Wompi
	Registrar los códigos de respuesta de cada ruta al API
	Registrar los datos de entrada obligatorios y opcionales para las rutas que los requieran

La solución y el camino por seguir se establece por medio de una propuesta inicial del arquitecto que pasa a ser expuesta y está sujeta a modificaciones por algunos motivos que los desarrolladores pueden encontrar antes de llevar a cabo la implementación. En caso de no existir se inicia la implementación, de lo contrario se realizan los cambios y cuando se llegue a una mejor solución por acuerdo del equipo ya se inicia la implementación. En la tabla 26 se describen las actividades:

Tabla 26 - Tercera iteración: actividades referentes al diseño de una solución para soportar el nuevo sistema de pagos Wompi

FINALIDAD	ACTIVIDADES
Diseño una solución para soportar el nuevo sistema de pagos Wompi	Analizar la solución del arquitecto de software
	Verificar la viabilidad de la implementación respecto a su aplicación en el código
	Proponer posibles mejoras a la solución inicial
	Crear el modelo ER de la solución
	Crear un diagrama de secuencia

A partir de la documentación realizada en las actividades de la Tabla 25 se deben probar las rutas del API para verificar el comportamiento y corroborar la información extraída de la documentación. Aunque esto puede parecer un proceso redundante es recomendable hacerlo para que el API de Rinn app sea lo más robusta posible y esté preparada para cualquier respuesta. Ver Tabla 27.

Tabla 27 - Tercera iteración: actividades referentes a la configuración y prueba de los servicios que provee el API de Wompi

FINALIDAD	ACTIVIDADES
Configuración y prueba de los servicios que provee el API de Wompi	Añadir y configurar las rutas al API de Wompi en Postman
	Registrar los cuerpos de respuesta para cada código HTTP
	Comprobar el comportamiento de las rutas al API de Wompi modificando los parámetros de configuración

Para crear una transacción con tarjeta de crédito se debe crear una fuente de pago asociada a los datos de esa tarjeta y es con esa fuente de pago que se realizan las operaciones. La tabla 28 especifica las actividades relacionadas:

Tabla 28 - Tercera iteración: actividades referentes a la implementación del pago con tarjetas en Rinn App

FINALIDAD	ACTIVIDADES

Implementar el pago con tarjetas en Rinn App	Implementar la funcionalidad para la toquenización de una tarjeta
	Definir los códigos de respuesta exitosa y con errores para la toquenización
	Implementar la funcionalidad que crea una fuente de pago con tarjeta
	Definir los códigos de respuesta exitosa y con errores en la creación de una fuente de pago
	Implementar la funcionalidad que crea una transacción con una fuente de pago con tarjeta
	Definir los códigos de respuesta exitosa y con errores en la creación de una transacción con fuente de pago
	Implementar la funcionalidad para obtener el detalle de una transacción creada
	Definir los códigos de respuesta exitosa y con errores al obtener los detalles de una transacción
	Prueba de las funcionalidades para el pago con tarjetas

Uno de los medios de pago más utilizados en Colombia y que ya existía en Rinn app funcionando en la pasarela de ePayco, necesita que se le provea una lista de bancos para continuar el proceso de pago. Funciona mediante la redirección a su portal web en donde el cliente completa dicha operación. Ver Tabla 29.

Tabla 29 - Tercera iteración: actividades referentes a la implementación del pago con PSE en Rinn App

FINALIDAD	ACTIVIDADES
Implementar el pago con PSE en Rinn App	Implementar la funcionalidad que obtiene la lista de bancos disponibles para procesar en PSE
	Definir los códigos de respuesta exitosa y con errores para la funcionalidad de listar bancos
	Implementar la funcionalidad que crea una transacción con PSE
	Definir los códigos de respuesta exitosa y con errores en la creación de una transacción con PSE
	Prueba de las funcionalidades para el pago con PSE

La capa de servicios para el API de Wompi requiere de unas funcionalidades que permitan crear los pagos, validarlos y entregar una respuesta estandarizada. Para ello se debe hacer uso de diferentes servicios creados anteriormente como la implementación de los pagos de PSE y tarjetas, la funcionalidad para obtener los detalles de una transacción y una nueva funcionalidad para dar manejo a todos los códigos de respuesta y estructurar la misma. Ver Tabla 30 para conocer el detalle de las actividades:

Tabla 30 - Tercera iteración: actividades referentes a la implementación de la capa de servicios para el API de Wompi en Rinn App

FINALIDAD	ACTIVIDADES
Implementación de la capa de servicios para el API de Wompi en Rinn App	Implementar el servicio que valida una transacción
	Implementar el servicio que permite la toquenización y creación de una fuente de pago
	Implementar el servicio que permite crear una transacción con tarjeta
	Implementar el servicio que permite crear una transacción con PSE

El botón Bancolombia es un método de pago relativamente nuevo que permite realizar la transferencia de dinero entre cuentas Bancolombia de forma rápida y segura. Tiene una funcionalidad similar a PSE en la que el proceso de pago se realiza a través de su portal web. Implementar este método requiere de las siguientes actividades:

Tabla 31 - Tercera iteración: actividades referentes a la implementación del pago con Botón Bancolombia en Rinn App

FINALIDAD	ACTIVIDADES
Implementar el pago con Botón Bancolombia en Rinn App	Implementar la funcionalidad que crea una transacción con Botón Bancolombia
	Definir los códigos de respuesta exitosa y con errores en la creación de una transacción con Botón Bancolombia
	Prueba de las funcionalidades para el pago con Botón Bancolombia

Un medio de pago muy popularizado en Colombia es Nequi y esto lo hace muy importante para ser agregado en Rinn App para aumentar el de clientes que puedan hacer uso de los servicios que ofrece. Requiere la creación de una fuente de pago para procesar las compras. Ver Tabla 32.

Tabla 32 - Tercera iteración: actividades referentes a la implementación del pago con Nequi en Rinn App

FINALIDAD	ACTIVIDADES
Implementar el pago con Nequi en Rinn App	Implementar la funcionalidad para la toquenización de una cuenta Nequi
	Definir los códigos de respuesta exitosa y con errores para la toquenización
	Implementar la funcionalidad que crea una fuente de pago con Nequi
	Definir los códigos de respuesta exitosa y con errores en la creación de una fuente de pago
	Implementar la funcionalidad que crea una transacción con una fuente de pago con Nequi
	Definir los códigos de respuesta exitosa y con errores en la creación de una transacción con fuente de pago

	Prueba de las funcionalidades para el pago con tarjetas

Rinn App ya implementó este método de pago usando una pasarela distinta, por tal razón es importante conocer la manera en que se dio la funcionalidad para reestructurar, verificar las buenas prácticas y evitar algunos errores en la nueva implementación. Ver Tabla 33.

Tabla 33 - Tercera iteración: actividades referentes al análisis de la lógica actual para realizar recargas por medio de PSE

FINALIDAD	ACTIVIDADES
Análisis de la lógica actual para realizar recargas por medio de PSE	Análisis y documentación de la funcionalidad actual para procesar las recargas con PSE
	Resaltar posibilidades de mejora para la nueva implementación

Tabla 34 - Tercera iteración: resumen comparativo entre la duración prevista de las actividades y su duración real

Actividad	Prioridad	Duración prevista	Duración Real	Completado %
Revisión de la documentación de la pasarela de pagos Wompi	Media	8	10	100
Documentación de la pasarela de pagos Wompi	Alta	16	16	100
Diseño de la arquitectura para soportar el nuevo sistema de pagos Wompi	Alta	40	32	100
Configuración y prueba de los servicios que provee el API de Wompi	Media	20	18	100
Implementar el pago con tarjetas en Rinn App	Alta	30	35	100
Implementar el pago con PSE en Rinn App	Alta	30	28	100
Implementación de la capa de servicios para el API de Wompi en Rinn App	Alta	24	20	50

Corrección de la paginación en endpoint que lista las transacciones con billetera	Baja	4	8	100
Análisis de la lógica actual para realizar recargas por medio de PSE	Media	16	24	100
		188	**191**	**75**

Diseñar el sistema de pagos que utilizara la nueva pasarela de pagos Wompi
Construir la API que soportará la nueva pasarela de pagos Wompi
Otros

Al terminar el tiempo de la tercera iteración se hace un análisis del desarrollo de esta, se realizaron 42 actividades y su duración total (real), es decir, el tiempo (en horas) para completarlas fue de 191 horas. Una de las actividades no se completó estimando que su avance está en aproximadamente el 50%, entonces se transfiere a la siguiente iteración (iteración 4) la finalización del otro 50% de la actividad. La estimación para las 42 actividades fue de 188 horas por tanto respecto a la duración real se invirtió aproximadamente 1.6% más del tiempo previsto. Debido a que aún hay una actividad por terminar se realiza una segunda estimación respecto a este sprint estimando cuánto costará terminar la actividad. Entonces sumando el tiempo que tomará realizar la actividad completa al tiempo real se tiene un valor de 211 horas (20 horas más para terminar la actividad). Por lo tanto, se calcula que aproximadamente se invirtió un 12.23% más del tiempo estimado lo que indica que se realizó una estimación regular. Además, el 11% de las actividades fueron de prioridad baja, el 33% de las actividades fueron de prioridad media y el 56% de prioridad alta.

4.2.4 CUARTA ITERACIÓN

La cuarta iteración tiene como objetivo principal implementar el pago con Botón Bancolombia y Nequi, diseñar el API para proveer el soporte de la nueva billetera de Rinn App, habilitar la recarga de la billetera con los métodos de pago

implementados con Wompi y la finalización de la actividad incompleta transferida de la iteración anterior.

La nueva billetera de Rinn App tiene la capacidad de recargarse con los medios de pago implementados en la anterior y la actual iteración: tarjetas, PSE, Nequi y botón Bancolombia. De esta forma la billetera está habilitada para que las personas puedan acceder a los productos que ofrece como la compra de SOAT, el pago de servicios públicos y las recargas de saldo de celular. Ver Tabla 35 para conocer las actividades desarrolladas.

Tabla 35 - Cuarta iteración: actividades referentes al diseño de un API para proveer el soporte de la nueva billetera de Rinn App

FINALIDAD	ACTIVIDADES
Diseño de un API para proveer el soporte de la nueva billetera de Rinn App	Analizar la solución del arquitecto de software
	Verificar la viabilidad de la implementación respecto a su aplicación en el código
	Crear el modelo ER de la solución
	Crear un diagrama de secuencia

Las actividades de las tablas 36, 37, 38 y 39 se relacionan a las actividades realizadas para habilitar la recarga de la nueva billetera de Rinn App.

Tabla 36 - Cuarta iteración: actividades referentes a la implementación de la recarga de la billetera con una fuente de pago de tarjeta

FINALIDAD	ACTIVIDADES
Implementar la recarga de la billetera con una fuente de pago de tarjeta	Implementar una funcionalidad que permita crear un pago para recarga de billetera usando una fuente de pago con tarjeta
	Definir los códigos de respuesta exitosa y con errores en la recarga de billetera
	Prueba de las funcionalidades para la recarga de billetera con una fuente de pago de tarjeta

Tabla 37 - Cuarta iteración: actividades referentes a la implementación de la recarga de la billetera con una fuente de pago Nequi

FINALIDAD	ACTIVIDADES

Implementar la recarga de la billetera con una fuente de pago Nequi	Implementar una funcionalidad que permita crear un pago para recarga de billetera usando una fuente de pago Nequi
	Definir los códigos de respuesta exitosa y con errores en la recarga de billetera
	Prueba de las funcionalidades para la recarga de billetera con una fuente de pago Nequi

Tabla 38 - Cuarta iteración: actividades referentes a la implementación de la recarga de la billetera con PSE

FINALIDAD	ACTIVIDADES
Implementar la recarga de la billetera con PSE	Implementar una funcionalidad que permita crear un pago para recarga de billetera con PSE
	Definir los códigos de respuesta exitosa y con errores en la recarga de billetera
	Prueba de las funcionalidades para la recarga de billetera con PSE

Tabla 39 - Cuarta iteración: actividades referentes a la implementación de la recarga de la billetera con Botón Bancolombia

FINALIDAD	ACTIVIDADES
Implementar la recarga de la billetera con Botón Bancolombia	Implementar una funcionalidad que permita crear un pago para recarga de billetera con Botón Bancolombia
	Definir los códigos de respuesta exitosa y con errores en la recarga de billetera
	Prueba de las funcionalidades para la recarga de billetera con Botón Bancolombia

Una funcionalidad adicional que se agregó en la creación de las fuentes de pago con tarjeta fue solicitar al usuario que se indique el valor que se cobró a su tarjeta producto de generar un valor aleatorio y hacer una transacción. Esto con la finalidad de añadir seguridad debido a que el propietario, quien agrega la fuente de pago recibirá un mensaje de texto del banco indicando que se ha realizado una transacción y el valor de ella. Si una persona no es el propietario esta será una dificultad para validar la fuente de pago en el sistema de Rinn App.

Tabla 40 - Cuarta iteración: actividades referentes a la validación en dos pasos de las fuentes de pago con tarjeta

FINALIDAD	ACTIVIDADES

Validación en dos pasos de las fuentes de pago con tarjeta	Implementar la funcionalidad que permite crear un pago con un monto aleatorio
	Definir los códigos de respuesta exitosa y con errores en la creación de un pago de monto aleatorio
	Funcionalidad que valida el pago de una fuente de pago de tarjeta y la habilita para realizar pagos
	Definir los códigos de respuesta exitosa y con errores en la creación de un pago de monto aleatorio
	Prueba de las funcionalidades para la creación de un pago con monto aleatorio y validación de la fuente de pago

Tabla 41 - Cuarta iteración: resumen comparativo entre la duración prevista de las actividades y su duración real

Actividad	Prioridad	Duración prevista	Duración Real	Completado %
Implementación de la capa de servicios para el API de Wompi en Rinn App	Alta	24	20	100
Implementar el pago con Botón Bancolombia en Rinn App	Media	16	17	100
Implementar el pago con Nequi en Rinn App	Alta	18	20	100
Implementar la recarga de la billetera con una fuente de pago con tarjeta	Alta	20	20	100
Implementar la recarga de la billetera con PSE	Alta	20	21	100
Implementar la recarga de la billetera con Botón Bancolombia	Media	20	16	100
Implementar la recarga de la billetera con una fuente de pago Nequi	Media	20	16	100
Diseño de un API para proveer el soporte de la nueva billetera de Rinn App	Alta	32	38	100
Validación en dos pasos de la tarjeta de crédito	Alta	20	26	100

191	**194**	**100**

Construir la API que soportará la nueva pasarela de pagos Wompi
Diseñar y construir el API de pagos que utilizara la nueva pasarela de pagos Diseñar una API para habilitar pagos de SOAT y servicios públicos
Otros

La cuarta iteración terminó satisfactoriamente y se hace un análisis del desarrollo de esta, se realizaron 21 actividades y su duración total (real), es decir, el tiempo (en horas) para completarlas fue de 194 horas. Por lo tanto, se calcula que aproximadamente se invirtió un 1.57% más del tiempo estimado lo que indica que se realizó una buena estimación. Además, el 33% de las actividades fueron de prioridad media y el 67% de prioridad alta.

4.2.5 QUINTA ITERACIÓN

La presente iteración está enfocada en la implementación de los productos que ofrece la nueva billetera de Rinn app. Además, se incluyen actividades de documentación y la adición de una funcionalidad que permite validar transacciones por medio del webhook de Wompi.

SOAT es el primer producto que se añade a la nueva billetera de Rinn App. Para su implementación se realizaron las siguientes actividades, ver Tabla 42:

Tabla 42 - Quinta iteración: actividades referentes a habilitar pagos con SOAT

FINALIDAD	ACTIVIDADES
Habilitar pagos con SOAT	Crear una funcionalidad que permita consultar una cotización de SOAT
	Definir los códigos de respuesta exitosa y con errores en una cotización de SOAT
	Crear una funcionalidad que permita realizar la compra del SOAT
	Definir los códigos de respuesta exitosa y con errores en la compra de SOAT

	Probar las funcionalidades para la cotización y compra de SOAT

El pago de servicios públicos conforma otro de los productos de billetera y les permite a los usuarios realizar este proceso sin necesidad de salir de casa o tener que desplazarse a un lugar específico y hacer filas. Esta funcionalidad es apoyada desde la pandemia del COVID 19 para evitar la propagación del contagio. Ver Tabla 43 para las actividades realizadas.

Tabla 43 - Quinta iteración: actividades referentes a habilitar pagos con Servicios Públicos

FINALIDAD	ACTIVIDADES
Habilitar Pagos con Servicios Públicos	Crear una funcionalidad que permita buscar proveedores de servicios
	Definir los códigos de respuesta exitosa y con errores en la búsqueda de proveedores
	Crear una funcionalidad que permita consultar el costo de una factura de servicios públicos
	Definir los códigos de respuesta exitosa y con errores en la consulta de una factura de servicios públicos
	Crear una funcionalidad que permita realizar el pago de una factura de servicios públicos
	Definir los códigos de respuesta exitosa y con errores en el pago de una factura de servicios públicos
	Probar las funcionalidades de búsqueda de proveedores, cotización y pago de servicios públicos

Recargas de celular se convirtió en el tercer producto que provee la nueva billetera de Rinn App y consiste en que los usuarios pueden comprar saldo a sus celulares o hacer una recarga. Especificando el monto deseado y procesando está transacción el usuario recibirá un mensaje de texto a su celular indicando que la recarga está en proceso y luego su operador se encargará de ofrecer los detalles. Ver Tabla 44 para ver las actividades realizadas.

Tabla 44 - Quinta iteración: actividades referentes a Habilitar Recargas de celular

FINALIDAD	ACTIVIDADES
Habilitar Recargas de celular	Crear una funcionalidad que permita listar los operadores móviles disponibles
	Definir los códigos de respuesta exitosa y con errores al listar los operadores
	Crear una funcionalidad que permita realizar una recarga de saldo de celular
	Definir los códigos de respuesta exitosa y con errores en la recarga de saldo de celular

	Probar las funcionalidades de listar operadores y realizar la recarga de celular

Tabla 45 - Quinta iteración: resumen comparativo entre la duración prevista de las actividades y su duración real

Actividad	Prioridad	Duración prevista	Duración Real	Completado %
Habilitar pagos con SOAT	Alta	45	50	100
Habilitar Pagos con Servicios Públicos	Media	45	41	100
Documentación de los nuevos servicios que provee el API de Rinn App	Media	25	30	100
Habilitar Recargas de celular	Baja	40	42	100
Implementar un webhook para la validación de las transacciones de Wompi	Media	36	42	100
		191	**205**	**100**

Diseñar una API para habilitar pagos de SOAT y servicios públicos
Otros

La quinta iteración terminó satisfactoriamente y se hace un análisis del tiempo de desarrollo de esta. En total 17 actividades fueron realizadas y su duración total (real), es decir, el tiempo (en horas) para completarlas fue de 147 horas. Por lo tanto, se calcula que aproximadamente se invirtió un 7.3% más del tiempo estimado lo que indica que se realizó una buena estimación. Además, el 20% de las actividades fueron de prioridad baja, el 60% de las actividades fueron de prioridad media y el 20% de prioridad alta. En ésta última iteración el tiempo restante para completar la iteración fue asignado

5 IMPLEMENTACIÓN DEL AMBIENTE DE PRUEBAS Y CONSTRUCCIÓN DE SOFTWARE

En esta sección se presenta el proceso llevado a cabo para implementar el ambiente de pruebas y las nuevas funcionalidades desarrolladas en el API de Rinn App. Este capítulo está dividido en dos secciones principales: en la primera sección se muestra el proceso realizado para la implementación del ambiente de pruebas y en la segunda el desarrollo de cada una de las funcionalidades explicando las soluciones propuestas y la implementación de cada una de estas.

5.1 IMPLEMENTACIÓN DEL AMBIENTE DE PRUEBAS

En el capítulo 3 en la caracterización se mostró que Rinn App es un sistema que está compuesto de seis servidores. Esto implica un costo en el servicio considerable y no es posible tener el mismo número de servidores para implementar el ambiente de pruebas por el precio elevado. Para esto se propuso implementar el entorno de pruebas en dos nuevos servidores para abordar toda la funcionalidad.

Son dos servidores los que se van a emplear para abordar toda la funcionalidad y es importante que el servicio sea en la nube para que se pueda acceder por diferentes programadores y con las herramientas que tienen actualmente esa tarea se puede hacer de manera sencilla. Entre las opciones de proveedores de servicios en la nube se encuentran varias opciones como GoogleCloud, Azure y Amazon Web Services (AWS) como tres de las más conocidas para suplir esa necesidad.

Se decidió contratar el servicio en la nube de AWS Instances EC2[4] producto de un estudio de factibilidad realizado por la empresa VICA Technology y determinó que AWS es una opción favorable para alcanzar los objetivos. En el contexto de AWS los servidores se representan como Instancias de las cuales se configuraron dos una de capacidad baja y otra de capacidad media [2].

Las instancias funcionan con una distribución de Linux basado en Debian llamado Ubuntu[5] en su versión 16. Se les asigna una dirección IP pública y se configura la seguridad de red para permitir el tráfico, es decir, que se les va a abrir un puerto específico en el firewall del sistema operativo Ubuntu. Dentro de las instancias se instalarán las tecnologías descritas (base de datos, balanceadores, colas, etc.), se desplegará el código de Rinn App tal como el API, el panel administrativo, el despachador de pedidos y se configurará el sistema completo.

[4] Amazon EC2 - https://aws.amazon.com/es/ec2/ (Disponible Ene. 08, 2021)
[5] Ubuntu 16.04 LTS - https://ubuntu.com/16-04 (Disponible Ene. 08, 2021)

5.1.1 CONFIGURACIÓN DEL SERVIDOR PARA LA GESTIÓN DE INFORMACIÓN

Este servidor se configurará para ofrecer los servicios de persistencia de datos, del motor de búsqueda y la base de datos en memoria para el manejo de eventos.

5.1.1.1 Instalación y configuración de MongoDB

Inicialmente se debe verificar la versión del motor de base de datos que está corriendo en producción Mongo DB v4.2. Es recomendable verificar también la versión de Mongo compatible para la versión del sistema operativo en el que se va a instalar, en este caso Ubuntu 16.

Se ejecutan los siguientes comandos:

```
wget -qO - https://www.mongodb.org/static/pgp/server-
4.2.asc | sudo apt-key add -

echo "deb [ arch=amd64,arm64 ]
https://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/4.2
multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-
org-4.2.list
```

Los cuales añaden el repositorio oficial de mongoDB al sistema y descargar isntalador en la versión especificada de MongoDB, luego se procede a su instalación ejecutando:

```
apt install -y mongodb-org
```

Después de completar la instalación se inicia el servicio y se prueba que esté funcionando, utilizando *systemctl* que es la herramienta de administración y agregando los comandos *status, start* y *stop* de Ubuntu.

Para iniciar el servicio se ejecuta:

```
systemctl start mongod
```

Luego, para que el servicio de MongoDB inicie automáticamente cuando el sistema, de igual forma inicie, escribir:

```
systemctl enable mongod
```

Y finalmente se verifica si el servicio de mongo está corriendo en el servidor con el comando:

```
systemctl status mongod
```

En la figura 6 se muestra una captura de pantalla del resultado de ejecutar el comando anterior:

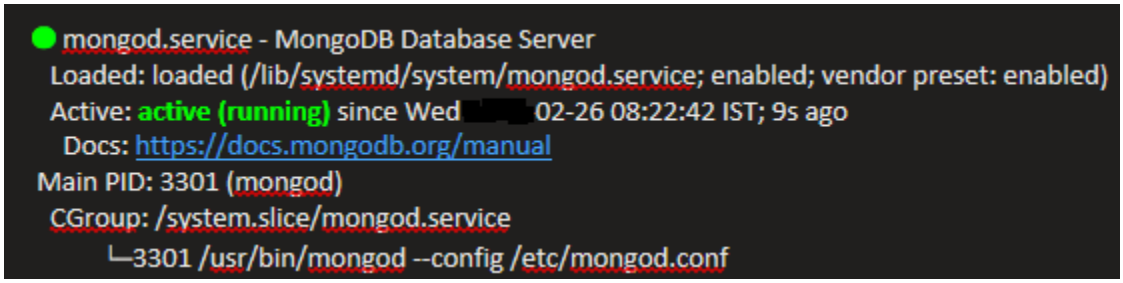


Figura 6. Servicio de MongoDB activo

Es importante tener en cuenta que, por defecto, MongoDB permite que los usuarios puedan leer y modificar datos sin autenticarse. Por lo tanto, se debe habilitar el control de acceso para hacer cumplir la autenticación y permitir que solo los usuarios autenticados realicen acciones en función de sus roles.

Utilizando un editor de texto se modifica el archivo ubicado en '*/etc/mongod.conf*' agregando la siguiente instrucción:

```
security:
authorization: enabled
```

Con la base de datos instalada, corriendo y la autenticación habilitada se crea un nuevo usuario para brindarle privilegios de administrador, se accede a la consola de mongo:

```
mongo
```

Se escoge la base de datos para trabajar sobre ella:

```
use admin
```

Y se ejecuta la siguiente instrucción:

```
db.createUser({
     user: <nombre del usuario>,

     pwd: <contraseña>,

     roles: [

           { role: "userAdminAnyDatabase", db: "admin" },

           { role: "readWriteAnyDatabase", db: "admin" },

           { role: "dbAdminAnyDatabase", db: "admin" }

     ]
```

```
});
```

Se reinicia el servicio de MongoDB para que tome la configuración:

```
systemctl restart mongod
```

Ahora se puede acceder a la base de datos con el siguiente comando:

- *mongo -u <nombre del usuario> -p <contraseña> <dirección IP>:27017/<nombre base de datos>*

Por seguridad se cambia el puerto por defecto que utiliza mongo el 27017 para el acceso a la base de datos recibir peticiones y el nuevo puerto se habilita para que pueda recibir peticiones de cualquier dirección IP, se modifica el archivo ubicado en '*/etc/mongod.conf*' *y* se agrega la siguiente instrucción:

```
net:

 port: <Nuevo puerto>

 bindIp:0.0.0.0
```

Se restaura un respaldo de la base de datos con todas las colecciones y se le dan permisos al usuario creado anteriormente para acceder a ella. La base de datos cuenta con información suficiente para hacer pruebas con esos datos. Se utiliza el siguiente comando:

```
mongorestore --host=<dirección IP> --port=<Nuevo puerto> --
-username=<nombre del usuario> --password=<contraseña> --
authenticationDatabase admin -d <nombre base de datos>
/<Ubicación del respaldo de la BD>
```

Algo adicional, para crear un respaldo de una base de datos existente se realiza ejecutando el comando siguiente:

```
mongodump --host=<dirección IP> --port=<Nuevo puerto> --
username==<nombre del usuario> --password=<contraseña> --
db=<nombre base de datos> --out=/opt/backup/rinnDBJul15
```

Algo adicional e importante es que se restringe el acceso a la base de datos únicamente por el balanceador de carga, y esto se hace configurando el firewall para que por el puerto de la base de datos solo reciba peticiones desde la IP del balanceador de carga y el servidor en que se instaló.

Primero, en la consola de Ubuntu se establece la configuración por defecto:

```
ufw default deny incoming
```

```
ufw default allow outgoing
```

Se habilitan las conexiones SSH:

```
ufw allow ssh
```

Se habilitan las conexiones HTTP y HTTPS:

```
ufw allow 80

ufw allow 443
```

Se configuran las conexiones IP específicas, en este caso la dirección IP del servidor (*3.230.208.73*) en donde estará el API de Rinn App. 5009 es el puerto configurado para MongoDB:

```
ufw allow from 3.230.208.73 to any port 5009
```

Finalmente se habilita UFW (*Uncomplicated Firewall*):

```
ufw enable
```

Y se comprueba la configuración anterior:

```
ufw status
```

```
root@ip-172-31-20-194:/home/ubuntu# ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
80                         ALLOW       Anywhere
443                        ALLOW       Anywhere
5009                       ALLOW       3.230.208.73
22 (v6)                    ALLOW       Anywhere (v6)
80 (v6)                    ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
```

Figura 7. UFW configurado

5.1.1.2 Instalación Elasticsearch

La versión que se emplea es la versión 6.x, esta requiere instalar en el servidor una versión específica del JDK de Java y establecer la variable de entorno para que los programas que lo utilicen puedan determinar en dónde está instalado Java. Para instalar JDK 8 se ejecuta el comando:

```
apt install openjdk-8-jdk
```

Luego modificar el archivo ubicado en '*/etc/enviroment' y* se agrega al final la siguiente instrucción:

```
JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64/jre/bin/"
```

Para verificar que se realizó el proceso correctamente ejecutar el comando:

```
echo $JAVA_HOME
```

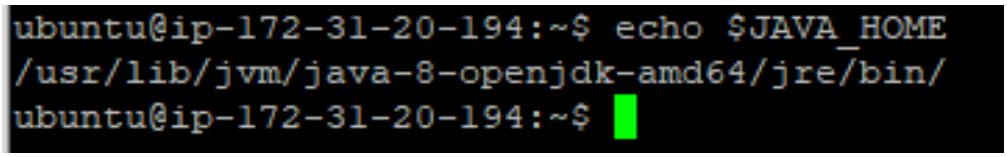


Figura 8. Verificación de la correcta configuración de Java Home

Para la instalación de Elasticsearch ejecutar en consola:

```
$ wget -qO - https://artifacts.elastic.co/GPG-KEY-
        elasticsearch | sudo apt-key add -


echo "deb https://artifacts.elastic.co/packages/6.x/apt
stable main" | sudo tee /etc/apt/sources.list.d/elastic-
                        6.x.list
```

Los cuales añaden el repositorio oficial de Elasticsearch al sistema y descargan el instalador en la versión especificada, luego se procede a su instalación ejecutando:

```
apt-get update && sudo apt-get install elasticsearch
```

y se debe configurar sus opciones para definir la asignación máxima y mínima de memoria la cual se hace tomando de referencia la que posee nuestro servidor o instancia. Por ejemplo: si la instancia tiene 2gb de memoria a Elasticsearch se le asignan 2gb de memoria también para su trbajao.

Para esto se modificar el archivo ubicado en '*/etc/elasticsearch/jvm.options'*:

```
-Xms2g
-Xmx2g
```

Esto significa que se le va a asignar mínimo y máximo 2gb de memoria RAM a un nodo de Elasticsearch.

Con el comando -*systemctl status elasticsearch.service*- se verifica el servicio de Elasticsearch corriendo en el servidor:

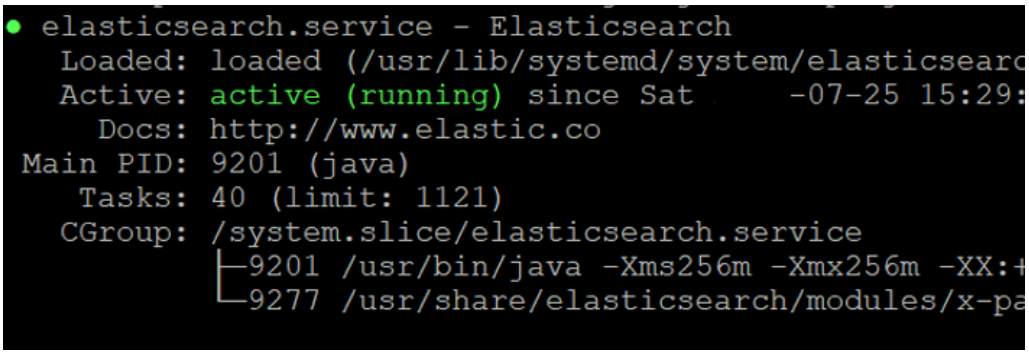


Figura 9. Servicio de Elasticsearch activo

Hay que habilitar su acceso remoto y esto se hace modificando el archivo ubicado en '*/etc/elasticsearch/elasticsearch.yml*' agregando la siguiente línea:

```
network.host: 0.0.0.0
```

Y deberá quedar como se observa en el Figura 10:

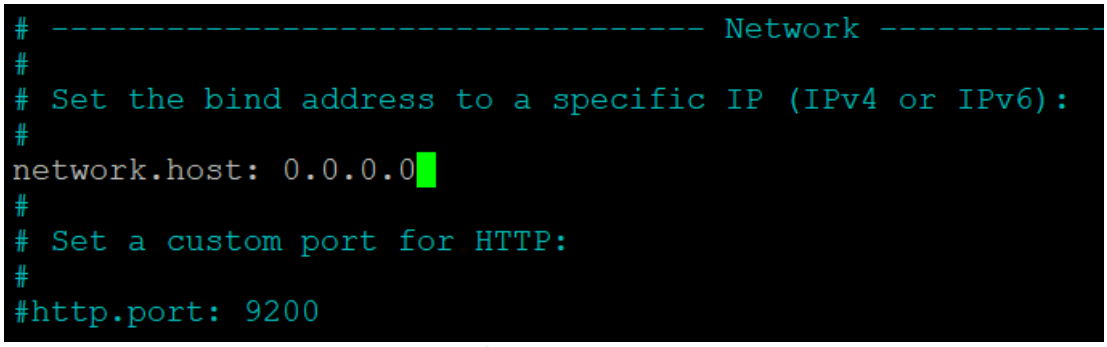


Figura 10. Configuración acceso remoto a Elasticsearch

Finalmente se reinicia el servicio y se restringe a que el acceso a Elasticsearch sea únicamente por el balanceador de carga, y esto se hace configurando el firewall para que por el puerto configurado para Elasticsearch solo reciba peticiones desde la IP del balanceador de carga.

```
ufw allow from 3.230.208.73 to any port 9200
```

5.1.1.3 Instalación Kibana

Kibana es una tecnología adicional que provee una interfaz para visualizar los datos de Elasticsearch (ES) y navegar en el Elastic Stack [2], en otras palabras se pueden gestionar los datos haciendo consultas al API de ES como por ejemplo para insertar, consultar, editar y eliminar.

Para su instalación, se ejecuta:

```
apt install kibana -y
```

Lo siguiente es configurar el acceso al *Dashboard* de Kibana. Primero se modifica el archivo ubicado en '*/etc/kibana/elasticsearch.yml'* agregando la siguiente línea:

```
server.host to 127.0.0.1
```

Por seguridad esto se realiza para restringir el acceso desde el servidor de forma local. Para iniciar el servicio se ejecuta:

```
systemctl start Kibana
```

Luego, para que el servicio de Kibana inicie automáticamente cuando el sistema, de igual forma inicie, escribir:

```
systemctl enable Kibana
```

Con el siguiente comando se verifica que la instalación se hizo correctamente:

```
netstat -plntu
```

Y en la Figura 11 se observa el resultado de la configuración, indica que el acceso al perto por defecto 5601 de kibana se hace de forma local

Figura 11. Verificación de la instalación de Kibana

Luego se configura un *SSH Tunnel* que tiene como función permitir que un *host* pueda acceder a los servicios internos de un servidor estando en una red exterior [3], en este caso el host que se va a utilizar para trabajar y la instancia configurada en AWS donde se encuentra ES.

Utilizando la aplicación PuTTY añadir una nueva conexión SSH, indicando el nombre de usuario, y la dirección IP de la instancia como se observa en la Figura 12:

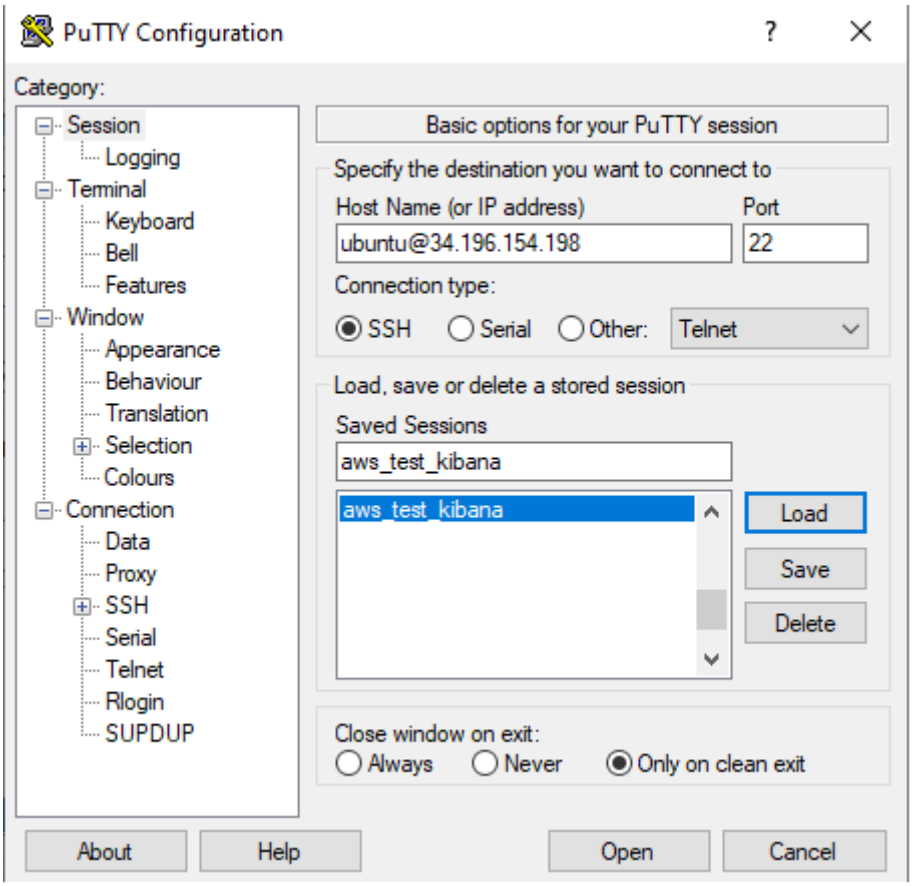


Figura 12. Conexión SSH en Kibana 1

Luego se configura en la sección *Connections – SSH – Tunnels*, como se ve en la Figura 13 y 14

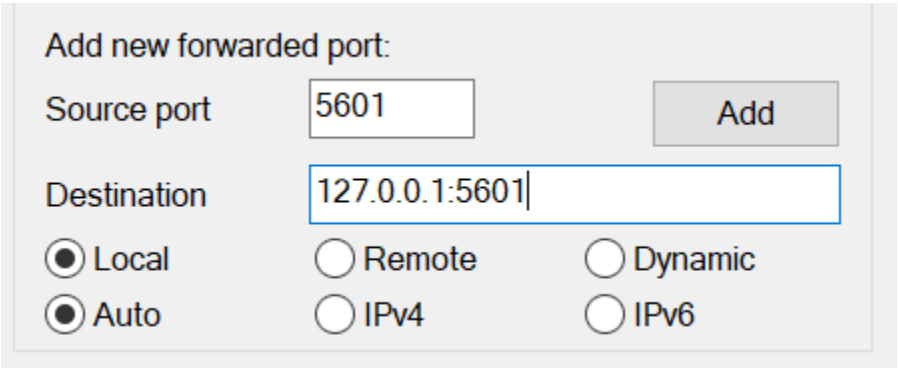


Figura 13. Conexión SSH en Kibana 2

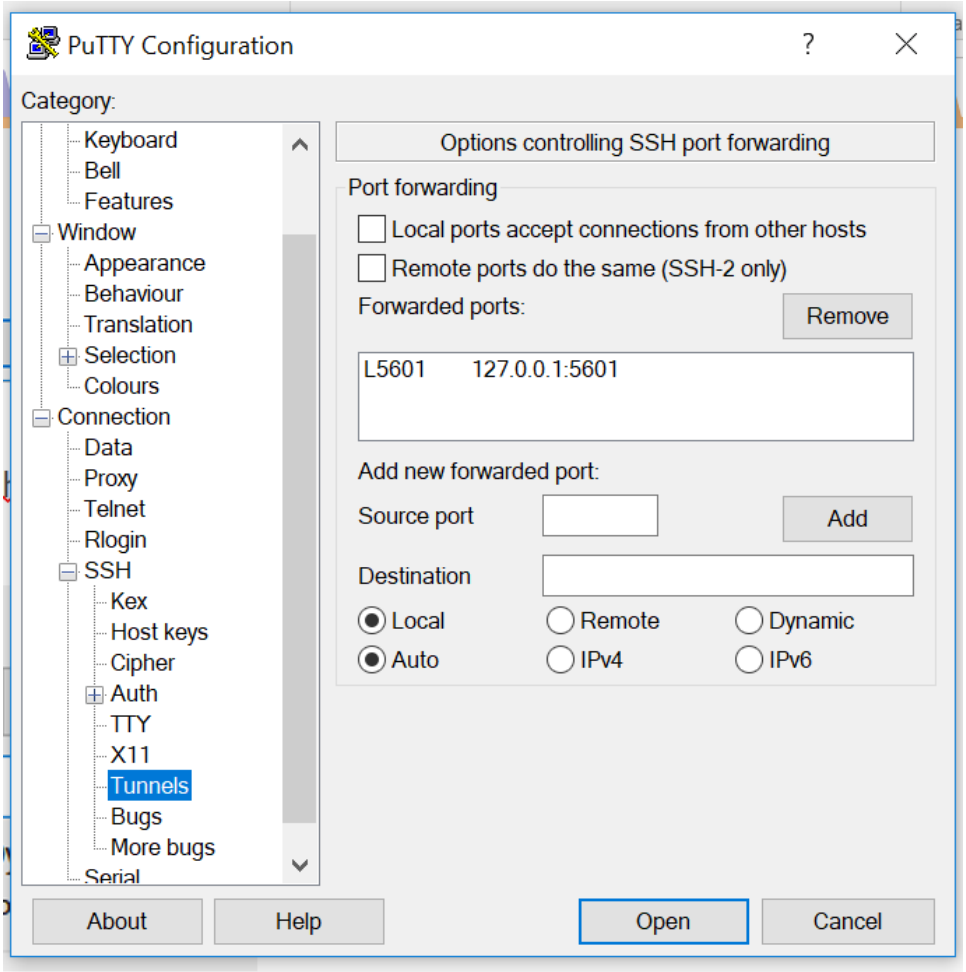


Figura 14. Conexión SSH en Kibana 3

Se guarda la configuración y desde un navegador se accede escribiendo la siguiente dirección *http://localhost:5601/*. El resultado será el Dashboard de Kibana como lo muestra la Figura 15.

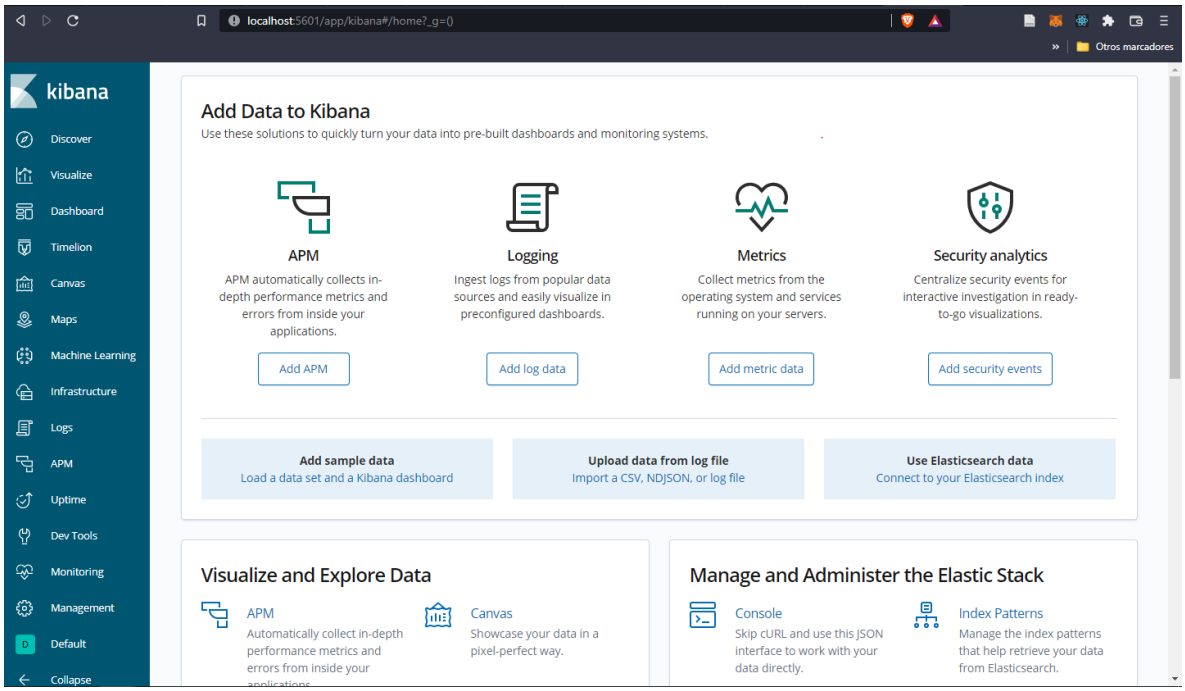


Figura 15. Dashboard de Kibana

Lo siguiente es restaurar una copia de los datos almacenados por ES de algunas colecciones de nuestra base de datos para optimizar la búsqueda.

Se define la ruta donde se va a ubicar la copia de respaldo, modificar el archivo ubicado en '*/etc/kibana/elasticsearch.yml*' agregando la siguiente línea:

```
path.repo: /opt/elastic/backup/
```

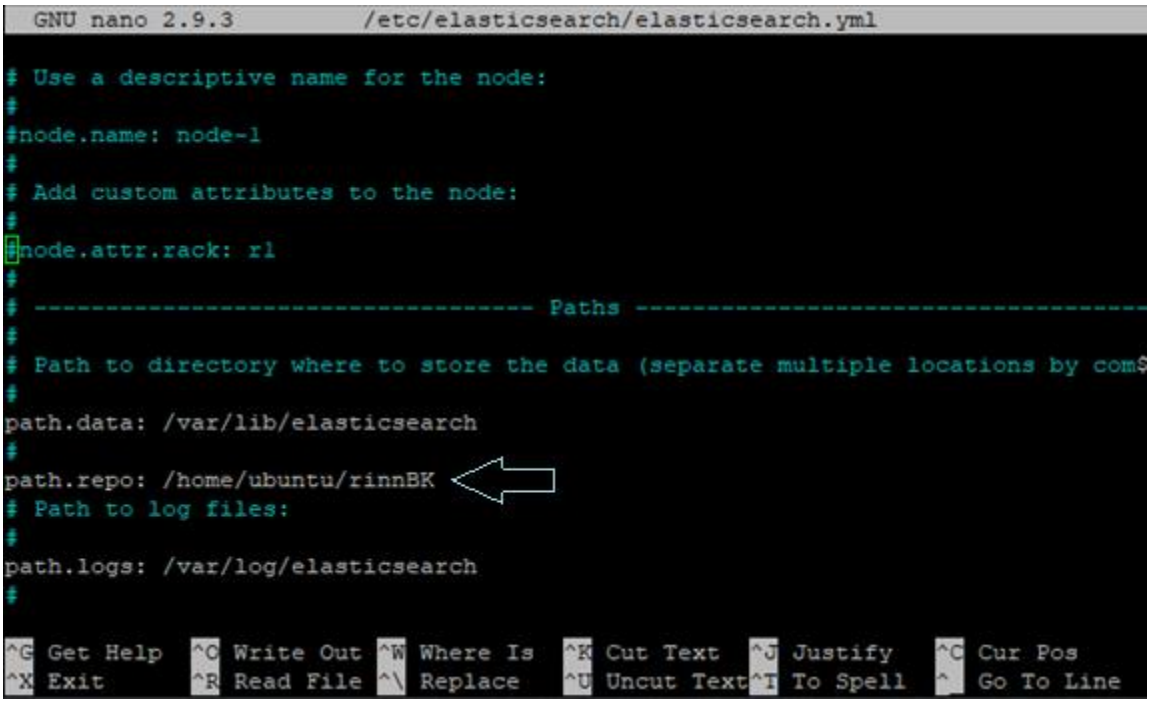


Figura 16. Dashboard de Kibana

Copiar en esa ruta la copia de respaldo y reiniciar el servicio.

```
systemctl restart elasticsearch.service
```

Luego se accede al dashboard y en la sección *DevTools* ejecutar los comandos siguientes:

Se crea un *snapshot* de la copia de respaldo, se define un nombre y la ubicación en que se encuentra el respaldo

```
PUT /_snapshot/<Nombre del snapshot>
{
    "type": "fs",
    "settings": {
        "location": "/home/ubuntu/backup",
        "compress": true
    }
}
```

Lo siguiente es cargar los índices almacenados en el snapshot creado anteriormente:

```
PUT /_snapshot/<Nombre del snapshot>/first-
snapshot?wait_for_completion=true


POST /_snapshot/<Nombre del snapshot>/first-
snapshot/_restore
```

De esta forma ya queda funcionando el respaldo y configurado en su totalidad Elasticsearch.

5.1.1.4 Instalación REDIS

La versión usada de Redis es la 3.0.6 y es importante tener instalado el paquete *Build Essential* de Ubuntu que contiene una lista de paquetes que son necesarios para satisfacer las dependencias de compilación que permiten compilar el software [4], en nuestro caso Redis. Se procede a instalarlo ejecutando el siguiente comando:

```
apt-get install build-essential tcl
```

Se procede a instalarlo y configurar aspectos de seguridad y comportamiento. Se descarga el instalador que viene comprimido en formato .tar:

```
cd /tmp && wget -c
"http://download.redis.io/releases/redis-3.0.6.tar.gz"
```

Descomprimir el archivo descargado con el comando:

```
tar xvzf "redis-3.0.6.tar.gz"
```

Dentro de la carpeta que se generó se ejecuta el siguiente comando para compilar los archivos de Redis:

```
make
```

Seguido, se recomienda ejecutar un comando que permite verificar si todo se ha compilado correctamente:

```
make test
```

Modificar el archivo ubicado en '*/etc/redis/redis.conf*' para especificar el directorio de configuración:

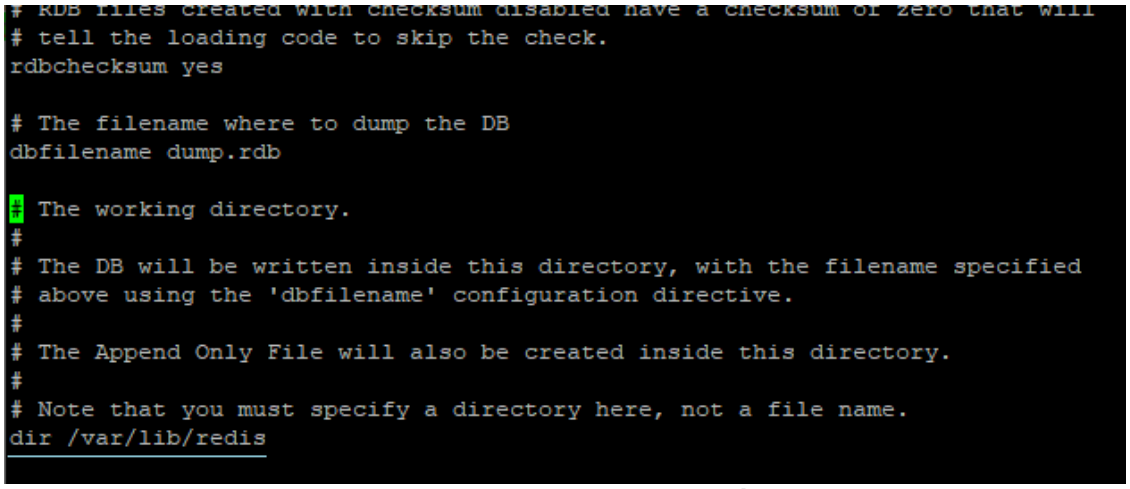


Figura 17. Directorio de configuración de Redis

Luego se crea el *systemd unit file* para administrar los procesos de Redis:

```
nano /etc/systemd/system/redis.service
```

Y se agregan las siguientes instrucciones:

```
[Unit]

Description=Redis In-Memory Data Store

After=network.target

[Service]

User=redis

Group=redis

ExecStart=/usr/local/bin/redis-server
/etc/redis/redis.conf

ExecStop=/usr/local/bin/redis-cli shutdown

Restart=always

[Install]

WantedBy=multi-user.target
```

En resumen, esta configuración indica que la conexión de red debe estar disponible antes de iniciar el servicio, se define un usuario, se indica el archivo de configuración al que debe apuntar para iniciar el servicio y que pueda recuperarse de fallos en caso de algún problema.

Añadir un usuario y un grupo de Redis con el siguiente comando

```
adduser --system --group --no-create-home redis
```

Se crea el directorio /var/lib/redis y se le da al usuario y al grupo Redis la propiedad sobre este directorio para que un usuario normal no pueda acceder a este directorio:

```
chown redis:redis /var/lib/redis
```

Finalmente se habilita su acceso remoto, modificando el archivo ubicado en '*/etc/redis/redis.conf*' y habilitando la línea seleccionada en la Figura 18:

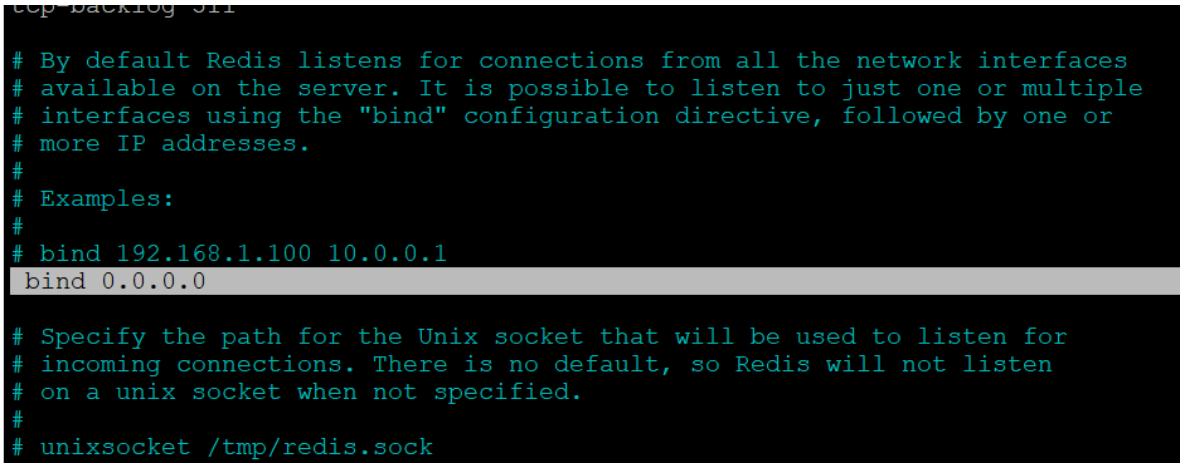


Figura 18. Directorio de configuración de Redis

Para iniciar el servicio se ejecuta:

```
systemctl start redis
```

Luego, para que el servicio de Redis inicie automáticamente cuando el sistema, de igual forma inicie, escribir:

```
systemctl enable redis
```

Se restringe el acceso a Redis únicamente por el balanceador de carga configurando el firewall para que por el puerto asignado a Redis solo reciba peticiones desde la IP del balanceador de carga.

La configuración final del firewall del servidor se verifica en la Figura 19:

```
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
80                         ALLOW       Anywhere
443                        ALLOW       Anywhere
5009                       ALLOW       3.230.208.73
6379                       ALLOW       3.230.208.73
9200                       ALLOW       3.230.208.73
22 (v6)                    ALLOW       Anywhere (v6)
80 (v6)                    ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
9200 (v6)                  ALLOW       Anywhere (v6)
5601 (v6)                  ALLOW       Anywhere (v6)
```

Figura 19. Resumen de configuración del Firewall del servidor de gestión de la Información

5.1.2 CONFIGURACIÓN DEL SERVIDOR PARA EL MANEJO DE LA LÓGICA Y COMUNICACIÓN ENTRE SERVIDORES

El segundo servidor que se configura tiene instalado el balanceador de carga HA Proxy, Rabbit MQ, Nginx, Verne MQ y las aplicación como el Administrativo (Superadmin), el API, el despachador de pedidos, notificaciones push, ofertas, búsqueda y gráficos. Se puede afirmar que este conjunto de aplicaciones conforma el núcleo central del sistema de Rinn App. A continuación, se muestra el proceso para la instalación de cada parte mencionada anteriormente:

5.1.2.1 Instalación del balanceador de carga HA Proxy

Rinn App está utilizando la versión 1.6.3. Se instala esa versión con el siguiente comando en la consola de Ubuntu:

```
apt-get install -y haproxy
```

se inicia el servicio y se comprueba su estado con los siguientes comandos.

```
service haproxy start

service haproxy status
```

```
● haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; vendor
   Active: active (running) since                15:54:17 UTC; 3min 5s
     Docs: man:haproxy(1)
           file:/usr/share/doc/haproxy/configuration.txt.gz
 Main PID: 4253 (haproxy)
    Tasks: 2 (limit: 1140)
   CGroup: /system.slice/haproxy.service
           ├─4253 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p
           └─4255 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p
```

Figura 20. Servicio de HA Proxy activo

En la configuración se habilita visualizar una página de estadísticas del proceso general y se cambia la contraseña del usuario. Modificando el archivo ubicado en *'/etc/haproxy/haproxy.cfg'* y agregando las siguientes líneas, como se observa en la Figura 21:

```
listen stats
        bind  *:1936
        # Enable the statistics page
        stats enable
        mode http
        stats uri /
        stats auth admin:auth
        stats hide-version
        stats realm Haproxy\ Statistics
```

En las líneas anteriores *'bind'* indica el puerto en el que se va a poder acceder al servicio y *stats auth* indica el usuario y la contraseña para acceder a la página de estadísticas.

ubuntu@ip-172-31-92-7: /etc/haproxy

```
  GNU nano 2.9.3

        ssl-default-bind-options no-sslv3

defaults
        log     global
        mode    http
        option  httplog
        option  dontlognull
        timeout connect 5000
        timeout client  50000
        timeout server  50000
        errorfile 400 /etc/haproxy/errors/400.http
        errorfile 403 /etc/haproxy/errors/403.http
        errorfile 408 /etc/haproxy/errors/408.http
        errorfile 500 /etc/haproxy/errors/500.http
        errorfile 502 /etc/haproxy/errors/502.http
        errorfile 503 /etc/haproxy/errors/503.http
        errorfile 504 /etc/haproxy/errors/504.http

listen stats
        bind  *:1936
        # Enable the statistics page
        stats enable
        mode http
        stats uri /
        stats auth admin:admin
        stats hide-version
        stats realm Haproxy\ Statistics
```

Figura 21. Configuración archivo haproxy.cfg

Guardar la configuración y reiniciar el servicio:

```
sudo service haproxy restart
```

Para acceder a la página de estadísticas de HAProxy desde el navegador ingresar la dirección IP de la instancia y el puerto que con la configuración actual serán 3.230.208.73 y 1936 respectivamente. Solicitará el usuario y contraseña especificados y luego de autenticarse se podrá tener acceso a la página de estadísticas.

En la Figura 22 se ve en funcionamiento la página de estadísticas.

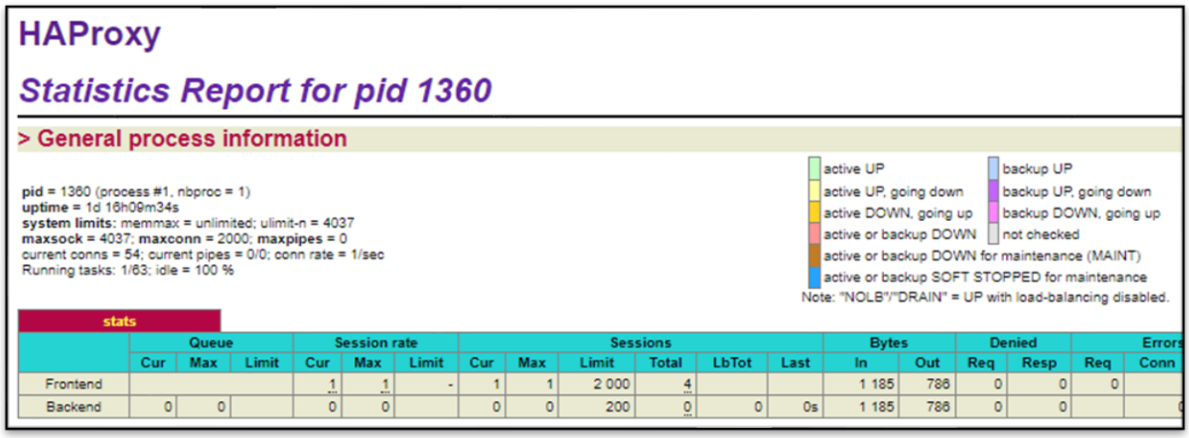


Figura 22. Página de estadísticas de HA Proxy

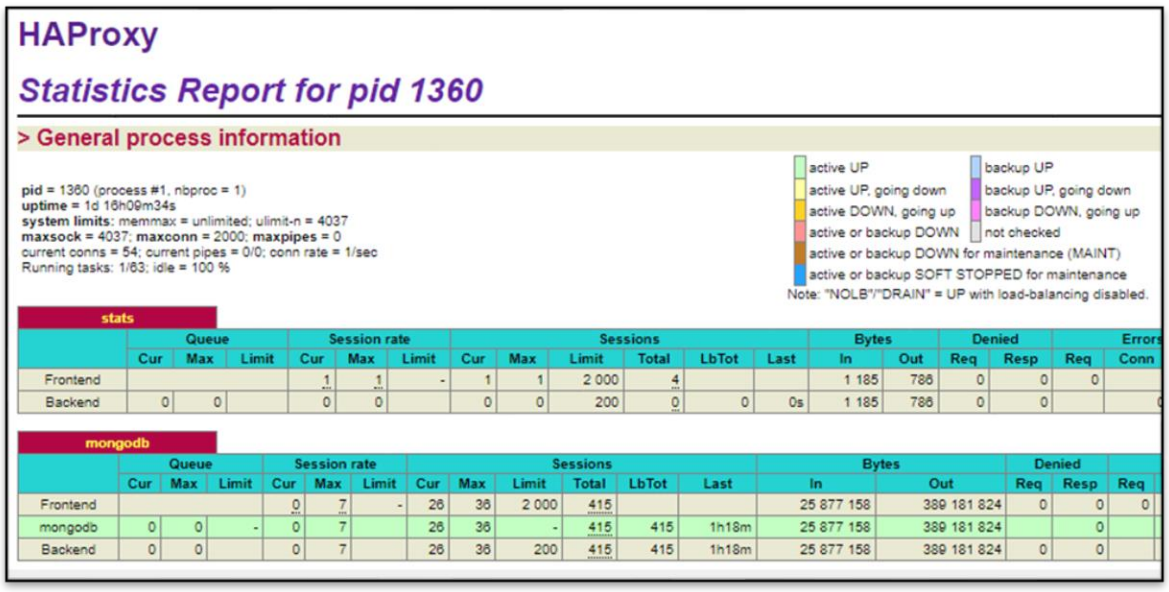
5.1.2.2 Configuración del balanceador de carga HA Proxy

Si el servicio HA Proxy (HAP) ya está funcionando lo primero que se realiza es registrar los servidores a los que va a redirigir el tráfico, básicamente se especifica:

1. Nombre
2. Un puerto
3. El protocolo
4. El algoritmo de balanceo de carga
5. Su dirección IP

Ejemplo de configuración del servicio de mongodb: modificar el archivo haproxy.cfg y agregar:

```
listen mongodb
        bind *:5009
        mode tcp
        option tcplog
        balance roundrobin
        server mongodb 34.134.201.155:5009 check
```



HAProxy

Statistics Report for pid 1360

> General process information

pid = 1360 (process #1, nbproc = 1)
uptime = 1d 16h09m34s
system limits: memmax = unlimited; ulimit-n = 4037
maxsock = 4037; maxconn = 2000; maxpipes = 0
current conns = 54; current pipes = 0/0; conn rate = 1/sec
Running tasks: 1/63; idle = 100 %

active UP | backup UP
active UP, going down | backup UP, going down
active DOWN, going up | backup DOWN, going up
active or backup DOWN | not checked
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

stats

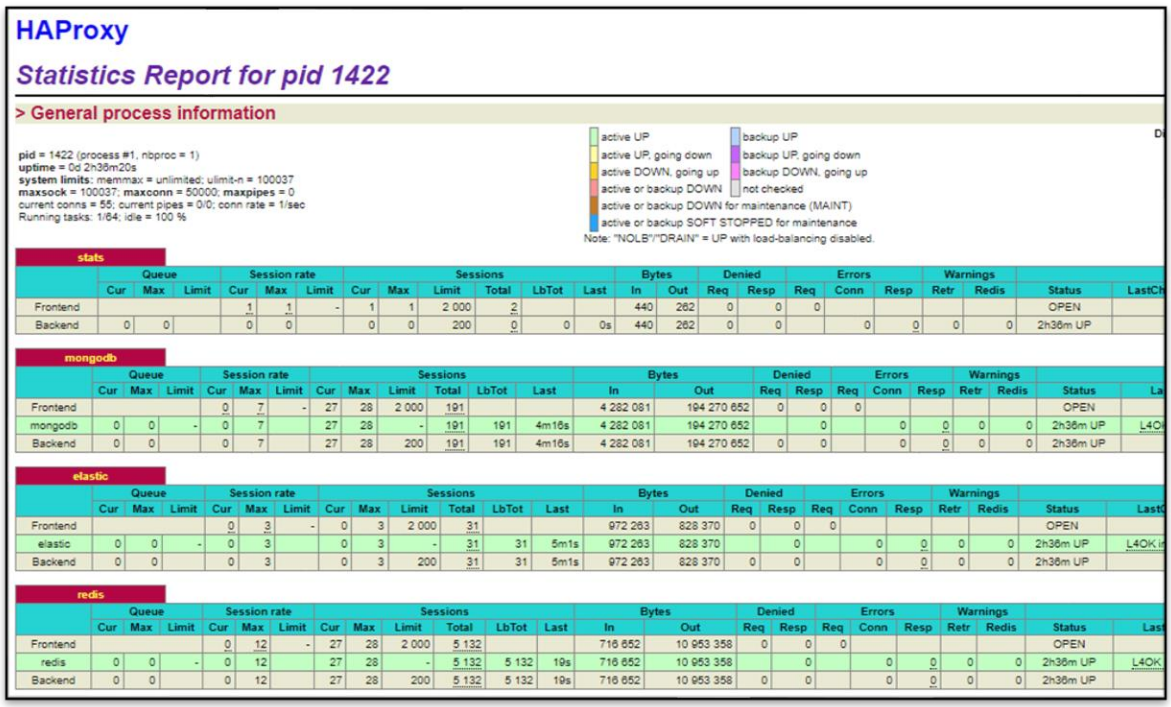
	Queue			Session rate			Sessions						Bytes		Denied		Errors	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn
Frontend				1	1	-	1	1	2 000	4			1 185	786	0	0	0	
Backend	0	0		0	0		0	0	200	0	0	0s	1 185	786	0	0		

mongodb

	Queue			Session rate			Sessions						Bytes		Denied		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req
Frontend				0	7	-	26	36	2 000	415			25 877 158	389 181 824	0	0	0
mongodb	0	0	-	0	7		26	36	-	415	415	1h18m	25 877 158	389 181 824		0	
Backend	0	0		0	7		26	36	200	415	415	1h18m	25 877 158	389 181 824	0	0	

Figura 23. Ejemplo de configuración MongoDB en HAP

Así mismo se agregan los servicios de Elasticsearch y Redis utilizando la lista de información antes mencionada y obtiene como resultado lo que se observa en la Figura 24:



HAProxy

Statistics Report for pid 1422

> General process information

pid = 1422 (process #1, nbproc = 1)
uptime = 0d 2h36m20s
system limits: memmax = unlimited; ulimit-n = 100037
maxsock = 100037; maxconn = 50000; maxpipes = 0
current conns = 55; current pipes = 0/0; conn rate = 1/sec
Running tasks: 1/64; idle = 100 %

active UP | backup UP
active UP, going down | backup UP, going down
active DOWN, going up | backup DOWN, going up
active or backup DOWN | not checked
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors		Warnings			
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status
Frontend				1	1	-	1	1	2 000	2			440	262	0	0	0					OPEN
Backend	0	0		0	0		0	0	200	0	0	0s	440	262	0	0		0	0	0	0	2h36m UP

mongodb

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status
Frontend				0	7	-	27	28	2 000	191			4 282 081	194 270 652	0	0	0					OPEN
mongodb	0	0	-	0	7		27	28	-	191	191	4m16s	4 282 081	194 270 652		0		0	0	0	0	2h36m UP
Backend	0	0		0	7		27	28	200	191	191	4m16s	4 282 081	194 270 652	0	0		0	0	0	0	2h36m UP

elastic

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status
Frontend				0	3	-	0	3	2 000	31			972 263	828 370	0	0	0					OPEN
elastic	0	0	-	0	3		0	3	-	31	31	5m1s	972 263	828 370		0		0	0	0	0	2h36m UP
Backend	0	0		0	3		0	3	200	31	31	5m1s	972 263	828 370	0	0		0	0	0	0	2h36m UP

redis

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status
Frontend				0	12	-	27	28	2 000	5 132			716 652	10 953 358	0	0	0					OPEN
redis	0	0	-	0	12		27	28	-	5 132	5 132	19s	716 652	10 953 358		0		0	0	0	0	2h36m UP
Backend	0	0		0	12		27	28	200	5 132	5 132	19s	716 652	10 953 358	0	0		0	0	0	0	2h36m UP

Figura 24. Configuración de HAP para MongoDB, Elasticsearch y Redis

5.1.2.3 Instalación de Rabbit MQ (RMQ)

RMQ es un gestor de colas desarrollado en lenguaje Erlang, por lo cual es necesario que se instale y configure Erlang antes de instalar RMQ. Como se ha hecho anteriormente se inicia verificando, en este caso, cual es la versión del lenguaje que está en la aplicación de producción.

```
erl -version
```

```
root@ip-172-31-30-192:/home/ubuntu# erl -version
Erlang (SMP,ASYNC_THREADS,HIPE) (BEAM) emulator version 10.4
root@ip-172-31-30-192:/home/ubuntu#
```

Figura 25. Versión Erlang

Y también conocer cuál es la version OTP con el siguiente comando

```
erl -noshell -eval
'erlang:display(erlang:system_info(system_version))' -eval
'init:stop()'
```

Como indica la figura 26 es la versión OTP 22:

```
root@ip-172-31-30-192:/home/ubuntu# erl -noshell -eval 'erlang:display(erlang:sy
stem_info(system_version))' -eval 'init:stop()'
"Erlang/OTP 22 [erts-10.4] [source] [64-bit] [smp:2:2] [ds:2:2:10] [async-thread
s:1] [hipe]\n"
root@ip-172-31-30-192:/home/ubuntu#
```

Figura 26. Versión OTP Erlang

Con las anteriores verificaciones se debe instalar el lenguaje Erlang, entonces, añadir el repositorio de Erlang:

```
wget https://packages.erlang-solutions.com/erlang-
solutions_2.0_all.deb


sudo dpkg -i erlang-solutions_2.0_all.deb
```

Y luego se ejecuta el siguiente comando para realizar la instalación:

```
sudo apt-get -y install esl-erlang=1:22.0.1-1
```

Se procede a instalar RMQ. Se recomienda buscar la versión compatible con la versión del sistema operativo que se esté utilizando y se puede hacer directamente desde el repositorio de paquetes de la página oficial de RMQ.

Añadir el repositorio de RMQ

```
curl -s
https://packagecloud.io/install/repositories/rabbitmq/rabb
itmq-server/script.deb.sh | sudo bash
```

E instalar:

```
sudo apt-get -y install rabbitmq-server=3.7.15-1
```

se inicia el servicio y se comprueba su estado:

```
service rabbitmq-server start

systemctl enable rabbitmq-server

service rabbitmq-server status
```

En la Figura 27 se observa el servicio de RMQ corriendo de forma correcta.

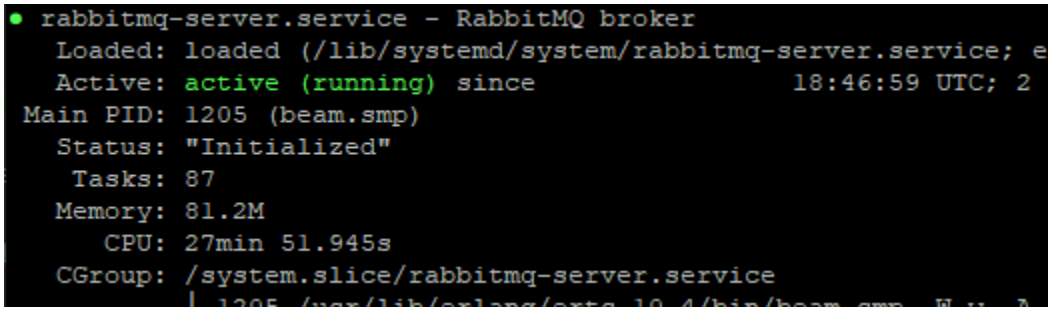


Figura 27. Servicio de Rabbit MQ activo

A continuación, se crea un usuario administrador, en este caso el *username* será '*admin*' y se habilita la consola web para poder restaurar la configuración anterior.

```
sudo rabbitmqctl add_user admin <Contraseña>

sudo rabbitmqctl set_user_tags admin administrator

sudo rabbitmqctl set_permissions -p / admin ".*" ".*" ".*"
```

Se habilita la página de administracioncon el comando siguiente:

```
sudo rabbitmq-plugins enable rabbitmq_management
```

Y se reinicia el servicio de RMQ para que tome la configuración

```
sudo systemctl restart rabbitmq-server.service
```

Para acceder a la página de administración desde un navegador web proporcionar la dirección IP de la instancia en que se instaló y el puerto por defecto de RMQ, el puerto 15672. Solicitará el usuario y contraseña configurados como se observa en la Figura 26.

Figura 28. Acceso a la página de administración de RMQ

Luego de acceder correctamente se puede observar en la Figura 29 en la lista de colas (*Queues*), está una tabla con la información del estado y funcionamiento.

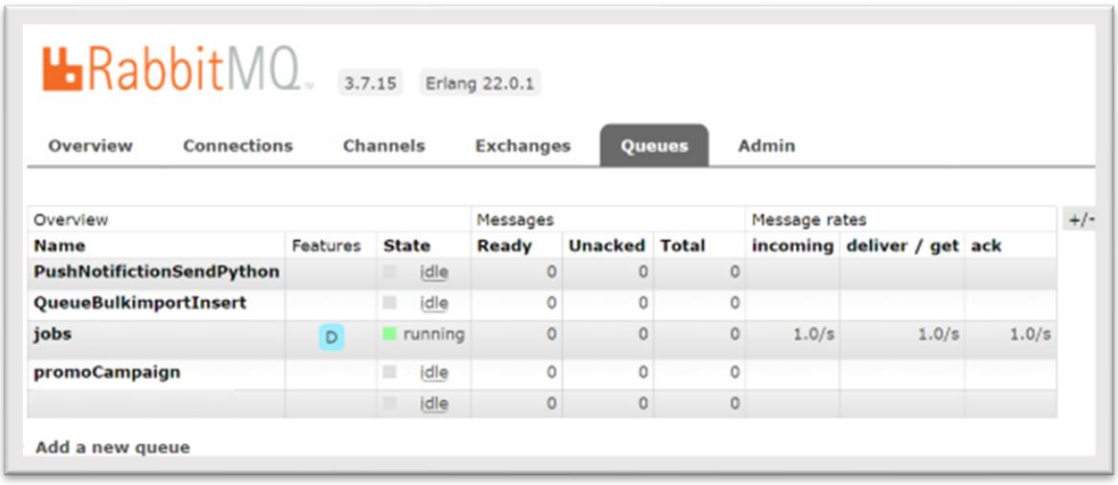


Figura 29. Consola web de Rabbit MQ

Al tener acceso a la página de administración de RMQ se pueden crear copias de las colas que se tienen configuradas para que se puedan cargar de nuevo en caso de reinstalar el servicio o crear una réplica para escalar.

5.1.2.4 Instalación de Nginx

Nginx se configuró para ser utilizado como un proxy reverso para dirigir el tráfico hacia el panel administrativo, el despachador de pedidos, la API y los servicios de notificaciones, ofertas, carga de gráficos y búsqueda. Así es que se instala Nginx y se establece su configuración.

En la Figura 30 se puede ver que se configura la dirección IP, que en este caso es local ya que Nginx y el API se encuentran en el mismo servidor, y el puerto.

```
# Socket balancer
upstream socket_balancer {
least_conn;
 server localhost:7002;
 }


server {
        listen 80;
        listen [::]:80;

listen 443 ssl;
        listen [::]:443;
```

Figura 30. Nginx configuración API 1

Se activa la validación de certificado SSL, se especifica la ruta de los certificados comprados para Rinn App y el nombre del dominio.

```
        ssl on;
        ssl_certificate /usr/ssl/star.rinnapp.co.crt;
        ssl_certificate_key /usr/ssl/STAR_rinnapp_co.pem;
        ssl_trusted_certificate /usr/ssl/star.rinnapp.co.ca-bundle;

 server_name testapi.rinnapp.co;

if ($http_x_forwarded_proto = "http") {
     return 301 https://$server_name$request_uri;
}
```

Figura 31. Nginx configuración API 2

La anterior configuración se realiza para el Panel Administrativo, el Despachador de Pedidos y los servicios de notificaciones, ofertas, carga de gráficos y búsqueda.

5.1.2.5 Desplegando el Panel Administrativo (PA)

El PA es una aplicación web desarrollada en PHP que permite a Rinn App gestionar toda la información del negocio. Para que funcione se deben ubicar las variables de entorno, habilitar que PHP pueda ser ejecutado por Nginx, instalar Mongo para PHP, clonar el código del PA en el servidor y reiniciar el servicio de Nginx y PHP.

En la Figura 29 se observa el panel administrativo desplegado con la dirección https://testsuperadmin.rinnapp.co/index.php?/superadmin/Dashboard

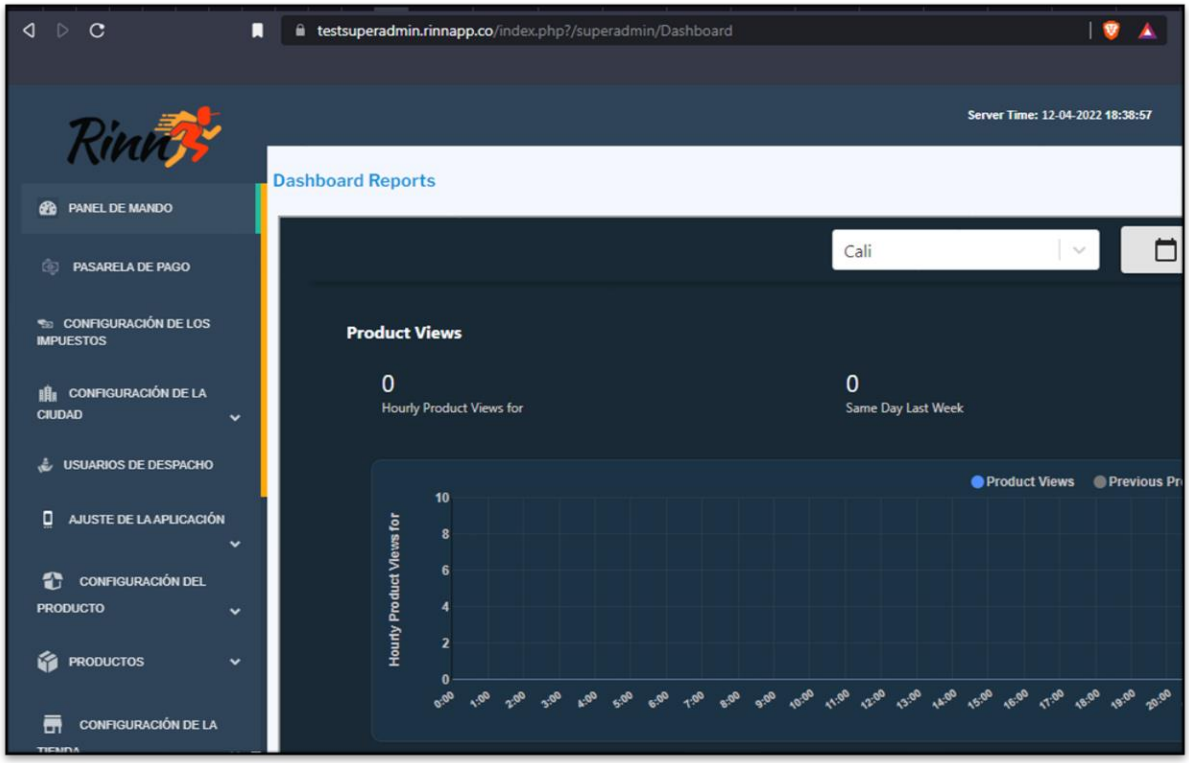


Figura 32. Superadmin desplegado

5.1.2.6 Desplegando el API de Rinn App

El API está desarrollada en NodeJs en la versión 8.17, se procede a su instalación. Se obtiene el paquete de instalación y luego se ejecuta la instalación:

```
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E
bash –
```

```
sudo apt install nodejs
```

Verificar la versión instalada, ejecutar el siguiente comando y la salida deberá verse como en la Figura 29:

```
node –version
```

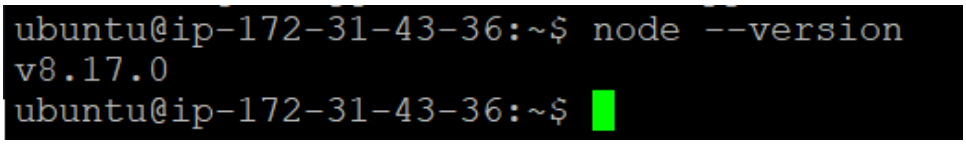


Figura 33. Versión NodeJs

Luego se ubican las variables de entorno que utiliza el API y se modifican las direcciones IP configuradas para el servidor de MongoDB y Elasticsearch los cuales deben establecerse con la dirección del balanceador de carga y para el servicio de MQTT con la misma dirección de la instancia actual (Servidor 2). Finalmente se clona el repositorio dentro del servidor, se instala PM2 en la versión 3.5 y se configura para que se inicie apenas el servidor esté funcionando.

5.1.2.7 Instalación Verne MQ

Se instala la versión 1.9 y se restaura la configuración existente que permite el chat entre los usuarios.

```
wget
https://github.com/vernemq/vernemq/releases/download/1.9.1
/vernemq-1.9.1.bionic.x86_64.deb

sudo dpkg -i vernemq-1.9.1.bionic.x86_64.deb

dpkg -s vernemq | grep Status
```

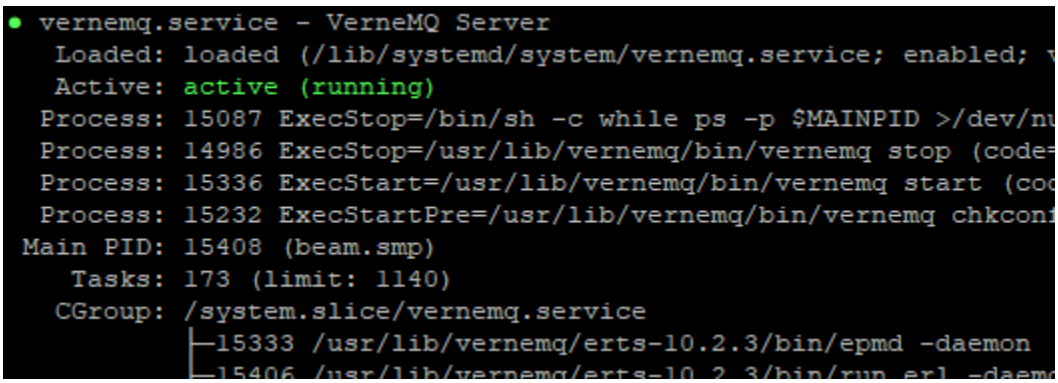


Figura 34. Servicio de Verne MQ activo

Iniciar el servicio y verificar el estado de la instalación con los siguientes comandos:

```
service vernemq start


service vernemq status
```

Luego se agregan los certificados SSL existentes para Rinn App en el directorio *'/etc/ssl/vernemq/'*, se puede utilizar una aplicación como WinSCP para transferir los archivos desde el escritorio a una máquina remota importando las conexiones SSH de PuTTY como se observa en la 32:

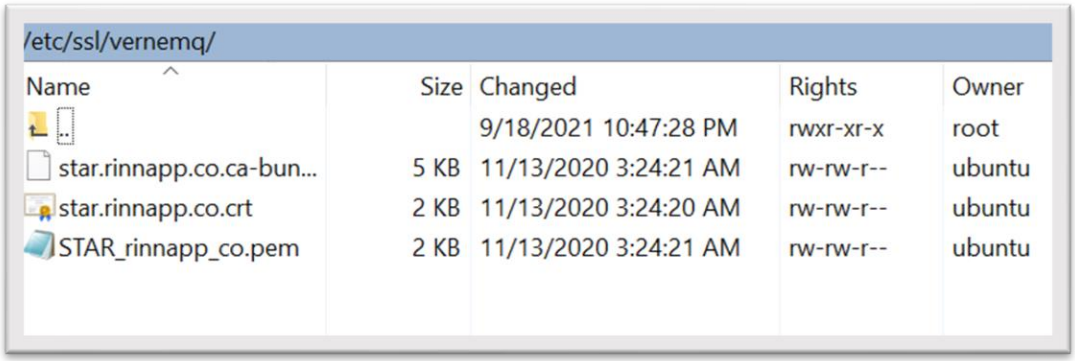


Figura 35. Certificados SSL para Vernemq

5.1.2.8 Configuración del servicio de notificaciones, ofertas, búsqueda y gráficos

Estos servicios están desarrollados en Python, se instala Python 3:

```
sudo apt install python3-pip
```

y se utiliza un ambiente virtual de Ubuntu para aislar las dependencias de configuración del resto de aplicaciones [5].

```
sudo pip3 install virtualenv==20.0.18
```

Se copia el código de las aplicaciones de notificaciones, ofertas, búsqueda y gráficos en una ubicación deseada y así mismo las variables de entorno como se observa en la Figura 33.

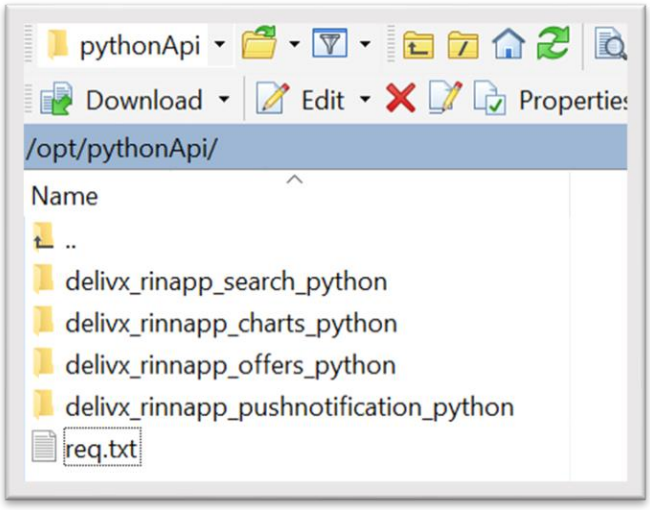


Figura 36. Aplicaciones de notificaciones, ofertas, búsqueda y gráficos y sus dependencias

En la ubicación en que se copiaron las aplicaciones se activa el entorno virtual:

```
source venv/bin/activate
```

y se instalan las dependencias que se encuentran en el archivo de req.txt. Este archivo contine los paquetes necesarios para el funcionamiento de las aplicaciones.

```
pip install -r req.txt
```

Por último, se añade cada aplicación como procesos en PM2 para que empiecen a funcionar.

Si todo el proceso se realizó correctamente desde la consola ejecutar el comando *pm2 list* lo cual mostrará una lista de los procesos corriendo bajo PM2 de los cuales *mainserver* es el proceso asignado para el API y los otros los procesos creados anteriormente.

App name	id	version	mode	pid	status
charts	2	N/A	fork	28016	online
entry	5	0.0.0	fork	0	online
mainServer	4	1.0.0	fork	0	online
pushnotification	6	1.0.0	fork	3729	online
offers/grocerOffres	1	N/A	fork	28015	online
search	3	N/A	fork	0	online

Figura 37. Procesos corriendo en ambiente de pruebas

Si se ejecuta el comando *pm2 logs* se observa el comportamiento de la API:

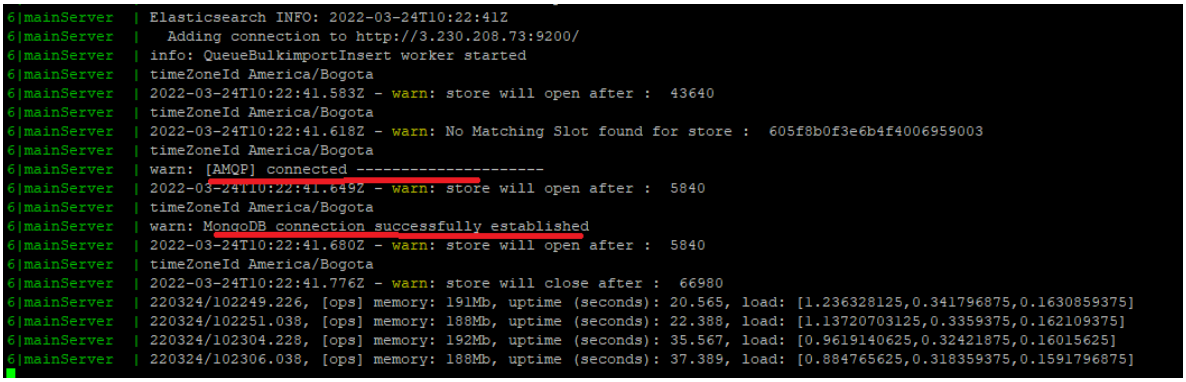


Figura 35. Logs del comportamiento del API de Rinn App

El anterior proceso de despliegue del ambiente de pruebas evidenció todo un ecosistema en funcionamiento para brindar el soporte a Rinn App, es importante resaltar que se redujo el número de servidores físicos, pero se mantuvo la arquitectura original.

5.2 CONSTRUCCIÓN DE SOFTWARE

5.2.1 Revisión de la documentación de la pasarela de pagos

Wompi, es una pasarela de pagos que ofrece sus servicios a los comercios y empresas para permitirles aceptar y procesar pagos digitalmente desde cualquier lugar del mundo. Es una nueva compañía del Grupo Bancolombia que tiene la cualidad de ser integrada fácilmente a las tiendas virtuales de las empresas por medio de un API amigable y código reutilizable [72]. En la página oficial de Wompi se encuentra la documentación para integrar la pasarela [73]. Se debe verificar qué requerimientos necesita la pasarela, conocer cuáles son las recomendaciones y observaciones para el correcto uso e implementación, cuáles son los códigos de respuesta HTTP de los servicios del API, la forma en que el API recibe los datos ya sea por medio de encabezados, parámetros o cuerpos y así mismo que formato de respuesta maneja. Finalmente se debe conocer cuál es el dominio del ambiente para hacer pruebas y cuál para realizar pagos reales.

Actualmente Rinn App cuenta con la pasarela de pagos ePayco, la cual está programada para realizar pagos con tarjetas de crédito y PSE.

5.2.2 Similitudes y diferencias entre pasarelas WOMPI y ePayco

Al analizar las dos pasarelas en cuestión se encuentran que desde el punto de vista conceptual, las mismas presentan varias similitudes, entre otras cosas, el

que ambas emplean los modelos *Gateway* y agregador, aceptan diversos medios de pago y en términos generales ofrecen servicios análogos: links de pago, para ventas sin contar con una página web; Plugin, para el módulo de pagos del *E-commerce*; API y *Checkout*, para aceptar pagos desde la página web de comerciante.[74][75]

Sin embargo, cabe señalar que existen ciertas diferencias entre las plataformas principalmente en lo relacionado con el costo de los servicios; ePayco por ejemplo ofrece un precio más favorable en lo que se refiere al Gateway, cobrando un valor por afiliación de $490.000 y un valor máximo de $269 por transacción exitosa, en tanto que para el modelo agregador si bien es cierto la afiliación la pasarela cobra el valor 2.99% + $900 + IVA por cada transacción exitosa [75].

Wompi, por su parte enfoca sus cobros a partir de la posibilidad de que el vendedor tenga acceso a los servicios y posibilidades que la pasarela ofrece. Así, el plan básico, completamente gratuito, ofrece unas funcionalidades limitadas de la plataforma que se circunscriben al link de pagos y al método de pago transferencia entre cuentas Bancolombia; los otros dos planes disponibles permiten al usuario acceder a las todas soluciones y funcionalidades Wompi para lo cual deberá pagar una comisión por cada transacción aprobada [16].

En relación las opciones pago, Wompi limita las posibilidades de pago en efectivo, pues únicamente admite el pago en corresponsales bancarios de Bancolombia, mientras que por su parte ePayco ofrece una amplia gama de posibilidades para el pago en efectivo, para lo cual pone a disposición del comprador 10 redes de recepción de pago en físico [76].

Por último, como diferencia importante, cabe mencionar que para el caso de Wompi solamente puede ser utilizada por las personas que tengan cuenta bancaria en Bancolombia y solamente acepta transacciones en moneda nacional, en tanto que ePayco soporta operaciones en dólares y puede ser utiliza teniendo cuentas bancarias en diferentes bancos [77].

5.2.3 Documentación de la pasarela de pagos Wompi

Los servicios se implementaron utilizando Postman, donde se definió una colección nueva, y se agregaron los servicios del API de Wompi. No se implementa de inmediato en el código, sino que se hace una especie de documentación con la finalidad de probar la forma de hacer las peticiones y cada variante de respuesta de los servicios de la pasarela. Para que cuando ya se vaya a codificar la documentación permite extraer flujos de comportamiento dependiendo de la respuesta.

5.2.4 Diseño de la arquitectura para soportar el nuevo sistema de pagos Wompi

El equipo de diseño está compuesto por un Arquitecto de software, un desarrollador móvil y un desarrollador backend, éste último es el rol que el practicante está desarrollando. El arquitecto propone una solución y un posible camino a seguir para la implementación del nuevo sistema, pero esto, es evaluado por todo el equipo y permite cambios, retroalimentación y posibles variantes que permitan llegar al objetivo. Se propone agregar una capa de servicios para mejorar la escalabilidad de la aplicación. Esto se debe a que el código presenta algunos errores en cuanto al uso de la arquitectura por capas ya que en la capa de presentación la implementación de la pasarela que se tiene actualmente (ePayco) está ejecutando acciones de lógica de negocio. Por tanto, el principal objetivo fue crear una abstracción de todos los servicios que provee la pasarela de pagos Wompi y los servicios de lógica de negocio propios de la aplicación para de esa forma permitir, mencionado anteriormente, escalar con nuevas funcionalidades que en futuro pudiesen deberse a una actualización propia de la pasarela o un cambio en el modelo de negocio de la empresa VICA Technology.

El proceso de desarrollo priorizó implementar los métodos de pago con tarjeta de crédito y PSE los cuales son los medios de pago actuales en Rinn App. Este tendrá en su implementación una capa de servicios y una capa de interacción quien establece una comunicación directa con el API de Wompi.

En su desarrollo se implementó una característica, definida como *long pooling*, que permitía obtener de forma periódica información de los pagos ya que estos toman su tiempo en ser confirmados y así validarlos en la aplicación. En la documentación de Wompi se enuncia que es necesario un mecanismo como este para obtener información de las transacciones: … “Activamente: haciendo una petición a nuestro API, por ejemplo, al endpoint…”.

Se añade una nueva entidad llamada *Payment* debido a que existe la necesidad de tener un registro con información adicional de las transacciones relacionadas al tipo recarga u orden de pedido, al medio de pago, a información del usuario más específica, de la transacción en la pasarela de pago, al estado de ella como por ejemplo si se pagó, si quedó pendiente, si se rechazó o se canceló y a los cambios de estado por los que pasó. Por tanto, el nuevo sistema de pagos usa como pilar la creación de este *Payment* para tener un mejor control y seguimiento de las transacciones. Cabe destacar que este puede estar relacionado a una o varias órdenes y a una recarga de billetera y que la información de su estado asociado al pago es la que se verifica para procesar las ordenes o procesar la recarga.

5.2.5 Implementación del pago con tarjetas

Realizar un pago con tarjetas de crédito (TC) a través de las pasarelas de pago tiene como finalidad hacer que el proceso de realizar una compra o un pago sea fácil haciendo que el cliente solo tenga que enfocarse en ese aspecto el de la compra, como bien se sabe puede involucrar escoger los productos, la cantidad, etc.

Para incluir los pagos con TC se debe primero enviar los datos de la tarjeta a la pasarela de pagos para que ella le asigne un identificador. Segundo, realizar un pago, en este caso de prueba, aleatorio entre 100 y 500 pesos para garantiza la validez de la tarjeta. Tercero, se almacena el identificador, pero con un estado de pendiente de validación. Cuarto, se comprueba que el pago realizado se hizo por el dueño de la tarjeta o al menos con su consentimiento pidiendo al usuario que ingrese el valor del pago aleatorio el cual podrá verificar generalmente por un mensaje de texto o correo que envía el banco con la información del pago. Y quinto, si se cumple lo anterior se actualiza el estado del identificador y se habilita para realizar pagos y compras.

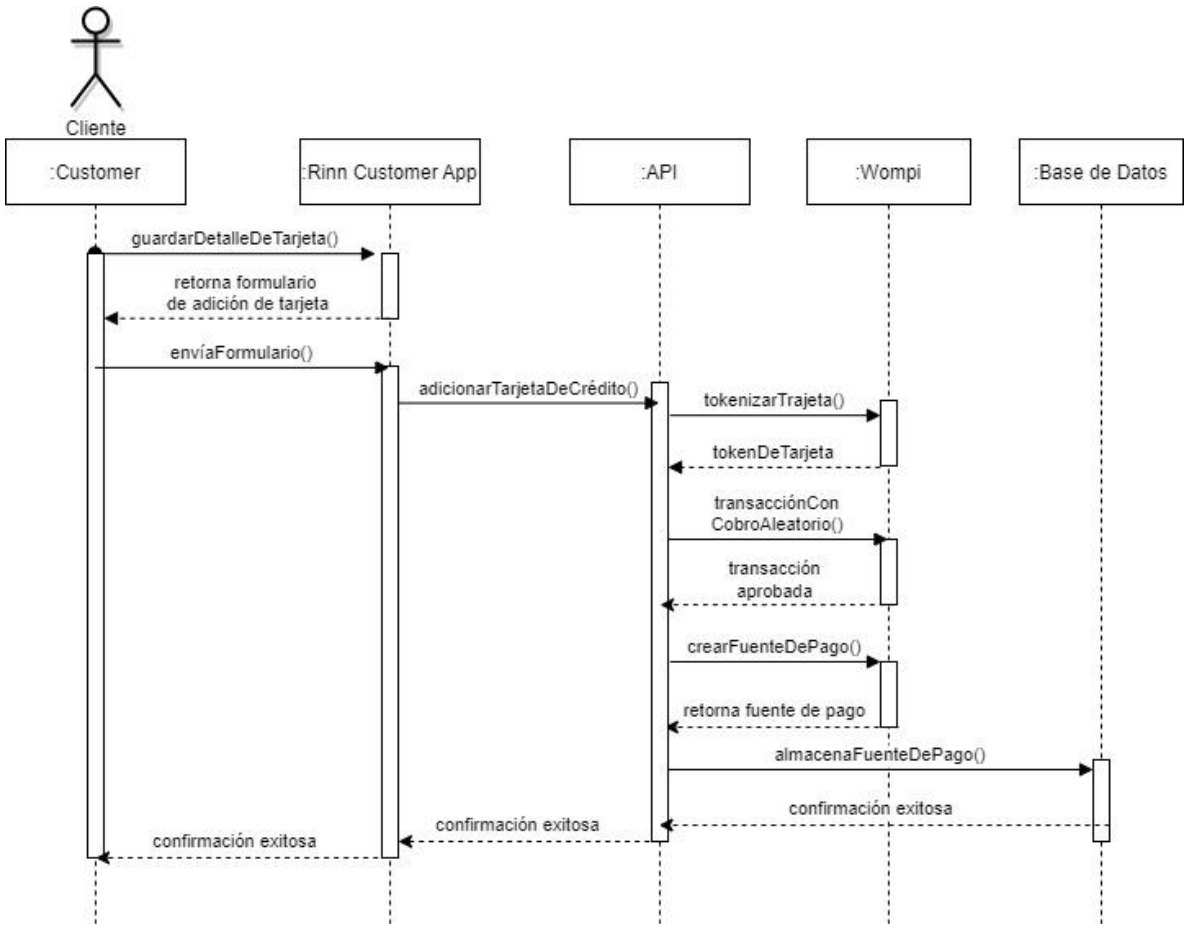


Figura 36. Diagrama de Secuencia - Guardar tarjeta de crédito

A partir de ahí todas las transacciones que se crean con TC incluirán este identificador y el cliente no tendrá la necesidad de volver a ingresar su número de tarjeta o su código de validación, siendo estos datos sensibles que nunca se deben almacenar en la base de datos.

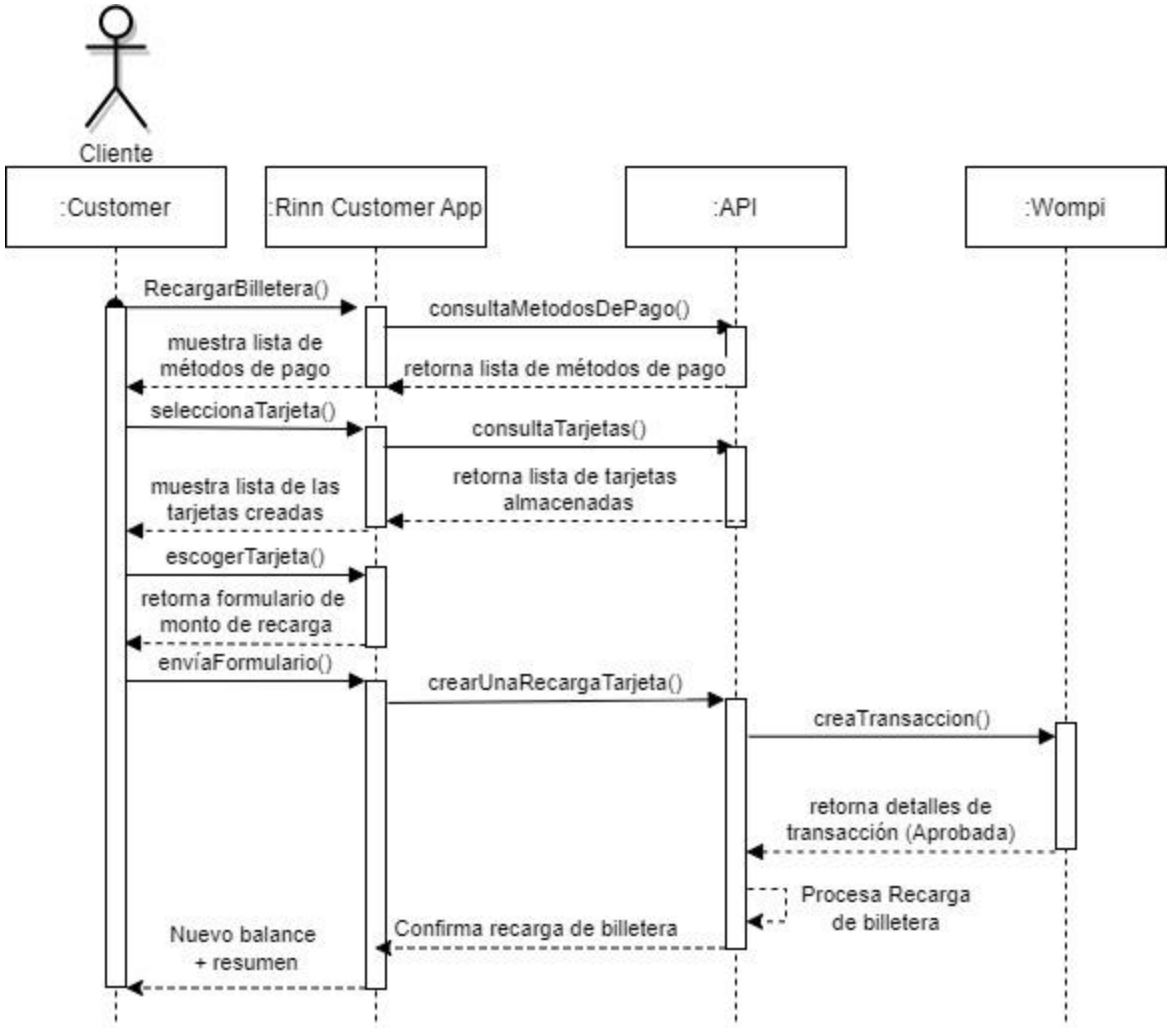


Figura 37. Diagrama de Secuencia - Recarga de billetera con tarjeta de crédito

5.2.6 Implementación del pago con PSE

PSE se considera una herramienta operativa de comercio electrónico, desarrollada por ACH Colombia e integrada al sistema bancario colombiano en el año 2005. Este sistema de pagos electrónico permite a los consumidores online pagar en línea y de manera ágil bienes adquiridos en el e-*commerce*, servicios y facturas. Su funcionamiento consiste en que a través de un procedimiento sencillo se debitan los recursos en línea de la entidad financiera en la cual el comprador

guarda su dinero, montos que se depositan en la entidad financiera establecida por la empresa o la tienda virtual [78].

La plataforma PSE, incorporada en el web site de la empresa vendedora, direcciona a las personas que estén realizando compras en línea desde el ambiente de los comercios o entidades gubernamentales a las páginas de internet del banco de los usuarios, para desde allí realizar el pago directamente a través de una operación débito [79].

Así las cosas, el Sistema Electrónico de Pago-PSE está integrado por varios actores a: Usuarios, empresas, entidades financieras. Los usuarios corresponden a personas naturales o jurídicas que disponen de una cuenta en una entidad financiera y que compran bienes y servicios a través de la página web de una empresa o comercio electrónico [80].

Las empresas, por su parte, se tratan de organizaciones comerciales que mediante sus páginas web o plataformas electrónicas ofrecen productos y/o servicios, para cuya gestión de los pagos en línea integran el sistema. Desde el comercio electrónico se envía y recibe información a PSE [80].

Las entidades financieras, corresponden a instituciones que forman parte del sistema financiero colombiano, las cuales a través de un negocio jurídico se asocian con ACH Colombia, para beneficiarse del Sistema Electrónico de pagos. Ahora, a través de tales entidades es que se realizan las transacciones dentro del PSE [81].

Ahora bien, la incorporación una página web de una empresa o comercio al botón de pagos PSE implica que deben poseer una infraestructura tecnológica que permita garantizar una conexión estable con el sistema electrónico de pagos y un esquema de comunicación segura entre la empresa y el operador del PSE. Concretamente se requieren los siguientes requisitos técnicos.

VPN: Implementar un canal de comunicación cifrado VPN con IPsec tipo Site to Site con dos direcciones IP públicas para establecer la comunicación. [81]

Web Service: Construcción de 4 métodos de comunicación estándar Web Service compuesto por SOAP, XML y WS-Security [81].

Certificado Digital: Utiliza un certificado digital de firma X 509 versión 3, con una clave publica RSA de 2048 Bits que cumpla con las características técnicas exigidas por ACH y sea emitido por las entidades certificadoras avaladas por la ONAC, para cifrar la información enviada a través del canal VPN por los *webs service* [81].

Las empresas de la misma forma, cuando no disponga del soporte tecnológico para incorporar el botón PSE o no deseen hacerlo, pueden acceder a dichos servicios a través de la pasarela de pagos a la que se encuentre vinculada, tanto en el modelo de Gateway como en el agregador de pago [81].

El funcionamiento de PSE es el siguiente: el cliente ingresa al sitio web o comercio electrónica, selecciona los productos o servicios que desea adquirir, así mismo liquida el pago; selecciona la opción PSE como medio de pago; selecciona la entidad financiera en la que tiene los fondos y da la opción de pagar; el cliente ingresa las credenciales y procede la autenticación y autoriza el débito y; finalmente, PSE informará a la entidad financiera recaudadora y empresa el pago realizado por el usuario [82].

Para la implementación de este medio de pago primero se realizó una validación de la forma en cómo se estaba realizando en Rinn app. Hay que recordar, como se mencionó antes, que los pagos con tarjeta y PSE eran los únicos en funcionamiento empleados para realizar recargas y pagos de órdenes. Se detectó que la forma en cómo se procesaba el pago generaba errores debido a que no existía una forma correcta de validar el estado de la transacción y además no se consideraban todos los posibles estados de la transacción. Por tanto, era necesario evitar estos errores en la nueva implementación del pago con PSE.

El proceso para realizar un pago con PSE difiere del método de pago con TC ya que se divide en dos fases muy notorias. La primera fase consiste en la creación de una transacción quien forma simplificada necesita un monto o valor y retornará un enlace de redirección proporcionado por el sistema ACH-PSE. La segunda fase consiste en llevar al cliente a este enlace para que realice el proceso de pago en su respectivo banco. Después de realizada la primer y segunda fase el sistema de pagos trae devuelta al cliente a la aplicación de Rinn app en donde el paso siguiente es validar el estado de la transacción.

Es importante mencionar que ahora ambos medios de pago TC y PSE se estarán procesando desde el servidor y no desde las aplicaciones de Android o iOS como se estaba realizando anteriormente. Implementar este proceso de pago en Wompi implicó la definición de dos nuevos *endpoints* uno para la creación de la transacción y otro para la validación de esta, la creación de una capa de servicios y una capa de interacción directa con la pasarela.

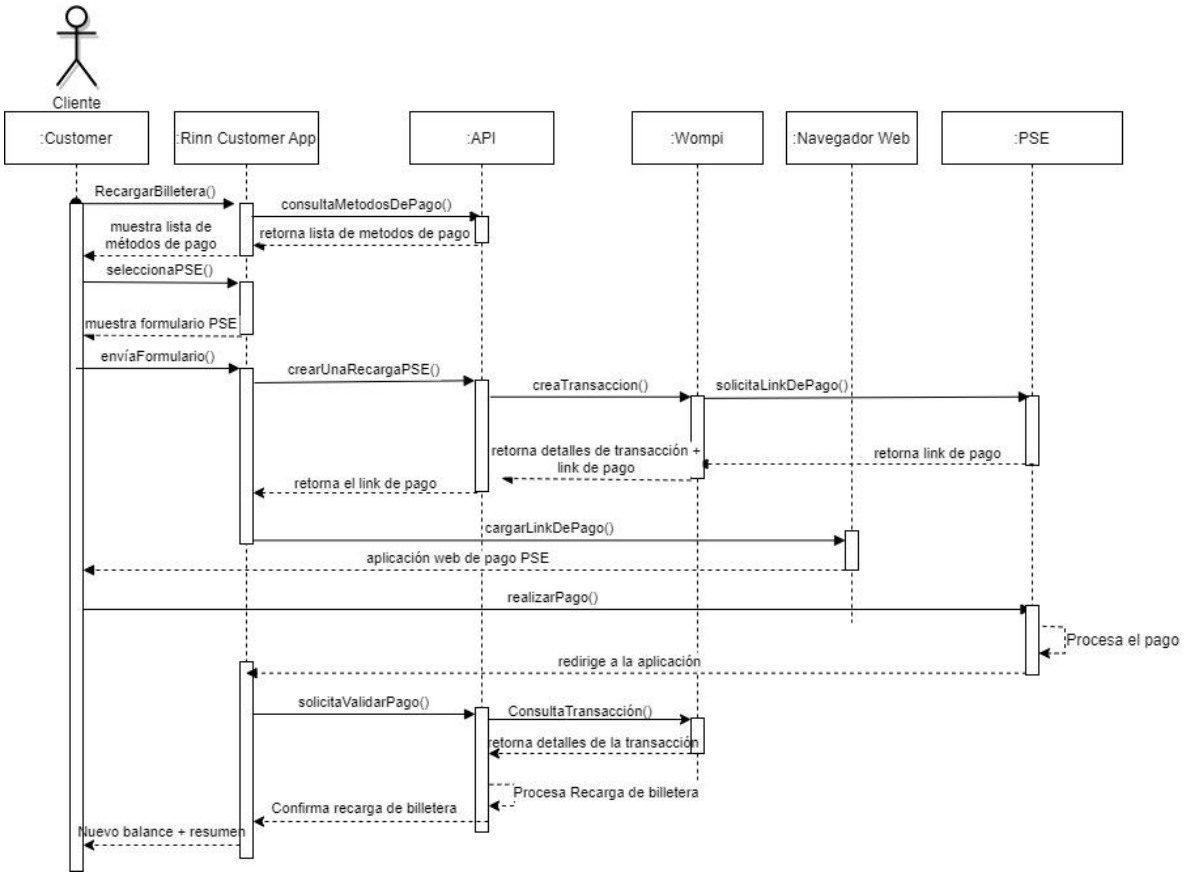


Figura 38. Diagrama de Secuencia - Recarga de billetera con PSE

5.2.7 Implementación del pago con Botón Bancolombia en Rinn App

Botón Bancolombia es uno de los nevos medios de pago digitales que provee Bancolombia. Todos los usuarios con una cuenta débito o crédito pueden acceder a este servicio y las transacciones pasan por un proceso de aprobación muy rápido pero seguro. Este medio de pago funciona de forma similar a PSE ya que lo primero que se debe hacer es crear una transacción con este método de pago. Luego retorna un enlace el cual redirecciona al sitio donde el cliente completa allí su proceso de pago. Cuando el cliente finaliza este proceso es redirigido a una URL que se ha especificado en la configuración de la transacción y el backend se encarga de validar la transacción con el mismo método anteriormente implementado, el *Long Pooling*, en donde se determina si la recarga de la billetera fue exitosa, rechazada o algún fallo y así mismo se le da la retroalimentación al cliente.

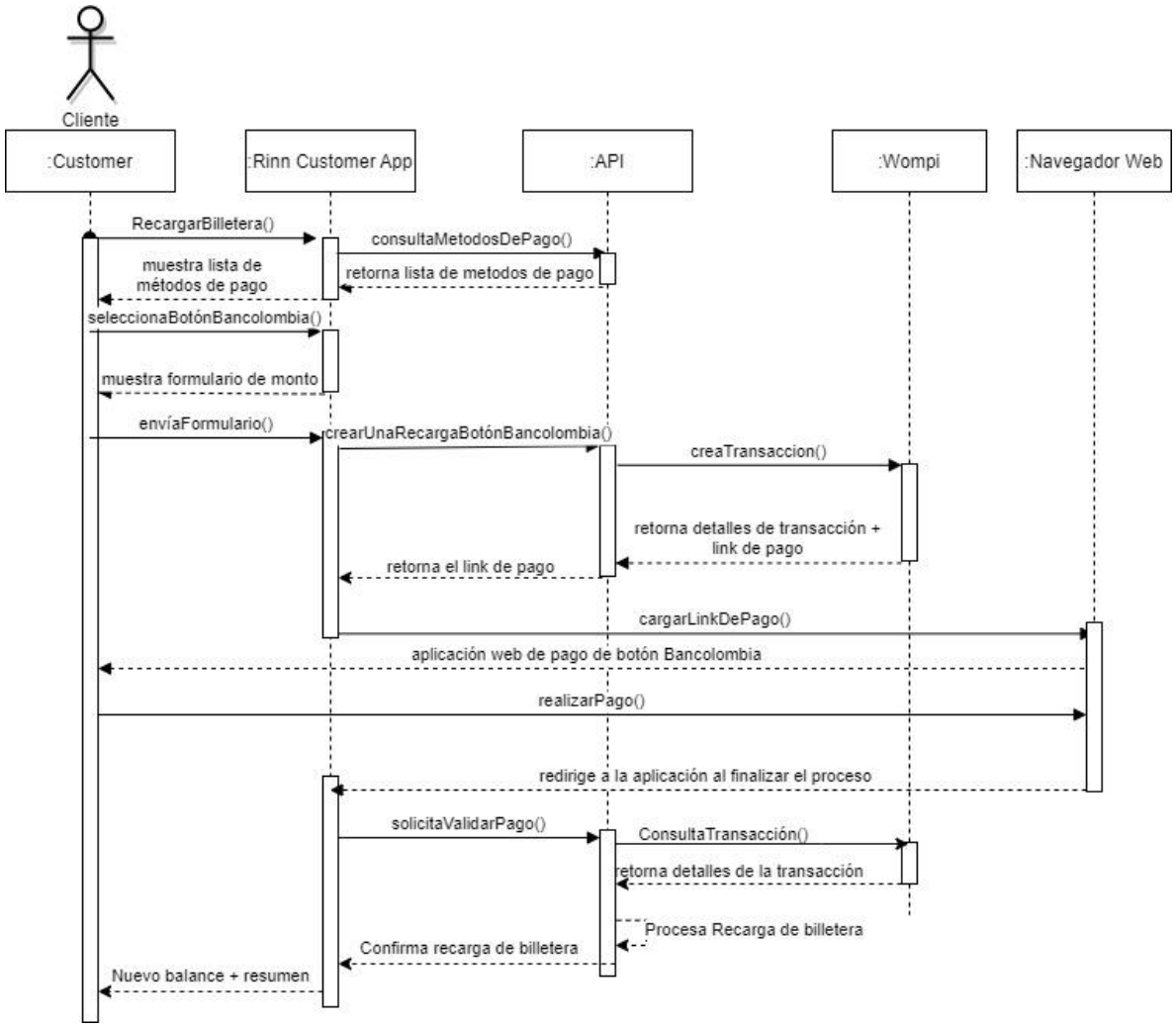


Figura 39. Diagrama de Secuencia - Recarga de billetera con Botón Bancolombia

5.2.8 Implementación del pago con Nequi en Rinn App

Se define como una billetera 100% digital o una Mobile wallet, es una línea de negocio del grupo Bancolombia y “Su propósito es que las personas tengan una mejor relación con su dinero, empoderándolas para que logren lo que quieren con ella” [Nequi]. En Colombia cuenta con aproximadamente 11’600.000 usuarios y esto hace que sea un medio de pago importante y de gran relevancia para incluirlo en el procesamiento de órdenes y recargas en Rinn App.

La implementación este medio de pago mediante el API de Wompi requiere primero se envíe el número de celular para verificar que este está asociado a una cuenta Nequi, si está validación se aprueba dentro de la aplicación de Nequi el cliente recibirá una notificación preguntándole si desea aceptar pagos desde el comercio de Rinn App, si el usuario acepta esto se notifica al API de Rinn y junto con un código que debe ser almacenado y anexado en las transacciones para

procesar pagos a futuro. Este proceso es muy sencillo para el usuario y además seguro debido a los protocolos que implementa Bancolombia.

Es importante tener en cuenta que la implementación de los métodos de pago Nequi y tarjeta de crédito son similares, así como Botón Bancolombia y PSE. Si en un comercio se desean implementar estos métodos de pago se recomienda implementar primero el procesamiento con tarjetas de crédito y luego Nequi, ya que el primero puede tener algunos pasos adicionales, pero llevarlos a cabo hace que implementar Nequi sea más fácil. En cuanto a PSE y Botón Bancolombia se recomienda primero implementar PSE, requiere un formulario más largo para crear la transacción, pero el proceso siguiente es prácticamente el mismo lo que permite reutilizar gran parte de la lógica haciendo que el proceso sea más rápido.

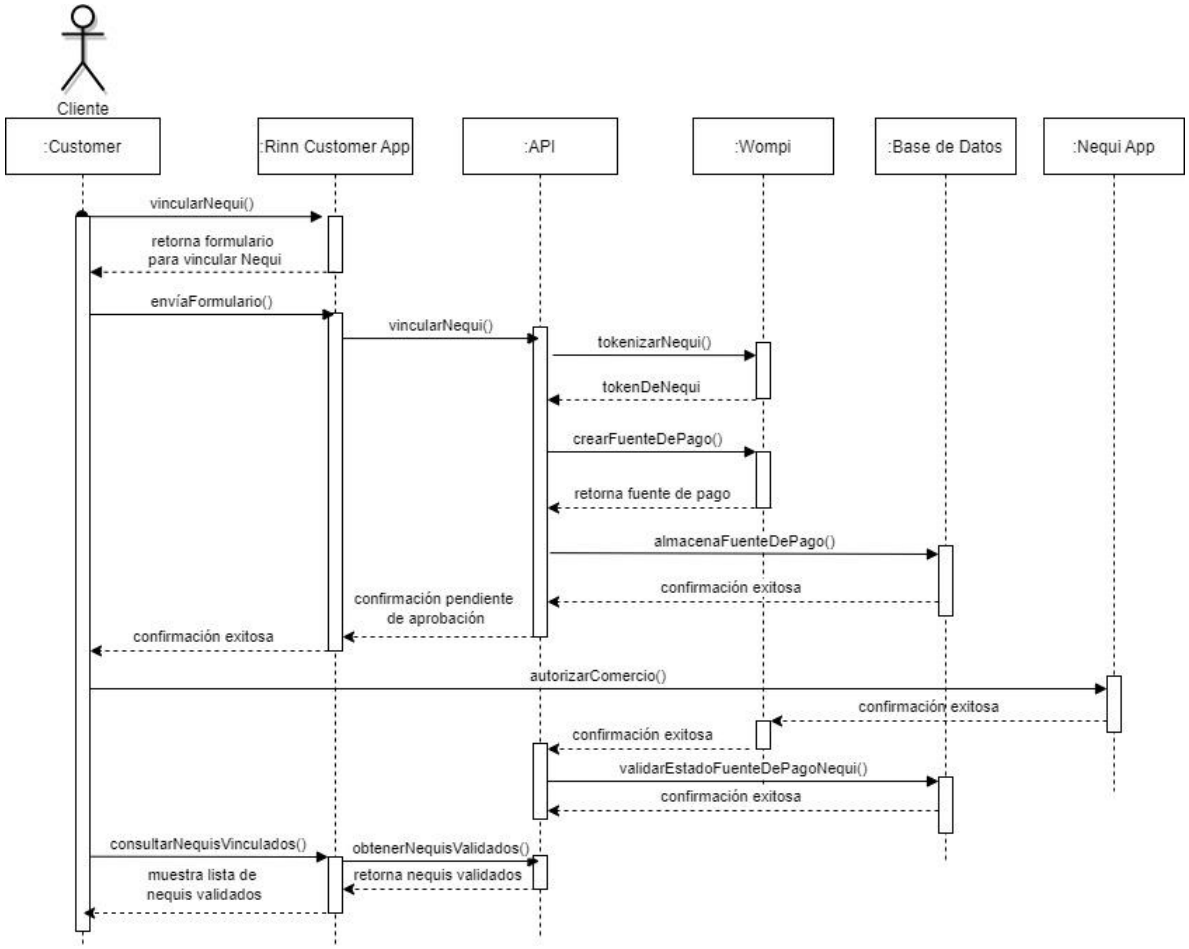


Figura 40. Diagrama de Secuencia - Vinculación de una cuenta Nequi

5.2.9 Validación en dos pasos de la tarjeta de crédito

Rinn App padecía de una brecha en la seguridad que ofrecía a sus clientes en el procesamiento de pagos con tarjeta de crédito debido a que, si el dueño de la

tarjeta pierde su tarjeta, o una persona cercana tiene acceso a ella sin autorización, con los datos que figuran en la tarjeta pueden vincular la tarjeta como medio de pago, lo cual consistía en que se realiza un cobro aleatorio menor a 500 pesos, y si el pago es aprobado ya podía realizar recargas u ordenes de pedidos a través de la plataforma. Es por esta razón que se implementa una solución definida como Validación en dos pasos, que consiste en que luego de realizado el cobro aleatorio éste cobro se notifica generalmente al celular o correo del dueño de la tarjeta y deberá ingresar este valor cobrado para poder habilitarla como medio de pago. Además, cuenta con 3 intentos únicos para ingresar el cobro exacto y de ser consumidos de manera errónea el código único que se generó para procesar pagos se inhabilita y el usuario tendrá que volver a iniciar el proceso de vinculación del medio de pago.

5.2.10 Diseño de un API para la nueva billetera de Rinn App

Actualmente la billetera de Rinn App solo funciona para realizar pagos de órdenes de restaurantes o pagar por realizar un Favor. Con la implementación de la pasarela de pagos Wompi se habilita la recargar con otros medios de pago como Nequi y Botón Bancolombia además de los ya existentes: tarjetas de crédito y PSE. Con esto el objetivo principal de la nueva billetera será ampliar sus funciones para que pueda permitir realizar la compra de diferentes servicios como: SOAT, Servicios públicos, recargas de celular, paquetes de celular y pines de entretenimiento siendo estos los más relevantes.

En el proceso de diseño se define un modelo en el cuál la nueva billetera de Rinn App está compuesta de varios tipos de productos que ella puede pagar como SOAT y Servicios Públicos, estos tipos de productos tienen unos subtipos de productos los cuales pueden ser los proveedores de ese tipo de producto y finalmente estos tienen el producto final que va a comprar el cliente.

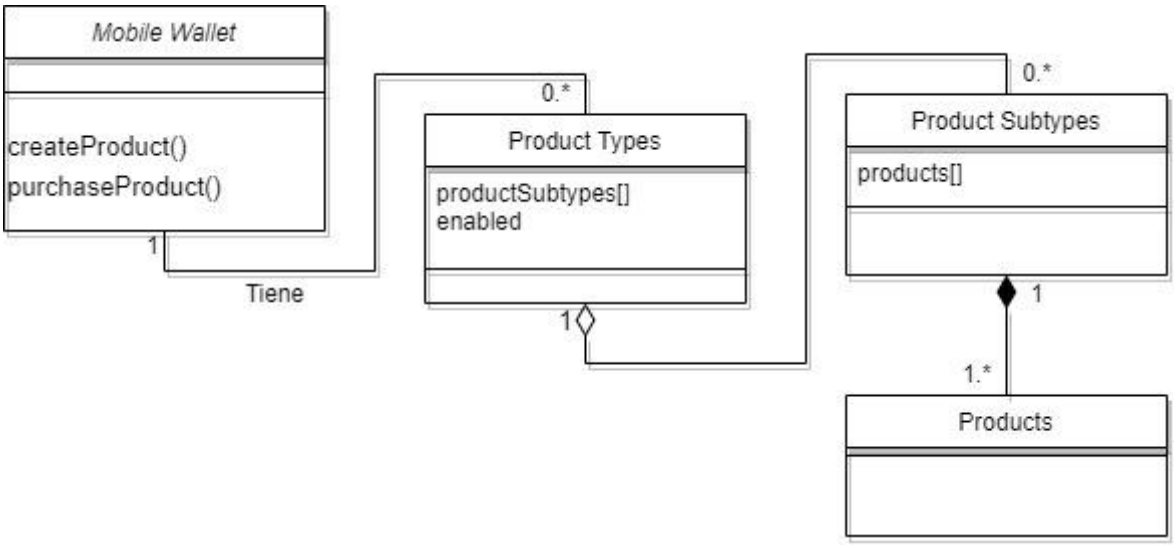


Figura 41. Diagrama E-R 1. Modelos de productos para la nueva billetera de Rinn App

Para la gestión de la información de la compra de productos de billetera se crea una nueva colección que guardará los datos del cliente, los detalles del producto, los costos y la fecha, siendo estos los datos más relevantes para mencionar.

5.2.11 Proceso de compra de SOAT

El proceso de compra de un SOAT requiere de gran cuidado ya que es una operación que tiene un intercambio monetario relativamente alto. En Colombia el valor de una póliza inicia desde los doscientos mil pesos hasta un millón quinientos mil pesos aproximadamente [seguros], por esta razón un error en la operación puede representar una pérdida significativa para el comercio, en caso de no ser posible una corrección del problema, y la desconfianza de los usuarios [83].

El flujo para la compra de un SOAT se define presentado un formulario al usuario para el ingreso de la placa del vehículo, el tipo y la cedula de la persona que figura en la tarjeta de propiedad del vehículo. Para esta funcionalidad se define una ruta de tipo POST que en su configuración recibir un cuerpo con los datos anteriores, se especifican los tipos de datos de cada valor enviado, se establecen todos los valores como obligatorios, se definen los códigos de respuesta para un caso exitoso, fallido y por error del sistema y dentro de la cabecera que envía la solicitud HTTP se recibirá un token de autenticación del usuario registrado.

Con esta información el API realiza una consulta al proveedor de servicios para el SOAT y obtiene la información relacionada al costo y dependiendo de la finalidad

del uso del vehículo esto puede variar, es decir, si es un vehículo familiar o de servicio público, por ejemplo.

Está primer interacción con la aplicación se define dentro del sistema como una cotización debido a que un usuario puede simplemente querer consultar el costo de su póliza sin llegar hasta el final del proceso, así como también conocer los diferentes costos de la póliza para finalidades distintas.

A partir de definir una cotización el sistema está preparado para crear un pago, pero antes de eso se deberán completar datos relevantes como la ubicación, el teléfono y el correo. Cabe destacar que, aunque para realizar esta cotización el usuario debe estar registrado en el sistema, es necesario que se complete esta información debido a que la póliza no necesariamente la va a comprar este mismo usuario registrado, sino que puede realizar la operación para otro usuario y los datos no serán los mismos que ya se tienen almacenados en la base de datos. En la Figura 42 el proceso anterior termina cuando se “Renderiza los datos de la cotización”. A partir de ahí se da el proceso de compra explicado a continuación:

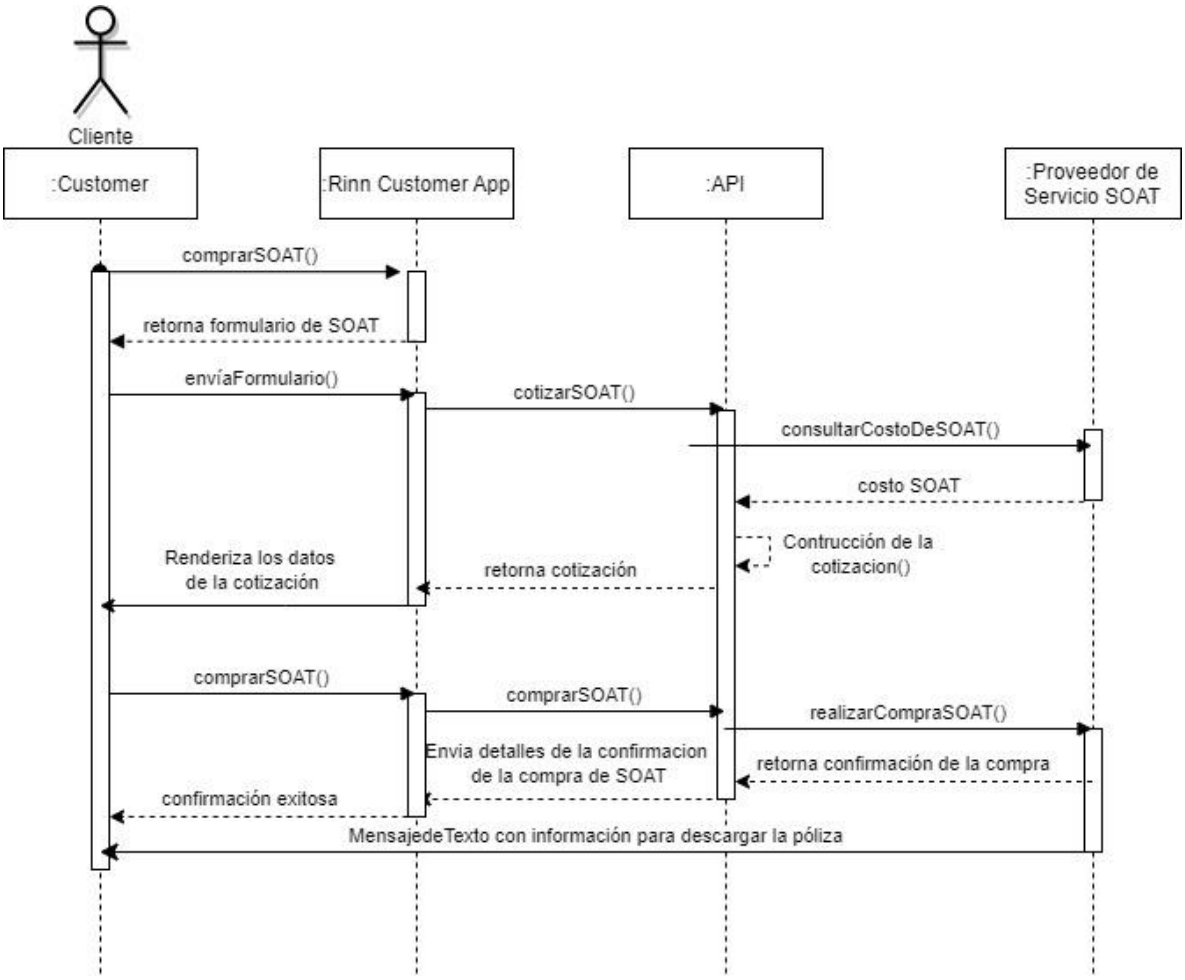


Figura 42. Diagrama de Secuencia – Proceso de compra de SOAT

Luego de diligenciada toda la información requerida cuando el usuario ejecuta la compra el API de Rinn App envía esa información al proveedor de servicios y es este quien procesa la información y establece si la compra fue exitosa o no, quien por defecto enviara una respuesta argumentando el motivo del rechazo o fallo. En caso de una transacción exitosa la póliza generalmente es enviada al correo electrónico o en un mensaje de texto que incluya ya sea un link de descarga o un link de redirección a una página web para permitir su descarga.

5.2.12 Proceso de pago de Servicios públicos y facturas

Pagar facturas de Servicios Públicos o servicios adquiridos como facturas de plan de datos, de internet para hogares, de televisión, entre otros, representa en muchos casos tener que hacer el pago en un banco o una sucursal física o de aliado y hacer una fila. Esta forma tradicional evita que el tiempo se invierta en cosas más importantes y genera en la mayoría de los casos gastos relacionados al trasporte por mencionar el más inmediato a qué se presente. Agregar este proceso de forma digital a la billetera móvil de Rinn App tiene la intención de evitar la incomodidad, ahorrar tiempo en el desplazamiento y el gasto adicional para pagar las facturas.

En Rinn se hizo un esfuerzo para que el flujo de pago de estas facturas sea muy sencillo para el usuario y le permita reconocer que es un beneficio claro utilizar la aplicación y no usar el modelo tradicional para pagar las facturas. El pago de facturas y servicios públicos inicia solicitando al cliente cual es el proveedor del servicio de su factura. Actualmente son miles de proveedores a nivel nacional y se debe proveer una búsqueda que vaya arrojando resultados a medida que se van digitando las letras del nombre del proveedor. Este enfoque es muy útil para el usuario ya que le permite no tener que digitar el nombre completo del proveedor, sino que una breve cadena que introduzca ya obtendrá los nombres de los proveedores que coinciden con dicha cadena, para que así lo pueda seleccionar de forma rápida y continuar al siguiente paso.

Hasta este momento lo primero que se implementa es un servicio de búsqueda que satisface el proceso anterior y se materializa en un endpoint que recibe como un *query param* la cadena de texto. Se define una solicitud HTTP de tipo GET con la siguiente estructura:

```
https://<dominio>/<Definición del enpoint>?text=<cadena de texto>
```

Los resultados obtenidos deberán contener principalmente el código que identifica al proveedor su descripción. En este caso si se quisiera mostrar la imagen del proveedor buscado depende mucho del proveedor de servicios debido a que, si esa información no es enviada y sumado a eso el hecho de que son miles de proveedores, se debería almacenar en base de datos un registro por cada uno de

ellos y agregarle una imagen, lo cual no resulta factible y la mejor opción es hacer la consulta y almacenar la data que sea enviada en cache de forma temporal.

A partir de ahí se necesitan dos servicios uno que obtiene el valor de la factura y otro que procesa el pago de esta. Para obtener el pago de la factura el usuario continúa el flujo y debe ingresar la referencia de su factura y con el dato anterior del proveedor y este se puede consultar esta información. Ya que un usuario puede tener la intención de solamente llegar hasta el punto de consultar el proveedor e ingresar la referencia para que le muestre el detalle de la factura el servicio que obtiene el valor de la factura básicamente hace una cotización únicamente. Se podría decir que si hasta este punto si solo se hacen consultas no habría necesidad de almacenar dicha información, pero, si no se hace, no se tendría registro de las intenciones que ha tenido el usuario al utilizar la aplicación. Ya se tiene un registro de que el usuario ha realizado una cotización y por ende a futuro sugerirle que finalice el proceso.

En la Figura 43 se representa el proceso en un diagrama de secuencia y se pueden evidenciar cada una de las secuencias comentadas anteriormente:

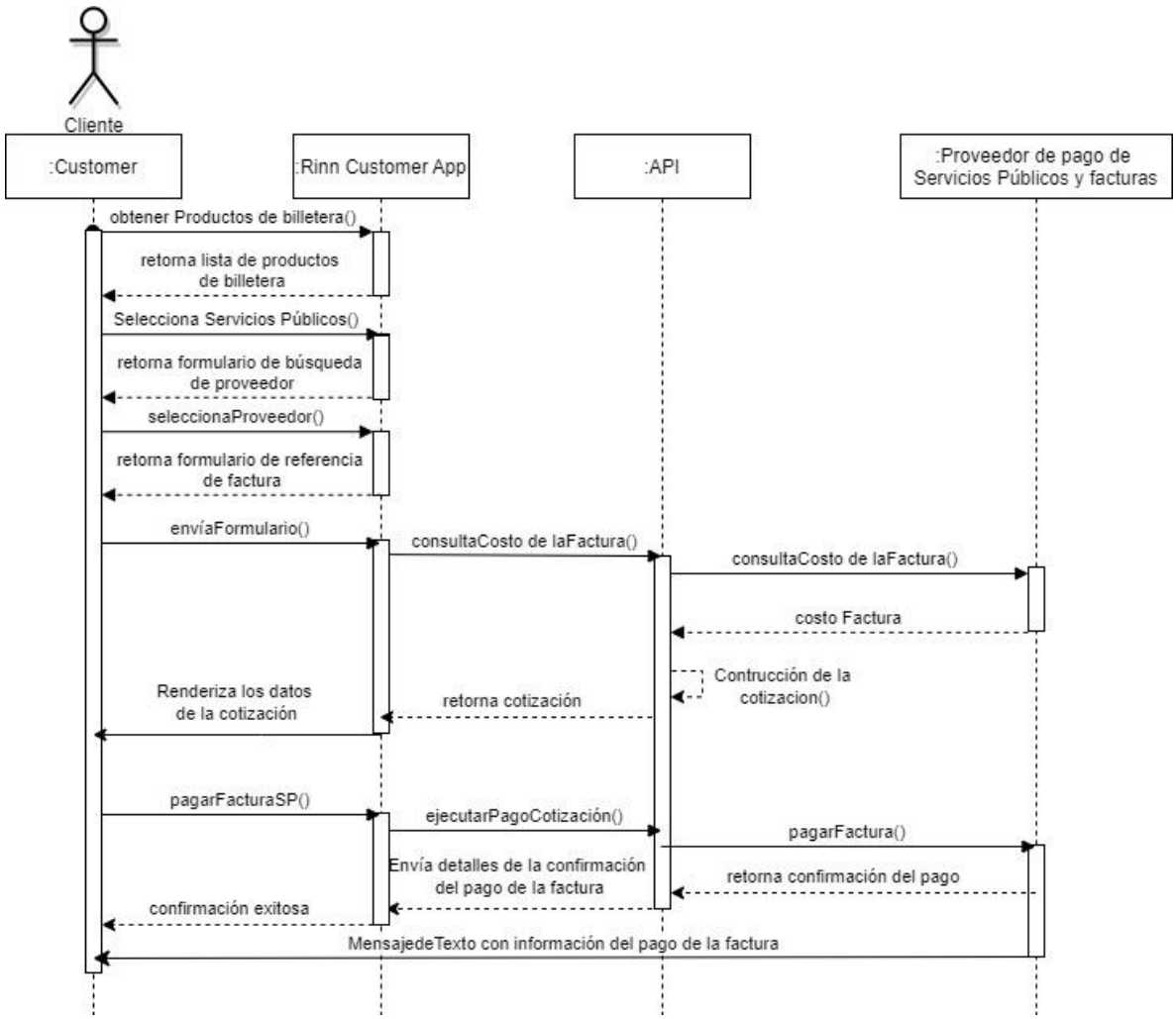


Figura 43. Diagrama de Secuencia – Pago de una factura de Servicios Públicos

Para el servicio que obtiene el valor de la factura se provee un endpoint de tipo POST que se encargará de realizar la cotización y recibirá en su cuerpo de petición el proveedor y la referencia de su factura.

Finalmente se implementa un endpoint que ejecutará el pago de la factura y solo será necesario enviarle el identificador de la cotización para que la información se consulte desde el *backend* y pueda procesar el pago.

6 LECCIONES APRENDIDAS

Para cualquier estudiante de ingeniería de sistemas deberá ser una experiencia enriquecedora realizar una práctica profesional en una empresa de desarrollo de software. Es un camino corto que se desarrolla en 6 meses pero que permite aprender muchas cosas valiosas que complementan la carrera profesionalmente y brinda la tendencia a la experticia en las áreas en que se debe trabajar. Sumado a esto se aprende de los errores más que de los aciertos así que es importante mencionar los más relevantes del proceso.

Hay que tener en cuenta, en algunos casos, que no se debe confiar ciegamente en los datos que envía el cliente al hacer una petición al API. En lo posible se debe corroborar la información que envía en la base de datos para evitar que se realicen consultas desde Postman por ejemplo y se puede manipular la información a gusto. Por ejemplo, cuando un usuario va a realizar una recarga de billetera lo primero que hace es consultar sus consulta sus fuentes de pago, selecciona una y luego define el monto a recargar. Entonces mediante una petición al API se va a enviar esta información. Lo que se debe hacer aquí es primero que todo validar si esa fuente de pago existe y segundo si esa fuente de pago pertenece al usuario que la creó. Es posible que desde Postman se intente hacer la misma consulta, pero desde un usuario distinto así que la validación permitirá que no se viole la seguridad del usuario al evitar que alguien procese un pago con su fuente de pago.

En el desarrollo de una API en NodeJs es importante que en los manejadores (*handlers*) de las rutas no se defina lógica de negocio, sino que se haga un manejo de errores e invocaciones a los servicios únicamente, es recomendable crear una capa adicional que maneje esa lógica. De esta forma se puede reutilizar estas funcionalidades en diferentes partes del código.

Cuando se desarrollan aplicaciones generalmente se está acostumbrando a retroalimentar a usuario de la aplicación con un mensaje que indique cual f el problema, pero, en una API cuando ocurren errores en el procesamiento de alguna solicitud, en la mayoría de los casos no se le debe enviar la respuesta directamente al usuario de los que ocurrió en el servidor, sino que se debe mapear este error y entregarle una respuesta relacionada a que no se pudo procesar la solicitud, por ejemplo: “Ups! Hubo un problema en nuestro sistema, intenta de nuevo por favor”. De esta forma se están creando respuestas genéricas y se evita que se conozca de cierta manera la forma en que se construyó el API.

Relacionado a los requerimientos, cuando se reciben, es una buena práctica ir documentado las tareas que se van realizando debido a que son varias las que en el proceso de desarrollo están completando. Las actividades de los requerimientos conforman básicamente la interfaz de lo que como desarrollador se va a trabajar y es relevante llevar un control de dichas tareas completadas. Esto sirve primero,

para poder definir una métrica del progreso que se ha tenido desde que se inicia en la empresa y segundo, para entregar un informe del trabajo realizado cuando sea requerido.

Al realizar modificaciones en el código que sea, se cambia el flujo de la operación o se definen nuevas condiciones lógicas que alteran la lógica y el comportamiento de la aplicación. Por experiencia lo recomendable es documentar por qué se realizan los cambios y la forma en que el nuevo flujo va a operar. Esto permite que, en el futuro, por motivos de mejoras, por ejemplo, al modificar el código podamos acceder a la documentación y recordar o verificar lo que en su momento se hizo y las razones e implicaciones lógicas para hacerlo de esa manera. Porque muy probablemente se puede olvidar y esto evita tener que repasar el código tratando de entender la lógica implementada.

7 CONCLUSIONES

La pandemia trajo consecuencias negativas y afortunadamente positivas también y por ello las empresas evidenciaron que el acceso a los servicios básicos como la alimentación, por medio de los dispositivos móviles en un aplicativo logra un gran impacto porque satisface una necesidad fundamental por medio de un acceso fácil y sencillo. De esta manera Rinn App, está generando un valor importante en la sociedad y específicamente en ciudades de Colombia como Cali, Jamundí y Tuluá, donde los servicios de delivery no están presentes.

Al incluir nuevos medios de pago para hacer uso de sus servicios se vuelve muy incluyente porque intenta que la mayoría de las personas puedan beneficiarse de la iniciativa y de participar de las comodidades que brinda el servicio de *delivery,* ya no es necesario que una persona tenga una tarjeta de crédito, sino que puede crear una de las billeteras digitales actuales como Nequi y Daviplata para poder ser beneficiada.

Los requerimientos desarrollados en la práctica profesional pudieron realizarse completamente debido a una buena estimación y alcance de los objetivos. Cabe resaltar que el proceso de adaptación del estudiante a un ritmo laboral es lento al principio y se convierte en un reto durante los primeros meses. Cuando se van adquiriendo las habilidades necesarias el ritmo de trabajo parece disminuir, pero lo que en realidad sucede es que se ha vuelto más competente y ha adquirido mejores formas de afrontar los problemas.

Cuando se construye un API es importante que la definición de las rutas, así como de las variables, sean autodescriptivas además que el uso de los métodos de las peticiones (GET, POST, PUT) estén siempre acordes a la funcionalidad que proveen. Por ejemplo: si se define un método GET lo ideal es que su funcionalidad sea la de recuperar información.

Al definir una ruta ésta va asociada a un manejador o *handler.* En la práctica se evidenció que los *handlers* entraban en interacción directa con la capa de datos lo que hacía que desarrollaran una funcionalidad muy amplia y esto impide que se pueda reutilizar las funciones dentro del código, esto se debe a que los parámetros del *handler* son la petición HTTP misma y la respuesta a esa petición (*request, reply*), por tanto, si se intenta reutilizar la funcionalidad de un *handler* en específico se deben de proveer estos mismos parámetros de entrada lo que desde dentro del código se hace inviable. Lo mejor es crear una capa de servicios que conecte los *handlers* con la capa de datos para que cualquier manejador pueda tener acceso definiendo parámetros de entrada sencillos, ya sean como datos primitivos o estructuras de datos más elaboradas, pero no como la información tan abundante que se presenta en una petición HTTP.

El desarrollo de las habilidades duras son un eje fundamental para el crecimiento dentro de la empresa, pero las habilidades blandas complementan un desarrollo personal tanto dentro de la empresa como fuera de ella y esto es lo que forma como ingenieros. La multidisciplinariedad en la empresa permite reforzar valores como la tolerancia y el respeto, sumado a esto desarrollar habilidades de comunicación para poder expresar las ideas a personas que no necesariamente son estudiadas en el mismo campo, de aprender a escuchar al otro, de trabajar en equipo y ser compañerista y de gestionar el tiempo para beneficio no solamente personal sino también para el bien común.

8 BIBLIOGRAFÍA

[1] "Conozca los emprendimientos con mayor potencial en Colombia - Infobae." https://www.infobae.com/america/colombia/2021/05/14/conozca-los-emprendimientos-con-mayor-potencial-en-colombia/ (Último acceso Nov. 08, 2021).

[2] "Plataformas robustas, el 'must have' del e-commerce – Consumotic." https://www.consumotic.mx/ecommerce/plataformas-robustas-el-must-have-del-e-commerce/ (Último acceso Nov. 08, 2021).

[3] "Tecnología, la verdadera ganadora de la era Post-Covid - Softtek." https://softtek.eu/tech-magazine/digital-transformation/tecnologia-la-verdadera-ganadora-de-la-era-post-covid/ (Último acceso Nov. 08, 2021).

[4] H. G. Doria, "Las Metricas de Software y su Uso en la Region," May 2001.

[5] L. A. Cardenas Ospina y J. A. Rodríguez Beltrán, Sistema de gestión de pruebas para productos de software, Trabajo de grado, Departamendo de Ingeniría de Sistemas, Bogotá: Universidad Libre, 2011."

[6] Vivek Venugopalan, «CVS Best Practices». Copyright © 2001 Vivek Venugopalan, oct. 15, 2005. Último acceso: nov. 25, 2021. [En línea]. Disponible en: https://tldp.org/REF/CVS-BestPractices/CVS-BestPractices.pdf

[7] D. A. Hurtado Chichande, Tecnología de contenedores de sofware en entornos de pruebas. Trabajo de Grado de Maestría, Esmeraldas, Ecuador: Escuela de Sistemas y Comunicación, Pontificia Universidad Católica de Escuador,, 2018.

[8] M. Callejas Cuervo, A. C. Alarcón Aldana y A. M. Álvarez Carreño, «Modelos de calidad del software, un estado del arte,» Entramado, vol. 13, nº 1, pp. 236-250, 2017.

[9] J. C. Franco Ochoa, Metodología para testing de software basado en componentes, Proyecto de grado, Medellín,: Universidad EAFIT, Ingeniería de Sistemas, 2010.

[10] Choucair Testing, «¿Sabes qué son los ambientes de prueba de desarrollo de software?,» CHOUCAIR, 2020. [En línea]. Available: https://www.choucairtesting.com/a-day-at-the-office/. [Último acceso: 24 mayo 2022].

[11] S. de Telecomunicación, U. Politécnica, D. E. Madrid, J. Manuel, and S. Peño, “ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y Pruebas de Software. Fundamentos y Técnicas,” 2015.

[12] Asociación Colombiana de Empresas de Tecnología e Innovación Financiera, “Estas son las pasarelas de pago para vender de forma segura en su negocio online|Colombiafintech.” https://colombiafintech.co/lineaDeTiempo/articulo/estas-son-las-pasarelas-de-pago-para-vender-de-forma-segura-en-su-negocio-online# (accessed Nov. 08, 2021).

[13] W. P. Cárdenas Mejía y A. C. Petro Martínez, Pasarelas de Pago al Servicio del E-commerce en las Empresas de Streaming Administración en Finanzas y Negocios Internacionales., Montería: Universidad de Cordoba, Facultad de Ciencias Económicas, Jurídicas y Administrativas, 2022.

[14] M. Palonino García, Provisión de servicio de comercio electrónico con soporte a pasarela de pago para verificación del pago a través de terminal lMS. Proyecto fin de carrera, Madrid: Universidad Carlos III de Madrid, Escuela Politécnica Superior, 2010.

[15] “Top 10: las mejores pasarelas de pagos online en Colombia (2021) - Marketing 4 Ecommerce - Tu revista de marketing online para e-commerce.” https://marketing4ecommerce.co/top-10-las-mejores-pasarelas-de-pagos-online-en-colombia/ (Último acceso Nov. 08, 2021).

[16] Wompi, «Tarifas,» Wompi, 2022. [En línea]. Available: https://wompi.co/tarifas/. [Último acceso: 12 julio 2022].

[17] L. Y. Paredes Chaves y J. R. Gonzales Gallego, Tendencias Tecnológicas en E-commerce y su tratamiento contable. Proyecto de grado Contaduría Pública, Facatativa: Universidad de Cundinamarca, 2020.

[18] ePayco, «Camara de Comercio de Medellín,» 2022. [En línea]. Available: https://www.camaramedellin.com.co/DesktopModules/EasyDNNNews/DocumentDownload.ashx?portalid=0&moduleid=569&articleid=615&documentid=403. [Último acceso: 11 julio 2022]

[19] ePayco, «Tarifas,» 2022. [En línea]. Available: https://epayco.com/tarifas/. [Último acceso: 11 julio 2022].

[20] A. T. U. of E. Neha Bansal, «E-wallets and Digitization». may 18, 2020. Último acceso: nov. 21, 2021. [En línea]. Disponible en: https://www.researchgate.net/publication/341452663_E-wallets_and_Digitization.

[21] "¿Qué es un e-wallet y cómo funciona? - Adyen." https://www.adyen.com/es_ES/blog/que-es-ewallet-como-funciona (Último acceso Nov. 08, 2021).

[22] "¿Qué es un e-wallet o billetera electrónica? | MyChoice2Pay." https://www.mychoice2pay.com/es/blog/que-es-ewallet (Último acceso Nov. 08, 2021).

[23] Tamizhvani S. y Saranya Alandur Srinivasan, «Mobile wallet: preferrence and satisfaction of its users' mobile wallet: preferrence and satisfaction of its users'», Department of Commerce Ethiraj - College for Women, 2020. Último acceso: nov. 21, 2021. [En línea]. Disponible en: https://www.researchgate.net/publication/342766629_MOBILE_WALLET_PREFERRENCE_AND_SATISFACTION_OF_ITS_USERS%27_MOBILE_WALLET_PREFERRENCE_AND_SATISFACTION_OF_ITS_USERS%27.

[24] "¿Qué son los servicios delivery y qué características tienen? | Tecnología10." https://tecnologia10.top/que-son-los-servicios-delivery-y-que-caracteristicas-tienen/ (Último acceso Nov. 08, 2021).

[25] "Definición de delivery - Qué es, Significado y Concepto." https://definicion.de/delivery/ (Último acceso Nov. 08, 2021).

[26] P. J. Soldi, Estudio de rendimiento en MongoDB sobre arquitecturas centralizadas y distribuidas. Tesina de Licenciatura, La Plata: Universidad Nacional de la Plata, Facultad de Informática, 2017.

[27] N. Román Seneque, Administración en MongoDB. Trabajo fin de grado, Madrid: Universidad de Valladolid, E.T.S Ingeniería Informática, 2014.

[28] K. Chodorow, Mongo DB The Definitive Guide, New York: O'Reilly Media, 2013.

[29] J. L. Coalla Cencerrado, Let´s get fotness- Desarrollo de una aplicación web con mean stack, Madrid: Universidad Politécnica de Madrid, Escuela Técnica Superior de Ingeniería de Sistemas Informaticos , 2018.

[30] L. G. Correa Real, Análisis comparartivo entre la base datos no relacional MONGODB con la base de datos POSTGRESQL, sistema para la gestión de clientes y registro de pagos clínica odontológica Ortho Dent, Ibarra, Ecuador: Universidad Técnica del Norte, Facultad de Ingeniería en Ciencias Aplicadas, 2015.

[31] Gobierno del Estado de Mexico, «Estrategia para el modelado de datos con MongoDB,» 2020. [En línea]. Available: https://dgsei.edomex.gob.mx/sites/dgsei.edomex.gob.mx/files/files/MongoDB.pdf. [Último acceso: 24 junio 2022].

[32] «Aproveche todo el potencial de MongoDB,» IBM, 2022. [En línea]. Available: https://www.ibm.com/co-es/cloud/databases-for-mongodb. [Último acceso: 24 junio 2022].

[33] C. Civantos Martos, Uso y ventajas de las Elasticsearch en bases de datos no relacionales. Trabajo Fin de Grado en Ingeniería Informática, Madrid: Universidad Autónoma de Madrid, 2019.

[34] D. Turner y Y. Welsch, «Una nueva era para la coordinación de clusters en Elasticsearch,» 16 marzo 2019. [En línea]. Available: https://www.elastic.co/es/blog/a-new-era-for-cluster-coordination-in-elasticsearch.

[35] IONOS Cloud S.L.U., «Elasticsearch: el motor de búsqueda flexible,» IONOS Cloud S.L.U., 2022. [En línea]. Available: https://www.ionos.es/digitalguide/servidores/configuracion/que-es-elasticsearch/. [Último acceso: 24 junio 2022].

[36] C. Carvajal Molinero, Desarrollo de Sistema Centralizado de recolección y correlación de eventos en red enviadas desde sondas Opensource. Trabajo fin de grado en Ingeniería Informática en Ingeniería de los Computadores, Extremadura, España: Universidad de Extremadura, 2018.

[37] Redis, «Redis.io,» 2020. [En línea]. Available: https://redis.io/. [Último acceso: 24 junio 2022].

[38] D. Moreno Bernal y R. González Sanchez, EVALUACIÓN DE SEGURIDAD DE GESTORES DE BASES DE DATOS NOSQL MONGODB, REDIS Y CASSANDRA. Trabajo de grado ingeniería de sistemas, Bogotá: Universidad Católica de Colombia, Facultad de Ingeniería, 2020.

[39] O. A. Vivas Reyes, Migración de Redis a RabbitQM en aplicación web de mantenimiento de enlaces rotos. Trabajo de grado para obtener el título de Ingeniero de Sistemas, Villavicencio: Universidad de los Llanos, Facultad de Ciencias Básicas e Ingeniería, 2020.

[40] L. Pérez, Estudio y análisis del protocolo de mensajería avanzado en el internet de las cosas para aplicación en el campo de la domótica. Trabajo de grado, Guayaquil, Ecuador: Universidad Católica de Santiago de Guayaquil, Facultad de educación técnica, 2019.

[41] V. Zabala Miranda, Desarrollo de una aplicación web utilizando el servidos Nginx en la compañía "Group Tektron", Riobamba, Ecuador: Escuela

Superior Politécnica de Chimborazo, Facultad de Informática y Electrónica, 2016.

[42] G. Bustos, «¿Qué es NGINX y cómo funciona?,» HOSTINGER TUTORIALES, 23 febrero 2022. [En línea]. Available: https://www.hostinger.co/tutoriales/que-es-nginx. [Último acceso: 25 junio 2022].

[43] B. Molina Coronado, Integación de un proxy inverso NGINX con un panel de control Virtualmin para crear una plataforma de hospedaje web. Trabajo fin de grado en Ingeniería Informática, Valencia, España: Universitat Politècnica de València, 2015.

[44] Octavo Labs AG, «Agrupación de MQTT para alta disponibilidad y escalabilidad.,» Octavo Labs AG, 2022. [En línea]. Available: https://vernemq-com.translate.goog/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=sc. [Último acceso: 25 junio 2022].

[45] J. Dompablo Tobar, DevOps para automatización de Gitlab en alta disponibilidad. Trabajo fin de grado en Informática, Madrid: Universidad Autónoma de Madrid, 2018.

[46] El balanceador de carga TCP/HTTP confiable y de alto rendimiento, «El balanceador de carga TCP/HTTP confiable y de alto rendimiento,» 2022. [En línea]. Available: https://www-haproxy-org.translate.goog/?_x_tr_sch=http&_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es-419&_x_tr_pto=op,sc. [Último acceso: 29 junio 2022].

[47] E. Pons Ballester, Inarts, la red social de contacto entre artistas y editoriales. Trabajo Fin de Grado Grado en Ingeniería Informática, Valencia, España: Universidad Politécnica de Valencia, Escola Tècnica Superior d'Enginyeria Informàtica, 2018

[48] G. Y. Salina Tomalá, Desarrollo de una aplicación web para el proceso de generación de órdenes de compra y venta de equipos fabricados en la empresa Baursa. Trabajo de grado en Ingeniería en tecnología de la información, La Libertad, Ecuador: Universidad Estatal Península de Santa Elena, 2021.

[49] E. . M. Salas González, Aplicando seguridad a una API REST con JSON Web Tokens. Trabajo fin de Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones MISTIC, Cataluña: Universidat Oberta de Catalunya, 2020

[50] M. Pérez Herrera, Arquitecturas basadas en microservicios. Trabajo fin de grado, Madrid, España: Universidad Politécnica de Madrid, 2015.

[51] J. R. León Gonzáles, Arquitectura basada en microservicios para un sistema de producción de hidrocarburos en la empresa BCL Venezuela, C.A. Trabajo

especial de grado, Puerto Ordaz: Universidad Católica Andrés Bello, Facultad de Ingeniería, 2021.

[52] F. F. S. García Victoria y F. E. Daza Gonzalez, Diseño y construcción de un componente para el generador de código ZathuraCode que genere proyectos con arquitectura basada en microservicios en Java. Trabajo de Grado presentado para optar al título de Ingeniero de Sistemas, Santiado de Cali: Universidad San Buenaventura, Facultad de Ingeniería, 2017

[53] A. Huerta Tajuelo, Transición del monolito de los servicios: cambio de paragidma en el desarrollo de aplicaciones. Trabajo Fin de Grado en Ingeniería en Tecnologías de Telecomunicación, Madrid, España: Universidad Pontificia Comillas, 2020.

[54] Amazon Web Services, «¿Qué son los microservicios?,» AWS, 2022. [En línea]. Available: https://aws.amazon.com/es/microservices/. [Último acceso: 13 julio 2022].

[55] M. P. Herrera Cuadrillero, Arquitecturas basadas en microservicios, Madrid: Universidad Politécnica de Madrid, 2015.

[56] Sociedad Argentina de Informática, «Arquitectura de microservicios distribuidos para una plataforma que orquesta actividades orientadas a la recolección de datos con intervención humana,» S.F.. [En línea]. Available: https://49jaiio.sadio.org.ar/pdfs/est/EST-05.pdf. [Último acceso: 12 junio 2022].

[57] O. Lizama, G. Kindley y J. . I. Jeria Morales, «Redes de computadores Arquitectura- Cliente Servidor,» 2018. [En línea]. Available: http://profesores.elo.utfsm.cl/~agv/elo322/1s16/projects/reports/Proyecto%20Cliente%20-%20Servidor.pdf.

[58] «What Is API Architecture? | Akana by Perforce», *Akana*. https://www.akana.com/blog/api-architecture (Último acceso 19 de enero de 2022).

[59] C. Cristián, Arquitectura Cliente-Servidor/Cliente-Servidor en Aplicaciones de Misión Crítica. Trabajo Final de Grado Ingeniería de Sistemas, Cordoba: Instituto Universitario Aeronáutico, Facultad de Ciencias de la Administración, 2017.

[60] Junta de Andalucía, «Marco de desarrollo de la Junta de Andalucía,» 2020. [En línea]. Available: https://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/arquitectura/capa-presentacion. [Último acceso: 13 julio 2022].

[61] A. Canico Hernández y O. Cortés Sánchez, «Programación en Capas,» Instituto Tecnológico Superior de Ciudad Serdán, Puebla, México,, 5 febero

2021. [En línea]. Available: https://itsciudadserdan.edu.mx/programacion-en-capas/. [Último acceso: 13 julio 2022].

[62] https://hapi.dev/ (Último acceso 21 de enero de 2022)

[63] «Snapshot». Último acceso: 1 de febrero de 2022. [En línea]. Disponible en: https://hapi.dev/

[64] «MongoDB Documentation». https://docs.mongodb.com/ (Último acceso 1 de febrero de 2022)

[65] «Capa gratuita de AWS | Cloud computing gratis |AWS», Amazon Web Services, Inc. https://aws.amazon.com/es/free/ (Último acceso 3 de marzo de 2022).

[66] «Tipos de instancias de Amazon EC2 - Amazon Web Services», Amazon Web Services, Inc. https://aws.amazon.com/es/ec2/instance-types/ (Último acceso 17 de marzo de 2022)

[67] «Official Ubuntu Documentation». https://help.ubuntu.com/ (Último acceso 10 de marzo de 2022)

[68] «Kibana: Explora, visualiza y descubre datos», *Elastic*. https://www.elastic.co/es/kibana (Último acceso 23 de marzo de 2022).

[69] «Tunel SSH | Facultad de Ingeniería». https://www.fing.edu.uy/es/sysadmin/configuraciones/t%C3%BAnel-ssh (Último acceso 23 de marzo de 2022).

[70] «Instale e implemente redis en Ubuntu, y configúrelo para arrancar (activar) - programador clic». https://programmerclick.com/article/1739802382/ (Último acceso 23 de marzo de 2022).

[71] «Creating a Virtual Environment for Python on Ubuntu 16.04 - Liquid Web». https://www.liquidweb.com/kb/creating-virtual-environment-ubuntu-16-04/ (Último acceso 24 de marzo de 2022).

[72] «Pasos para recibir pagos del exterior con Wompi». https://www.bancolombia.com/wps/portal/empresas/capital-inteligente/actualidad-economica-sectorial/recibir-pagos-del-exterior-con-wompi (Último acceso 5 de abril de 2022)

[73] «Inicio rápido · Wompi Docs». https://docs.wompi.co/ (Último acceso 31 de marzo de 2022).

[74] Wompi, «Plan de información,» 2022. [En línea]. Available: https://wompi.co/. [Último acceso: 11 julio 2022].

[75] ePayco, «Funcionalidades,» 2022. [En línea]. Available: https://epayco.com/pagos/. [Último acceso: 11 julio 2022].

[76] L. Montoya, «Los pros y los contras de las pasarelas de pago en Colombia,» Tudatos.co, 16 junio 2020. [En línea]. Available: https://www.tusdatos.co/blog/los-pros-y-los-contras-de-las-pasarelas-de-pago-en-colombia. (Último acceso: 12 de abril 2022)

[77] D. Millan, «Las 5 mejores pasarelas de pago en Colombia,» 3 julio 2021. [En línea]. Available: https://www.davidmillan.co/pasarelas-de-pago-colombia/#:~:text=A%20diferencia%20de%20Wompi%2C%20podr%C3%A1s,Pagos%20por%20PSE. (Último acceso: 12 de abril2022).

[78] M. Urrego Álvarez y E. J. Yepes Sanchez, Transformación Digital de la Banca: Modelo basado en Machine Learning para la clasificación de transacciones bancarias realizadas a través de PSE, Medellín: Institución Universitaria Tecnológico de Antioquia, Facultad de Ingeniería, 2021.

[79] Unidad de Regulación Financiera, «Estudio sobre los sistemas de pago de bajo valor y su regulación,» junio 2018. [En línea]. Available: https://www.urf.gov.co/webcenter/ShowProperty?nodeId=%2FConexionContent%2FWCC_CLUSTER-106608%2F%2FidcPrimaryFile&revision=latestreleased. (Último acceso: 15 de abril 2022).

[80] Fundación Universitaria de Ciencias de la Salud, «¿Qué ES PSE?,» enero 2017. [En línea]. Available: https://www.fucsalud.edu.co/sites/default/files/2017-01/QUE-ES-PSE-Y-PREGUNTAS-FRECUENTES.pdf. (Último acceso: 20 de abril 2022).

[81] ACH Colombia, «Requisitos básicos para implementar PSE,» 2020. [En línea]. Available: https://www.pse.com.co/empresa-adquiere-pse-para-tu-negocio#Agregador_de_pagodpgk_51. (Último acceso: 22 de abril 2022).

[82] PSE, «¿Qué es PSE?,» 2020. [En línea]. Available: https://www.pse.com.co/documents/1176700/1193759/Brochure_pse.pdf/e2b0d8ed-dda7-279a-c9df-61687bddf81c?t=1637770867999. (Último acceso: 23 de abril 2022).

[83] Seguros del Estado y Seguros de Vida del Estado - segurosdelestado.com, segurosdevidadelestado.com. (s. f.-b). segurosdelestado.com. https://www.segurosdelestado.com/pages/SOAT_Tarifas. (Último acceso: 12 mayo 2022).