Universidad Hispanoamericana

ESCUELA DE INGENIERÍA INFORMÁTICA

TESIS PARA OPTAR POR EL GRADO DE BACHILLERATO, EN LA CARRERA DE INGENIERÍA INFORMÁTICA

Servicio de trazabilidad y transparencia en cadenas de suministro y productores agrícolas

Sustentante:

Lirroy José Coto Obando

Tutor:

Ing. Cristian Paz Campos Agüero

Febrero, 2022

Tabla de contenido

Índice de ilustraciones

Índice de tablas

Carta de autorización

Martes, 16 de marzo de 2021

Escuela de Ingeniería de Sistemas

Facultad de Tecnología de Información y Comunicación

Universidad Hispanoamericana de Heredia

Estimados,

El objetivo de la presente es para confirmar la aceptación del proyecto de graduación de Lirroy José Coto Obando portador de la cedula No. 1-1524-0724, estudiante de la Universidad Hispanoamericana de Heredia, en la especialidad de Ingeniería en Informática, en la Red de Agricultura Sostenible Costa Rica, S.A. Dicho proyecto estará iniciando en el mes de abril y tiene un periodo de duración no mayor a 365 días.

El estudiante estará bajo la supervisión de Enzo Jiménez Herra con el rol de Gerente de IT y desarrollará el proyecto titulado: Servicio de trazabilidad y transparencia en cadenas de suministro y productores agrícolas por medio del uso de cadenas de bloques. Incluyendo todas las actividades definidas en el plan de trabajo.

Atentamente,

ENZO FERNANDO JIMENEZ HERRA (FIRMA)

Digitally signed by ENZO FERNANDO JIMENEZ HERRA (FIRMA)
Date: 2021.03.24 17:50:04 -06'00'

Enzo Jiménez Herra

Gerente de TI

JOSE JOAQUIN CAMPOS ARCE (FIRMA)

Firmado digitalmente por JOSE JOAQUIN CAMPOS ARCE (FIRMA)
Fecha: 2021.03.24 20:08:39 -06'00'

José Joaquín Campos

Director Ejecutivo

Carta de aprobación del tutor

CARTA DEL TUTOR

San José,05 de noviembre del 2021

Señora:
María Isabel Losilla Barrientos
Directora de Carrera
Ingeniería Informática
Universidad Hispanoamericana

Estimada señora:

El estudiante Lirroy José Coto Obando**,** cédula de identidad número 1-1524-0724, me ha presentado, para efectos de revisión y aprobación, el trabajo de investigación denominado *Servicio de trazabilidad y transparencia en cadenas de suministro y productores agrícolas,* el cual ha elaborado para optar por el grado académico de Bachillerato.

En mi calidad de tutor, he verificado que se han hecho las correcciones indicadas durante el proceso de tutoría y he evaluado los aspectos relativos a la elaboración del problema, objetivos, justificación; antecedentes, marco teórico, marco metodológico, tabulación, análisis de datos; conclusiones y recomendaciones.

De los resultados obtenidos por el postulante, se obtiene la siguiente calificación:

a)	ORIGINAL DEL TEMA	10%	10
b)	CUMPLIMIENTO DE ENTREGA DE AVANCES	20%	20
C)	COHERENCIA ENTRE LOS OBJETIVOS, LOS INSTRUMENTOS APLICADOS Y LOS RESULTADOS DE LA INVESTIGACION	30%	30
d)	RELEVANCIA DE LAS CONCLUSIONES Y RECOMENDACIONES	20%	20
e)	CALIDAD, DETALLE DEL MARCO TEORICO	20%	20
	TOTAL		100

En virtud de la calificación obtenida, se avala el traslado al proceso de lectura.

Atentamente,
CRISTIAN PAZ CAMPOS AGUERO (FIRMA)
Firmado digitalmente por CRISTIAN PAZ CAMPOS AGUERO (FIRMA)
Fecha: 2021.11.05 21:41:35 -06'00'

Ing. Cristian Campos Agüero
Cédula de residencia 160400100307
Carné CPIC 3568

Carta de aprobación del tutor

CARTA DE LECTOR

Universidad Hispanoamericana
Carrera Ingeniería Informática

Dirección

El estudiante Lirroy José Coto Obando, cédula de identidad: 1-1524-0724, me ha presentado la Tesina para efectos de revisión y aprobación, denominada "**Servicio de trazabilidad y transparencia en cadenas de suministro y productores agrícolas**", el cual ha elaborado para obtener su grado de Bachillerato en Ingeniería Informática.

He revisado y he hecho las observaciones relativas al contenido analizado, particularmente lo relativo a la coherencia entre el marco teórico y análisis de datos, la consistencia de los datos recopilados y la coherencia entre éstos y las conclusiones; así mismo, la aplicabilidad y originalidad de las recomendaciones, en términos de aporte de la investigación. También se realizaron las modificaciones solicitadas a nivel de contenido y forma.

Por consiguiente, este trabajo cuenta con mi aval para ser presentado en la defensa pública.
Atte.

Firma 

Nombre Ing. Luis Navarro S

Cédula 2-0484-0537

Declaración jurada

DECLARACIÓN JURADA

Yo **Lirroy Coto Obando**, mayor de edad, portador de la cédula de identidad número **1-1524-0724** estudiante de la carrera de Ingeniería de Software de la Universidad Hispanoamericana, hago constar por medio de este acto y debidamente apercibido y entendido de las penas y consecuencias con las que se castiga en el Código Penal el delito de perjurio, ante quienes se constituyen en el Tribunal Examinador de mi trabajo de tesis para optar por el título de Bachillerato, juro solemnemente que mi trabajo de investigación titulado: **Servicio de trazabilidad y transparencia en cadenas de suministro y productores agrícolas**, es una obra original que ha respetado todo lo preceptuado por las Leyes Penales, así como la Ley de Derecho de Autor y Derecho Conexos número 6683 del 14 de octubre de 1982 y sus reformas, publicada en la Gaceta número 226 del 25 de noviembre de 1982; incluyendo el numeral 70 de dicha ley que advierte; articulo 70. Es permitido citar a un autor, transcribiendo los pasajes pertinentes siempre que éstos no sean tantos y seguidos, que puedan considerarse como una producción simulada y sustancial, que redunde en perjuicio del autor de la obra original. Asimismo, quedo advertido que la Universidad se reserva el derecho de protocolizar este documento ante Notario Público. En fe de lo anterior, firmo en la ciudad de Heredia, a los 10 días del mes de Noviembre del año dos mil veintiuno.

Firma del estudiante

Cédula: 115240724

Autorización de publicación

UNIVERSIDAD HISPANOAMERICANA
CENTRO DE INFORMACION TECNOLOGICO (CENIT)
CARTA DE AUTORIZACIÓN DE LOS AUTORES PARA LA CONSULTA, LA REPRODUCCION PARCIAL O TOTAL Y PUBLICACIÓN ELECTRÓNICA DE LOS TRABAJOS FINALES DE GRADUACION

San José, 09 de abril 2022

Señores:
Universidad Hispanoamericana
Centro de Información Tecnológico (CENIT)

Estimados Señores:

El suscrito (a) Liroy Coto Obando con número de identificación 1-1524-0724 autor (a) del trabajo de graduación titulado Servicio de trazabilidad y transparencia en cadenas de suministro y productores agrícolas presentado y aprobado en el año 2022 como requisito para optar por el título de Bachillerato Ing. Informática, (SI / NO) autorizo al Centro de Información Tecnológico (CENIT) para que con fines académicos, muestre a la comunidad universitaria la producción intelectual contenida en este documento.

De conformidad con lo establecido en la Ley sobre Derechos de Autor y Derechos Conexos N° 6683, Asamblea Legislativa de la República de Costa Rica.

Cordialmente,

1-1524-0724
Firma y Documento de Identidad

CAPÍTULO I: PROBLEMA DEL PROYECTO

1.1 Antecedentes y justificación del Proyecto

1.1.1 Marco de Referencia Empresarial y Contextual

1.1.1.1 Historia de RAS (Red de Agricultura Sostenible)

La RAS nació hace más de 20 años como una coalición de organizaciones conservacionistas sin fines de lucro en América, África, Europa y Asia.

La RAS es una asociación de organizaciones no gubernamentales que trabajan para la protección de la biodiversidad y el desarrollo humano sostenible. Su secretaría cuenta con sedes en Estados Unidos, Reino Unido y Costa Rica.

Por más de dos décadas la RAS ha mejorado las vidas y las prácticas de producción de más de 1,2 millones de productores en todo el mundo. La RAS ha ayudado a transformar la agricultura en más de 3,3 millones de hectáreas certificadas, especialmente en las regiones tropicales y subtropicales del planeta. Su concepto de sostenibilidad reconoce que el bienestar de las sociedades y el de los ecosistemas están entrelazados y dependen de la conservación ambiental, así como de un desarrollo socialmente justo y económicamente viable.

Por muchos años han trabajado para alcanzar su visión a través del lente de la certificación agrícola y trabajando en colaboración con Rainforest Alliance. La RAS al poner en práctica los mecanismos de certificación, ha reconocido el valor y los múltiples beneficios de esta herramienta, pero también han sido

conscientes de las limitaciones inherentes. Los desafíos que el mundo enfrenta son enormes y están creciendo rápidamente; han concluido que, para cumplir mejor su misión, necesitan evolucionar. Ahora se han movido más allá de la certificación y trabajan directamente con empresas, donantes y otras organizaciones para acelerar y profundizar el impacto positivo que pueden hacer como socios, juntos en un viaje de cambio.

La RAS siempre ha sido una organización única. Fueron formados cuando un grupo de ONG de diferentes países y continentes decidieron trabajar juntas basadas en la convicción de que las prácticas agrícolas sostenibles podían ser aprovechadas como un eficaz instrumento para la protección del medio ambiente y para mejorar la vida de la población rural.

La RAS siempre ha sido una organización visionaria. Dos décadas atrás, introdujeron el concepto de los “Tres pilares de la sostenibilidad” para la certificación, redefiniendo la sostenibilidad como algo que incluye no sólo la protección del medio ambiente o el bienestar de los trabajadores, sino que también promueve y mejora la economía de las operaciones agrícolas.

(Red de Agricultura Sostenible, 2021)

1.1.1.2 Misión

La misión de la RAS es ser una red global que transforma la agricultura para asegurar un futuro sostenible de los alimentos, la naturaleza y las comunidades.

Los desafíos que el planeta enfrenta son enormes y crecen rápidamente; desde el cambio climático y la destrucción de la biodiversidad, hasta el trabajo infantil y los derechos laborales; el mundo se encuentra en riesgo de muchas maneras. En la RAS existen oportunidades claras para transformar la agricultura desde las fincas y a través de las cadenas de abastecimiento asociadas. La RAS es un acelerador del cambio positivo que se necesita.

La RAS es una red global de ONGs enfocada en ayudar a los productores, empresas y donantes a avanzar con sus agendas de sostenibilidad de manera práctica y eficiente. Son un aliado poderoso y eficaz para alcanzar y monitorear metas, transformar prácticas agrícolas y crear valor en el terreno.

1.1.1.3 Visión

La RAS se enorgullece de ser una organización liderada por miembros que combina el alcance global y local para tener un impacto colectivo. Están comprometidos con actuar acorde a sus valores en todo lo que hacen. Son transparentes, confiables, innovadores, colaborativos y buscan defender la mejora continua. Sus servicios se caracterizan por su calidad, flexibilidad y rentabilidad.

(Red de Agricultura Sostenible, 2021)

1.1.1.4 Gobernanza de la RAS

Los miembros de la RAS se agrupan en dos grupos y tres categorías de acuerdo con la naturaleza de su trabajo y el papel que desempeñan dentro de la red. La estructura está diseñada para facilitar la interacción entre grupos y con la Secretaría.

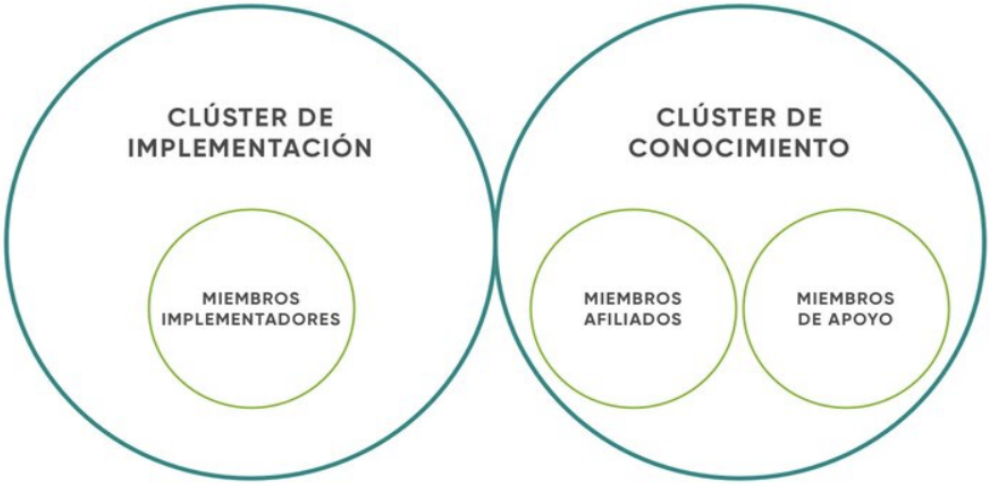


Figura 1 Clúster y categorías de la RAS

FUENTE: Sitio Web: https://www.agriculturasostenible.eco/gobernanza (Red de Agricultura Sostenible, 2021)

1.1.1.5 Clúster de Implementación

El clúster de implementación se centra en el trabajo de campo agrícola. Está compuesto por ONG locales e internacionales, de conservación ambiental y de enfoque social, con experiencia demostrada en agricultura.

Los miembros de este clúster son organizaciones sin fines de lucro únicamente, capaces de implementar proyectos y otras actividades en el campo. Tienen contacto directo con los agricultores y tienen un fuerte enfoque en el apoyo.

Los miembros implementadores tienen un papel clave en el desarrollo de propuestas de proyectos y su implementación, y son parte de la gobernanza como miembros de la Asamblea General y el Consejo Directivo. Pueden participar en todas las actividades y plataformas RAS.

1.1.1.6 Clúster de Conocimiento

El Clúster de Conocimiento se centra en el intercambio de información y experiencias, facilitando un diálogo abierto hacia la innovación y brindando apoyo en los esfuerzos de promoción e incidencia.

Este clúster está integrado por todo tipo de organización, que son voces líderes en el mundo agrícola y actúan impulsadas por una misión. El clúster agrupa a los Miembros Afiliados y de Apoyo.

1.1.1.7 Cuerpos de gobierno de la RAS

La RAS trabaja utilizando un proceso participativo de toma de decisiones con todos los actores. Esta estructura permite la creación de consensos confiables y alienta debates democráticos, inclusivos y transparentes. Los órganos rectores son:

- Asamblea General: La Asamblea General está compuesta por todos los miembros de la red y es la autoridad suprema de la RAS. Tiene un representante por cada una de las organizaciones miembros y elige al Consejo Directivo.
- Consejo Directivo: La RAS es administrada por un Consejo Directivo, compuesto por un máximo de 11 representantes de los miembros elegidos por la Asamblea General. El Consejo Directivo aprueba los planes anuales, los objetivos y las estrategias de la organización.
- Secretaría: La Secretaría ejecuta las decisiones tomadas por el Consejo Directivo de la RAS. Sus funciones principales son administrar las operaciones diarias de la organización, facilitar la comunicación, promover la cooperación con las iniciativas existentes, coordinar las actividades de apoyo y supervisar la implementación eficiente de los servicios y proyectos de la RAS.

1.1.1.8 Junta Directiva de la RAS en Costa Rica

Según (Red de Agricultura Sostenible, 2021) la junta directiva de la RAS en Costa Rica se constituye de la siguiente manera:

- Director Ejecutivo: José Joaquín Campos Arce
- Director Técnico y Desarrollo: Mona McCord
- Director de Servicios Corporativos: Carlos Cortés Tormo
- Gerente de Manejo de Proyectos: Catalina Mora
- Gerente de Innovación y Conocimiento: Oliver Bach
- Coordinador Técnico: Edwin Vargas
- Especialista Técnica: Jana Dietershagen
- Gerente de TI: José Rodríguez
- Especialista de TI: Enzo Jiménez
- Coordinador de Programas en India: Prashanth Muniyappa
- Contadora: Xinia Morales
- Asistente Ejecutiva: Catalina Rivera

1.1.1.9 Objetivos de la RAS

Según el plan estratégico (Red de Agricultura Sostenible, 2021), la RAS se define por los siguientes objetivos:

1. Expandir y fortalecer una red global para asegurar un futuro sostenible de los alimentos, la naturaleza y las personas.

2. Contribuir a lograr cadenas de valor sostenibles y productores y comunidades rurales prósperas.

3. Desarrollar aplicaciones innovadoras y prácticas de soluciones tecnológicas en agricultura.

4. Asegurar que la producción agrícola contribuya a sistemas alimentarios que sean saludables para las personas y el planeta.

1.1.2 Justificación del proyecto

Como parte de uno de los objetivos definido en (Red de Agricultura Sostenible, 2021) la RAS requiere mejorar la transparencia sobre el origen de los productos agrícolas mediante el desarrollo de soluciones innovadoras y tecnológicas.

Como parte de sus procedimientos la RAS vela por la certificación de la producción agrícola, para que exista un desarrollo armónico entre el medio ambiente y las comunidades, pero se han encontrado dificultades para para poder garantizar y trazar cada producto que sale de estas áreas de producción, llega al mercado y es adquirido por el consumidor (E. Jiménez, comunicación personal, 11 de marzo de 2021).

Desde Julio de 2017 la RAS ha utilizado un sistema web en PHP, elaborado por el equipo de tecnologías de información de la RAS, que genera de manera dinámica una plantilla en Excel, esta plantilla contiene:

- Los metadatos requeridos para dar significado e identidad a los datos exportados.
- Hojas ocultas para la configuración de idiomas, ya que soporta inglés, español, francés y portugués.
- Hoja con los criterios de evaluación, los cuales se dividen en áreas, metas, objetivos e indicadores subsecuentemente. Lo que da como resultado un cuestionario jerárquico donde se debe indicar si el campo agrícola en estudio cumple, no cumple o no aplica con un criterio de evaluación.

- Hoja con la información general de la evaluación que se está realizando, en esta hoja se completan datos tales como:
    - Área de la finca auditada
    - Área de producción
    - Cantidad de empleados (Hombres y mujeres por separado)
    - Uso de la tierra
    - Consumo de combustible
    - Consumo de electricidad
    - Si es ganadería, el tipo de animales que se tiene
    - Si es cultivo, que tipo de cultivo es. Ejemplo, café, banano, maíz.
    - Estimación de producción.

Una vez que se recopila la información descrita anteriormente el sistema permite cargar la plantilla de Excel para poder almacenar la información en una base de datos.

Importante mencionar, que se utilizaron las tecnologías de PHP y hojas de Excel, dado que los usuarios de esta herramienta difícilmente contaran con acceso a internet en los campos agrícolas, por lo que esta plataforma implementa una solución para trabajo fuera de línea.

A pesar de que este desarrollo ha sido de mucha utilidad, no se ha logrado el objetivo de contar con un servicio que permita firmar, legitimar y trazar cada transacción que gira en torno a la certificación de la producción agrícola, dado que

el alcance de esta herramienta es de garantizar la persistencia de la información y brindar una herramienta para captura de datos para la certificación agrícola.

(E. Jiménez, comunicación personal, 05 de julio de 2021)

1.2 Definición del problema

1.2.1 Problemática

Como parte de su objetivo primordial, la RAS busca controlar y regular la producción agrícola basado en el desarrollo armónico ambiental y social, dado que a nivel mundial se ha dado un crecimiento desmedido de la población y los campos de producción agrícola se han reemplazado por civilizaciones, según (E. Jiménez, comunicación personal, 11 de marzo de 2021) esto ha traído consigo que los diferentes productores agrícolas utilicen técnicas no aprobadas tales como:

- Uso de terrenos no certificados.
- Utilización productos artificiales no saludables.
- Explotación de la mano de obra agrícola.

Lo descrito anteriormente, trae consigo 3 agentes fundamentales que requieren ser mejorados con el uso de la tecnología para tener un mayor control y trazabilidad de la producción:

- Auditorías iniciales del campo agrícola.
- Control sobre la transacción de venta producción agrícola.
- Control sobre la compra de dicha producción.

La evaluación de tierras es el medio por el cual un agente de la RAS certifica que un área agrícola especifica sigue los estándares y reglas de esta entidad, en esta certificación se establece una de las principales variables que no ha podido ser salvaguardada y respetada a lo largo del tiempo, la cual es la cantidad permitida de producción agrícola, esto significa que muchos de los productores a nivel mundial sobrepasa estas cantidades, por lo tanto, a nivel de mercado se vende producción agrícola no certificada, como si lo fuera.

Una vez que la producción agrícola es completada, lo siguiente que se requiere mejorar es el proceso de venta y el proceso de compra de la producción agrícola, según (E. Jiménez, comunicación personal, 11 de marzo de 2021) identifican como regiones productoras:

- África
- Asia
- Suramérica
- Centroamérica

Por otro lado, las regiones compradoras se localizan en:

- Estados Unidos
- Europa

Para controlar las transacciones de venta y compra, no existe un sistema de software que permita vincular el proceso inicial de evaluación a los procesos de

compra y venta, por lo que fácilmente los vendedores y compradores pueden cometer violaciones a procesos y trayendo al consumidor productos sin alguna procedencia conocida.

1.2.2 Diagrama Causa – Efecto

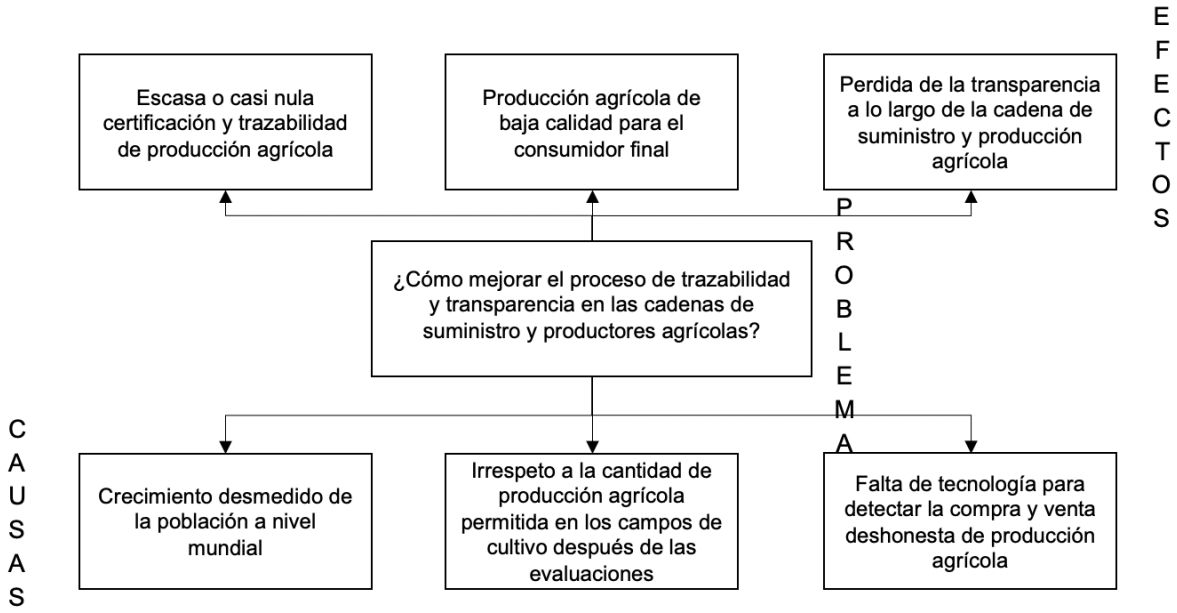


Figura 2 Diagrama Causa-Efecto

FUENTE: Diagrama creado con la información suministrada por la RAS.

1.2.3 Problema General

¿Cómo mejorar el proceso de trazabilidad y transparencia en las cadenas de suministro y productores agrícolas?

1.2.4 Problemas específicos

1. ¿Cómo mejorar la trazabilidad en las transacciones de auditoria en los campos de producción?
2. ¿Cómo mejorar la transparencia en las transacciones de venta de la producción agrícola?
3. ¿Cómo se puede dar un mejor seguimiento de las transacciones de compra de la producción agrícola?
4. ¿Cómo se puede dar visibilidad, de las diferentes transacciones que suceden en la cadena de suministro de producción agrícola?

1.3 Objetivo General y específico

1.3.1 Objetivo general

Implementar un servicio que permita el registro de los procesos de evaluación, compra y venta en cadenas de suministro y productores agrícolas, mediante la incorporación de la tecnología de *Blockchain*.

1.3.2 Objetivos específicos

1. Investigar acerca de los beneficios del uso del *Blockchain* y los elementos de arquitectura requeridos para implementar esta tecnología como solución al problema de investigación.
2. Definir el conjunto de requerimientos para el almacenamiento y operatividad de la transacción de evaluación, compra y venta agrícola, dentro de cada bloque cifrado de *Blockchain*.
3. Diseñar un diagrama de infraestructura que permita unificar todos los servicios necesarios en la nube para poder tener una solución que cumpla con el requerimiento de software.
4. Programar un API REST que permita la creación y obtención de bloques en el servicio de *Blockchain* incluyendo las pruebas unitarias requeridas para asegurar el correcto funcionamiento del API.
5. Implementar un plan piloto de lanzamiento de la solución, que permita hacer uso del microservicio para efectos de pruebas y mantenimientos.

1.4 Alcances y limitaciones

1.4.1 Alcance del proyecto

1. El primer entregable será llevar a cabo una investigación de la tecnología *Blockchain*, en este caso orientada a servicios en la nube de Microsoft Azure, ya que este es el servicio en la nube contratado por la RAS.
2. El segundo entregable estará relacionado al levantamiento de requerimientos para la definición de los datos que se almacenarán por cada transacción de auditoría, compra y venta agrícola.
3. El tercer entregable será un diagrama de arquitectura en la nube que incluya todos los diversos servicios requeridos para poder soportar un API REST y que este a su vez pueda consumir un servicio de *Blockchain*.
4. El cuarto entregable será la programación de un API que contenga los estándares REST, este entregable contempla:
    a. Configuración y establecimiento del servicio *Blockchain* utilizando servicios en la nube.
    b. Configuración de un API REST con la arquitectura de Microservicio.
    c. Programación del proceso de autenticación y autorización del API.
    d. Creación de los puntos de acceso para la creación de bloques en *Blockchain*.
    e. Creación de los puntos de acceso para la lectura de los bloques de *Blockchain* existentes.
    f. Programación de pruebas unitarias para garantizar el correcto funcionamiento del API.

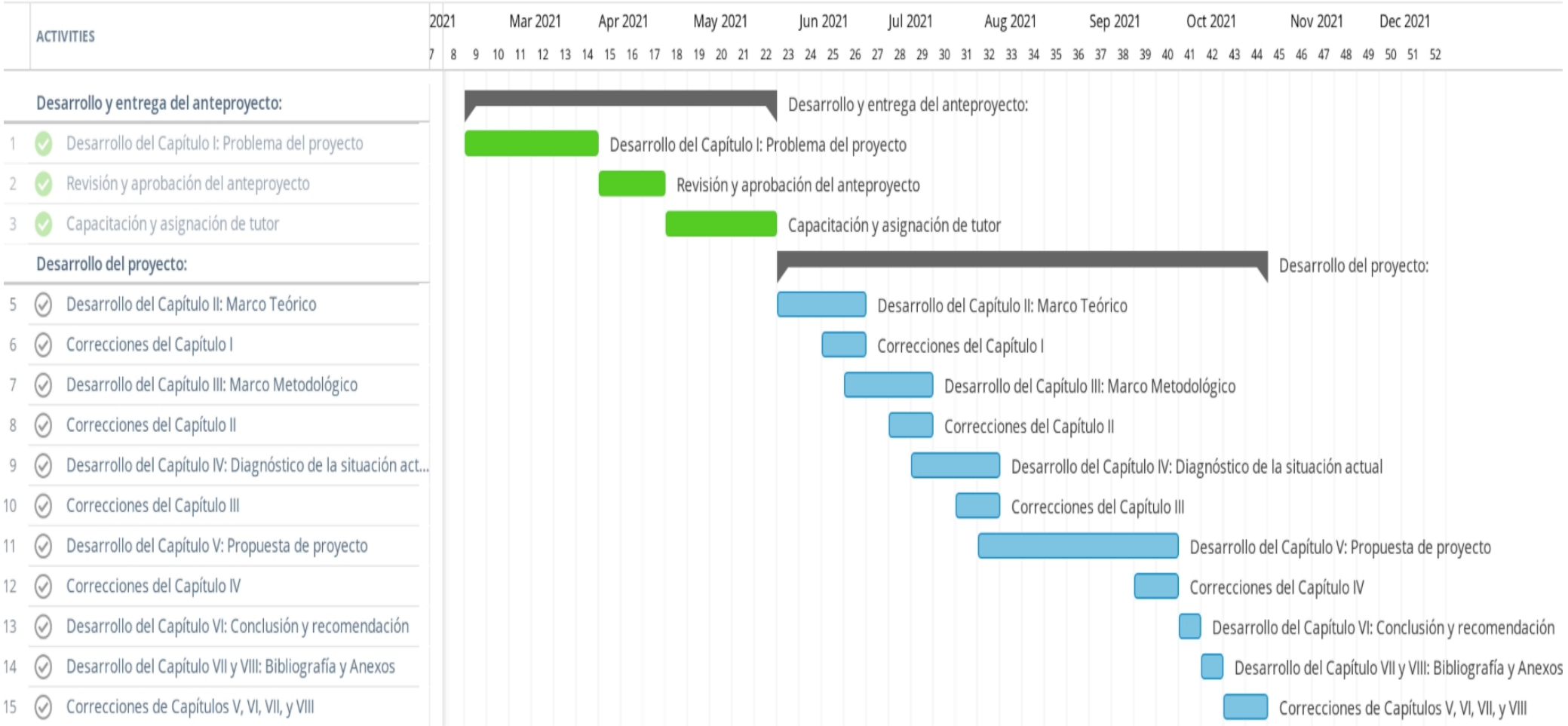
g. Documentación del API para orientar posibles clientes del API.

5. El quinto entregable será el diseño de un plan de piloto de lanzamiento del microservicio, este entregable contempla:
    a. Capacitación técnica al equipo de aplicaciones de la RAS, para poder recorrer la solución completa y puedan hacer el mantenimiento de la solución.
    b. Recomendaciones a futuro para la solución.
    c. Lanzamiento del microservicio en la infraestructura en la nube de Microsoft, a manera de prueba.
6. Estará fuera del alcance cualquier mejora a las aplicaciones existentes de la RAS para que puedan hacer uso del microservicio de *Blockchain*, esto ya será integrado por el equipo de TI (Aplicaciones) de la RAS en el futuro.

1.4.2 Limitaciones del proyecto

1. El uso de librerías de terceros para la programación del API, no pueden ser utilizadas sin antes no haber pasado por el proceso de aprobación de-seguridad de la RAS.
2. Los recursos y servicios de hardware que pueden ser utilizados para cualquier requerimiento del software, serán proveídos por Microsoft Azure, por lo tanto, no se podrá hacer uso de ningún otro proveedor en términos de infraestructura.

1.5 Cronograma del proyecto



CAPÍTULO II: MARCO TEÓRICO

Este capítulo tiene como objetivo definir y desarrollar de una manera integral y secuencial los conceptos necesarios para entender la investigación efectuada.

Todos los conceptos estarán alineados a la carrera de Ingeniería Informática y buscarán de manera objetiva construir vínculos fundamentados con el tema de investigación, mediante la comparación de visiones de diferentes autores.

Adicionalmente, como eje tecnológico principal de esta investigación se desarrollará el concepto de *Blockchain* en la actualidad y cuáles han sido algunos de los más claros ejemplos donde se ha utilizado dicha tecnología.

2.1 Ingeniería

Desde tiempos inmemorables, el ser humano ha buscado la forma de poder solucionar los diferentes problemas y retos que se presentan en el día a día, dando paso a su creatividad e ingenio para poder establecer soluciones que le permitan evolucionar con el tiempo.

Según (Rojas López, 2011, p. 15) define la ingeniería como:

> "La manera que se tiene para llevar las ideas a hechos concretos por medio del conocimiento científico (…)".

La cita anterior hace referencia a la capacidad e ingenio que con el tiempo el ser humano ha logrado desarrollar para implementar objetivamente soluciones que permitan mejorar la humanidad, y no solo llegar a posibles conclusiones u opciones

de como solventar dichos problemas, al contrario, se base en llevar a cabo un plan estratégico de solución y su implementación.

Para un mejor entendimiento del concepto de la ingeniería es importante repasar sus objetivos, para ello se tomará como referencia a (Rojas López, 2011, pp. 17-18) :

- Usar herramientas tecnológicas para garantizar el progreso en la investigación, para ello se debe buscar el desarrollo analítico como cualidad en el profesional.
- Promover el desarrollo de la versatilidad y adaptación del profesional, para que pueda responder a cualquier situación o ambiente, siempre orientado al cumplimiento de objetivos.
- Buscar el avance objetivo en los proyectos mediante equipos multidisciplinarios, es decir, busca cambiar el individualismo por el trabajo en equipo.
- Promover la constante capacitación, aprendizaje del profesional. Esto a su vez promueve el uso de tecnologías que facilitara al profesional emplearse en diferentes ámbitos laborales.
- Buscar la formación en la planeación, con esto el profesional deberá anticipar el cambio. Sumado a esto el profesional deberá desarrollar sus capacidades de coordinación, resolución de problemas y manejo de presupuesto financiero para la búsqueda de resultados.

2.2 Ingeniería de Software

El desarrollo de software como rama de la ingeniería se define según (Falgueras, 2003, p. 15) como:

> "(…) es un conjunto integrado de programas que en su forma definitiva se pueden ejecutar, pero comprende también las definiciones de estructuras de da- tos (por ejemplo, definiciones de bases de datos) que utilizan estos programas y también la documentación referente a todo ello (tanto la documentación de ayuda en el uso del software para sus usuarios como la documentación generada durante su construcción, parte de la cual también servirá para su mantenimiento posterior)".

La definición anterior permite describir desde la perspectiva de software, una opción de solución de problemas de la vida cotidiana. La construcción de software a su vez engloba y aplica todos objetivos anteriormente descritos como parte de la ingeniería.

2.2.1 Fases de desarrollo de software

En la actualidad se pueden encontrar diferentes procedimientos para garantizar que el desarrollo del software se lleve a cabo de manera ordenada y secuencial. Estos procesos según (Falgueras, 2003) se ven representados como etapas que preceden y proceden a la programación como tal, esto sumado a los métodos y técnicas de la ingeniería del software forman parte del marco del ciclo de vida del software.

2.2.1.1 Modelo clásico

Según (Falgueras, 2003, p. 20) describe este ciclo como:

> "A veces, el ciclo de Vida clásico también se denomina ciclo de Vida en cascada, lo cual quiere decir que en cada etapa se obtienen unos documentos (en inglés, *deliverables*) que son las bases de partida de la etapa siguiente —que, por tanto, no puede comenzar antes de que haya terminado la anterior— y nunca se regresa a etapas pasadas."

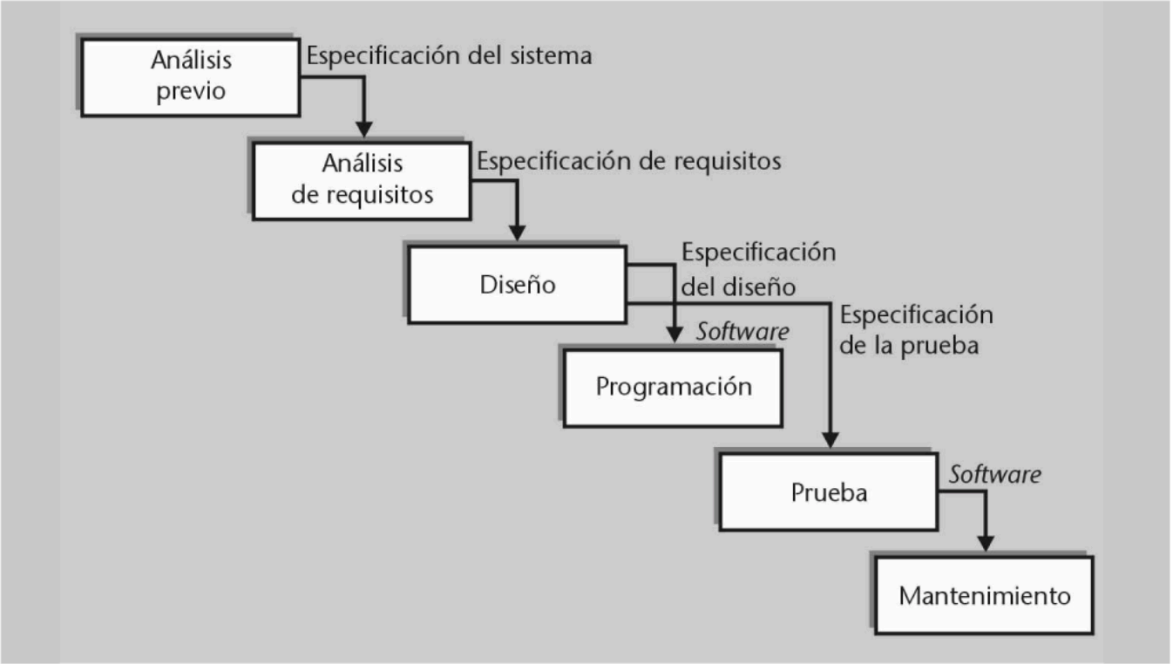


Figura 3 Etapas del ciclo de vida clásico del software

FUENTE: Falgueras, B. C. (2003). *Ingeniería del software.* Barcelona: Editorial UOC.

Como se aprecia en la figura y tomando en cuenta la definición del modelo cascada anterior, se puede inferir que el funcionamiento de dicho modelo es secuencial, es decir, para poder avanzar al siguiente paso, se debe finalizar el paso anterior y así sucesivamente. También es importante mencionar, que este modelo no permite regresar en a un paso anterior, lo cual puede provocar restricciones y desventajas al momento de liberar un producto de software en producción.

Para detallar el objetivo de cada uno de los pasos en el ciclo de vida de software clásico, se tomará como referencia a (Rojas López, 2011) :

- Análisis previo: Se definen a gran escala las características del software, que soportara una serie de actividades determinadas, dentro del marco contextual de la empresa u organización.
- Análisis de requisitos: En esta esta etapa se busca definir los requerimientos de información del software de manera mas detallada, sin la definición del como en la parte técnica.
- Diseño: Una vez definido el problema en el paso anterior, este paso tiene como objetivo, buscar la solución técnica a los problemas descritos anteriormente. En este paso se definen aspectos como la arquitectura del sistema en general, estructura de la base de datos, definición de interfaces de usuario, identificación de servicios de terceros y de igual manera se busca diseñar las pruebas necesarias para garantizar el correcto funcionamiento de la solución.

- Programación: Consiste en transformar la etapa del diseño en lenguaje procesable por un ordenador. Todo el desarrollo se toma como un único entregable con todas las especificaciones requeridas.
- Prueba: En esta etapa se busca ejecutar todas las pruebas definidas con el fin de garantizar el correcto funcionamiento del software desde diferentes perspectivas.
- Mantenimiento: Hace referencia a la inspección de software para determinar errores y nuevas funcionalidades que deben ser agregados al software.

2.2.1.2 Modelo iterativo e incremental

Según (Falgueras, 2003) los ciclos iterativos e incrementales se definen como:

> "(…) Parece que la opción más razonable sea estudiar a fondo una pequeña parte de los requisitos que tenga una cierta autonomía, y diseñarla, programarla y probarla, y una vez que el cliente la haya dado por buena, hacer lo mismo con otra parte, y otra. (…) esto es lo que denominamos ciclo de Vida iterativo e incremental (iterativo porque se repite dentro de un mismo proyecto e incremental porque procede por partes)."

La inspección y adaptación son las principales características de un ciclo de vida iterativo, estos promueven la creación de un producto de software bajo

el desarrollo de pequeñas porciones, como se indica en la definición anterior, la idea es priorizar el desarrollo con lo requerimientos que se encuentran mejor definidos y detallados, además tienen una autonomía para poder llevar a cabo su programación y las pruebas del software. Esto permite a su vez, poder contar con entregables que pueden ser evaluados por el usuario de manera mas rápida, y esto permitirá definir posibles mejoras de manera eficaz y anticipada.

2.3 Desarrollo ágil del Software

Antes de entender el concepto de un desarrollo ágil de software, es importante definir en que consiste una metodología, que (Real Academia Española, 2021) la define de la siguiente manera:

"Conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal."

Con la definición anterior se puede entonces definir que una metodología de desarrollo ágil consiste en una serie de prácticas que siguen las organizaciones, equipos o cualquier profesional que de alguna manera interactúan con el desarrollo de aplicaciones con el fin de obtener una mejora sustancial de la eficiencia.

Existe una gran variedad de metodologías ágiles y cada una de estas presenta el uso de diferentes prácticas y "creencias" que hacen a cada metodología única; pero en la mayor parte de los casos estas metodologías de desarrollo pretenden llegar al mismo objetivo, lograr entregar un producto de alta calidad, de una manera más

rápida y siempre presentando una estructura flexible durante el proceso de desarrollo, la cual se adapta a la naturaleza cambiante de los requerimientos y prioridades los proyectos.

La principal característica de las metodologías ágiles son los ciclos de desarrollo cortos en los cuales los desarrolladores o cualquier profesional que realice trabajo directamente relacionado con el progreso del proyecto, trabajan en tareas relativamente pequeñas que toman generalmente menos de dos semanas en completarse y generalmente al final de cada ciclo las nuevas piezas del sistema pueden empezar a ser utilizadas y generar valor para la empresa con un costo relativamente bajo y específicamente este es el punto que hace que las metodologías ágiles sean tan atractivas para las empresas.

Las empresas en su mayor parte optan por metodologías ágiles en lugar de su contraparte, la metodología cascada, ya que, en el área económica, administrativa y de innovación de las empresas tiene mucho más sentido para la empresa, equipos o profesionales de desarrollo.

Se sabe que hay una gran variedad de metodologías ágiles y diferentes recetas que son una combinación de dos o más metodologías, pero hay dos que sobresalen como lo son Scrum y Kanban.

2.3.1 Scrum

Posiblemente la mejor forma de definir Scrum es como ellos mismos la describen en la pagina oficial (Scrum.org, 2021) :

“Scrum es un marco de trabajo dentro del cual las personas pueden abordar problemas complejos de adaptación, al tiempo que entregan productos de manera productiva y creativa del mayor valor posible.”

Para poder lograr lo antes mencionado, Scrum recurre a una serie de prácticas que actúan como una guía para la correcta implementación del marco de trabajo, entre estas prácticas se puede resaltar una en específico que posiblemente es la más importante, los *Sprints*, tal como se menciona en (Atlassian, 2021):

“Un sprint es un período breve de tiempo fijo en el que un equipo de scrum trabaja para completar una cantidad de trabajo establecida. Los sprints se encuentran en el corazón de las metodologías scrum y ágil.”

Scum también implementa las llamadas ceremonias ágiles, estas ceremonias son una serie de reuniones que toman lugar antes, durante y después de los *sprints*, estas reuniones se utilizan para poder planear los *sprints*, adaptarse durante el desarrollo del sprint y buscar cómo mejorar al final del mismo.

Mediante la utilización de los sprints y las ceremonias ágiles, los equipos logran entregar productos funcionales de una manera muy rápida y eficiente.

2.3.2 Kanban

(Agile Alliance, 2021) define Kanban como:

“un medio para diseñar, gestionar y mejorar los sistemas de flujo para el trabajo del conocimiento. El método también permite a las organizaciones comenzar con su flujo de trabajo existente e impulsar un cambio evolutivo. Pueden hacer esto visualizando su flujo

de trabajo, limitar el trabajo en progreso (WIP) y dejar de comenzar y comenzar a terminar."

Tal como se menciona en la descripción anterior, Kanban se enfoca más en los flujos de trabajo continuos a diferencia de Scrum que utiliza sprints para controlar el desarrollo del proyecto, una de las principales características de esta metodología son los "Kanban boards", estos permiten obtener una visualización completa del trabajo, permite ver en que se está trabajando y maximiza la eficiencia.

Kanban se puede percibir como una metodología más sencilla en la cual los miembros del equipo tienen muy pocas tareas en progreso pero están completamente concentrados en completar la tarea actual y al finalizar, se procede a trabajar en la siguiente tarea que tenga más relevancia. Determinar cuál metodología ágil utilizara un equipo es una decisión difícil que generalmente está guiada por preferencia y por la naturaleza de trabajo, muchos equipos deciden utilizar Kanban debido a que ciertos tipos de proyectos y la naturaleza continua de ciertas actividades.

2.4 Calidad del Software

Como parte de la definición de los requerimientos del software, nace la necesidad de poder garantizar que cada requerimiento y especificación del software cumpla con el resultado esperado por cada interesado del proyecto, teniendo en consideración variables previamente definidas como el tiempo y el costo, tal como se menciona en (Laporte & April, 2018, p. 20):

"Therefore, quality software is software that meets the true needs of the stake-holders while respecting any predefined cost and time constraints."

Como parte del estándar (ISO, 2014), establece una serie de características que califican la calidad del Software:

- Funcionalidad: Atributos relacionados entre si mediante funciones y procedimientos para satisfacer las necesidades implícitas y explicitas del software.
- Fiabilidad: Hace referencia al nivel de estabilidad en condiciones predefinidas en un lapso de tiempo.
- Usabilidad: Hace referencia al esfuerzo necesario para su uso, y la evaluación de este ejercicio mediante un grupo de usuarios.
- Eficiencia: Es la relación que existe entre el nivel de desempeño del software con la cantidad de recursos necesitados bajo las condiciones definidas.
- Mantenibilidad: Hace referencia a la facilidad de poder corregir o mejorar la versión actual del software.
- Portabilidad: Capacidad de reacción de software para poder ser utilizado en diferentes plataformas.
- Calidad en uso: Aceptación por parte del usuario final y la seguridad.

2.4.1 Tipos de pruebas funcionales para mejorar la calidad del software

A continuación, se describirán tres opciones de pruebas funcionales que permiten mejorar la calidad del uso del software, siendo una de ellas las pruebas unitarias,

que por el alcance y arquitectura de este proyecto se ajustan de manera eficaz para el aseguramiento de la calidad de este proyecto de investigación.

2.4.1.1 Pruebas unitarias

Para un mejor entendimiento de que es una prueba unitaria, se toma como referencia la definición según (Microsoft, 2020) que dice:

> "Una *prueba unitaria* es una prueba que ejercita componentes o métodos de software individuales, también conocidos como "unidad de trabajo". Estas solo deberían probar código que pueda controlar el desarrollador. No se usan para comprobar problemas con la infraestructura. Estos problemas incluyen la interacción con bases de datos, sistemas de archivos y recursos de red."

Es importante recalcar ciertos elementos fundamentales de las pruebas unitarias, presentes en la definición anterior, tales como:

- Desacoplamiento: Cada prueba unitaria debe ser creada para probar un único método lógico u unidad de trabajo, y no debe ser compartida para probar otros métodos lógicos.
- Responsabilidad del desarrollador: Las pruebas unitarias deben ser creadas por la figura que tendrá a cargo el desarrollo lógico del código, en este caso el programador.
- No involucra agentes externos: Esto quiere decir, que las pruebas unitarias no tienen como objetivo velar por que recursos externos

como la red, base de datos, interacción con otros sistemas, entre otros. Por el contrario, busca dar calidad a la capa de negocio del software.

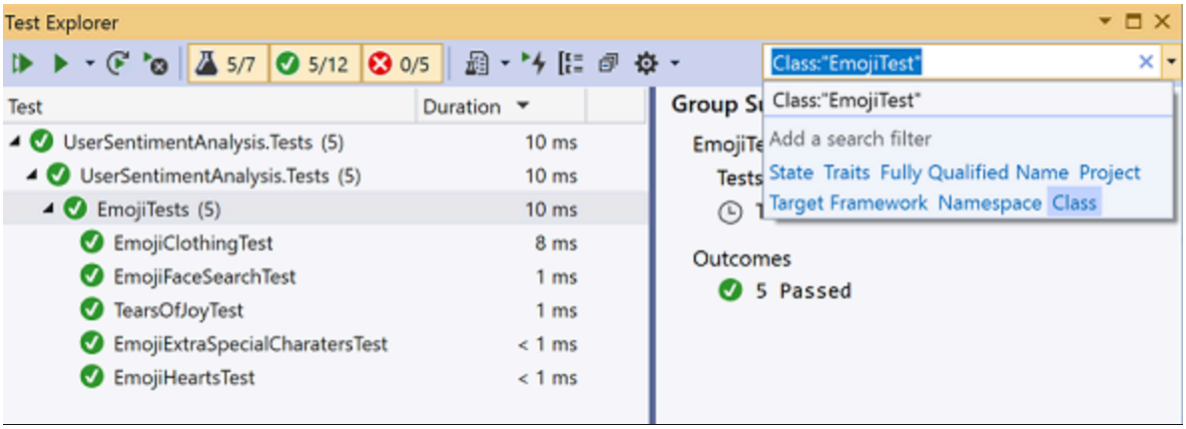


Figura 4 Ejemplo de ejecución de pruebas unitarias con Visual Studio

FUENTE: Sitio Web: https://docs.microsoft.com/es-es/visualstudio/test/unit-test-basics?view=vs-2019 (Microsoft, 2019)

2.4.1.2 Pruebas de integración

Según (Microsoft, 2020) la definición de una prueba de integración es:

"Una *prueba de integración* se diferencia de una prueba unitaria en que ejercita la capacidad de dos o más componentes de software de funcionar juntos, a lo que se conoce también como su "integración". Estas pruebas operan en un espectro más amplio del sistema sometido a prueba, mientras que las pruebas unitarias se centran en componentes individuales. Las pruebas de integración suelen incluir problemas con la infraestructura."

En este caso, una prueba de integración busca ampliar el alcance del comportamiento del software, relacionándolo con aspectos externos como las bases de datos, acceso a red entre otros.

2.4.1.3 Pruebas de carga

En este caso, según (Microsoft, 2020) una prueba de carga es:

> "El objetivo de una prueba de carga es determinar si un sistema puede controlar una carga especificada, por ejemplo, el número de usuarios simultáneos que usan una aplicación y la capacidad de esta de controlar las interacciones de forma más responsable"

Tal como se cita la definición anterior, una prueba de carga vela por llevar el software a sus puntos extremos de interacción y consumo, esto con el fin de garantizar un comportamiento responsable del software bajo estas condiciones.

2.5 Arquitecturas de Software

A lo largo del tiempo se ha buscado una definición consistente y que englobe el concepto desde diferentes perspectivas, para este proyecto de investigación se definirá la arquitectura según (IEEE SA, 2000, p. 6), quien define la arquitetura como:

> "La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución."

La arquitectura de software busca brindar un "ecosistema" funcional, donde todos los componentes que constituyen el software puedan cumplir los objetivos por los cuales fueron diseñados, sin descuidar el comportamiento de los mismos conforme el tiempo y la evolución.

2.5.1 Principios de la arquitectura

A continuación, se describirán algunos de los principios fundamentales de la arquitectura tomando como referencia a (Microsoft, 2020).

- Separación de intereses: Consiste en la separación lógica y funcional del trabajo que realiza el software, esto significa, que se debe tratar de independizar las capas que residen en la aplicación, lo más común ha sido la separación por capas, donde se busca que la lógica de negocio no dependa de ninguna otra capa de la aplicación.

- Encapsulación: Con la encapsulación, se busca la versatilidad y aislamiento de las partes de la aplicación. Con esto los diferentes objetos y paquetes se pueden remplazar con implementaciones alternativas, esto a su vez ayudará a la mantenibilidad y evolución del software.
- Inversión de dependencias: El objetivo de este principio es buscar la segmentación mediante módulos y la implementación de código fácil de mantener, esto mediante la introducción de interfaces y su implementación.
- Responsabilidad única: Significa que los objetos presentes en nuestra solución deben estar diseñados bajo una única responsabilidad y solo con una razón para cambiar.

2.5.2 Estilos de arquitectura de software

A continuación, se describirán algunos de los principales estilos de arquitectura, los cuales se clasifican de esta forma por ser familias de arquitecturas que comparten determinadas características (Microsoft, 2019).

2.5.2.1 N niveles

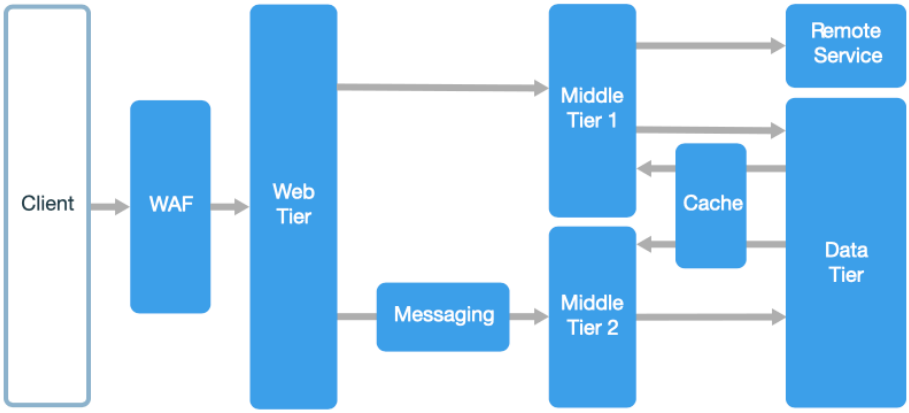


Figura 5 Arquitectura de n niveles/capas

FUENTE: Sitio Web: https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/n-tier (Microsoft, 2019)

Consiste en una arquitectura que permite separar las funcionalidades del software mediante capas, es decir, cada una cumple una responsabilidad en específico. Esto a su vez conlleva a que una capa superior pueda consumir los servicios de una capa inferior pero no al contrario.

Este tipo de arquitectura es muy común en aplicaciones web simples o en las aplicaciones monolíticas tradicionales.

2.5.2.2 Web-Cola-Trabajo

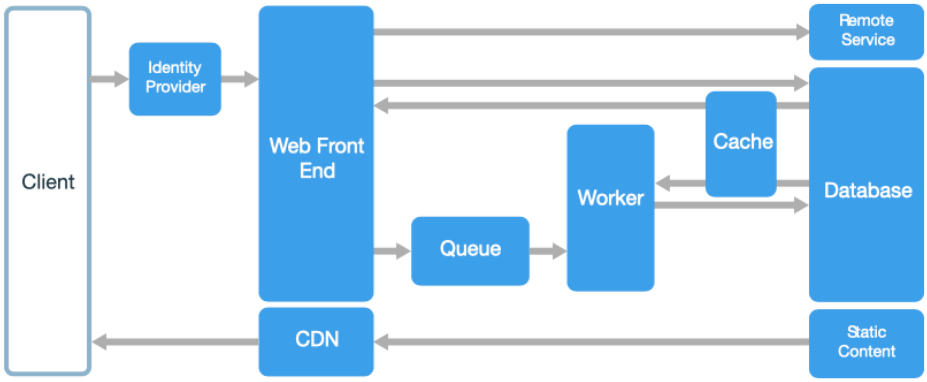


Figura 6 Arquitectura de Web-Cola-Trabajo

FUENTE: Sitio Web: https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/web-queue-worker (Microsoft, 2019)

Esta arquitectura se compone de un *frontend* web principal que atiende las solicitudes del cliente, adicionalmente posee un componente llamado "trabajador" que se encarga de llevar a cabo las funciones cuya ejecución toman mayor tiempo o se requiere mediante lotes.

Este tipo de arquitectura se ve aplicada en aplicaciones de dominio sencillo, es decir, que no tengan tanta lógica de negocio aplicada en si, además en soluciones donde se necesiten llevar a cabo tareas prolongadas o que se requiera segmentar una solicitud por lotes.

2.5.2.3 Basada en eventos

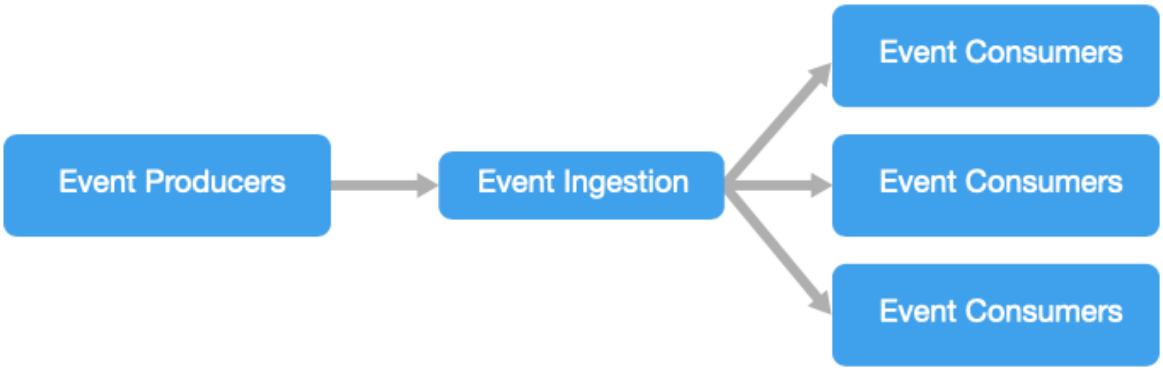


Figura 7 Arquitectura basada en eventos

FUENTE: Sitio Web: https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/event-driven (Microsoft, 2019)

Esta arquitectura se basa en dos elementos principales:

- Productores de eventos
- Consumidores de eventos

La idea de esta arquitectura es que los eventos sean entregados de manera inmediata para que los consumidores puedan responder de manera inmediata a dichos eventos. Tal como se aprecia en la imagen anterior, existe una desconexión entre los consumidores y productores, y los consumidores entre sí Esto con el fin de que los consumidores extraigan los mensajes de una cola unificada y así evitar la duplicación de procesamiento.

Este tipo de arquitecturas se pueden encontrar en escenarios donde varios sistemas o subsistemas requieren procesar los mismos eventos.

Adicionalmente, en escenarios donde se requiera procesamiento en tiempo real. Muy usado en procesamiento datos para IoT.

2.5.2.4 Microservicios

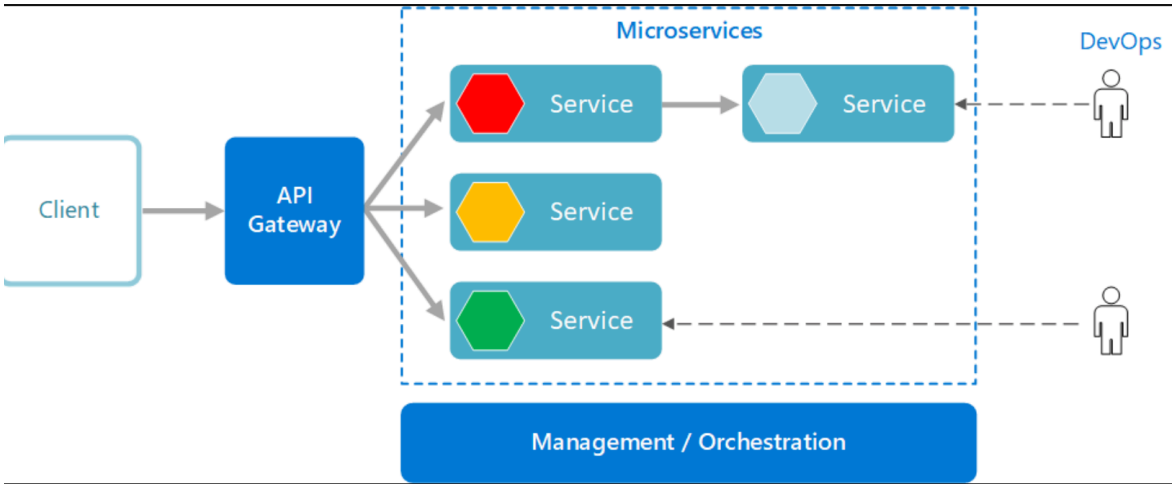


Figura 8 Arquitectura de microservicios

FUENTE: Sitio Web: https://docs.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices (Microsoft, 2019)

Para definir un microservicio se tomarán como referencias las definiciones descritas por (Microsoft, 2019):

"Una arquitectura de microservicio consta de una colección de servicios autónomos y pequeños. Cada uno de servicio es independiente y debe implementar una funcionalidad de negocio individual dentro de un contexto delimitado"

Y adicionalmente (Atlassian, n.d.):

"(…) es un término moderno utilizado para describir un patrón tradicional de "separación de intereses" dentro de un proyecto distribuido y conectado en red"

Buscando un término integral basado en las dos definiciones anteriores, un microservicio tiene como objetivo representar y solventar un problema en específico que tiene el negocio, dándole autonomía y singularidad ha dicha solución, esto quiere decir, que un microservicio debe estar separado de otros servicios y no verse afectado por cualquier tarea, mejora o mantenimiento de otro microservicio.

Adicionalmente a lo descrito anteriormente, se puede encontrar que los microservicios son independientes…

- … en su código fuente, es decir, un grupo de desarrolladores puede trabajar en sus modificaciones sin afectar otros servicios.
- … en su implementación, es decir, los cambios o mejoras se pueden liberar para un microservicio sin necesidad de volver a liberar todos los otros microservicios.
- … en la persistencia de los datos, esto significa que cada microservicio es responsable de velar por la creación, edición o eliminación de los datos que interactúan con el domino representado por el microservicio.

- … en las tecnológicas con que son construidos, esto permite que cada microservicio pueda ser diseñado y programado utilizando un diferente grupo de tecnologías.

Como parte de la arquitectura de los microservicios es común encontrar dos elementos fundamentales:

- Administración e implementación: Este componente tiene como objetivo colocar los servicios en los nodos, el monitoreo de errores en dichos nodos y velar por un equilibrio por todos los nodos existentes.
- Puerta de enlace de API: Este componente es el canal por el cual los clientes utilizarán los recursos presentes en cada microservicio, ya que se encarga de reenviar la solicitud al servicio correspondiente, evitando que los clientes se conecten directamente a los microservicios.

2.5.3 Comparación de estilos de arquitectura

El cuadro a continuación tiene como objetivo realizar una comparación de los diferentes estilos de arquitectura y sus características, para determinar sus ventajas y desventajas en relación con este proyecto de investigación.

Estilo de arquitectura	Portabilidad	Escalabilidad	Curva de aprendizaje	Implementación
N niveles	Puede ser liberado en entornos en la nube y locales, y mezclando ambos medios.	Al ser una arquitectura monolítica dificulta la separación de componentes y servicios, y la asignación de recursos.	Fácil aprendizaje para los desarrolladores.	Las nuevas funcionalidades requerirán de un despliegue completo de toda la solución.
Web-cola-trabajo	De preferencia utilización de plataformas en la nube.	El *frontend* se desacopla de los servicios por lo que permite la escalabilidad de manera individual.	Fácil aprendizaje para los desarrolladores.	Implementación de mejoras de manera independiente entre los servicios.
Basado en eventos	De preferencia utilización de plataformas en la nube.	Desvinculación de productores y consumidores, lo que facilita la distribución y escalabilidad.	Moderado grado de complejidad para desarrolladores.	Productores y consumidores son desplegados de manera aislada.
Microservicios	Puede ser liberado en entornos en la nube y locales, y mezclando ambos medios.	Permite escalar horizontalmente los subsistemas de manera independiente.	Moderado grado de complejidad para desarrolladores.	Cada subsistema puede ser implementado de manera aislada sin desplegar todos los otros microservicios.

Tabla 1 – Comparación de arquitecturas de software

FUENTE: Tabla creado con la información suministrada por (Microsoft, 2019)

2.6 Computación en la nube

Según (Real Academia de Ingeniería, 2021) se define la computación en la nube como:

"Utilización de las instalaciones propias de un servidor web albergadas por un proveedor de Internet para almacenar, desplegar y ejecutar aplicaciones a petición de los usuarios demandantes de las mismas."

Adicionalmente según (Microsoft, 2021) la computación en la nube es:

"… el suministro de servicios informáticos (incluidos servidores, almacenamiento, bases de datos, redes, software, análisis e inteligencia) a través de Internet ("la nube"), cuyo objetivo es ofrecer una innovación más rápida, recursos flexibles y economías de escala. Lo habitual es pagar solo por los servicios en la nube utilizados, de tal forma que lo ayude a reducir los costos operativos, a ejecutar la infraestructura con más eficacia y a escalar a medida que cambian las necesidades de su negocio."

Por último, pero no menos importante según (Amazon, 2021) la computación en la nube es:

"La informática en la nube es la distribución de recursos de TI bajo demanda a través de Internet mediante un esquema de pago por uso. En vez de comprar, poseer y mantener servidores y centros de datos físicos, puede obtener acceso a

servicios tecnológicos, como capacidad informática, almacenamiento y bases de datos, en función de sus necesidades a través de un proveedor de la nube (…)"

Tal como se explica en las definiciones anteriores, el internet es el mecanismo clave para suministrar este tipo de servicios, que tienen como objetivo simplificar la creación, acceso, modificación o eliminación de recursos informáticos.

2.6.1 Características de la computación en la nube

A continuación, se explican una serie de características para la computación en la nube según (Ávila Mejía, 2011):

- Accesibilidad: Lo único que requiere el usuario para acceder a los recursos o servicios en la nube, es un dispositivo con acceso a internet, no se requiere ningún equipo potente a nivel de hardware, dado que todo el procesamiento ocurre en la plataforma desplegada en la nube.
- Auto reparable: Múltiples opciones para manejar caídas de los servicios, mediante procesos de recuperación utilizando respaldos.
- Escalabilidad: La infraestructura, servicios o recursos, se desplegarán según el consumo y la demanda de estos, permitiendo de esta manera la eliminación de tiempos de espera y cuellos de botella.
- Virtualización: Las aplicaciones que se necesiten ejecutar son totalmente independientes al hardware en el que están corriendo.

- Seguridad: Los servicios en la nube ofrecen esquemas de seguridad entre los consumidores de los servicios, lo que evita que los datos o la información importante sea comprometida.

2.6.2 Tipos de servicio en la nube

Los servicios que ofrece actualmente la informática en la nube se pueden agrupar en tres categorías, que según (Microsoft, 2021) se pueden describir de la siguiente manera.

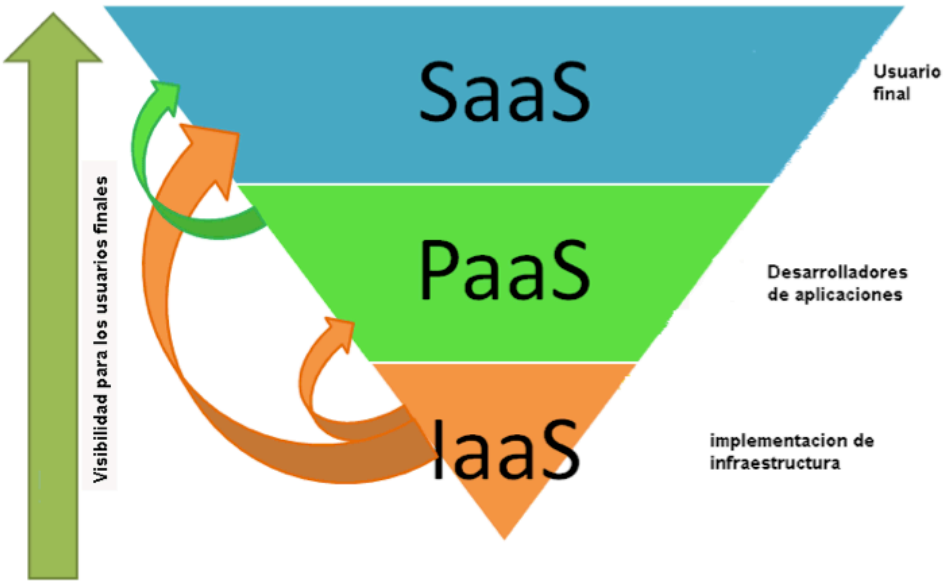


Figura 9 Clúster y categorías de la RAS

FUENTE: Sitio Web: http://www2.izt.uam.mx/newpage/contactos/anterior/n80ne/nube.pdf (Ávila Mejía, 2011)https://www.agriculturasostenible.eco/gobernanza

2.6.2.1 Infraestructura como servicio (IaaS)

Se conoce como el servicio más básico en la informática en la nube. Mediante esta categoría se puede alquilar servicios de TI, tales como:

- Servidores
- Máquinas virtuales
- Almacenamiento
- Redes
- Sistemas Operativos

Todos los servicios anteriores se rigen bajo un esquema de costo por uso.

2.6.2.2 Plataforma como servicio (PaaS)

Este tipo de servicio ofrecen un ecosistema para desarrollar, probar, desplegar y administrar aplicaciones de software. Su objetivo es facilitar a los desarrolladores la creación de aplicaciones web o móviles, sin necesidad de preocuparse por elementos ajenos al software, tales como la administración o la configuración de la infraestructura, configuraciones de red o configuraciones de base de datos.

2.6.2.3 Software como servicio (SaaS)

Este servicio tiene como objetivo entregar aplicaciones a petición generalmente acompañada de una suscripción. La diferencia de este servicio es que los proveedores de estas aplicaciones se encargan de las

actualizaciones y mantenimiento, dejando al usuario un producto final para ser accedido mediante sus computadores o dispositivos móviles. Un ejemplo de este tipo de servicio es la plataforma Office 365 de Microsoft.

2.6.3 Ventajas de la computación en la nube

La aparición de la computación en la nube ha traído consigo numerosos beneficios a las diferentes áreas empresariales, industriales y familiares, tales como:

- Reducción en la inversión de adquisición de software y hardware, y el costo de mantenimiento que trae consigo dicha inversión. Además del gasto por temas de espacio, refrigeración, consumo eléctrico y mano de obra técnica para operar estos servicios.
- Flexibilidad y escalabilidad global, lo que permite hacer uso de los recurso o servicios que se necesitan únicamente desde la zona global que mejor se adecue a la necesidad de negocio.
- El rendimiento de los sistemas computacionales y el hardware se maximiza dado que todos estos servicios están en un constante proceso de mantenimiento, para garantizar que la tecnología se eficaz y de ultima generación.
- Se ofrecen esquemas completos para la administración de seguridad, por lo que todo se centraliza y permite una mejor administración de accesos a los recursos o servicios.

- Velocidad, esto permite que el usuario pueda solicitar sus recursos mediante plataformas amigables, y que tan solo en cuestión de minutos pueda contar con lo requerido.
- Los centros de datos locales traen consigo muchas tareas de mantenimiento, actualización, constante monitoreo y desde la perspectiva financiera procesos de depreciación. Lo anterior desaparece con el uso de estos servicios en la nube.
- Ofrece un mejor manejo de esquemas de recuperabilidad y estabilidad del negocio ante situaciones de desastre.

2.6.4 Desventajas de la computación en la nube

A pesar de que los servicios en la nube describen un sinfín de beneficios, existen varios aspectos que según (Ávila Mejía, 2011) son importantes de mencionar. Algunos de ellas son:

- Reducción en la demanda de dispositivos periféricos, por ejemplo, discos duros, dispositivos de almacenamiento USB, unidades ópticas. Esto debido a que todo puede ser accedido en la nube siempre y cuando se tenga acceso a internet.
- La dependencia de millones de usuarios a estos servicios, para su negocio del día a día, lo que exige a estos proveedores a garantizar un servicio constante y sin derecho a fallas.
- Existe una dependencia ineludible por el internet, es decir, sin este no se puede acceder a los servicios ni los recursos, lo que obliga al usuario a contar

con un buen proveedor de internet para evitar afectaciones de productividad esto a nivel empresarial.

- Si el proveedor de este servicio no ofrece flujos robustos de escalabilidad, los recursos pueden verse afectados, ya que cada día crece la cantidad de usuarios que utilizan estos servicios.

2.6.5 Proveedores de computación en la nube

A pesar de que existen muchos proveedores de estos servicios, a continuación, se mencionaran 3 de los más grandes que han acaparado este mercado.

2.6.5.1 Microsoft Azure

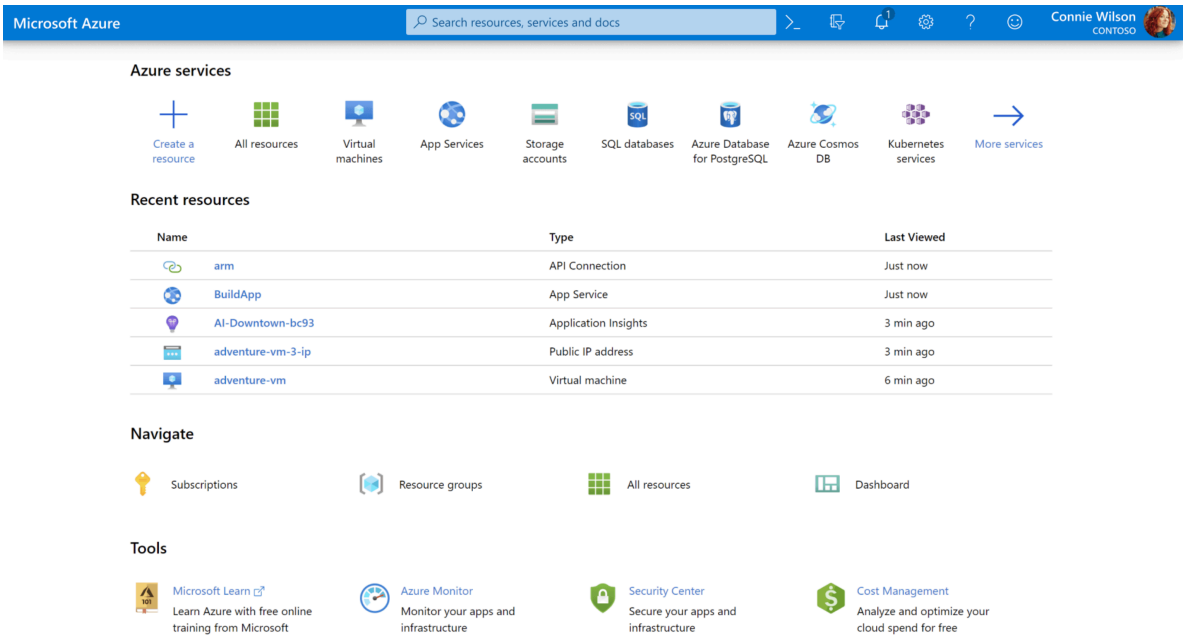


Figura 10 Portal de administración de recursos de Azure

FUENTE: Sitio Web: https://azure.microsoft.com/es-es/

Microsoft Azure es el nombre que recibe la plataforma creada por Microsoft para ofrecer los servicios de computación en la nube.

En octubre de 2008, este gigante de la tecnología hizo la presentación previa de este servicio bajo el nombre de Windows Azure Platform, en su evento “Professional Developers Conference”. Sin embargo, fue hasta febrero de 2010 donde se comercializó de manera oficial este producto, incluyendo servicios tales como: bases de datos relacionales de SQL, Full trust PHP, CDN entre otros.

En junio de 2010, se iniciaron con mejoras a la plataforma, donde se incluyeron nuevas actualizaciones de SQL Azure, así como novedades en el sistema operativo con .NET Framework 4 o CDN.

Para 2014, además del crecimiento que había tenido la plataforma a nivel de clientes, se decide cambiar el nombre por Microsoft Azure (Cabezudo, 2019).

Al 2021, según (Microsoft, 2021) posee más de 200 productos y servicios en la nube que permiten crear soluciones para solventar dificultades actuales y ayudar a las empresas a afrontar el futuro.

Además, Microsoft invierte más de 1000 millones de dólares en seguridad y expertos en ciberseguridad para analizar más de 8 billones de señales de amenazas diarias, lo que deja en claro su preocupación por salvaguardar la información de sus clientes.

Algunas de las categorías en productos y servicios que se ofrecen por Microsoft Azure son (Microsoft, 2021):

- Administración y Gobernanza
- Almacenamiento
- Análisis
- Bases de datos
- *Blockchain*
- Contenedores
- DevOps
- Herramientas para desarrolladores
- Hybrid + Multicloud
- IA y Machine Learning
- Identidad
- Integración
- Internet de las cosas
- Migración
- Movilidad
- Multimedia
- Proceso
- Realidad mixta
- Redes
- Seguridad
- Web
- Windows Virtual Desktop

2.6.5.2 AWS (Amazon Web Services)

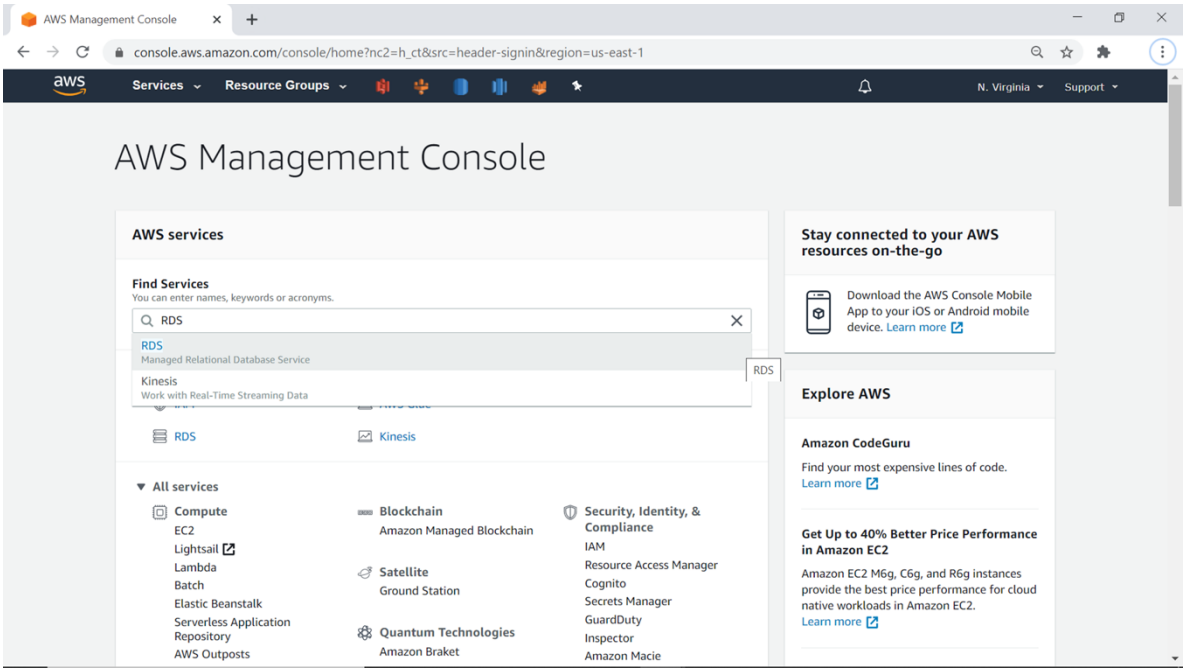


Figura 11 Consola de administración de recursos de AWS

FUENTE: Sitio Web: https://www.sqlshack.com/getting-started-with-aws-rds-aurora-db-clusters/

Amazon Web Services es el nombre que recibe la plataforma creada por Amazon para ofrecer los servicios de computación en la nube.

Según (Amazon, 2021), en 2006, Amazon Web Services (AWS) comenzó a proporcionar servicios de infraestructura de TI para empresas en forma de servicios web, más conocido hoy como informática en la nube.

Para el 2021, Amazon proporciona una infraestructura escalable, confiable, y de bajo costo, que ha permitido potencializar miles de negocios a lo largo de 190 países en todo el mundo. Esto se debe a que posee centros de datos en:

- Estados Unidos
- Europa
- Brasil
- Singapur
- Japón
- Australia

Entre las categorías de servicio que ofrecen, se pueden mencionar:

- Alojamiento de aplicaciones
- Respaldos y almacenamiento
- Entrega de contenido
- Sitios web
- TI empresarial
- Bases de datos

2.6.5.3 GCP (Google Cloud Platform)

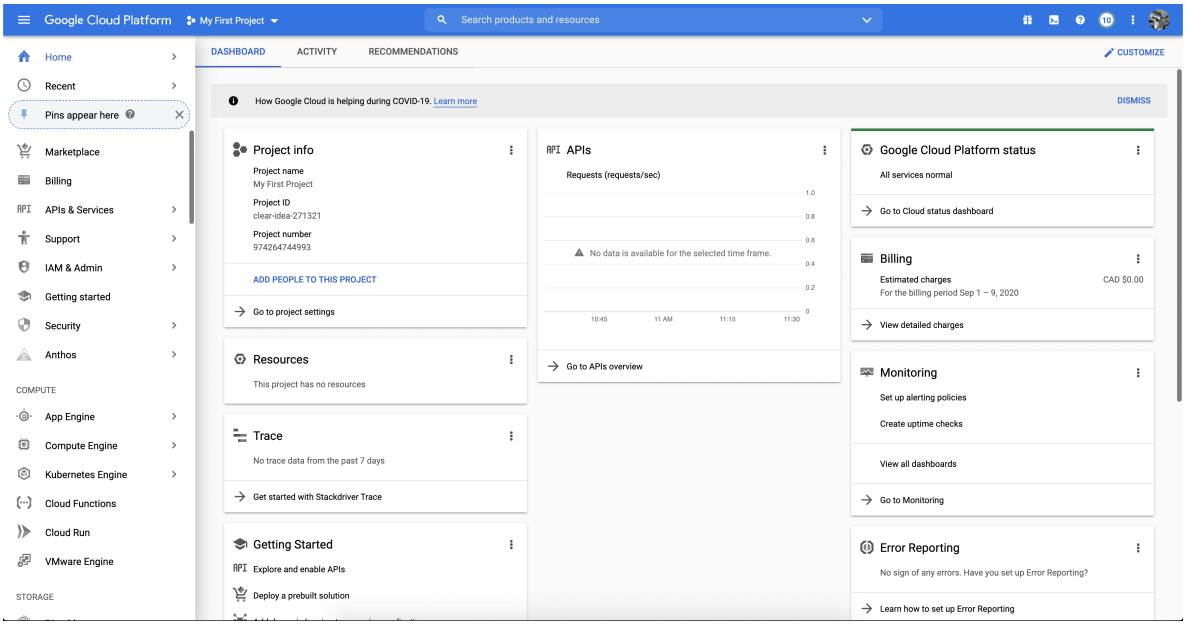


Figura 12 Consola de administración de recursos de GCP

FUENTE: Sitio Web: https://tyk.io/getting-started-with-tyk-on-google-cloud-platform-and-debian/

La primera versión de GCP, fue lanzada 2 años después del lanzamiento de Amazon, esto en 2008. La primera versión de GCP fue presentada como “Google App Engine”, y su objetivo principal fue facilitar a los desarrolladores en general el como iniciar una aplicación y como fácilmente escalar sus recursos conforme el tráfico de esta aumentara (Google, 2008).

Esta primera versión generada por Google se habilitó para 10,000 desarrolladores con el objetivo de obtener retroalimentación y así poder mejorar los servicios.

Finalmente, en noviembre del 2011, la plataforma que había sido lanzada como plan piloto se oficializo como un producto mas de esta compañía, dando paso así a la plataforma de GCP (Paul, 2018).

A continuación, algunos de los productos destacados de esta plataforma:

- Compute Engine
- Google Kuberneters Engine
- Operaciones
- Cloud Storage
- BigQuery
- Cloud Run
- SDK de Google Cloud
- Cloud CDN
- Anthos
- Cloud SQL
- Dataflow

2.7 Interfaces de comunicación de Software

En esta sección se buscará conceptualizar un API y un servicio web, siendo ambas tecnologías de comunicación e integración entre aplicaciones, concluyendo con un cuadro comparativo y resaltando ventajas y desventajas de dichas tecnologías.

2.7.1 Servicio Web

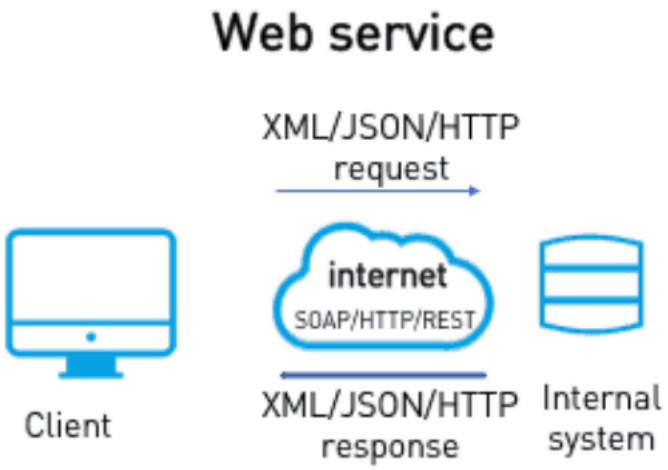


Figura 13 Esquema general de arquitectura de un Web API

FUENTE: Sitio Web: https://medium.com/beltranc/diferencia-entre-api-y-servicio-web-5f204af3aedb

Un servicio web según (Real Academia de Ingeniería, 2021) se define como:

"Aplicación basada en protocolos web que pueden combinarse y ajustarse para proporcionar funcionalidad de carácter empresarial a través de una conexión internet."

Un servicio web es un mecanismo por el cual, un cliente puede realizar peticiones a un servidor y esperar una respuesta con datos de este, esto utilizando un protocolo de comunicación.

Según (IBM, 2021), los servicios web buscan una arquitectura orientada en servicios, esto quiere decir, su enfoque es crear servicios que puedan ser invocados para llevar a cabo una tarea. Adicionalmente, los servicios web utilizan tecnologías

existentes como por ejemplo HTTP para el transporte de información y XML para invocar la implementación.

2.7.1.1 Componentes de un Servicio Web

Existen dos componentes que hacen posible el correcto funcionamiento de esta tecnología, los cuales son:

- Lenguaje de descripción de servicios web (WSDL): Es un archivo con formato XML que describe el servicio web, aquí se definen los diferentes servicios u operaciones que pueden ser efectuados.
- Protocolo SOAP: Protocolo por el cual se invoca un servicio del servicio web.

2.7.1.2 Características de un servicio web

Algunas de las principales características de un servicio web son:

- Interoperable y multiplataforma, esto implica que no existe una dependencia de tecnología empleada para la creación del cliente y servidor para poder comunicarse entre ellos.
- Su formato se basa en texto, por eso su uso mediante HTTP.
- Intercambio de mensajes entre el cliente y servidor utilizando el protocolo SOAP.
- La interfaz se expone mediante WSDL.

2.7.2 API (Application Programming Interface)

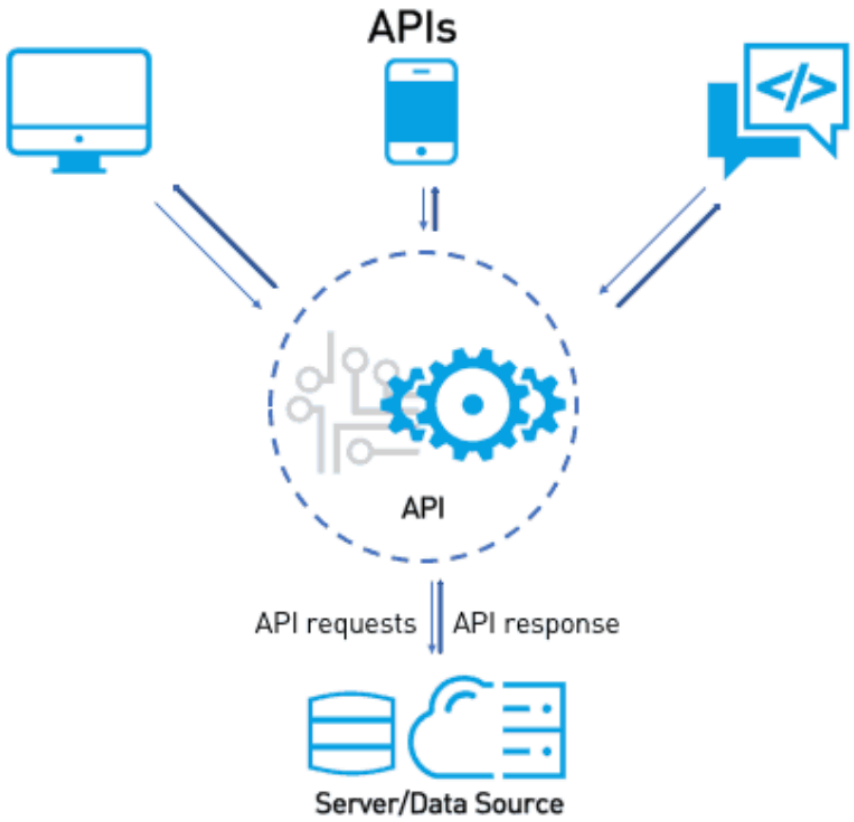


Figura 14 Esquema general de arquitectura de un Web API

FUENTE: Sitio Web: https://medium.com/beltranc/diferencia-entre-api-y-servicio-web-5f204af3aedb

Traducido al español las siglas API significan Interfaz de programación de aplicaciones, y según (Real Academia de Ingeniería, 2021) se define como:

"Conjunto de funciones o procedimientos de una aplicación accesibles desde un programa externo."

Esta tecnología tal como se describe anteriormente, es un mecanismo por el cual las aplicaciones se pueden comunicar entre si, permitiendo de esta manera, una

comunicación multiplataforma, es decir, es irrelevante la tecnología del emisor y del receptor en este caso.

Para que un API sea funcional, según (Microsoft, 2018) se debe cumplir con los siguiente:

- Independencia de la plataforma: Significa que cualquier cliente debe ser capaz de consumir el API, sin importar la forma en la que fue diseñado e implementado. Por lo que es necesario el uso de protocolos estándar y la definición de contratos para garantizar la forma en la que cliente y el servicio se comunicaran.
- Evolución del servicio: La API debe de evolucionar y con ello implementar nuevas funcionalidades, pero esto no debe afectar el funcionamiento de las aplicaciones cliente. Además, se debe garantizar una correcta documentación de las funciones para que los clientes sepan como utilizar o consumir la API.

2.7.2.1 Principios REST

Los principios REST fueron propuestos en el año 2000, la transferencia de estado representacional (REST), es un estilo de arquitectura para el diseño de API. Si bien es cierto, REST no es un protocolo, los principios se ven implementados mediante otros protocolos, por ejemplo: HTTP o HTTPS.

Algunos de los principios REST implementados con el protocolo HTTP:

- El diseño de APIs REST giran entorno a recursos, estos son todos aquellos objetos, datos o servicios que pueden ser accedidos por el cliente.
- Cada recurso debe tener un único identificador, el cual es una URI (Uniform Resource Identifier) una secuencia de caracteres que identifica lógica o físicamente un recurso. Por ejemplo: https://test-works.com/orders/1.
- Los clientes interactúan con el API mediante el intercambio de representaciones de los recursos. La mayoría de los API utilizan el formato JSON, para facilitar estas representaciones.
- Uso de interfaces comunes, para desacoplar los clientes de las implementaciones de los servicios, y cuando se utiliza el protocolo HTTP, lo normal es utilizar los verbos de este protocolo para efectuar operaciones sobre los recursos. Los verbos HTTP más comunes son:
    - GET: Este permite obtener información de los recursos.
    - POST: Permite crear un nuevo recurso.
    - PATCH: Permite actualizar propiedades específicas del recuso.
    - PUT: Funciona como la actualización completa de todo un recurso, por lo general involucra una eliminación del recurso anterior y la creación del nuevo recurso.
    - DELETE: Permite la eliminación de un recurso.
- Las solicitudes al API deben ser “stateless”, esto quiere decir, que no debe existir una relación entre una solicitud y otra, la única información que se debe almacenar o afectar es la del recurso únicamente.

- REST APIs son manejados por vínculos de hipermedia, que se encuentran en la representación del recurso, estos a su vez facilitan a saber que otras operaciones de recursos relacionados se pueden realizar y como hacerlo.

2.7.2.2 OpenAPI

Según indica (Open API, 2021):

> "La Iniciativa OpenAPI (OAI) fue creada por un consorcio de expertos de la industria con visión de futuro que reconocen el inmenso valor de estandarizar cómo se describen las API. Como estructura de gobernanza abierta bajo la Fundación Linux, la OAI se centra en la creación, evolución y promoción de un formato de descripción neutral para el proveedor."

En palabras más simples, la iniciativa de OpenAPI define una especificación que permite de una forma neutral, sin ser específica de un lenguaje de programación, describir un API y su comportamiento, esto permite que cualquier desarrollador pueda entender la capacidades de un servicio sin tener que acceder a documentación o hacer una revisión de código.

Al OpenAPI ser un estándar, cualquier API que lo implemente de una manera apropiada adquiere un funcionamiento muy predecible lo cual es una gran ventaja a la hora de interactuar con este.

2.7.2.3 Swagger

OpenApi y Swagger se encuentran muy relacionados, OpenApi es el estándar y Swagger es la forma en la que comúnmente se implementa ese estándar.

(Swagger.io, 2021) menciona que Swagger permite crear la descripción de un API para que esta sea luego interpretada por una máquina. La habilidad de las APIs para describir su propia estructura es lo que hace que Swagger sea tan útil. La ventaja de poder leer la estructura del API, es que le permite a una serie de herramientas que forman parte de Swagger crear la documentación del API, generar librerías para el API en varios lenguajes de programación y también permite crear pruebas automáticas, todo esto es muy útil ya que se traduce en más fiabilidad, mayor eficiencia y reducción de costos.

Muchas de las principales tecnologías para implementar APIs como lo son los marcos de trabajo de los diferentes lenguajes (*frameworks*), servidores HTTP como Nginx o Apache y servicios en la nube como AWS API Gateway o Google Cloud API Gateway soportan de manera nativa Swagger, esto también es una gran ventaja en el caso de que se requiere migrar de una plataforma a otra.

2.7.3 API vs Servicios Web

Con base a la investigación e información anterior, a continuación, se comparan ambas tecnologías:

Servicios Web	API
Todos los servicios web son API.	No todos los API son servicios web.
Los servicios web utilizan únicamente formato XML para envío y recepción de información.	Los API soportan diferentes formatos para envío de respuestas, tales como: JSON, XML o cualquier otro formato.
Los servicios web solo utilizan el protocolo SOAP para para enviar y recibir datos sobre la red. Como consecuencia, hace que su arquitectura sea mas compleja.	Los API utilizan una arquitectura mas simple.
Los servicios web solo utilizan soporte a protocolo HTTP.	Los API proporcionan soporte a los protocolos HTTP/HTTPS y también soporta encabezados en esta comunicación.

Tabla 2 – Comparación de servicios Web y API

FUENTE: Tabla creado con la información suministrada por (Microsoft, 2018) y (IBM, 2021)

2.8 Blockchain

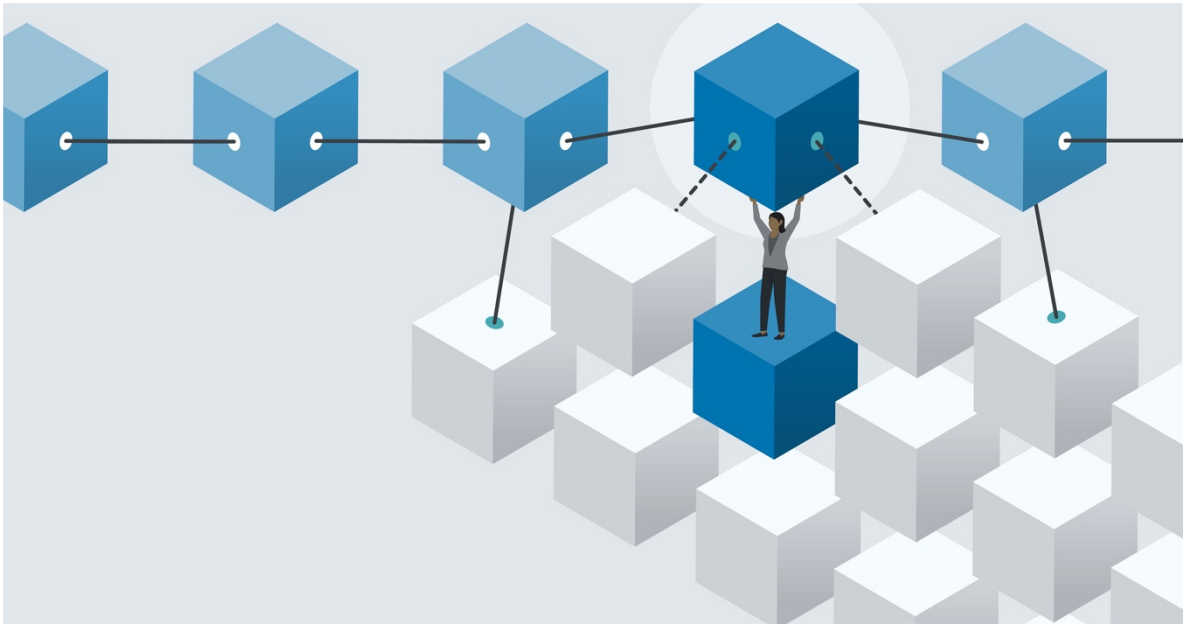


Figura 15 Blockchain

FUENTE: Sitio Web: https://shopinga.net/item/advance-your-skills-in-the-blockchain/

2.8.1 Definición y elementos del Blockchain

Según la explicación inicial e introducción de (Haber & Stornetta, 1991) se puede definir *Blockchain* como una lista de registros que crece continuamente, dichos registros reciben el nombre de bloques, y dichos bloques se vinculan y aseguran entre si mediante la criptografía.

Los bloques de registros están constituidos normalmente por tres elementos fundamentales:

1. Datos: Registros que se quieren almacenar en el bloque cifrado.
2. Hash previo: Almacena el hash del bloque anterior.
3. Hash: Identificador del bloque actual.

En este contexto del Blockhain, se define Hash como la representación única de cada bloque de datos, comportándose como una huella digital cifrada.

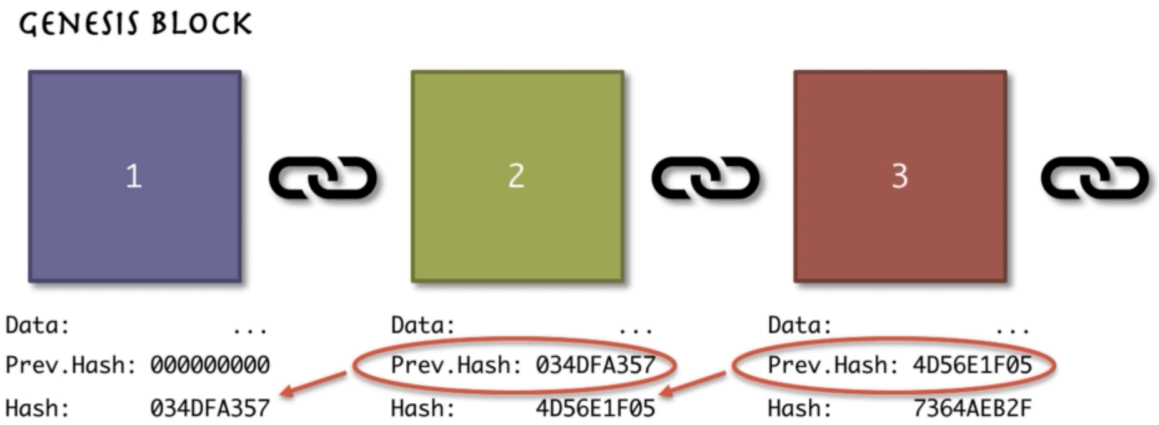


Figura 16 *Blockchain* Concepto general

FUENTE: Sitio Web: https://www.udemy.com/course/build-your-blockchain-az/

2.8.1.1 Algoritmo SHA256 Hash

Tomando como referencia (Tel, 2008), para poder definir y entender en que consiste el algoritmo SHA256 Hash, se puede hacer la analogía con la huellas digitales que permiten identificar a los seres humanos, es decir, en la actualidad las bases policiales utilizan las huellas digitales de las personas como un identificador único, si bien es cierto existe una remota posibilidad de que una huella digital se repita es casi imposible, según (Scientific American, 2020) la posibilidad es de 1 en 64 trillones, por lo que nos da un gran nivel de fiabilidad como identificador único de los seres humanos.

Este mismo principio es aplicado a documentos o cualquier representación digital, mediante el algoritmo SHA256 Hash, el cual fue desarrollado por la NSA (Agencia de Seguridad Nacional, U.S).

El nombre de SHA como acrónimo en ingles de “Secure Hash Algorithm” y los 256 por la cantidad de bits que ocupa en memoria, además siempre tiene una longitud de 64 caracteres donde se pueden ver incluidas letras y números ya que su base es hexadecimal.

Este sería un ejemplo de cómo se vería el texto “Probando algoritmo SHA256” convertido en SHA256:

“7bb64d6c8cfcf7e85b5102c9b44228e35e3b7c7f2dade35d4506b91d54940923”

Es importante que los algoritmos Hash cumplan con los siguientes requerimientos:

1. Unidireccional: El algoritmo Hash no debe permitir la ingeniería inversa, es decir, de un valor hash reconstruir el elemento cifrado.
2. Determinista: El algoritmo Hash debe de ser capaz de siempre devolver el mismo valor cifrado siempre y cuando el input sea el mismo.
3. Computación rápida: El algoritmo debe ser eficaz y bien optimizado.
4. Efecto avalancha: Cualquier cambio al input por mas pequeño que sea, la salida del algoritmo debe ser complemente diferente. Para demostrar este concepto, se utilizará el texto “Probando algoritmo

SHA256.”, nótese que solo se agregó un punto al final de la cadena de caracteres, y el resultado es completamente diferente al anterior: *“155a4e55c5ea467ad7d5369731ed9169c0f8454e8c5c5c8d9b7fc0b73cca2706”*

5. Debe resistir colisiones: Esto quiere que en el algoritmo no pueden existir colisiones que permitan tener más de un elemento relacionado un mismo hash.

2.8.1.2 Registro inmutable (Libro mayor)

Según (IBM, 2021) el elemento de registro inmutable consiste en un libro mayor compartido donde las transacciones pueden ser registradas una única vez, y estas no pueden ser modificadas o bloqueada por otros participantes, si por alguna razón la transacción presenta algún error, una nueva transacción debe ser registrada revertiendo el error, y posteriormente ambas transacciones serán visibles.

2.8.1.3 Redes distribuidas Peer-2-Peer

Este elemento de *Blockchain*, representa una capa importante de seguridad para esta tecnología, las redes de distribución P2P permiten tener una copia del registro inmutable descentralizado en diferentes equipos localizados en la red. Esto quiere decir que, bajo un ataque a un bloque o secuencia de bloques, el elemento de P2P constantemente revisa que los bloques sean iguales en todos los pares conectados a la red, y si encuentran un problema

automáticamente restauran los bloques dañados en el equipo correspondiente, lo que cierra la brecha de ataques.

2.8.1.4 Minería

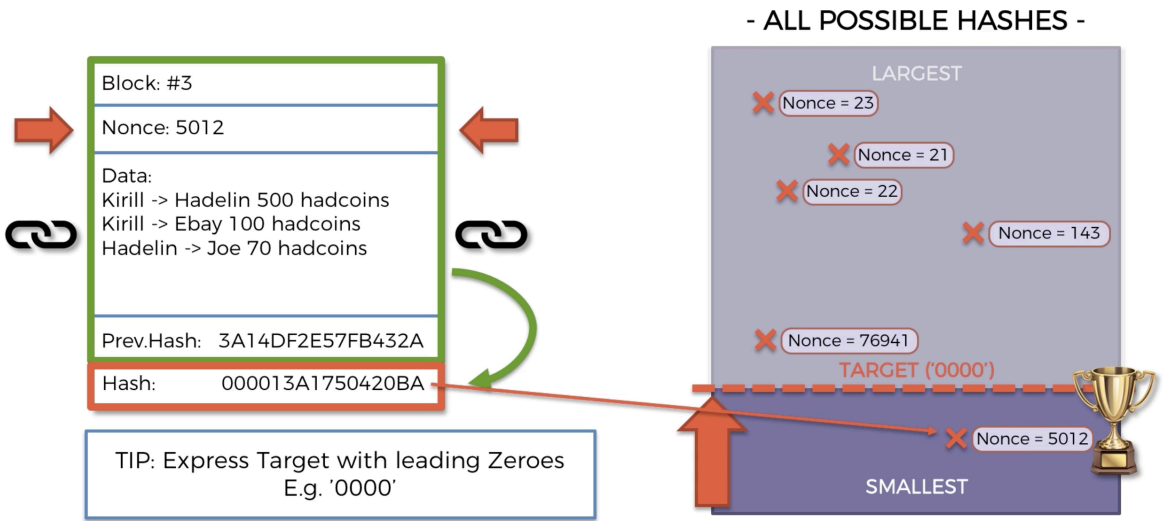


Figura 17 Minería en Blockchain

FUENTE: Sitio Web: https://www.udemy.com/course/build-your-blockchain-az/

Según (Raj, 2021) la minería en *Blockchain* consiste en el proceso de agregar un bloque nuevo a la cadena de bloques existentes. Esta tarea es llevada a cabo por una comunidad en mundo conocida como los mineros de *Blockchain*.

La tarea de los mineros consiste en generar una Hash valido para el bloque, para que este pueda ser agregado al registro inmutable de transacciones. La forma que utilizan para generar Hash distintos sin afectar los datos del bloque es cambiando un elemento conocido como “Nonce”, esta propiedad del bloque puede ser cambiado, sin afectar ningún registro contenido en el

bloque, y según la teoría del efecto avalancha al cambiar este elemento, el algoritmo Hash generara una salida totalmente diferente. El objetivo de los mineros es obtener un Hash que este dentro de la meta establecida para agregar bloques.

2.8.2 Casos de usos del *Blockchain*

Según (IBM, 2021) algunos de los ejemplos claros de la implementación de *Blockchain* en la actualidad que se pueden mencionar son:

1. IBM está ayudando a Raw Seafood en la confiabilidad y trazabilidad de los alimentos, desde el momento que estos son extraídos del mar y son llevados a los supermercados o restaurantes.
2. Golden State Foods, se está apalancando de la inmutabilidad del *Blockchain* para garantizar su proceso de cadena de suministros y garantizar la calidad de sus productos.
3. En el área de salud en general, la plataforma que ofrece IBM para *Blockchain*, genera confianza de la procedencia de la información y la eficiencia, con el fin de mejorar el cuidado del paciente y la rentabilidad del negocio.

2.8.3 Plataformas de Blockchain

	EOS	Ethereum	Hyperledger Fabric	Quorum	Corda
Industry focus	Cross-industry	Cross-industry	Cross-industry	Financial services	Financial services
Consensus mechanism	Delegated Proof-of-Stake	Proof-of-Work	Permissioned voting based ordering service	QuorumChain	Validity or Uniqueness Consensus
Enterprise Features	Scalability and energy-efficiency	Large developer base	Modular architecture	Privacy and confidentiality	Strict privacy model
Programming Languages	C++	Solidity	Java, Golang, Node.js	Solidity	Kotlin, Java

Figura 18 Plataformas de *Blockchain*

FUENTE: Sitio Web: https://eoscostarica.medium.com/how-to-choose-an-enterprise-blockchain-platform-7c3665994ad6

En esta sección se buscará dar contexto de los diferentes servicios que ofrecen los diferentes proveedores de plataformas de Blockchain:

- EOS.IO
- Ethereum
- HyperLedger Fabric
- Quorum
- Corda

2.8.3.1 EOS.io

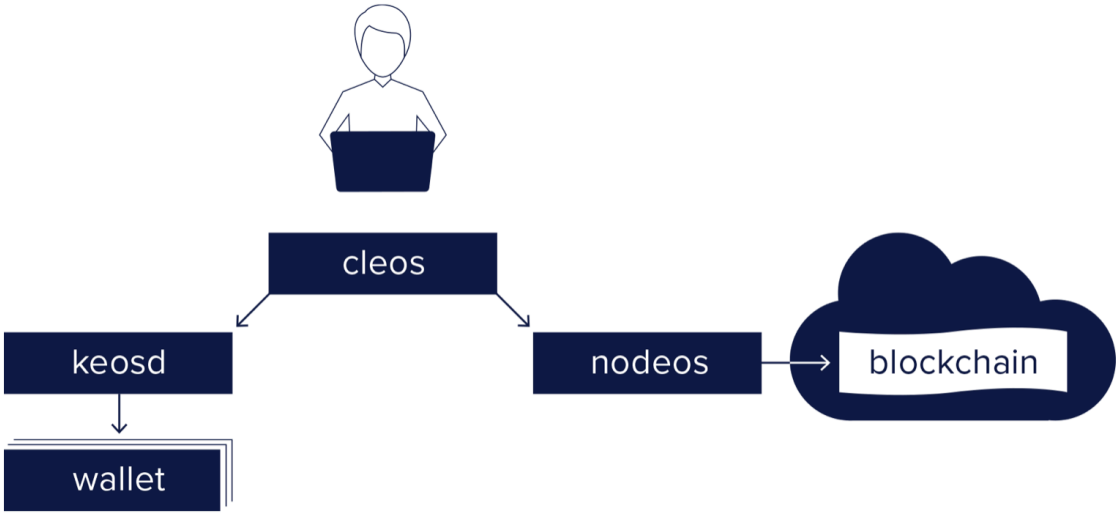


Figura 19 Minería en Blockchain

FUENTE: Sitio Web: https://developers.eos.io/manuals/eos/latest/index

Según (EOSIO, 2021) EOS.IO es una plataforma para la creación y lanzamientos de aplicaciones distribuidas y contratos inteligentes.

Esta plataforma trae consigo una serie de herramientas, que se describen a continuación:

- *Nodeos*: Tal como se aprecia en la imagen anterior, *Nodeos* en el servicio principal que corre en segundo plano en cada nodo, y puede ser configurado para procesar contratos inteligentes, validar transacciones, producir bloques con transacciones válidas y confirmar bloques para guardarlos en la cadena de bloques.
- *Cleos*: Es una herramienta de línea de comando que interactúa con el API REST expuesto por *Nodeos.* Los desarrolladores pueden utilizar esta herramienta para liberar y probar los contratos inteligentes.

- *Keos*: Es un servicio que corre en segundo plano, que se encarga de almacenar las llaves privadas y de firmar mensajes digitales. Además, proporciona un medio de almacenamiento de claves seguro para que las claves se cifren en reposo en los archivos de billeteras asociados

2.8.3.2 Ethereum.

Desarrollador por Vitall Butterin, Ethereum es una plataforma de Blockchain fundada en 2014 que ofrece flexibilidad y adaptabilidad para múltiples industrias. Apareció poco después del *Bitcoin* pero introdujo nuevos conceptos como contratos inteligentes y *Dapps* que significa aplicaciones descentralizadas. Su creador construyó Ethereum como un Blockchain público para aplicaciones B2C; por lo tanto, su libro de mayor sin permisos puede ser difícil para el uso empresarial.

Esta plataforma usa el consenso *Proof-of-Work,* en el cual los mineros deben competir por los códigos de acceso digitales (llamados “Ether”) para completar las transacciones. La minería requiere grandes cantidades de energía para correr computadores con hardware especializado que resolverá algoritmos complejos, por lo que se aumenta el costo de mantener las redes.

2.8.3.3 Consensys Quorum

Según (EOS Costa Rica, 2021) Quorum fue lanzado en 2016 por la institución financiera J.P Morgan, es una plataforma de código abierto basada en *Ethereum* para soluciones en diferentes industrias. Enfocada en la industria, esta plataforma

actúa como una red con permisos que soporta transacciones publica y privadas que requieren una completa seguridad de los datos.

Quorum pretende solventar múltiples preocupaciones en el sector financiero, por lo tanto, ofrecen una alta privacidad y confidencialidad de los registros, un simple mecanismos de censo por votación llamado *QuorumChain*, y un rendimiento rápido cuando se procesan las transacciones a comparación de *Etherum*.

2.8.3.4 HyperLedger Fabric

Según (EOS Costa Rica, 2021) esta plataforma fue lanzada en 2016 por *Linux Foundation*, *Digital Asset* e *IBM*. Es un *Blockchain* de código abierto con una serie de funcionalidades modulares, escalables y seguros apuntando a soluciones de tamaño industrial. Una arquitectura modular le permitiría a los desarrolladores conectar componentes adicionales como ordenamiento o servicios de membrecías.

Soporta un libro de mayor basado en permisos, lo que significa es operado por un grupo de participantes conocidos e identificados, ofreciendo una manera segura para manejar interacciones entre las entidades. Esta plataforma soporta contratos inteligentes escritos en lenguajes de propósito general tales como *Java*, *Go* y *Node.js.* Esta plataforma como tal, no soporta una moneda por defecto pero motiva a los usuarios a desarrollar una usando *chaincode* o contratos inteligentes.

2.8.3.5 Corda

Según (EOS Costa Rica, 2021) es una plataforma de código abierto, lanzada en 2015 por una empresa de *Blockchain R3.* Esta plataforma permite la ejecución de contratos inteligentes para múltiples industrias, pero principalmente para el sector financiero. *Corda* integra una pared de fuego que permite conexiones entrantes solo de los nodos de *Corda* y usa un modelo estricto de privacidad que previene acceso no autorizado al *Blockchain*.

Corda fue construida como un libro de mayor distribuido, en el cual solo las entidades que participan en el negocio tendrán accesos para ver o almacenar transacciones. Esto permitirá mejorar la confidencialidad comparado con el *Hyperledger Fabric* que usa un canal separado por cada relación. *Corda* a su vez es más escalable en ese sentido dado que solo guarda negocios entre su red.

2.8.4 Ventajas de usar *Blockchain*

Según (IBM, 2021) se pueden identificar y resumir 3 beneficios del *Blockchain*:

1. Mayor confiabilidad: Como miembro de una red única de miembros, Blockchain permite descansar tranquilo de que se esta recibiendo información precisa y a tiempo, y que la información será compartida únicamente a los miembros de la red a los que se le otorgo permisos.
2. Mayor seguridad: Genera una mayor conciencia en la importancia de la precisión de los datos brindada por todos los miembros de la red, y

que todas las transacciones son inmutables ya que todas las transacciones son guardadas de manera permanente, y nadie, absolutamente nadie puede borrar una transacción, ni siquiera un perfil administrador.

3. Con un registro inmutable compartido entre los miembros de una red, tiempos perdidos por temas de reconciliación son eliminados, y para acelerar las transacciones, un conjunto de reglas llamadas contratos inteligentes, pueden ser almacenados en los bloques y ser ejecutados de manera automática.

2.8.5 Desventajas de usar *Blockchain*

De la misma forma se pueden identificar algunas desventajas del *Blockchain* tomando como referencia a (Universidad de Alcalá, 2018):

1. Dificultad de implementación: Al ser una tecnología disruptiva, puede presentar complejidad para implementar todos los protocolos requeridos para garantizar un correcto funcionamiento del servicio.
2. Desempleo: Eliminación de *intermediarios* para llevar a cabo diferentes transacciones.
3. Anonimidad: Al ser una red abierta, las transacciones que una persona realiza de compra y venta estarán disponibles para los demás, perdiendo así el anonimato de transacción.
4. Ineficiencia: La creación de bloques pueden representar horas y horas de consumo de electricidad por parte de los mineros de *Blockchain*

tratando de resolver u obtener un nuevo Hash valido para un bloque de datos.

CAPÍTULO III: MARCO METODOLÓGICO

Este capítulo tiene como objetivo identificar el tipo de enfoque de la investigación y la definición de los instrumentos utilizados para la misma. Adicionalmente se detallará la forma en la que se realizará la investigación.

3.1 Enfoques de la investigación

Según (Sampieri, 2014) existen tres tipos de enfoque para una investigación, que comparten ciertas características, y otras que son exclusivas de cada enfoque. A continuación, se describirán cada una de ellas y se definirá el enfoque de este proyecto de investigación.

3.1.1 Enfoque cualitativo

Según (Bernal, 2016) dice:

> "Su preocupación no es prioritariamente medir, sino cualificar, describir e interpretar el fenómeno (situación o sujeto) social a partir de rasgos determinantes, según sean percibidos por los elementos que están dentro de la situación estudiada. Los investigadores que utilizan el método cualitativo buscan entender una situación social como un todo, teniendo en cuenta sus propiedades y su dinámica (pág.72)".

3.1.2 Enfoque cuantitativo

Según (Bernal, 2016) afirma:

> "Fundamenta en la medición de las características de los fenómenos sociales, lo cual supone derivar de un marco conceptual pertinente al problema analizado una

serie de postulados que expresen relaciones entre las variables estudiadas de forma deductiva. Este método tiende a generalizar y normalizar resultados" (pág.72).

3.1.3 Enfoque mixto

Según (Sampieri, 2014) se define como:

"(…) la integración sistemática de los métodos cuantitativo y cualitativo en un solo estudio con el fin de obtener una "fotografía" más completa del fenómeno, y señala que éstos pueden ser conjuntados de tal manera que las aproximaciones cuantitativa y cualitativa conserven sus estructuras y procedimientos originales ("forma pura de los métodos mixtos"); o bien, que dichos métodos pueden ser adaptados, alterados o sintetizados para efectuar la investigación y lidiar con los costos del estudio ("forma modificada de los métodos mixtos")" (pág.534).

En conclusión y con base a las tres definiciones anteriores este proyecto de investigación busca desde un enfoque cualitativo obtener la percepción de los colaboradores de la RAS, que están estrictamente relacionados a los procesos de evaluación de campos agrícolas, a las transacciones de compra y venta de la producción agrícola.

3.2 Tipo de investigación

Según (Sampieri, 2014) existen tres tipos de investigación: la exploratoria, descriptiva y correlacional. A continuación, se explicará cada una de ellas y se definirá el tipo de investigación para este proyecto

3.2.1 Exploratoria

Según (Sampieri, 2014):

> "Los estudios exploratorios se realizan cuando el objetivo es examinar un tema o problema de investigación poco estudiado, del cual se tienen muchas dudas o no se ha abordado antes. Es decir, cuando la revisión de la literatura reveló que tan solo hay guías no investigadas e ideas vagamente relacionadas con el problema de estudio, o bien, si deseamos indagar sobre temas y áreas desde nuevas perspectivas" (pág.91).

3.2.2 Descriptiva

Según (Sampieri, 2014):

"Con los estudios descriptivos se busca especificar las propiedades, las características y los perfiles de personas, grupos, comunidades, procesos, objetos o cualquier otro fenómeno que se someta a un análisis" (pág.92).

3.2.3 Correlacional

Según (Sampieri, 2014) afirma:

> "Este tipo de estudios tiene como finalidad conocer la relación o grado de asociación que exista entre dos o más conceptos, categorías o variables en una muestra o contexto en particular. En ocasiones sólo se analiza la relación entre dos variables, pero con frecuencia se ubican en el estudio vínculos entre tres, cuatro o más variables. Para evaluar el grado de asociación entre dos o más variables, en los estudios correlacionales primero se mide cada una de éstas, y después se cuantifican, analizan y establecen las vinculaciones. Tales correlaciones se sustentan en hipótesis sometidas a prueba" (pág.93).

Para efectos de este proyecto, el tipo de investigación será descriptiva, ya que se trabajará con el perfil de tres procesos de la RAS, los cuales son la evaluación agrícola, compra y venta de las producciones agrícolas.

3.3 Métodos de investigación

Este apartado define las diferentes etapas que existen para la recolección de información, ya que este será el insumo para llevar a cabo un análisis, depuración y obtención de resultados.

3.3.1 Análisis de contenido

Según (Sampieri, 2014) lo define como:

"(…) una técnica para estudiar cualquier tipo de comunicación de una manera "objetiva" y sistemática, que cuantifica los mensajes o contenidos en categorías y subcategorías, y los somete a análisis estadístico" (pág.251).

3.3.2 Fuentes de información

Esta sección busca detallar los dos tipos de fuentes primarias que existen: primarias y secundarias; y al mismo tiempo poder definir las fuentes para este proyecto de investigación.

3.3.2.1 Fuentes primarias o de primera mano

Tal y como lo dice (Sampieri, 2014):

> "Las referencias o fuentes primarias proporcionan datos de primera mano, pues se trata de documentos que incluyen los resultados de los estudios correspondientes. Ejemplos de fuentes primarias son: libros, antologías, artículos de publicaciones periódicas, monografías, tesis y disertaciones, documentos oficiales, reportes de asociaciones, trabajos presentados en conferencias o seminarios, artículos periodísticos, testimonios de expertos, documentales, videocintas en diferentes formatos, foros y páginas en internet, etcétera" (pág.61).

Y de manera más conceptual (Life Pacific University, 2022) lo define:

> "(…) es una fuente que se origina en el momento de un evento, un testigo del evento que describe el evento en sus propias palabras"

Para este proyecto de investigación se tomará como fuente primaria la información brindada por el equipo de tecnología y aplicaciones, y demás colaboradores relacionados a los procesos de evaluación, compra y venta agrícola de la RAS acerca del caso en estudio además de artículos científicos o de casos de éxito acerca de *Blockchain*.

3.3.2.2 Fuentes secundarias o de segunda mano

Una fuente secundaria a su vez se puede definir como una reorganización o interpretación de una fuente primaria.

Según (Life Pacific University, 2022):

> "Una fuente **secundaria** es aquella que fue creada más tarde por alguien que no tuvo experiencia de primera mano o participó en los eventos".

Como fuente secundaria se tomarán en cuenta todos aquellos artículos, libros, sitios web y documentaciones técnicas disponibles que se encuentren relacionadas al tema en análisis, adicionalmente cualquier documento interno suministrado por la RAS.

3.4 Población y muestra

3.4.1 Muestra

La muestra (Sampieri, 2014) la define como:

> "un subgrupo de la población de interés sobre el cual se recolectarán datos, y que tiene que definirse y delimitarse de antemano con precisión, además de que debe ser representativo de la población. El investigador pretende que los resultados encontrados en la muestra se generalicen o extrapolen a la población (en el sentido de la validez externa que se comentó al hablar de experimentos). El interés es que la muestra sea estadísticamente representativa" (pág.173).

3.4.2 Población

Según (Pimienta & De la Orden, 2017) la población se de

> "el conjunto de elementos que son parte del fenómeno o problemática a estudiar, y que poseen características similares, pues éstos serán la base de dicho estudio. A dicho conjunto, compuesto por la totalidad de los elementos, individuos o factores que forman parte de nuestro objeto de estudio y, en un lugar y tiempo determinados, poseen cualidades similares y observables, se le denomina población" (pág. 84).

Como conclusión y para efectos de este proyecto de investigación, la población estará conformada por 8 colaboradores, involucrando a los técnicos agrónomos,

equipo de TI y 3 de logística de ventas, los cuales tienen relación con el problema de investigación de este proyecto. Estas personas son parte de la junta directiva que se definió en el apartado 1.1.1.8.

Dicha población, se vera segmentada en muestras para poder abarcar los diferentes objetivos de la investigación, de la siguiente forma:

Población **Equipo de la RAS**	
Equipo de operaciones de la RAS (Agrónomos y logística de ventas)	6
Equipo de Tecnologías de la Información	2
Total	**8**

Tabla 3 Población y muestras

FUENTE: Elaboración propia.

3.5 Definición de instrumentos

En este apartado, se definirán los diferentes instrumentos con los que se cuenta en un proyecto de investigación para la recolección de datos que tienen un propósito en específico.

3.5.1 Cuestionario

(Sampieri, 2014) lo define como:

> "Conjunto de preguntas respecto de una o más variables que se van a medir". (pág.217)

Según (Martínez Ruíz, 2018) se define como:

> "(…) instrumento de investigación estructurado a partir de un conjunto de ítems o preguntas redactadas y estructuradas de forma coherente para ser planteado a los informantes de una determinada unidad de observación". (pág.110)

3.5.2 Encuesta

Según (Martínez Ruíz, 2018) es una técnica que consiste en la elaboración de un cuestionario compuesto por un conjunto de preguntas estandarizadas, es decir, ajustadas a un modelo o norma común, para conocer la opinión de un grupo amplio de personas.

3.5.3 Observación

Para (Martínez Ruíz, 2018):

> "La observación es la técnica etnógrafa por excelencia. Como su nombre lo indica, consiste en contemplar con atención el fenómeno social, generar información y registrarla para su posterior análisis e interpretación. Se utiliza para recabar datos empíricos (producto de la realidad), los cuales deben ser obtenidos sin que

interfieran prejuicios (etnocentrismo, dogmatismo) que suelen distorsionar la información que puede surgir de ellos" (pág.108).

3.5.4 Entrevista

Según (Martínez Ruíz, 2018):

"La entrevista es otro recurso para generar datos en una práctica investigativa.

Consiste en obtener información mediante una conversación entre dos o más personas para identificar la percepción, conocimiento y enfoque respecto de la cuestión que se estudia entre los involucrados o principales afectados. Suele emplearse para llevar a cabo estudios de carácter exploratorio, ya que permite recabar información de manera rápida" (pág.110).

3.6 Validación de instrumentos

Todo instrumento elegido para un proyecto de investigación debe de cumplir dos requisitos primordiales los cuales son:

- Confianza
- Validez

Según (Sampieri, 2014) lo define como:

> “La confiabilidad se calcula y evalúa para todo el instrumento de medición utilizado, o bien, si se administraron varios instrumentos, se determina para cada uno de ellos. Asimismo, es común que el instrumento contenga varias escalas para diferentes variables o dimensiones, entonces la fiabilidad se establece para cada escala y para el total de escalas” (pág.294).

Así mismo, (Sampieri, 2014) afirma para el segundo requisito:

> “la evidencia sobre la validez del contenido se obtiene mediante las opiniones de expertos y al asegurarse de que las dimensiones medidas por el instrumento sean representativas del universo o dominio de dimensiones de las variables de interés (a veces mediante un muestreo aleatorio simple). La evidencia de la validez de criterio se produce al correlacionar las puntuaciones de los participantes, obtenidas por medio del instrumento, con sus valores logrados en el criterio”

Los instrumentos que se emplearán para este proyecto de investigación serán las entrevistas y los cuestionarios, que serán aplicados a los grupos o muestras definidas en el apartado anterior, cumpliendo de esta manera los dos requisitos descritos anteriormente para dar seguridad de la información obtenida y así poder llevar a cabo los objetivos de este proyecto de investigación.

3.7 Operacionalización de las variables

A continuación, se detallan las variables para la recolección, medición y análisis de los datos, que permitirán llevar a cabo los objetivos definidos en este proyecto de investigación.

Objetivos Específicos	Variables	Indicadores	Conceptualización	Instrumentalización
Investigar acerca de los beneficios del uso del *Blockchain* y los elementos de arquitectura requeridos para implementar esta tecnología como solución al problema de investigación.	Beneficios del *Blockchain* y su implementación	• Casos de éxito del uso del *Blockchain* • Ejemplos de arquitecturas de *Blockchain*	*Blockchain*: estructura de datos, cuya información se agrupa en bloques los cuales son vinculados a un bloque precedente. Implementación: Poner en funcionamiento o llevar a cabo una cosa determinada.	Artículos científicos
Definir el conjunto de datos que requieren ser almacenados para la transacción de evaluación compra y venta agrícola, dentro de cada bloque cifrado de *Blockchain.*	Transacción de evaluación agrícola, compra y venta agrícola.	• Datos requeridos en la transacción de evaluación. • Datos requeridos en la transacción de venta agrícola. • Datos requeridos en la transacción de compra agrícola.	Datos: Información concreta sobre hechos, elementos, etc., que permite estudiarlos, analizarlos o conocerlos. Transacción: Trato o convenio por el cual llegan a un acuerdo comercial, generalmente de compraventa.	Cuestionario y entrevista

Objetivos Específicos	Variables	Indicadores	Conceptualización	Instrumentalización
Diseñar un diagrama de infraestructura que permita unificar todos los servicios necesarios en la nube para poder tener una solución que cumpla con el requerimiento de software.	Diagrama de arquitectura con la solución de software en la nube	• Creación del diagrama de arquitectura	Diagrama: Representación gráfica de las variaciones de un fenómeno o de las relaciones que tienen los elementos o las partes de un conjunto. Arquitectura: Estructura o forma en que algo está ordenado, dispuesto o construido.	Diagrama de arquitectura
Programar un API REST que permita la creación y obtención de bloques en el servicio de *Blockchain* incluyendo las pruebas unitarias requeridas para asegurar el correcto funcionamiento del API.	Programación del API REST para el uso del *Blockchain*	El correcto funcionamiento del API, haciendo uso de los recursos de *Blockchain*	Programación: Dar las instrucciones necesarias a una máquina o aparato para que realice su función de manera automática.	API REST

Objetivos Específicos	Variables	Indicadores	Conceptualización	Instrumentalización
Implementar un plan piloto de lanzamiento de la solución, que permita hacer uso del microservicio para efectos de pruebas y mantenimientos.	Plan piloto para el uso del Microservicio de *Blockchain*	Diseño de un plan piloto para utilización del microservicio	Implementación: Poner en funcionamiento o llevar a cabo una cosa determinada. Plan piloto: esfuerzo temporal que se asume para probar la viabilidad de una solución exclusiva del sistema propuesta	Capacitación técnica y documentación

3.8 Diseño de la investigación

A continuación, se describen las fases y etapas que comprenden este proyecto, y las técnicas y herramientas que se utilizaron en cada etapa.

Figura 20 Fases de la investigación
FUENTE: Elaboración propia

3.8.1 Investigación

En esta fase se llevará a cabo una investigación acerca de la tecnología de *Blockchain,* y cuales son los beneficios que esta tecnología ha traído a nuestra vida cotidiana.

Para llevar a cabo dicha investigación se hizo uso de las siguientes técnicas:

1. Búsqueda de artículos científicos fundamentados y publicados por fuentes confiables de información.
2. Búsqueda de información y documentación de los servicios de *Blockchain* que existen en el mercado actualmente, el cual permitió sustentar elementos conceptuales en el marco teórico.

3.8.2 Análisis

En esta fase se definirán el conjunto de requerimientos para el almacenamiento y operatividad de la transacción de evaluación, compra y venta agrícola, dentro de cada bloque cifrado de Blockchain.

Para poder lograr lo que se describió anteriormente se harán uso de las siguientes herramientas y técnicas:

1. Técnica de entrevista: se hará uso de una entrevista, la cual será aplicada al equipo de TI, que fueron identificados como muestra de la población en estudio. Esta entrevista, tendrá como objetivo poder obtener los datos requeridos para poder cumplir los objetivos 2 y 3 definidos en el apartado 1.3.2. Esta entrevista, se llevará a cabo de manera virtual, utilizando la plataforma de Office 365 que la Universidad Hispanoamericana pone a disposición del estudiante, específicamente la herramienta Teams, y se buscará responder una serie de preguntas previamente establecidas, y que se encuentran en el anexo 7.3.

2. Técnica envío de cuestionario: se aplicará un cuestionario al equipo de operaciones de la RAS para complementar y llevar a cabo el objetivo 2 definido en el apartado 1.3.2. Para este cuestionario se hará uso de la plataforma Office 365 que la Universidad Hispanoamericana pone a disposición del estudiante, en esta se encuentra una herramienta llamada Microsoft Forms que permite crear formularios con preguntas y que dichos formularios sean respondidos por diferentes personas. La tabulación de las respuestas también se puede obtener de esta herramienta, exportando las respuestas en un archivo CSV.
3. Uso del marco de trabajo *Scrum:* Para la definición y documentación de los requerimientos, se hará uso de las historias de usuario las cuales contienen un estructura que se adopta de manera efectiva a los alcances y tiempos del proyecto.

3.8.3 Diseño

En esta fase se diseñará un diagrama de infraestructura que permita unificar todos los servicios necesarios en la nube para poder tener una solución que cumpla con el requerimiento de software.

Para poder llevar a cabo lo descrito anteriormente se utilizaron las siguientes técnicas y herramientas:

1. Técnica de entrevista: se hará uso de una entrevista, al igual que la fase anterior, esta entrevista de igual manera será aplicada al equipo de TI, que

fueron identificados como muestra de la población en estudio. Esta entrevista, se llevará a cabo de manera virtual, utilizando la plataforma de Office 365 que la Universidad Hispanoamericana pone a disposición del estudiante, específicamente la herramienta Teams, y se buscará responder una serie de preguntas previamente establecidas, y que se encuentran en el anexo 7.3.

2. Técnica de diagrama: Para poder identificar y diseñar los diferentes componentes que integran el diagrama de arquitectura, se utilizará la herramienta Draw.io que contiene iconos representativos de servicios en la nube y que facilitan la interpretación de la solución a un nivel general.
3. Uso de arquitectura de Microservicio: Como se describió en el marco teórico en la sección 2.5, existen diversas arquitecturas de software, pero tomando en cuenta las respuestas de la entrevista expuesta en el primer punto, se acordó utilizar la arquitectura de microservicio, ya que con esto se logrará contar con un software independiente, no dependerá de otras herramientas previamente desarrolladas por la RAS, pero si podrá ser utilizada por todas estas aplicaciones.

3.8.4 Desarrollo

Para esta fase se programará un API REST que permita la creación y obtención de bloques en el servicio de *Blockchain* incluyendo las pruebas unitarias requeridas para asegurar el correcto funcionamiento del API.

Para llevar a cabo la fase de desarrollo se utilizarán diferentes técnicas y herramientas que se definieron en el marco conceptual.

1. Técnica de entregables por *sprint*: Según se definió en la fase de análisis, se utilizará el marco de trabajo de *Scrum,* para efectos de desarrollo se utilizaran *sprints* de trabajo con una duración de 7 días, donde se abarcarán las historias de usuario priorizadas y que puedan completarse en esta caja de tiempo. Adicionalmente, al finalizar la semana de trabajo, se presentarán los avances al equipo de TI de la RAS y seguidamente se planeará el trabajo de la siguiente semana, estas sesiones se llevarán a cabo de manera virtual , nuevamente utilizando la herramienta de Microsoft Teams.
2. Desarrollo de API REST: Como parte de las interfaces de comunicación de software, se utilizará un API utilizando los estándares REST. Para desarrollar este API se utilizará la tecnología de .Net Core y el lenguaje de programación C#. Adicionalmente, se utilizarán herramientas como *Swagger* para la especificación del API y se utilizara NUnit para desarrollar las pruebas unitarias.
3. Software en la nube: Es importante recalcar que la programación del API se orientará a la utilización de servicios en la nube, en este caso se utilizará Microsoft Azure como proveedor de servicios en la nube.
4. Uso de servicio EOS.IO: El desarrollo del API se basará en la utilización del servicio de *Blockchain* EOS.IO, ya que este servicio ofrece la flexibilidad de creación de contratos inteligentes, que pueden ser adaptados para los requerimientos que se analizaron con la RAS.

3.8.5 Implementación

En esta fase se implementará un plan piloto de lanzamiento de la solución, que permita hacer uso del microservicio para efectos de pruebas y mantenimientos.

Para llevar a cabo lo descrito anteriormente, se utilizarán las siguientes herramientas y técnicas:

1. Sesiones de traspaso de conocimiento: Se llevará a cabo una sesión técnica con el equipo de TI de la RAS, donde se explicará toda la solución y como esta constituida, adicionalmente, se revisará la estructura de programación para asegurar el entendimiento del equipo. Esta sesión se llevará a cabo de manera virtual, utilizando la herramienta de Microsoft Teams.
2. Técnica de documentación: Se hará uso de la técnica de archivos Readme.md los cuales buscan la documentación técnica de los componentes que integran la solución, además buscan dar una guía al programador de como dar seguimiento al desarrollo de la aplicación y como poder utilizarla de manera local.
3. Implementación en la nube: Haciendo uso de la herramienta Microsoft Azure DevOps, se desarrollará un flujo que permita publicar el API REST de manera automática luego de cada integración de nuevas funcionalidades, esto permitirá reducir el tiempo que se requiere para liberar la solución en los servicios en la nube.

3.9 Matriz de coherencia

Objetivo	Entregable	Fase o Etapa	Técnicas recolección de la información	Instrumentos	Temas relacionados para marco teórico
Investigar acerca de los beneficios del uso del *Blockchain* y los elementos de arquitectura requeridos para implementar esta tecnología como solución al problema de investigación	El primer entregable será llevar a cabo una investigación de la tecnología *Blockchain*, en este caso orientada a servicios en la nube de Microsoft Azure, ya que este es el servicio en la nube contratado por la RAS.	Investigación	- Búsqueda de artículos científicos fundamentados y publicados por fuentes confiables de información. - Búsqueda de información y documentación de los servicios de *Blockchain* que existen en el mercado actualmente	Artículos científicos	- Blockchain

Definir el conjunto de requerimientos para el almacenamiento y operatividad de la transacción de evaluación, compra y venta agrícola, dentro de cada bloque cifrado de *Blockchain*	El segundo entregable estará relacionado al levantamiento de requerimientos para la definición de los datos que se almacenarán por cada transacción de auditoría, compra y venta agrícola	Análisis	- Uso de la herramienta *Microsoft Forms*. - Uso de la herramienta *Microsoft Teams.*	Cuestionario y entrevista	- Ingeniería - Ingeniería de Software - Desarrollo ágil del software
Diseñar un diagrama de infraestructura que permita unificar todos los servicios necesarios en la nube para poder	El tercer entregable será un diagrama de arquitectura en la nube que incluya todos los diversos servicios requeridos para	Diseño	- Uso de la herramienta *Microsoft Teams.* - Uso de la herramienta Draw.io	Cuestionario y diagrama de arquitectura	- Arquitecturas de Software

tener una solución que cumpla con el requerimiento de software.	poder soportar un API REST y que este a su vez pueda consumir un servicio de *Blockchain*.				
Programar un API REST que permita la creación y obtención de bloques en el servicio de *Blockchain* incluyendo las pruebas unitarias requeridas para asegurar el correcto	El cuarto entregable será la programación de un API que contenga los estándares REST	Desarrollo	- Sesiones de planeación y revisión utilizando Microsoft Teams	API REST	- Blockchain - Interfaces de comunicación de Software - Computación en la nube - Desarrollo ágil del software

funcionamiento del API					
Implementar un plan piloto de lanzamiento de la solución, que permita hacer uso del microservicio para efectos de pruebas y mantenimientos	El quinto entregable será el diseño de un plan de piloto de lanzamiento del microservicio	Implementación	- Sesión de traspaso de conocimiento mediante Microsoft Teams	Capacitación técnica y documentación	- Computación en la nube - Desarrollo ágil del software

CAPÍTULO IV: DIAGNÓSTICO DE LA SITUACIÓN ACTUAL

4.1 Diagnóstico administrativo u operativo

La RAS en Costa Rica tal como se menciona en la sección 1.1.1.8, se encuentra constituida por 1 director ejecutivo, 1 director técnico y de desarrollo, también cuenta con 1 director de servicios corporativos, 1 gerente de manejo de proyectos, 1 gerente de innovación y conocimiento, 1 coordinador técnico, 1 especialista técnica, 2 miembros del equipo de tecnología y desarrollo de aplicaciones, finalmente cuenta con 1 coordinador de programas en India, 1 contadora y una asistente ejecutiva.

4.1.1 Proceso de evaluación de campo agrícola

A continuación, se describe el paso a paso del proceso de evaluación en los campos agrícolas:

1. Los expertos de campo deben ingresar y autenticarse en la plataforma de la RAS para poder descargar la plantilla de evaluación según sea el alcance de la evaluación que se requiera, los expertos de campo no necesariamente son funcionarios directos de la RAS.
2. Los expertos de campo descargan de la plataforma de la RAS una plantilla en Excel antes de realizar su inspección en una determinada finca, se realiza con Excel por el limitado acceso de internet desde los campos de cultivo agrícola.
3. Los expertos de campo completan la hoja de criterios de evaluación, los cuáles se encuentran segmentados en áreas, metas, objetivos e indicadores. Lo que da como resultado, el completar un cuestionario organizado

jerárquicamente que permitirá determinar si el campo agrícola cumple con los criterios de evaluación.

4. Los expertos de campo completan la información general de la evaluación que se encuentra en otra hoja en la plantilla de Excel, aquí se completan datos tales como:
    a. Área de la finca auditada
    b. Área de producción
    c. Cantidad de empleados (Hombres y mujeres por separado)
    d. Uso de la tierra
    e. Consumo de combustible
    f. Consumo de electricidad
    g. Si es ganadería, el tipo de animales que se tiene
    h. Si es cultivo, que tipo de cultivo es. Ejemplo, café, banano, maíz.
    i. Estimación de producción con diferentes unidades de medida.
    j. Cumplimiento de prácticas con base al cultivo y geografía.
5. Una vez que se recopila la información descrita anteriormente, los encargados de campo cargan la plantilla de Excel nuevamente al sistema para almacenar la información en una base de datos y poder obtener resultados de la evaluación.
6. El sistema realiza una serie de validaciones con base a reglas definidas por la RAS.
7. Una vez se completa el proceso de resultados, el experto de campo entrega un reporte al dueño de la finca o a la cooperativa de las fincas con los

aspectos a mejorar ya sea para una certificación o para la mejora continua de la producción agrícola en relación con el desarrollo del medio ambiente.

Este proceso no se encuentra documentado por parte de la RAS, por lo tanto se toma referencia según (Jiménez & Rodríguez, 2021).

4.1.2 Proceso de compra y venta agrícola

Tal como se identificó en la justificación del proyecto en la sección **Error! Reference source not found.**, el proceso de compra y venta lo realiza cada finca o cooperativa de fincas que se encuentra asociadas a la RAS, lo que da como resultado una variedad de ejecuciones de estos procesos, ya que algunas fincas utilizan papel y lápiz para llevar la transacción y otras utilizan sistemas informáticos como hojas de cálculo con macros, herramientas desarrolladas a la medida sin grandes especificaciones, esto para sus respectivos controles.

4.2 Diagnóstico técnico

El equipo de tecnología esta conformado por 2 miembros, 1 de ellos es el gerente de tecnología y el otro sería el especialista, ambos tienen a cargo el funcionamiento lógico y operativo del departamento.

A continuación, se describe el funcionamiento lógico y de desarrollo de software que lleva a cabo el equipo de TI de la RAS según (Jiménez & Rodríguez, 2021) y (Jiménez & Rodríguez, Entrevista de requerimientos, 2021):

1. El equipo de TI se encarga de llevar a cabo los procesos de soporte, mantenimiento y desarrollo de aplicaciones.
2. Cuando los proyectos de desarrollo implican un involucramiento grande de tiempo buscan invertir mediante terceros para el desarrollo de nuevas aplicaciones que eventualmente necesitare la RAS.
3. La RAS cuenta con su infraestructura en la nube, utilizan Microsoft Azure donde publican sus aplicaciones y los diferentes recursos de tecnología que necesitan. No cuentan con servidores físicos.
4. Las RAS cuenta con una red privada configurada en Microsoft Azure, para poder acceder a sus recursos tales como almacenamiento y aplicaciones.
5. Utilizan herramientas como Office 365 para la administración y la ofimática de la RAS.
6. Para el almacenamiento de código fuente de las aplicaciones se utiliza Microsoft Azure DevOps, desde esta herramienta también controlan el ciclo de vida del software y la liberación de este a los ambientes productivos.
7. Para efectos de desarrollo de aplicaciones no cuentan con un patrón para desarrollo, ni documentos donde se especifiquen sus estándares de programación.
8. Actualmente no cuentan con una federación de usuarios a lo largo de todas sus plataformas, por lo que cada aplicación o futuro desarrollo se debe incluir una capa de seguridad para dar acceso a sus usuarios.

4.3 Diagnóstico de percepción

4.3.1 Entrevista no estructurada al equipo de TI

Esta entrevista es la más importante para determinar y ubicar el proyecto en la matriz de necesidad y prioridad que tiene la RAS, se puede revisar el anexo 7.3. Esta entrevista fue efectuada a los miembros de TI y es importante recalcar que la misma permitió enfocar el alcance que se esperaba de una tecnología moderna como lo es *Blockchain*.

Adicionalmente, permitió elaborar el diagnóstico técnico, que brindó una mejor definición de lo que se requería construir a nivel de uso de servicios en la nube de Microsoft Azure de la RAS.

Por último, la entrevista con el equipo de TI permitió elaborar la lista de requerimientos base a nivel operativo, y así determinar los datos que debían validarse y almacenarse en los bloques cifrados de *Blockchain* para las transacciones de evaluación, compra y venta agrícola.

4.3.2 Cuestionario no estructurado al equipo operativo de la RAS

También se envía un cuestionario al equipo operativo de la RAS, con el objetivo de reforzar la información obtenida de la entrevista con el equipo de TI, se puede revisar el anexo 7.1.

Gracias a este cuestionario se logró identificar la posibilidad de unificar unidades de medición de producción y la importancia de estandarizar las unidades de medida que deberían ser aceptadas por la solución que se quiere desarrollar.

Además, se logró observar una coincidencia de respuestas con respecto a la modificación de la información a lo largo del tiempo, ya que esto sin duda representa la importancia de contar con un registro inmutable para garantizar la fiabilidad de los datos.

4.4 Conclusiones del diagnóstico

Tomando como base la información recopilada mediante la entrevista realizada al equipo de TI y el cuestionario enviado al equipo operativo de la RAS, a continuación se presenta un análisis de brechas:

1. La RAS no cuenta con una federación de usuarios para la autenticación en las diferentes aplicaciones que poseen: Para resolver esta brecha se podría incorporar un servicio de autenticación de usuarios como por ejemplo Okta, el cual permite tener usuarios centralizados y estructurados por rol.
2. Las RAS utiliza archivos de Excel para poder llevar a cabo las evaluaciones en campo: Pese a que esta brecha se encuentra fuera del objeto de estudio y del alcance por ser una aplicación ya existente en la RAS, es importante proponer una mejora a este proceso mediante algún desarrollo nativo para dispositivos móviles que pueda trabajar sin internet y posteriormente una sincronización al sistema principal.

3. Para las evaluaciones de fincas existen una gran cantidad de datos que se recopilan, no pueden ser alterados con el tiempo y se necesita tener una trazabilidad de los hechos: Para esta brecha se podría registrar un bloque inmutable de Blockchain que contenga solo los datos requeridos que hacen referencia a la transacción que se llevo a cabo por el sistema de la RAS, por lo tanto se podría desarrollar un API que exponga un *endpoint* que a su vez pueda ser utilizado por la aplicación de la RAS para establecer un nuevo bloque en la cadena, de esta manera no se podría cambiar la información capturada.
4. Todas las evaluaciones que se realizan en las diferentes fincas cuentan con una variedad de unidades de medida dependiendo del cultivo o producción que se tenga: Para solventar esta brecha, se pueden estandarizar las unidades de medida, por ejemplo, utilizar las toneladas como medida estándar pero de igual manera guardar los datos originales para no perder ninguna trazabilidad de la información. Adicionalmente, se puede implementar un proceso de validación para que el API solo permita unidades de medida previamente establecidas.
5. Para las transacciones de compra y venta existen una gran cantidad de datos que se recopilan y de diferentes maneras por parte de las fincas o cooperativas de fincas: Para solucionar esta brecha se puede desarrollar un módulo adicional a la aplicación existente de la RAS para que los usuarios de la plataforma puedan registrar sus transacciones de compra y venta, y adicionalmente elaborar un API que exponga un *endpoint* que a su vez pueda

ser utilizado por la aplicación de la RAS para establecer un nuevo bloque en la cadena, que contenga todos los datos requeridos en la transacción de compra y venta, y de esta manera no se podría cambiar la información capturada.

6. Constante validación y control de los saldos de producción de los integrantes de la RAS: Para controlar la existencia de producción certificada por la RAS y el constante monitoreo de las capacidades de producción, se puede desarrollar un nuevo contrato de *Blockchain*, que permita llevar el control de las producciones agrícolas mediante *tokens*, ya que estos representaran las unidades disponibles de producción, y adicionalmente controlaran la producción vendida y comprada por parte de las fincas y cooperativas de fincas que forman parte del *IHub* de la RAS.

CAPÍTULO V: PROPUESTA DE PROYECTO

5.1 Investigación de casos de éxito del uso de *Blockchain*

Este apartado tiene como objetivo investigar acerca de los beneficios del uso del *Blockchain* y los elementos de arquitectura requeridos para implementar esta tecnología como solución al problema de investigación.

A continuación, se presentan tres casos de uso del *Blockchain* que han sido exitosos y que han permitido darla a conocer a nivel mundial, adicionalmente estos casos permitirán fundamentar los beneficios que llevaron a elegir esta tecnología como eje principal de la solución.

5.1.1 The Plastic Bank

Figura 21 Iniciativa de "*The plastic bank"*

FUENTE: Sitio Web: https://www.forbes.com/sites/annefield/2017/11/29/the-plastic-bank-using-plastic-to-create-a-global-currency-for-the-poor/?sh=81b324e74332

Según (Field, 2017) "The plastic bank" es una plataforma para los indigentes que hace uso de la tecnología blockchain. La plataforma fue fundada en el año 2013, y la compañía tiene como objetivo pagar a las personas de bajos recursos o que viven en las calles, por la recolecta y entrega de desechos plásticos a centros de reciclaje. Esta iniciativa se lleva a cabo en países como Haití y Filipinas, con planes de expansión a Brasil e Indonesia, y posteriormente a Sur África, Panamá y la India.

Utilizando blockchain y trabajando de la mano con IBM, The plastic bank creó una aplicación bancaria que está dirigida a los recolectores, la cual les permite saber cuánto dinero recolectaron y les proporciona una billetera digital que puede ser utilizada para retirar los fondos. Trayendo consigo un beneficio importante a este grupo social, ya que muchas de estas personas no califican para obtener una cuenta bancaria.

Este artículo describe una serie de beneficios del Blockchain, siendo una tecnología que contribuye al desarrollo social, brindando una fuente de ingreso a personas con escasos recursos en países afectados por la pobreza. Adicionalmente brinda un beneficio al medio ambiente, impulsando el reciclaje en países con altas cuotas de contaminación a nivel mundial.

Tomando como referencia este artículo, existen varias aristas que comparten características con este proyecto de investigación, por ejemplo:

- Implementación de procesos de control y manejo de transacciones.
- Velar por el desarrollo armónico de las comunidades, grupos sociales y medio ambiente.
- Transacciones de compra y venta.
- Escalabilidad y alcance global.

5.1.2 Bitcoin

Figura 22 Ilustración popular del Bitcoin

FUENTE: Sitio Web: https://www.reuters.com/technology/us-becomes-largest-bitcoin-mining-centre-following-china-ban-2021-10-13/

Posiblemente el caso de éxito más grande de blockchain es Bitcoin, en el 2021 con una capitalización de mercado de más de un trillón de dólares y con cientos de miles de transacciones al día (Yahoo Finance, 2021).

La ventaja que le ha permitido a Bitcoin llegar a este punto es blockchain, el cual le permite poder operar sin la necesidad de tener una autoridad central, esto reduce el riesgo de manipulación y reduce considerablemente las tarifas de transferencia. Bitcoin también presenta un componente de anonimato a diferencia de los bancos

u otras instituciones financieras, estos factores han contribuido a la rápida adopción de Bitcoin por parte de gran cantidad de personas e instituciones.

Bitcoin logró dar a conocer que es blockchain a nivel mundial y el potencial que tiene, gracias a esto han surgido incontables proyectos que utilizan blockchain para resolver diferentes problemas.

Otro gran logro de Bitcoin y posiblemente el más grande fue demostrar que es posible crear una plataforma totalmente descentralizada que no tenga dependencia de una institución, persona o gobierno lo cual pone en una posición más equiparada a las personas que buscan involucrarse en el proyecto.

De igual manera Bitcoin tiene algunos puntos en los cuales debe mejorar, su naturaleza descentralizada y anónima hace que sea más difícil su regulación, lo cual atrae a la plataforma personas que realizan actividades al margen de la ley o poco éticas.

Otro punto en el cual Bitcoin necesita mejorar es su huella ambiental, según (Coindesk & Feign, 2021) en Julio del 2021, Bitcoin requiere de 1719.51 kilovatio hora para confirmar una transacción, para ponerlo en perspectiva, esta es la cantidad de energía que consume un hogar durante 59 días en Estados Unidos, la principal preocupación es que con el crecimiento exponencial que tiene bitcoin (y las criptomonedas en general) que se agrave el daño ambiental ocasionado por satisfacer la demanda energética.

Figura 23 Consumo de energía del Bitcoin

FUENTE: Sitio Web: https://www.coindesk.com/business/2021/08/18/how-much-energy-does-bitcoin-use/

5.1.3 BurstIQ

Figura 24 BurstIQ mejora de salud

FUENTE: Sitio Web: https://www.forbes.com/sites/andreatinianow/2019/09/23/blockchain-technology-is-already-improving-lives-at-22-hospitals/?sh=59b82ce26c7d

Según (BurstIQ, 2021) es un proveedor de una red de datos para la industria de la salud basada en *Blockchain*. BurstIQ le permite a doctores y pacientes transferir datos médicos de manera segura mediante contratos de *Blockchain*.

BurstIQ utiliza *Blockchain* para crear lazos entre los diferentes registros de los pacientes, estos pueden ser consultas médicas, enfermedades o condiciones específicas del paciente, medicamentos entre otros. Para la industria de la salud esto tiene mucha utilidad ya que le permite a los doctores tomar mejores decisiones utilizando la información que se encuentra guardada de cada paciente, de igual

manera, le permite a los investigadores tener acceso a gran cantidad de información y perfiles lo cual eleva la posibilidad de alcanzar nuevos descubrimientos médicos.

Gracias a la utilización de *Blockchain*, BurstIQ puede asegurar la integridad de los datos y saber que no hay alteraciones en los mismos, esto es algo vital para la industria de la salud ya que el manejo de información incorrecta puede costar la vida de una o varias personas.

Según (Tinianow, 2019) *BurstIQ* usando sus datos en combinación con los sistemas de análisis de otra empresa llamada *Empiric*, lograron reducir el costo de las cirugías, ahorrándole al hospital decenas de millones de dólares, esto demuestra la utilidad de poder manejar este tipo de datos en gran cantidad y de manera segura.

De esta forma, se puede concluir que la utilización de la tecnología de Blockchain, brinda seguridad, confianza y trazabilidad de la información, evitando modificaciones o cambios inesperados de los datos que pueden generar desconfianza y poca exactitud de la información que se obtiene, que a su vez podría afectar tomas de decisión importantes en el cuidado del paciente y su salud.

5.2 Definición de datos requeridos para la transacción de evaluación, venta y compra agrícola.

El objetivo de este apartado es definir el conjunto de datos que requieren ser almacenados para la transacción de evaluación, compra y venta agrícola, dentro de cada bloque cifrado de *Blockchain*. Por lo tanto, se recopilarán los datos obtenidos de la entrevista aplicada al equipo de tecnología, y se integrarán las respuestas obtenidas del cuestionario aplicado al equipo de operaciones de la RAS, siendo ambos muestra de la población en estudio según el apartado 3.4.

Para la recolección de requerimientos, se hará uso de las técnicas descritas por el marco de trabajo de *Scrum* el cual se apalanca en historias de usuario para poder definir los requerimientos del sistema de Software. En este caso las muestras en estudio se definirán como “Usuario” de ahora en adelante para este capítulo y para la redacción de dichas historias de usuario.

Para tener más detalle de las preguntas realizadas en la entrevista al equipo de TI y el cuestionario realizado puede consultar en el anexo 7.1.

Adicionalmente, con el objetivo de validar y dar confianza de los datos obtenidos con los instrumentos de cuestionario y entrevista, se puede revisar el anexo 7.2 donde los resultados son aprobados por los miembros de la RAS, específicamente el equipo de IT.

5.2.1 Transacción de evaluación para certificación de campos agrícolas

A continuación, se enlistan los requerimientos definidos y aprobados por la RAS para tomar en consideración cuando se quiere registrar una transacción de auditoría:

ID: 1	**Épica:** Transacción de evaluación para certificación de campos agrícolas
Yo como	Usuario
Quiero	Que el servicio de escritura en los bloques registre los siguientes datos: • Id de Finca • Nombre de la Finca • Tipo de cultivo • Área disponible para cultivo
Para	Controlar y dar persistencia a los datos que se obtienen al auditar un campo agrícola y evitar que sean modificados.
Descripción • Crear un punto de acceso que permita guardar la información descrita anteriormente. • Todos los datos para esta transacción son requeridos. • Estos datos provienen de otros sistemas, estos datos ya fueron previamente revisados y validados, con base a las reglas de negocio de la RAS, por lo que no se deben validar nuevamente.	

ID: 2	**Épica:** Transacción de evaluación para certificación de campos agrícolas
Yo como	Usuario
Quiero	Que el servicio de escritura en los bloques registre los siguientes datos de evaluación de sistemas: • ID usuario de la transacción • Nombre del usuario en la transacción • Fecha de la transacción
Para	Dar trazabilidad y seguimiento de como suceden las transacciones y quien las lleva a cabo.

Descripción

- Los datos de ID usuario de la transacción y el nombre del usuario, serán tomados según del sistema que provenga la petición, mientras que la fecha de transacción será capturada en el momento que ocurre la transacción en el microservicio de *Blockchain*.
- Todos los datos para esta transacción son requeridos.

5.2.2 Transacción de compra y venta de producción agrícola

A continuación, se enlistan los requerimientos definidos y aprobados por la RAS para tomar en consideración cuando se quiere registrar una transacción de venta y compra:

ID: 3	**Épica:** Transacción de compra y venta de producción agrícola
Yo como	Usuario
Quiero	Que el servicio de escritura en los bloques registre los siguientes datos: • Id de comprador • Nombre del comprador • Id del vendedor • Nombre del vendedor • Volumen comprado • Unidad volumen comprado • Volumen vendido • Unidad volumen vendido • Tipo de cultivo • Monto vendido local • Monto vendido dólar • Monto comprado local • Monto comprado dólar
Para	Controlar y dar persistencia a los datos que se obtienen al momento en que se lleva a cabo una venta y compra de producción agrícola.
Descripción • Crear un punto de acceso que permita guardar la información descrita anteriormente. • Todos los datos para esta transacción son requeridos.	

• Estos datos provienen de otros sistemas, estos datos ya fueron previamente revisados y validados, con base a las reglas de negocio de la RAS, por lo que no se deben validar nuevamente.

ID: 4	**Épica:** Transacción de compra y venta de producción agrícola
Yo como	Usuario
Quiero	Que el servicio de escritura en los bloques registre los siguientes datos de evaluación de sistemas: • ID usuario de la transacción • Nombre del usuario en la transacción • Fecha de la transacción
Para	Dar trazabilidad y seguimiento de como suceden las transacciones y quien las lleva a cabo.
Descripción Los datos de ID usuario de la transacción y el nombre del usuario, serán tomados según del sistema que provenga la petición, mientras que la fecha de transacción será capturada en el momento que ocurre la transacción en el microservicio de *Blockchain*.	

ID: 5	**Épica:** Transacción de compra y venta de producción agrícola
Yo como	Usuario
Quiero	Que el servicio de escritura en los bloques calcule automáticamente los saldos del comprador y del vendedor con base a las transacciones históricas.
Para	Implementar Contratos Inteligentes para controlar el saldo solicitado.

ID: 6	**Épica:** Transacción de compra y venta de producción agrícola
Yo como	Usuario
Quiero	Que el servicio de escritura en los bloques valide los códigos de unidad de medida para los campos: • Unidad volumen comprado • Unidad volumen vendido
Para	Controlar y dar persistencia a los datos que se obtienen al momento en que se lleva a cabo una venta y compra de producción agrícola.
Descripción • El punto de acceso para registrar la transacción debe validar que los códigos ingresados se encuentren en esta lista de códigos: o t (Tonelada) o kg (Kilogramo) o g (gramos) o ton (Tonelada métrica) o lb (Libras) o oz (Onzas)	

5.2.3 Lectura de información guardada en el servicio de *Blockchain*

Como parte importante del desarrollo de esta solución, se requiere tener accesos a la información que se va a almacenar con las historias de usuario descritas anteriormente, por lo tanto, para resolver esta necesidad del negocio, se creó la siguiente historia de usuario:

ID: 7	**Épica:** Lectura de información del servicio de Blockchain
Yo como	Usuario
Quiero	Que el servicio de Blockchain permita consultar los bloques creados.
Para	Controlar y dar seguimiento a la información escrita.
Descripción Crear un punto de acceso en el API que permita retornar los datos ingresados al servicio de Blockchain.	

Con la definición de las historias usuario anteriores, es importante describir el mecanismo por el cual se dará seguimiento, se controlará el avance de las mismas.

Según (Jiménez & Rodríguez, Entrevista de requerimientos, 2021) se acordó el uso de la herramienta *Azure* DevOps, la cual es parte de *Microsoft Azure*. La RAS cuenta con todo el licenciamiento respectivo de esta herramienta, y objetivo de utilizarla será contar con un tablero, que permita visualizar todas las historias de usuario definidas y acordadas, adicionalmente poder ver el avance del proyecto y poder dar visibilidad a la RAS del trabajo realizado.

A continuación, se presenta un ejemplo de cómo se visualizan las historias de usuario creadas en la herramienta de *Azure* DevOps, las cuales se encuentran en la lista de requerimientos de la solución, conocido como *Backlog* según Scrum**:**

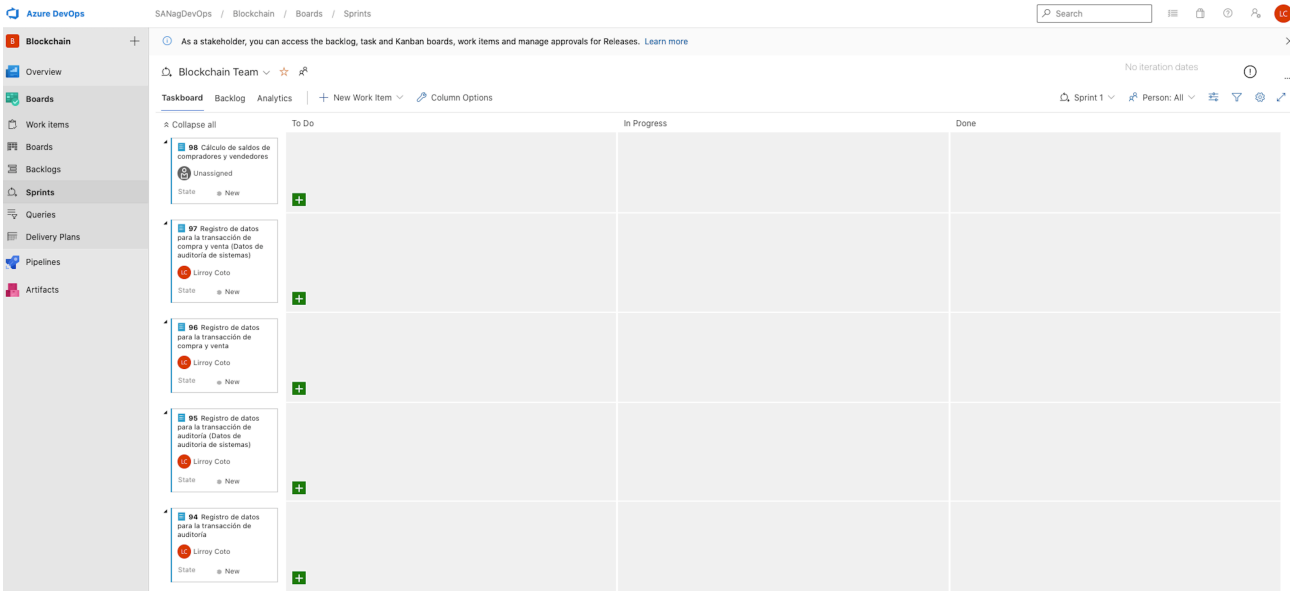
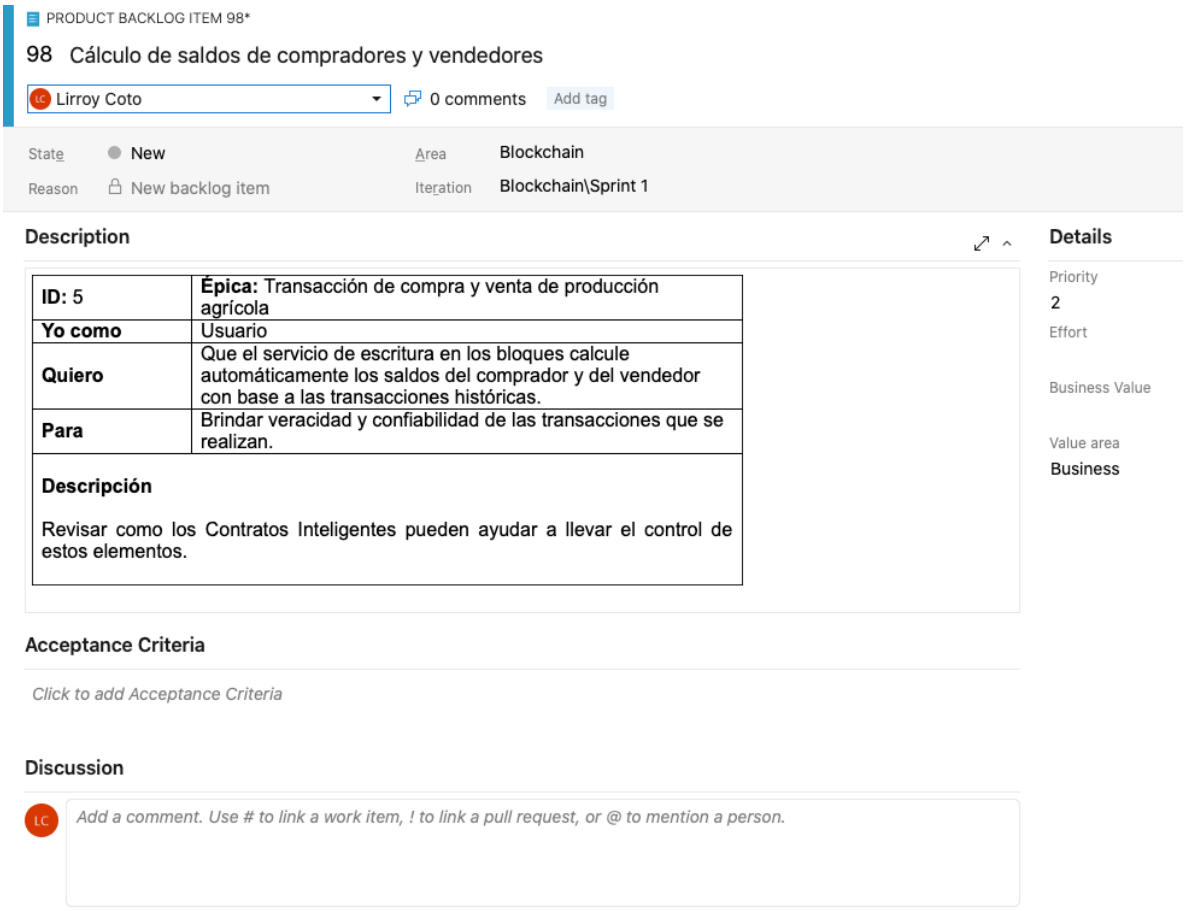


Figura 25 Dashboard Azure DevOps

FUENTE: Sitio Web: Sitio de Azure DevOps de la RAS

A continuación, se puede visualizar como ejemplo el contenido de la historia de usuario:



PRODUCT BACKLOG ITEM 98*

98 Cálculo de saldos de compradores y vendedores

Lirroy Coto | 0 comments | Add tag

State: New | Area: Blockchain

Reason: New backlog item | Iteration: Blockchain\Sprint 1

Description

ID: 5	**Épica:** Transacción de compra y venta de producción agrícola
Yo como	Usuario
Quiero	Que el servicio de escritura en los bloques calcule automáticamente los saldos del comprador y del vendedor con base a las transacciones históricas.
Para	Brindar veracidad y confiabilidad de las transacciones que se realizan.
Descripción Revisar como los Contratos Inteligentes pueden ayudar a llevar el control de estos elementos.	

Acceptance Criteria

Click to add Acceptance Criteria

Discussion

Add a comment. Use # to link a work item, ! to link a pull request, or @ to mention a person.

Details

Priority: 2

Effort

Business Value

Value area: Business

Figura 26 Detalle de historia de usuario

FUENTE: Sitio Web: Sitio de Azure DevOps de la RAS

5.3 Diagrama de arquitectura de la solución

El objetivo de este capítulo es diseñar un diagrama de infraestructura que permita unificar todos los servicios necesarios en la nube para poder tener una solución que cumpla con el requerimiento de software.

Según (Jiménez & Rodríguez, Entrevista de requerimientos, 2021) se confirmó que la RAS utiliza la infraestructura en la nube de Microsoft, por lo tanto, el diseño del diagrama se realizó con base a los servicios que ofrece este proveedor.

Adicionalmente, se buscará explicar cada uno de los elementos que conforman el diagrama, y como interactúan entre si para poder cumplir la funcionalidad del software.

Por otro lado, a lo largo de este capítulo se mencionarán las historias de usuario creadas a raíz del diseño de la arquitectura, y que responden a requerimientos no funcionales del sistema. El rol de dichas historias de usuario será representado por el alias “Sistema”.

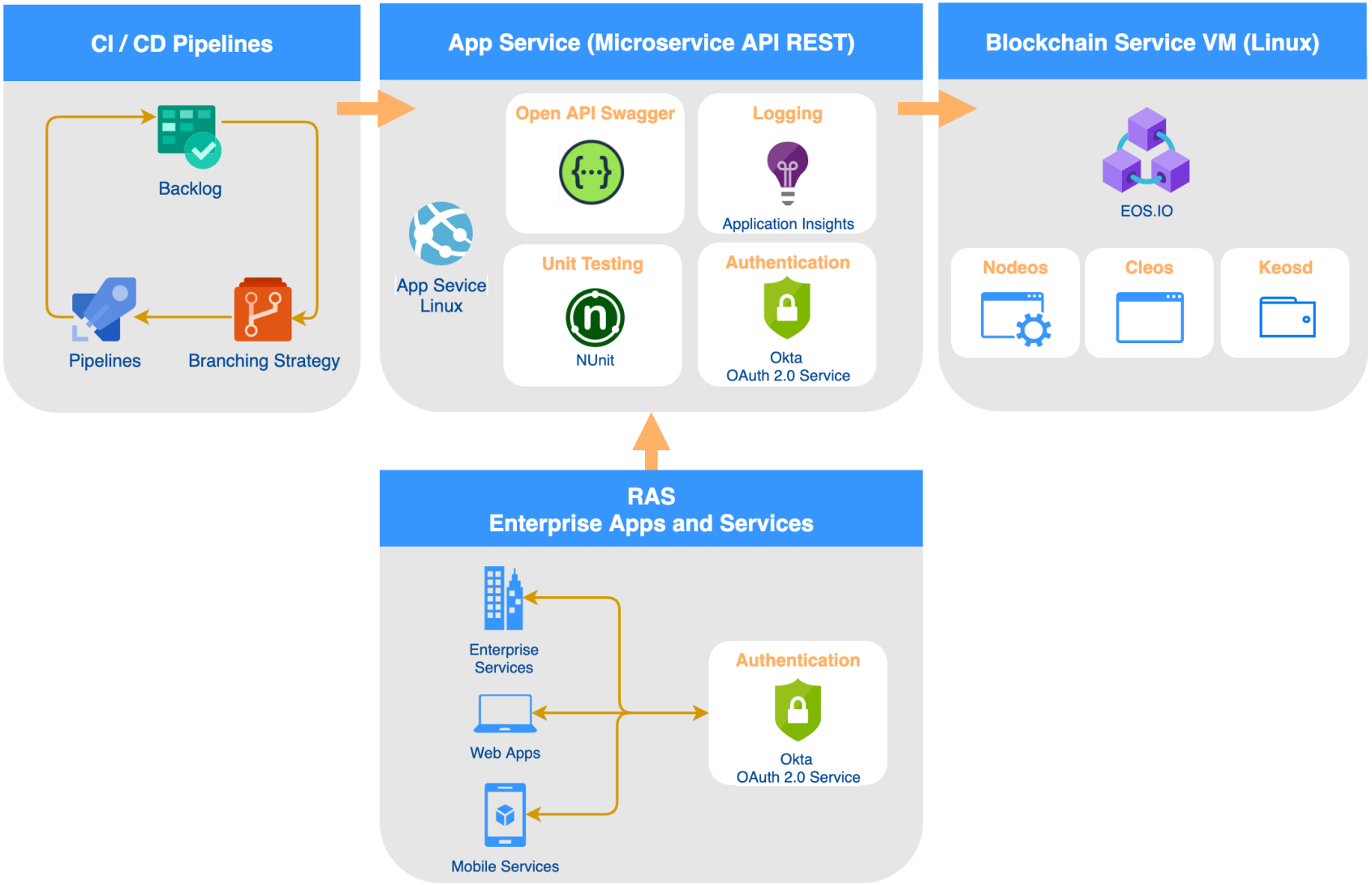


Figura 27 Diagrama de arquitectura del Microservicio de *Blockchain*

FUENTE: Elaboración propia

5.3.1 CI / CD Pipelines

Este apartado tiene como objetivo definir los componentes que permitirán dar un ciclo de vida del Software mediante la mejora continua y automatizaciones de lanzamiento de versiones de la solución.

5.3.1.1 Backlog

Este componente representa la lista de requerimientos o en este contexto las historias de usuario que componen esta solución. Si bien es cierto, este proyecto de investigación definirá una lista historias de usuario para poder liberar una primera versión del software, no se puede limitar su continuo mantenimiento y constante mejora del servicio, por lo tanto, se busca canalizar cualquier funcionalidad nueva, corrección o mantenimiento mediante Azure DevOps – Backlog.

5.3.1.2 Estrategia de ramas (Branching Strategy)

Para un correcto control de versiones de la solución, existe una herramienta que Azure DevOps pone a disposición, la cual es *Repos.*

En este componente se almacenará el código fuente de la solución, permitiendo así tener el control de la versión mas reciente la aplicación, la versión que esta siento probada para luego ser liberada a producción y además las versiones que incluyen nuevas funcionalidades o correcciones de la solución.

Esta herramienta se integra con el componente anterior Backlog, permitiendo relacionar una nueva historia de usuario con una versión de la solución que esta en proceso de desarrollo y revisión.

5.3.1.3 Pipelines

Una vez que se trabajó con la rama que o nueva versión de la aplicación, *Pipelines* se integra para poder hacer el lanzamiento de las nuevas funcionalidades o correcciones al ambiente que se requiera de manera automática, los cuales podrían ser:

- Producción
- QA
- UAT

Importante mencionar que en este componente se ejecutaran todas las pruebas unitarias creadas para garantizar que la nueva versión de la solución que se publica no dañe las funcionalidades existentes.

5.3.2 Servicio de aplicaciones (*App Services*)

Esta sección del diagrama representa el ecosistema el que residirá el microservicio de *Blockchain*, específicamente el API REST que se encargará de canalizar las peticiones al servicio de *Blockchain* que se explicará más adelante.

5.3.2.1 Microservicio API REST (Microservice API REST)

Este servicio detecta automáticamente las características y dependencias del software para brindar el hardware requerido para que esta pueda ser ejecutada sin problema.

Específicamente, en este componente se publicará la versión más reciente del API REST que se desarrollará.

Entre las principales características y dependencias que este servicio soportará, como resultado del desarrollo del API será .Net Core 5.0, siendo C# el lenguaje para el desarrollo del código.

Según (Jiménez & Rodríguez, Entrevista de requerimientos, 2021) se conversó acerca de si la RAS cuenta con algún estándar que se deba seguir, pero la respuesta que se obtuvo fue que quedaba a criterio del desarrollador de la solución.

5.3.2.2 Rastros de la aplicación (Application Insights)

El servicio de aplicaciones permite hacer uso de un componente llamado *Application Insights.* Este componente tiene como objetivo registrar todos aquellos eventos que fueron programados en el API REST para informar, alertar o registrar un error mediante una bitácora. Estos eventos se escriben en este componente, y podrán ser revisados en cualquier momento en caso de algún fallo o que se requiera alguna mejora en el sistema.

Tomando como base que el API REST se desarrollará con .Net Core 5.0 y C#, se realizará las implementaciones respectivas de la interfaz *ILogger* la cual, permite el registro de eventos de varios tipos:

- Información
- Advertencia
- Error
- Personalizados

5.3.2.3 Autenticación (Okta OAuth 2.0 Service)

El servicio de autenticación será suministrado por el equipo de la RAS, según (Jiménez & Rodríguez, Entrevista de requerimientos, 2021). Este servicio consiste la utilización de federación de identidad, esto quiero decir, que para acceder a las funciones del microservicio es requerido que la solitud contenga un *token* válido generado por Okta, este a su vez, será validado por el microservicio de igual manera con *Okta*.

Para llevar a cabo este requerimiento a nivel de programación se hará uso de un Paquete *NuGet* que proporciona .Net Core/C# para el manejo de autenticación.

5.3.2.4 Pruebas unitarias

Para el desarrollo de las pruebas unitarias se hará uso del framework NUnit, las mismas tendrán el objetivo de velar por el correcto funcionamiento de la lógica de negocio aplicada a la solución.

5.3.2.5 Open API esquema y documentación

Basado en la especificación de Open API se utilizará la herramienta *Swagger*, para documentar y definir la estructura del API, para que pueda ser consumido por los diferentes clientes o aplicaciones de la RAS.

5.3.3 Servicio de Blockchain (*Blockchain Service*)

El servicio de EOS será configurado en una máquina virtual, cuyo sistema operativo será Linux.

5.3.3.1 Nodeos

Este servicio admitirá peticiones mediante el puerto 8888, de esta manera el API para interactuar con la cadena de bloques podrá ser accedida por el microservicio liberado en el servicio de aplicación descrito en el punto anterior.

5.3.3.2 Cleos

La terminal de Cleos podrá ser accedida mediante una conexión remota utilizando SSH a la maquina virtual.

5.3.3.3 Keosd

El servicio estará disponible en la misma máquina virtual, y podrá ser accedido mediante Cleos, tal cual se menciona en el punto anterior.

5.3.4 Servicios y aplicaciones empresariales de la RAS (*RAS Enterprise Apps and Services*)

Este elemento del diagrama busca representar la forma en la que las herramientas existentes en la RAS podrán utilizar el microservicio de Blockchain ya se para la escritura o la lectura de los bloques cifrados.

Es importante que las aplicaciones de la RAS que requieran hacer uso del microservicio obtengan un *token* válido mediante *Okta*, para luego enviarlo en la solicitud, de otra manera, el servicio retornará un código de error 401, indicando que la petición no esta autorizada.

5.3.5 Requerimientos no funcionales de la solución con base en el diagrama de arquitectura

A continuación, se enlistarán todos los requerimientos no funcionales que fueron identificados en la elaboración del diagrama de arquitectura en conjunto con los requerimientos recolectados según (Jiménez & Rodríguez, Entrevista de requerimientos, 2021):

ID: 8	**Épica:** Requerimientos no funcionales
Yo como	Sistema
Quiero	Contar con un repositorio de código
Para	Controlar las versiones del código
Descripción El servicio de aplicaciones será construido con base en el presupuesto de la RAS y sus especificaciones de hardware.	

ID: 9	**Épica:** Requerimientos no funcionales
Yo como	Sistema
Quiero	Contar con un servicio de aplicaciones configurada en Microsoft Azure con Application Insights habilitado
Para	Publicar el microservicio en la nube de Microsoft
Descripción El servicio de aplicaciones será construido con base en el presupuesto de la RAS y sus especificaciones de hardware.	

ID: 10	**Épica:** Requerimientos no funcionales
Yo como	Sistema
Quiero	Contar con un pipeline configurado para liberar el microservicio en el servicio de aplicaciones de Microsoft Azure
Para	Brindar un proceso automático de lanzamiento y publicación
Descripción La idea del pipeline es que pueda compilar la aplicación, correr las pruebas unitarias y si todo es correcto liberar la aplicación a un ambiente previo a producción.	

ID: 11	**Épica:** Requerimientos no funcionales
Yo como	Sistema
Quiero	Implementar un modulo de autenticación y validación de accesos utilizando el SDK de Okta en el microservicio
Para	Velar por la seguridad de la solución
Descripción Este requerimiento será integrado en el desarrollo de la aplicación.	

ID: 12	**Épica:** Requerimientos no funcionales
Yo como	Sistema
Quiero	Configurar el servicio de Blockchain en la cuenta de EOS.IO
Para	Consumir el servicio mediante el microservicio de Blockchain
Descripción Levantamiento del servicio de Blockchain mediante la plataforma de EOS.	

ID: 13	**Épica:** Requerimientos no funcionales
Yo como	Sistema
Quiero	Configurar el servicio de documentación de API utilizando el patrón de Open API
Para	Brindar soporte documental a posibles consumidores de esta solución
Descripción Levantamiento del servicio de Blockchain mediante la plataforma de EOS.	

ID: 14	**Épica:** Requerimientos no funcionales
Yo como	Sistema
Quiero	Contar con la estructura y organización de componentes para un microservicio
Para	Organizar la solución de manera efectiva y fácil de entender
Descripción Creación de la solución con toda la estructura de carpetas y organización para un microservicio.	

ID: 15	**Épica:** Requerimientos no funcionales
Yo como	Sistema
Quiero	Configurar el servicio que alojara las pruebas unitarias en la solución
Para	Garantizar el correcto funcionamiento de la solución
Descripción Incluir el proyecto de .Net que manejara las pruebas unitarias y dicha estructura.	

ID: 16	**Épica:** Requerimientos no funcionales
Yo como	Sistema
Quiero	Configurar el servicio que permitirá el registro de eventos y la captura de errores de la solución.
Para	Garantizar el correcto funcionamiento de la solución
Descripción Configuración de los filtros para registro y captura de errores.	

5.4 Programación del API REST

Este apartado tiene como objetivo programar un API REST que permita la creación y obtención de bloques en el servicio de *Blockchain* incluyendo las pruebas unitarias requeridas para asegurar el correcto funcionamiento del API.

Para sustentar el objetivo descrito anteriormente se recorrerá la solución creada utilizando una segmentación por *sprint*, dando visibilidad de lo que se entregó cada iteración, tomando como base la priorización del trabajo y las dependencias de las historias de usuario. Se hará uso de *Postman* para poder ejecutar el microservicio y obtener capturas que comprueben la funcionalidad de cada requerimiento.

Adicionalmente, cada historia de usuario incluirá las pruebas unitarias que se llevaron a cabo para garantizar el correcto funcionamiento lógico de la aplicación. Es importante aclarar, que no todas las historias de usuario deben tener una o varias pruebas unitarias asociadas, ya que responden a configuraciones externas a la programación del código.

5.4.1 Sprint # 1

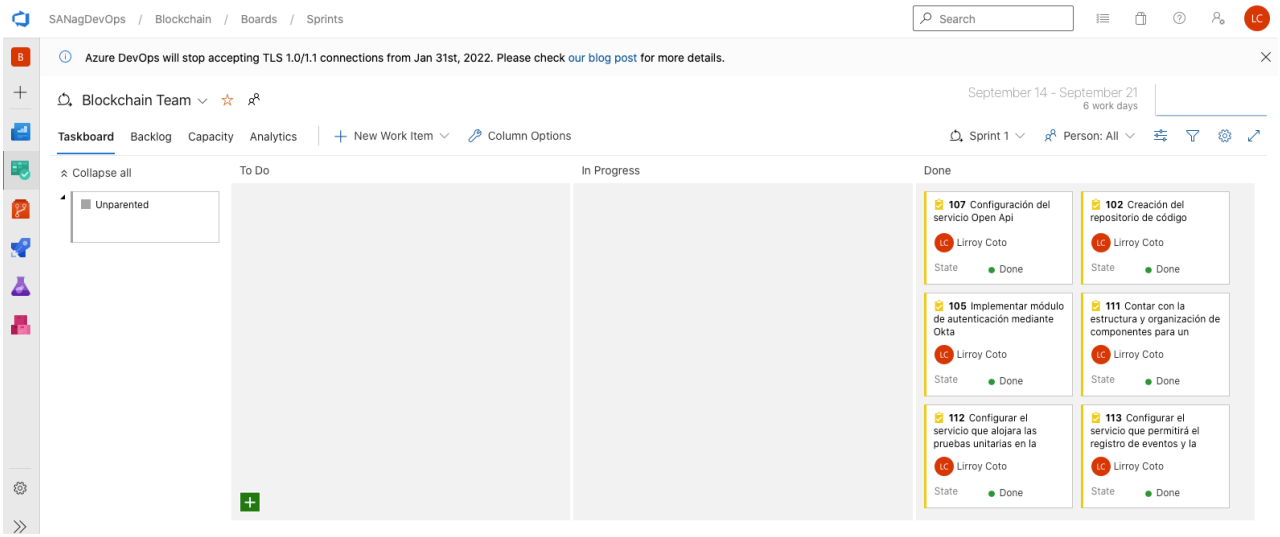


Figura 28 Pizarra de trabajo sprint #1

FUENTE: Sitio Web: Sitio de Azure DevOps de la RAS

Este sprint comprendió del 14 al 21 de setiembre de 2021, y se entregaron las siguientes historias de usuario:

5.4.1.1 Creación del repositorio de código

Para referencia de la historia de usuario se puede revisar la sección 5.3.5, requerimiento con ID 8.

A continuación, se muestra el repositorio de código creado para la solución utilizando el Azure DevOps de la RAS.

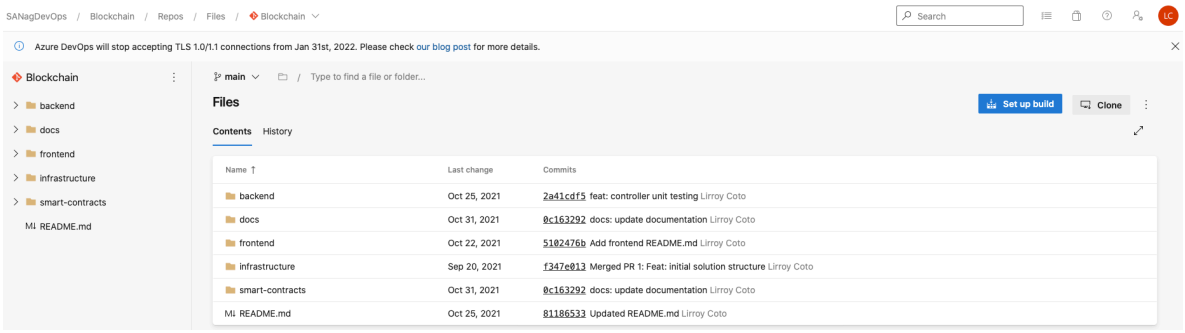


Figura 29 Repositorio de código

FUENTE: Sitio Web: Sitio de Azure DevOps de la RAS

5.4.1.2 Contar con la estructura y organización de componentes para un microservicio

Para referencia de la historia de usuario se puede revisar la sección 5.3.5, requerimiento con ID 14.

En la siguiente imagen se puede observar la estructuración de carpetas que se realizó para garantizar un orden de los elementos técnicos que componen la solución.

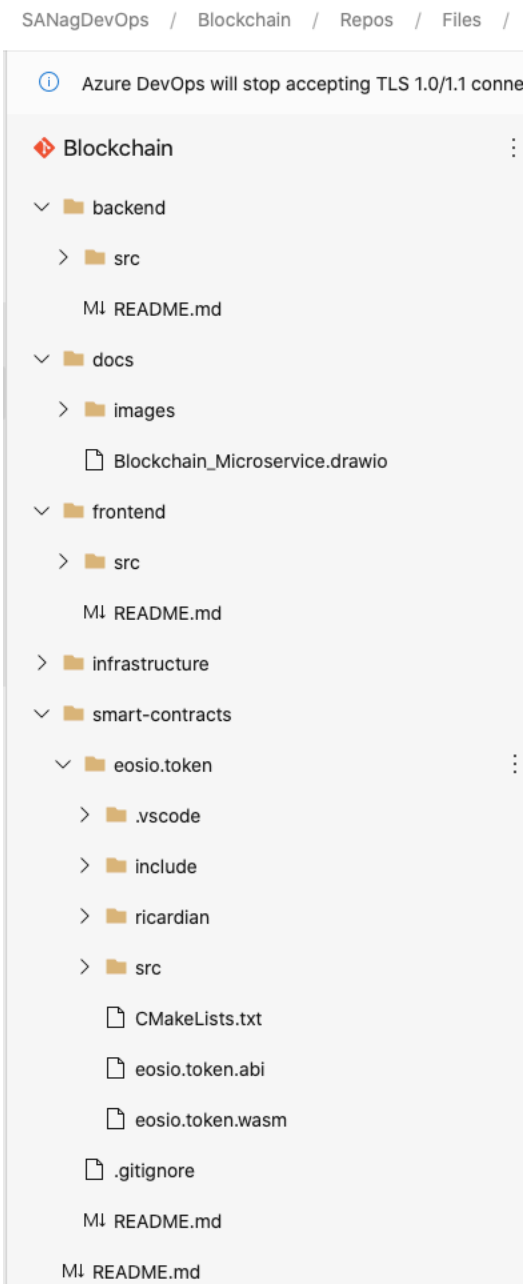


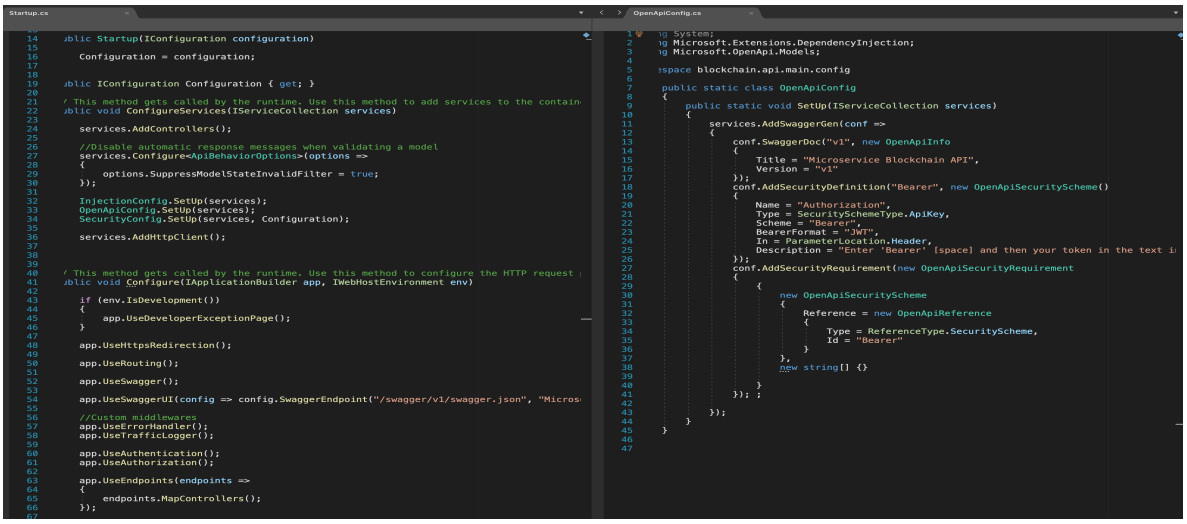
Figura 30 Estructuración de carpetas en el repositorio de código

FUENTE: Sitio Web: Sitio de Azure DevOps de la RAS

5.4.1.3 Configuración del servicio Open Api

Para referencia de la historia de usuario se puede revisar la sección 5.3.5, requerimiento con ID 13.

Tal como se definió en la sección 5.3.2.5, se implementó *Swagger* en el API, a continuación, se ilustra cómo se configuró *Swagger* y cómo se visualiza para el usuario.



```
Startup.cs
public Startup(IConfiguration configuration)

    Configuration = configuration;


public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the contain
public void ConfigureServices(IServiceCollection services)

    services.AddControllers();

    //Disable automatic response messages when validating a model
    services.Configure<ApiBehaviorOptions>(options =>
    {
        options.SuppressModelStateInvalidFilter = true;
    });

    InjectionConfig.SetUp(services);
    OpenApiConfig.SetUp(services);
    SecurityConfig.SetUp(services, Configuration);

    services.AddHttpClient();


// This method gets called by the runtime. Use this method to configure the HTTP request
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseHttpsRedirection();

    app.UseRouting();

    app.UseSwagger();

    app.UseSwaggerUI(config => config.SwaggerEndpoint("/swagger/v1/swagger.json", "Micros

    //Custom middlewares
    app.UseErrorHandler();
    app.UseTrafficLogger();

    app.UseAuthentication();
    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
```

```
OpenApiConfig.cs
ng System;
ng Microsoft.Extensions.DependencyInjection;
ng Microsoft.OpenApi.Models;

space blockchain.api.main.config

public static class OpenApiConfig
{
    public static void SetUp(IServiceCollection services)
    {
        services.AddSwaggerGen(conf =>
        {
            conf.SwaggerDoc("v1", new OpenApiInfo
            {
                Title = "Microservice Blockchain API",
                Version = "v1"
            });
            conf.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme()
            {
                Name = "Authorization",
                Type = SecuritySchemeType.ApiKey,
                Scheme = "Bearer",
                BearerFormat = "JWT",
                In = ParameterLocation.Header,
                Description = "Enter 'Bearer' [space] and then your token in the text i
            });
            conf.AddSecurityRequirement(new OpenApiSecurityRequirement
            {
                {
                    new OpenApiSecurityScheme
                    {
                        Reference = new OpenApiReference
                        {
                            Type = ReferenceType.SecurityScheme,
                            Id = "Bearer"
                        }
                    },
                    new string[] {}
                }
            }) ;
        });
    }
}
```

Figura 31 Configuración de Swagger

FUENTE: Elaboración propia

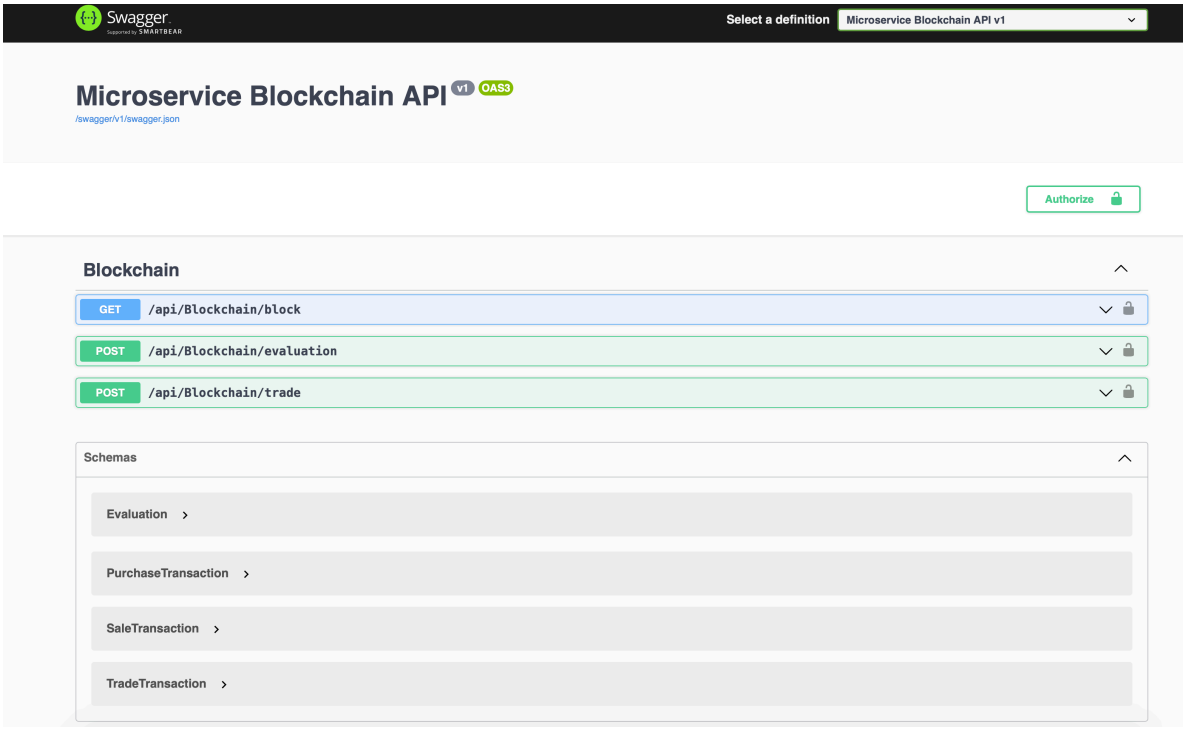


Figura 32 Sitio de Swagger

FUENTE: Elaboración propia

5.4.1.4 Implementar módulo de autenticación mediante Okta

Para referencia de la historia de usuario se puede revisar la sección 5.3.5, requerimiento con ID 11.

Tal como se describió en la sección 5.3.2.3, la autenticación y autorización del microservicio se realizará mediante *Okta*. A continuación, se presenta la configuración y programación realizada para la autenticación en el microservicio.

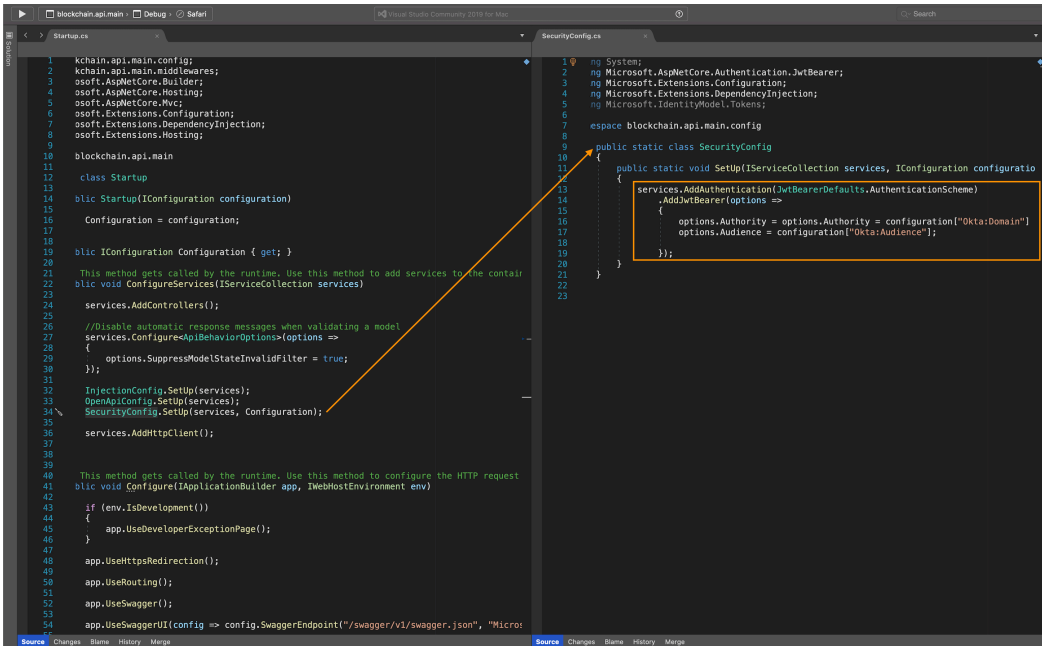


Figura 33 Configuración de Okta

FUENTE: Elaboración propia

Adicionalmente en la siguiente imagen se puede ver un ejemplo del código de estado del API si no se envía un *token* valido generado por Okta.

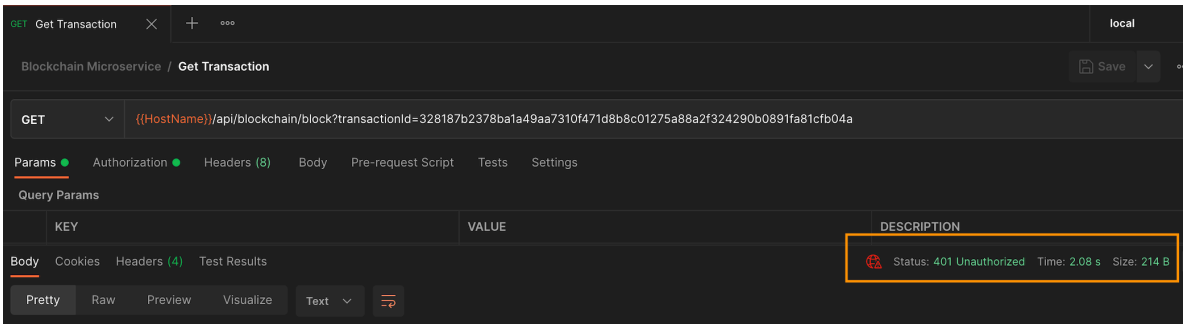


Figura 34 Ejecución no autorizada desde Postman

FUENTE: Elaboración propia

5.4.1.5 Configurar el servicio que permitirá el registro de eventos y la captura de errores de la solución

Para referencia de la historia de usuario se puede revisar la sección 5.3.5, requerimiento con ID 16.

Tal como se aprecia en la siguiente imagen, se programó un manejador de eventos y errores en la solución. Este manejador de eventos registrará la información en las bitácoras de la aplicación.

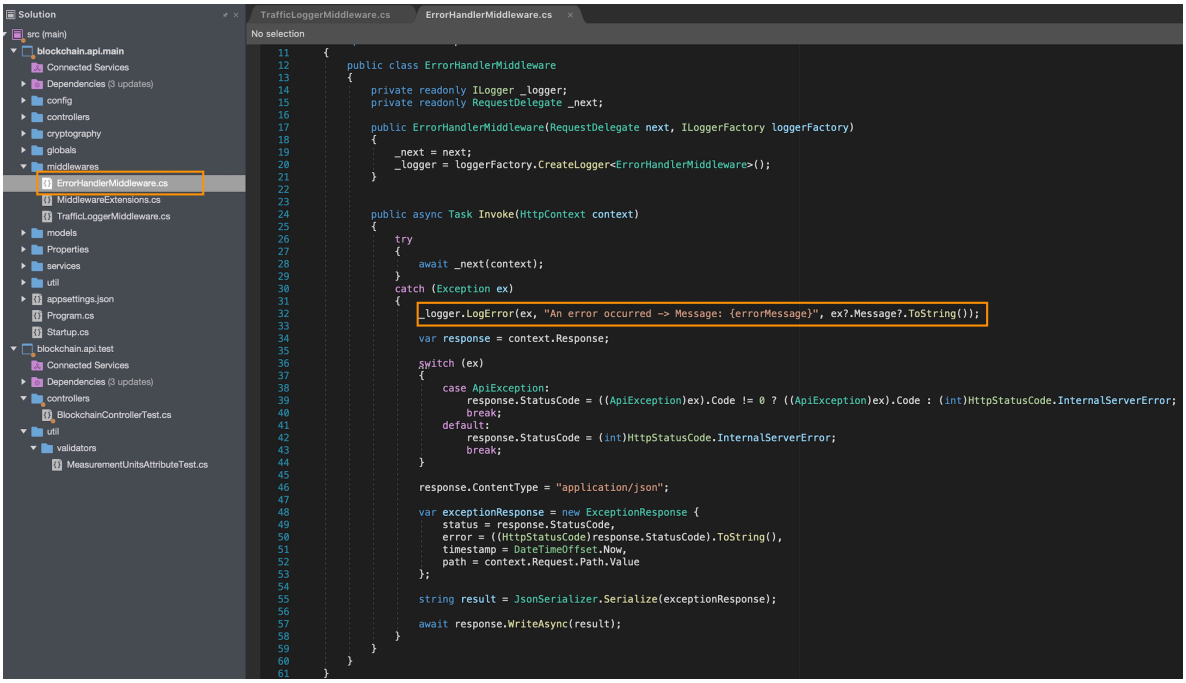


Figura 35 Registro de eventos y errores

FUENTE: Elaboración propia

5.4.1.6 Configurar el servicio que alojara las pruebas unitarias en la solución

Para referencia de la historia de usuario se puede revisar la sección 5.3.5, requerimiento con ID 15.

La siguiente imagen, sitúa las pruebas unitarias a nivel de estructura dentro de la solución que se llevó a cabo.

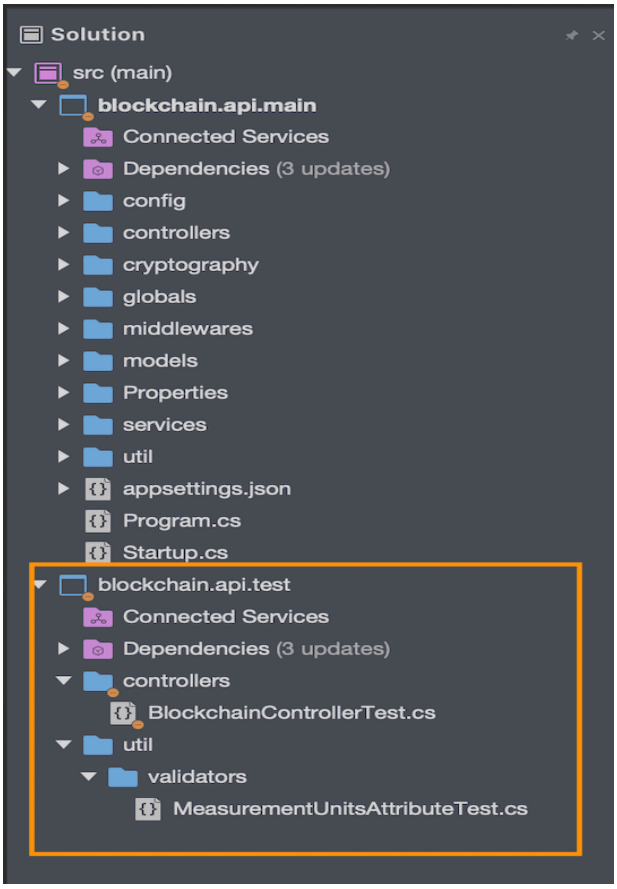


Figura 36 Ubicación de las pruebas unitarias en la solución

FUENTE: Elaboración propia

5.4.2 Sprint #2

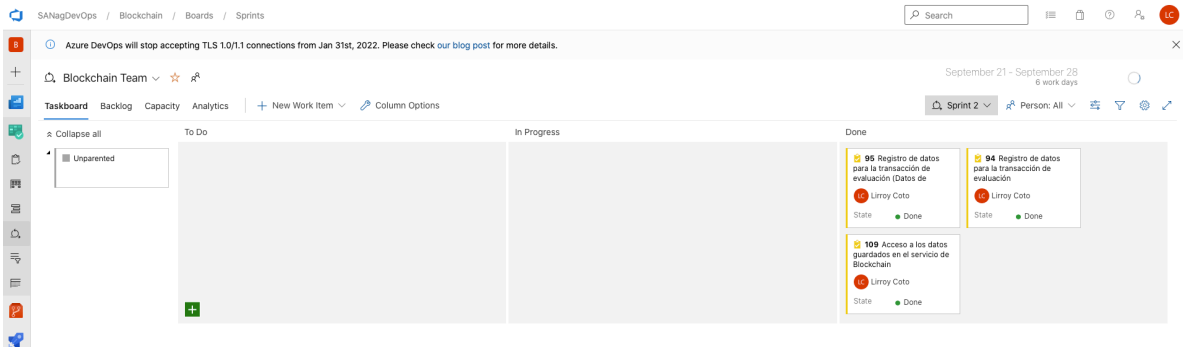


Figura 37 Pizarra de trabajo sprint #2

FUENTE: Sitio Web: Sitio de Azure DevOps de la RAS

Este sprint comprendió del 21 al 28 de setiembre de 2021, y se entregaron las siguientes historias de usuario:

5.4.2.1 Configuración del Smart Contract Eosio.Token y control de saldos de compradores y vendedores

Para referencia de la historia de usuario se puede revisar la sección 5.3.5, requerimiento con ID 12 y en la sección 5.2.2 requerimiento con ID 5.

Para el control de los saldos de las producciones agrícolas de las diferentes fincas, se utilizó el contrato de *Eosio.Token*, este contrato esta predefinido por EOS. Para poder cumplir el requerimiento de la RAS se ajustó este contrato, agregando dos nuevas acciones que contemplan las evaluaciones, compras y ventas de producción. Tal como se muestra a continuación:

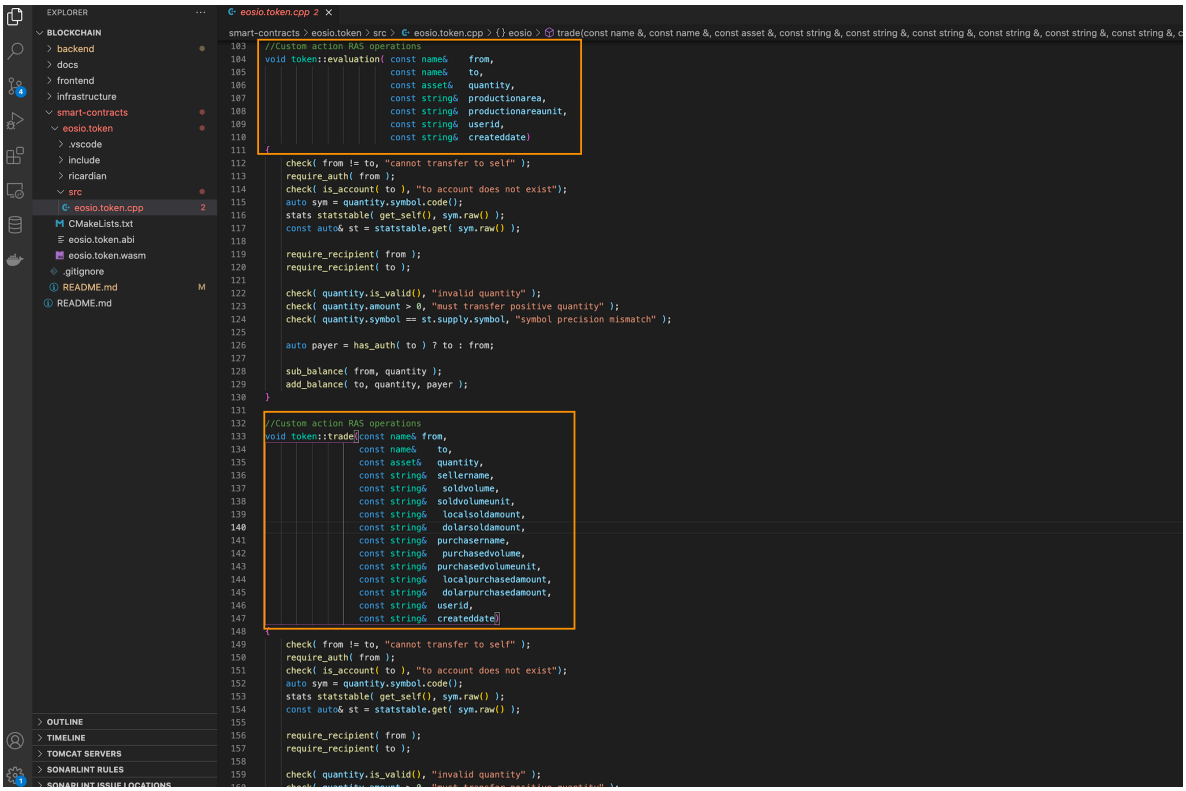


Figura 38 Acción en contrato EOS.IO datos para transacción de evaluación

FUENTE: Elaboración propia

Adicionalmente, este contrato define el concepto de *Tokens*, los cuales pueden representar un objeto de valor. Para este proyecto los *Tokens* serán creados tomando como base los tipos de cultivos que existen y su producción en toneladas como unidad estándar.

Para explicar lo anterior, se tiene el siguiente ejemplo:

La RAS posee como tipo de cultivo el café, a su vez las fincas al ser evaluadas estiman una posible cantidad de producción al año siguiendo las normativas de la RAS lo que da como resultado el proceso de evaluación. Posteriormente esta finca

puede vender su producción o comprar otro café de alguna otra finca asociada a la red.

Con el ejemplo anterior el objeto de valor que se crea en el *smart-contract* será “TCF” (Toneladas de Café) lo cual simbólicamente expone un comportamiento muy similar a una moneda en el mercado. Al completar la evaluación la RAS destinará, n cantidad de toneladas de café a la finca específica basado en su evaluación y esta será la cantidad de toneladas que la finca podrá vender a otras fincas, y así comenzar el flujo de transacciones que generará un control de saldos en los bloques de Blockchain.

Con base a la explicación anterior, es importante entender que esa implementación será controlada a su vez por el microservicio de Blockchain que se programó.

Para poder ejecutar *Nodeos,* una vez que está instalado se ejecuta el servicio mediante la terminal, la siguiente imagen muestra dicho servicio en ejecución:

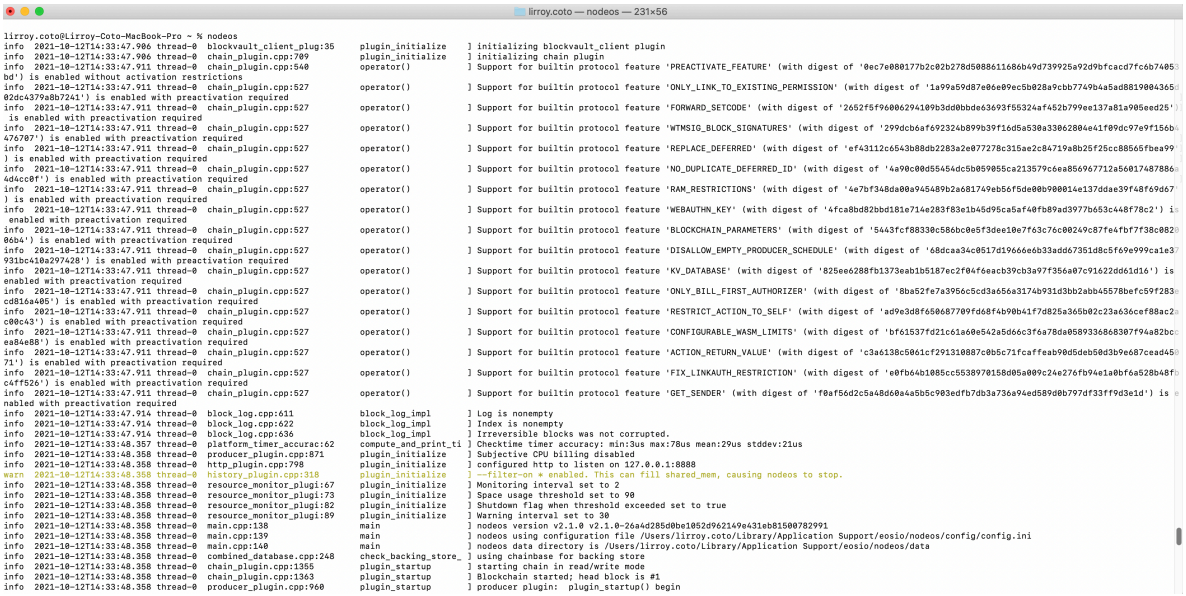


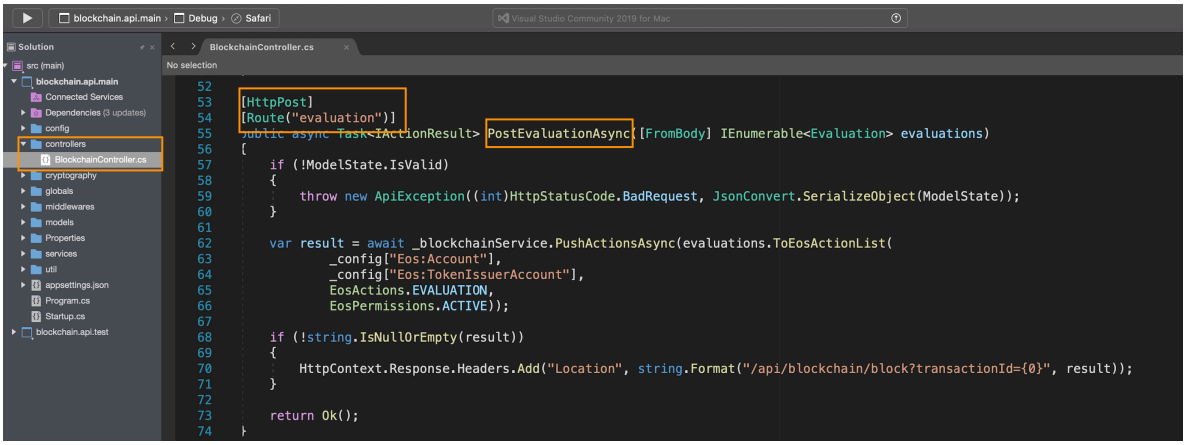
Figura 39 Ejecución del servicio Nodeos

FUENTE: Elaboración propia

5.4.2.2 Registro de datos para la transacción de evaluación y datos de auditoría de sistemas

Para referencia de la historia de usuario se puede revisar la sección 5.2.1, requerimientos con ID 1 y ID 2.

A continuación, se puede observar el *endpoint* que se creó con el fin registrar la transacción de evaluación de los campos agrícolas:

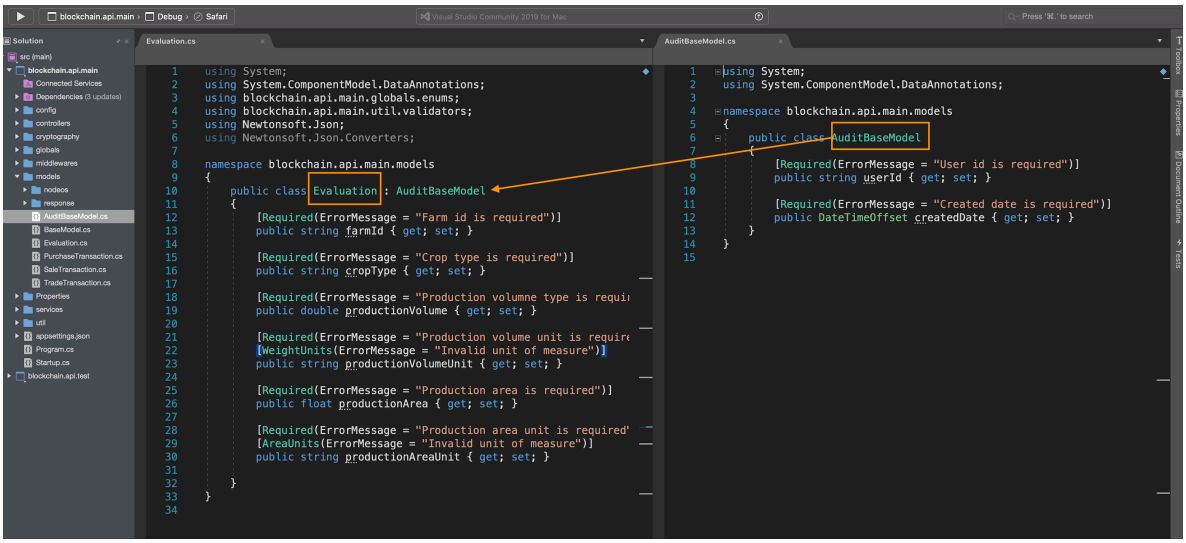


```
[HttpPost]
[Route("evaluation")]
public async Task<IActionResult> PostEvaluationAsync([FromBody] IEnumerable<Evaluation> evaluations)
{
    if (!ModelState.IsValid)
    {
        throw new ApiException((int)HttpStatusCode.BadRequest, JsonConvert.SerializeObject(ModelState));
    }

    var result = await _blockchainService.PushActionsAsync(evaluations.ToEosActionList(
            _config["Eos:Account"],
            _config["Eos:TokenIssuerAccount"],
            EosActions.EVALUATION,
            EosPermissions.ACTIVE));

    if (!string.IsNullOrEmpty(result))
    {
        HttpContext.Response.Headers.Add("Location", string.Format("/api/blockchain/block?transactionId={0}", result));
    }

    return Ok();
}
```

Figura 40 *Endpoint* para transacción de evaluación

FUENTE: Elaboración propia

Adicionalmente se construyó un objeto que permite abstraer los datos requeridos para dicha transacción y que hereda de otro objeto las propiedades para almacenar los datos de auditoría de sistemas:



```
using System;
using System.ComponentModel.DataAnnotations;
using blockchain.api.main.globals.enums;
using blockchain.api.main.util.validators;
using Newtonsoft.Json;
using Newtonsoft.Json.Converters;

namespace blockchain.api.main.models
{
    public class Evaluation : AuditBaseModel
    {
        [Required(ErrorMessage = "Farm id is required")]
        public string farmId { get; set; }

        [Required(ErrorMessage = "Crop type is required")]
        public string cropType { get; set; }

        [Required(ErrorMessage = "Production volumne type is requi
        public double productionVolume { get; set; }

        [Required(ErrorMessage = "Production volume unit is require
        [WeightUnits(ErrorMessage = "Invalid unit of measure")]
        public string productionVolumeUnit { get; set; }

        [Required(ErrorMessage = "Production area is required")]
        public float productionArea { get; set; }

        [Required(ErrorMessage = "Production area unit is required
        [AreaUnits(ErrorMessage = "Invalid unit of measure")]
        public string productionAreaUnit { get; set; }

    }
}
```

```
using System;
using System.ComponentModel.DataAnnotations;

namespace blockchain.api.main.models
{
    public class AuditBaseModel
    {
        [Required(ErrorMessage = "User id is required")]
        public string userId { get; set; }

        [Required(ErrorMessage = "Created date is required")]
        public DateTimeOffset createdDate { get; set; }
    }
}
```

Figura 41 Modelo lógico de datos para transacción de evaluación

FUENTE: Elaboración propia

En la siguiente captura se puede observar el funcionamiento del *endpoint,* para dicha ejecución se utilizó *Postman*:

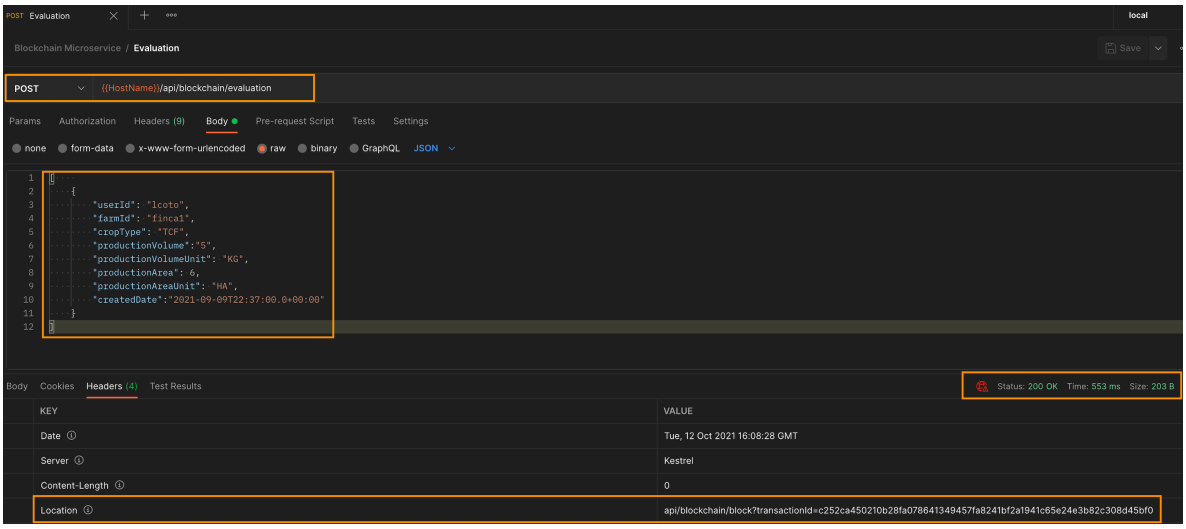


Figura 42 Prueba de ejecución punto de acceso para evaluaciones

FUENTE: Elaboración propia

Finalmente, para esta historia de usuario se llevaron a cabo las siguientes pruebas unitarias:

1. *PostEvaluationAsync_WhenIsInvalidModel_ReturnApiException*: Esta prueba unitaria tiene como objetivo garantizar que cuando la solicitud de creación de una nueva transacción de evaluación contenga un problema con la información recibida devuelva un mensaje de *Bad Request* con un código de error 400.

```
[Test]
public void PostEvaluationAsync_WhenIsInvalidModel_ReturnApiException()
{
    //Arrange
    HttpStatusCode expectedStatusCode = HttpStatusCode.BadRequest;

    blockchainController.ModelState.AddModelError("Bad model state","Trigger state model error");

    //Act
    ApiException ex = Assert.ThrowsAsync<ApiException>(() => blockchainController.PostEvaluationAsync(null));

    //Assert
    Assert.IsNotNull((HttpStatusCode)ex.Code, "Missing status code");
    Assert.AreEqual((HttpStatusCode)ex.Code, expectedStatusCode, "Unexpected status code");
    Assert.IsNotNull(ex.Message, "Missing exception message");

}
```

Figura 43 Prueba de unitaria PostEvaluationAsync_WhenIsInvalidModel_ReturnApiException

FUENTE: Elaboración propia

2. *PostEvaluationAsync_WhenIsValidModelAndResultNotEmpty_ReturnOkAndHeaderLocation:* Esta prueba unitaria garantiza que el API responda con un mensaje *OK* y un código de 200 cuando la información enviada es procesada correctamente.

```
[Test]
public async Task PostEvaluationAsync_WhenIsValidModelAndResultNotEmpty_ReturnOkAndHeaderLocation()
{
    //Arrange
    string expectedTransactionId = "";
    HttpStatusCode expectedStatusCode = HttpStatusCode.OK;

    _blockchainService.Setup(x => x.PushActionsAsync(It.IsAny<List<Action>>())).ReturnsAsync(expectedTransactionId);

    //Act

    var result = await blockchainController.PostEvaluationAsync(new List<Evaluation>()) as OkResult;

    //Assert
    Assert.IsNotNull(result.StatusCode, "Missing status code");
    Assert.AreEqual((HttpStatusCode)result.StatusCode, expectedStatusCode, "Unexpected status code");

}
```

Figura 44 Prueba unitaria

PostEvaluationAsync_WhenIsValidModelAndResultNotEmpty_ReturnOkAndHeaderLocation

FUENTE: Elaboración propia

Finalmente, se puede observar en la siguiente imagen la ejecución de las dos pruebas descritas anteriormente:

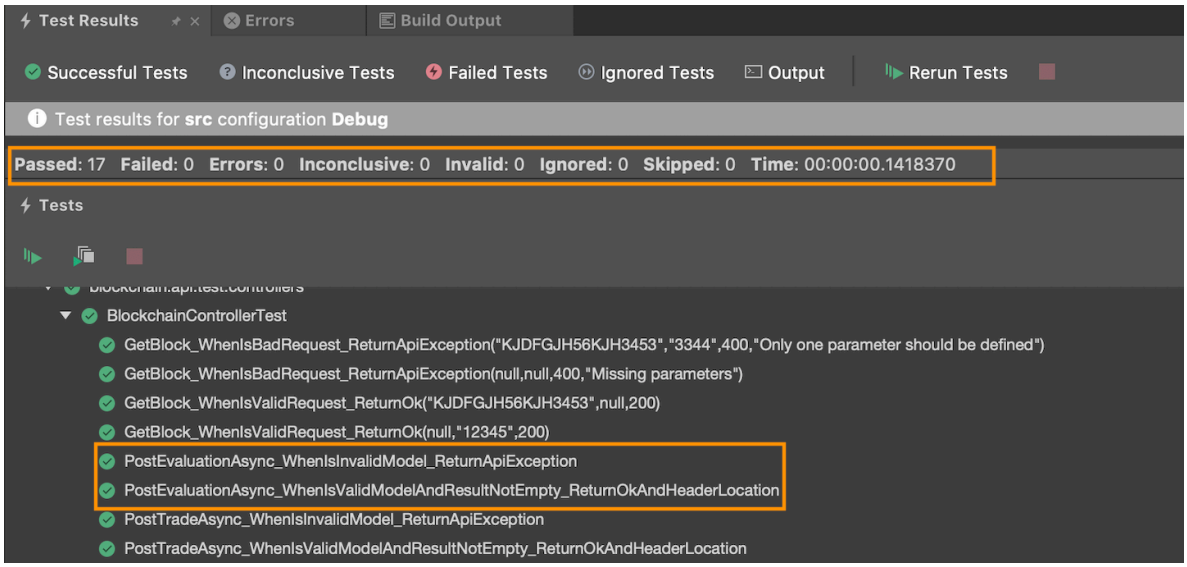


Figura 45 Ejecución exitosa de las pruebas unitarias

FUENTE: Elaboración propia

5.4.3 Sprint #3

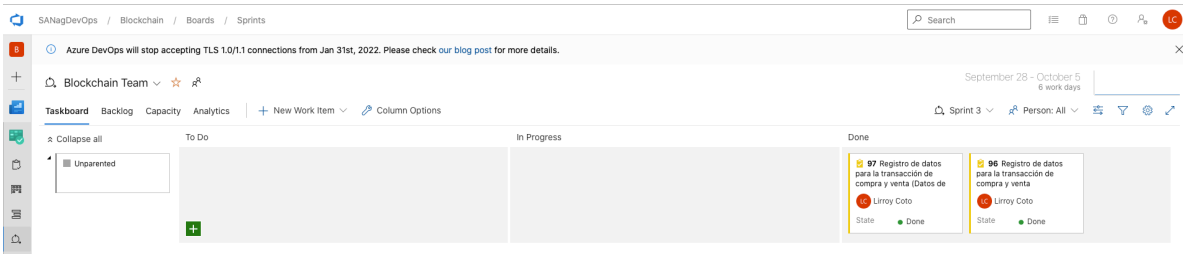


Figura 46 Pizarra de trabajo sprint #3

FUENTE: Sitio Web: Sitio de Azure DevOps de la RAS

Este sprint comprendió del 28 de setiembre al 5 de octubre de 2021, y se entregaron las siguientes historias de usuario:

5.4.3.1 Registro de datos para la transacción de compra y venta, y datos de auditoría de sistemas

Para referencia de la historia de usuario se puede revisar la sección 5.2.2, requerimientos con ID 3 y ID 4.

A continuación, se puede observar el *endpoint* que se creó con el fin registrar la transacción de compra y venta en los campos agrícolas, y una prueba de ejecución desde *Postman*:

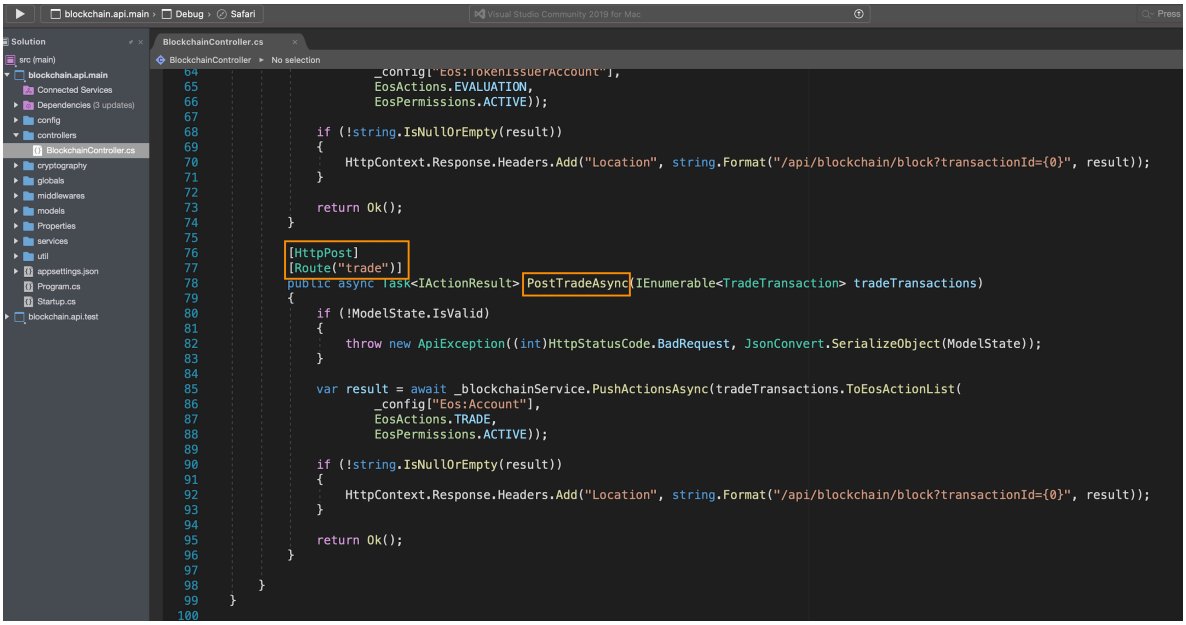


Figura 47 *Endpoint* para registro de compra y venta

FUENTE: Elaboración propia

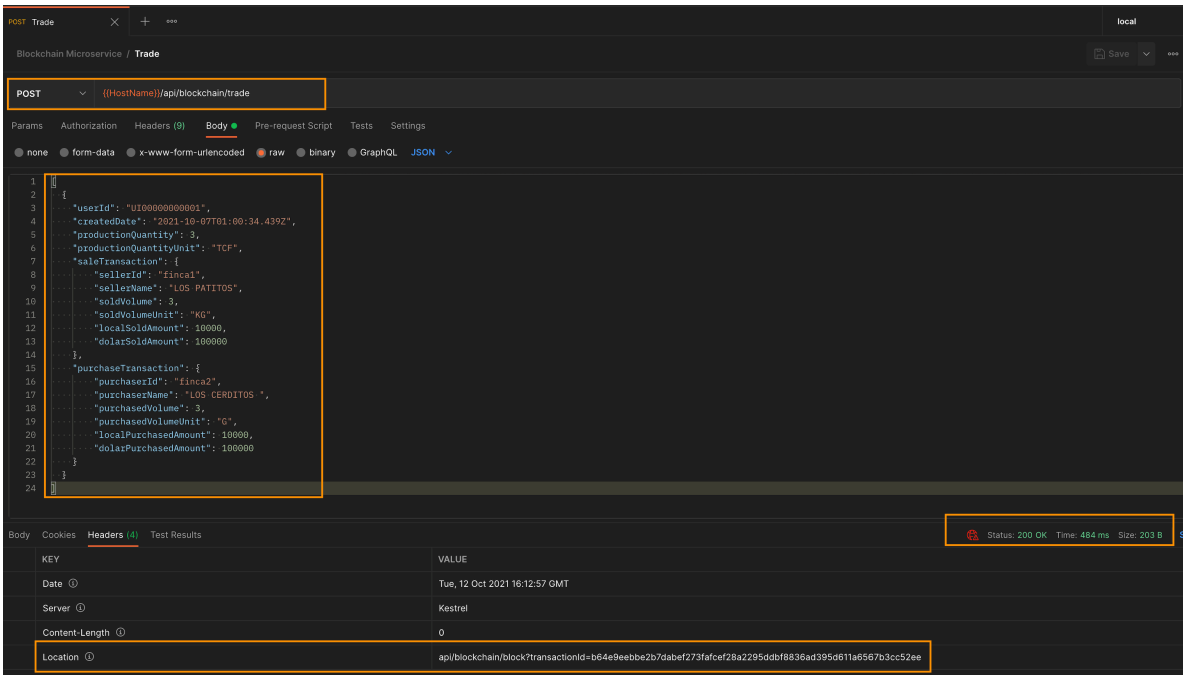


Figura 48 Ejecución desde *Postman* para registrar un compra y venta

FUENTE: Elaboración propia

Adicionalmente se construyó un objeto que permite abstraer los datos requeridos para dicha transacción y que hereda de otro objeto las propiedades para almacenar los datos de auditoría de sistemas:

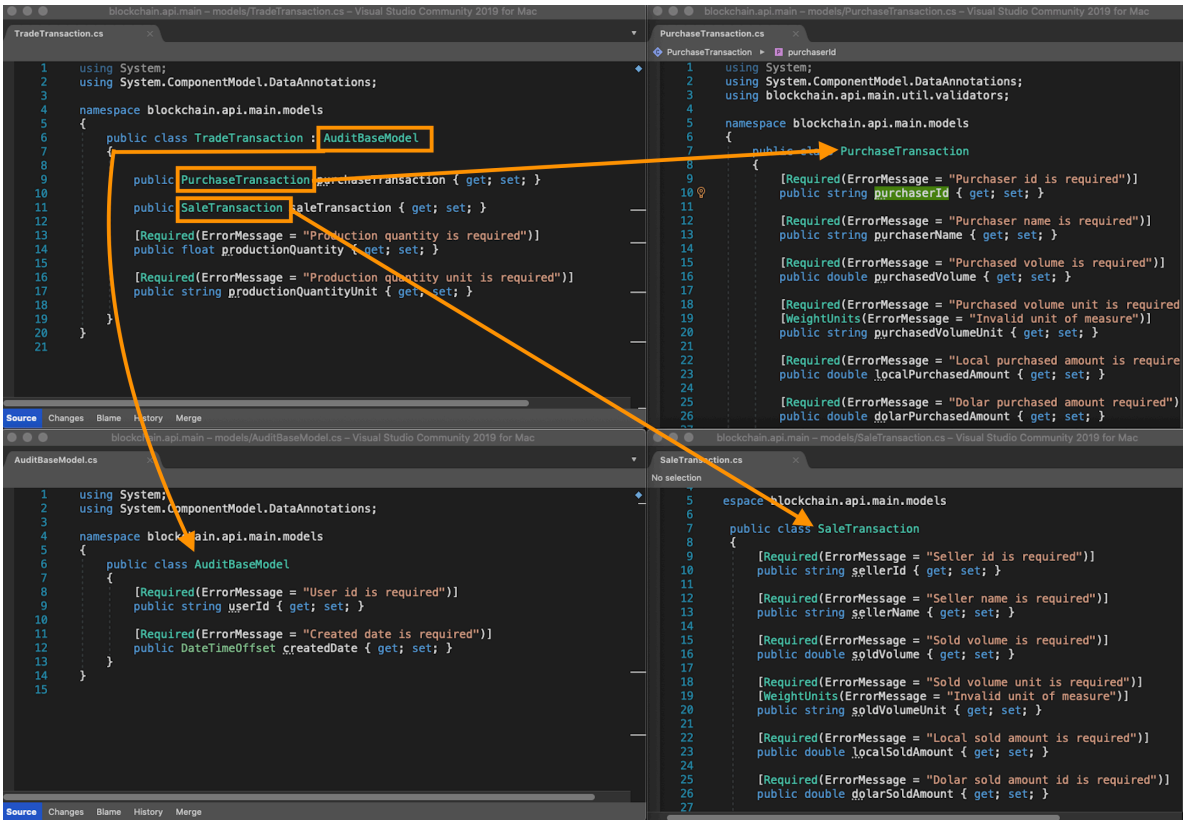


Figura 49 Modelo lógico de datos para transacción de compra y venta

FUENTE: Elaboración propia

Para esta historia de usuario se llevaron a cabo las siguientes pruebas unitarias:

1. *PostTradeAsync_WhenIsInvalidModel_ReturnApiException*: Esta prueba unitaria tiene como objetivo garantizar que cuando la solicitud de creación de una nueva transacción de compra y venta contenga un problema con la información recibida devuelva un mensaje de *Bad Request* con un código de error 400.

```
[Test]
public void PostTradeAsync_WhenIsInvalidModel_ReturnApiException()
{
    //Arrange
    HttpStatusCode expectedStatusCode = HttpStatusCode.BadRequest;

    blockchainController.ModelState.AddModelError("Bad model state", "Trigger state model error");

    //Act
    ApiException ex = Assert.ThrowsAsync<ApiException>(() => blockchainController.PostEvaluationAsync(null));

    //Assert
    Assert.IsNotNull((HttpStatusCode)ex.Code, "Missing status code");
    Assert.AreEqual((HttpStatusCode)ex.Code, expectedStatusCode, "Unexpected status code");
    Assert.IsNotNull(ex.Message, "Missing exception message");

}
```

Figura 50 Prueba unitaria PostTradeAsync_WhenIsInvalidModel_ReturnApiException

FUENTE: Elaboración propia

2. *PostEvaluationAsync_WhenIsValidModelAndResultNotEmpty_ReturnOkAndHeaderLocation:* Esta prueba unitaria garantiza que el API responda con un mensaje *OK* y un código de 200 cuando la información enviada es procesada correctamente.

```
[Test]
public async Task PostTradeAsync_WhenIsValidModelAndResultNotEmpty_ReturnOkAndHeaderLocation()
{
    //Arrange
    string expectedTransactionId = "";
    HttpStatusCode expectedStatusCode = HttpStatusCode.OK;

    _blockchainService.Setup(x => x.PushActionsAsync(It.IsAny<List<Action>>())).ReturnsAsync(expectedTransactionId);

    //Act

    var result = await blockchainController.PostEvaluationAsync(new List<Evaluation>()) as OkResult;

    //Assert
    Assert.IsNotNull(result.StatusCode, "Missing status code");
    Assert.AreEqual((HttpStatusCode)result.StatusCode, expectedStatusCode, "Unexpected status code");

}
```

Figura 51 Prueba unitaria

PostEvaluationAsync_WhenIsValidModelAndResultNotEmpty_ReturnOkAndHeaderLocation

FUENTE: Elaboración propia

Finalmente, se puede observar en la siguiente imagen la ejecución de las dos pruebas descritas anteriormente:

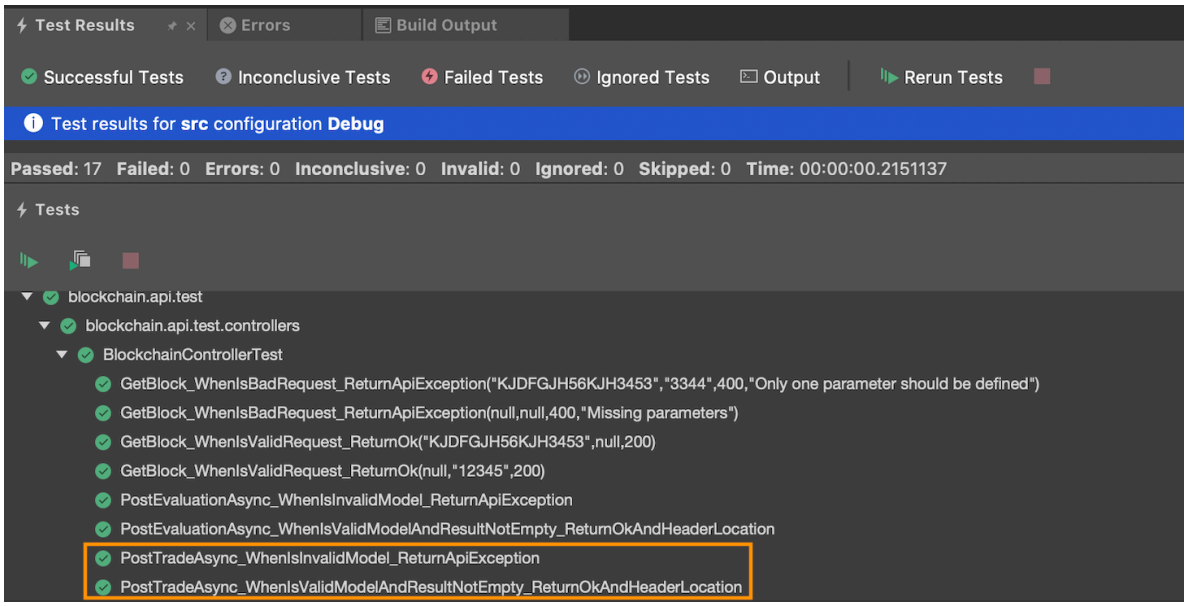


Figura 52 Ejecución exitosa de las pruebas unitarias

FUENTE: Elaboración propia

5.4.4 Sprint #4

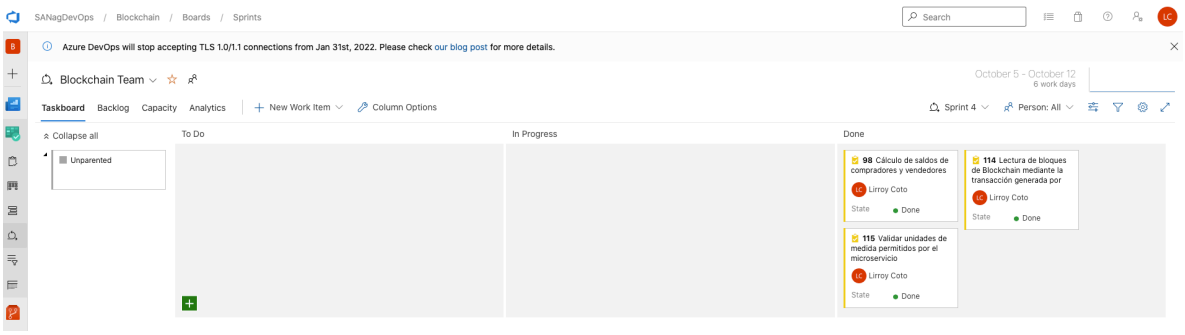


Figura 53 Pizarra de trabajo sprint #4

FUENTE: Sitio Web: Sitio de Azure DevOps de la RAS

Este sprint comprendió del 5 al 12 de octubre de 2021, y se entregaron las siguientes historias de usuario:

5.4.4.1 *Lectura de bloques de Blockchain mediante la transacción generada por EOS*

Para referencia de la historia de usuario se puede revisar la sección 5.2.3, requerimientos con ID 7.

Se creó un *endpoint* que permite obtener la información de un bloque específico mediante el id de la transacción o mediante el número del bloque. A continuación, se muestra dicho *endpoint* y una prueba de ejecución desde *Postman*.

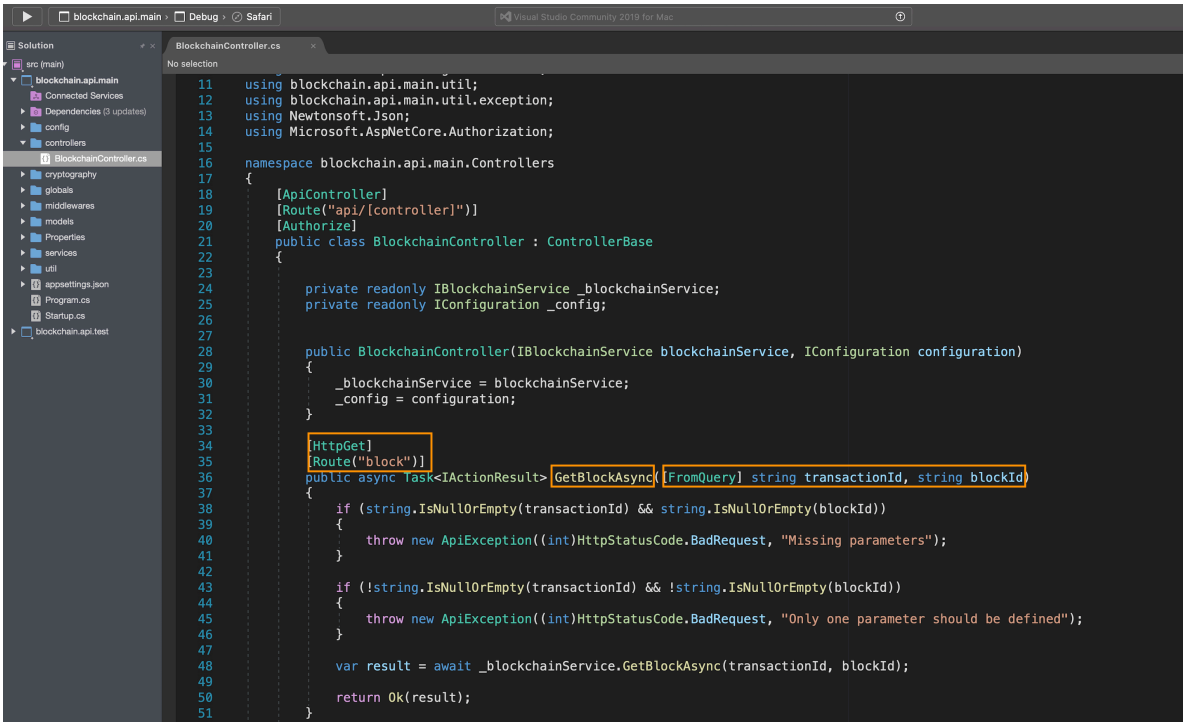


Figura 54 *Endpoint* para obtener un bloque

FUENTE: Elaboración propia

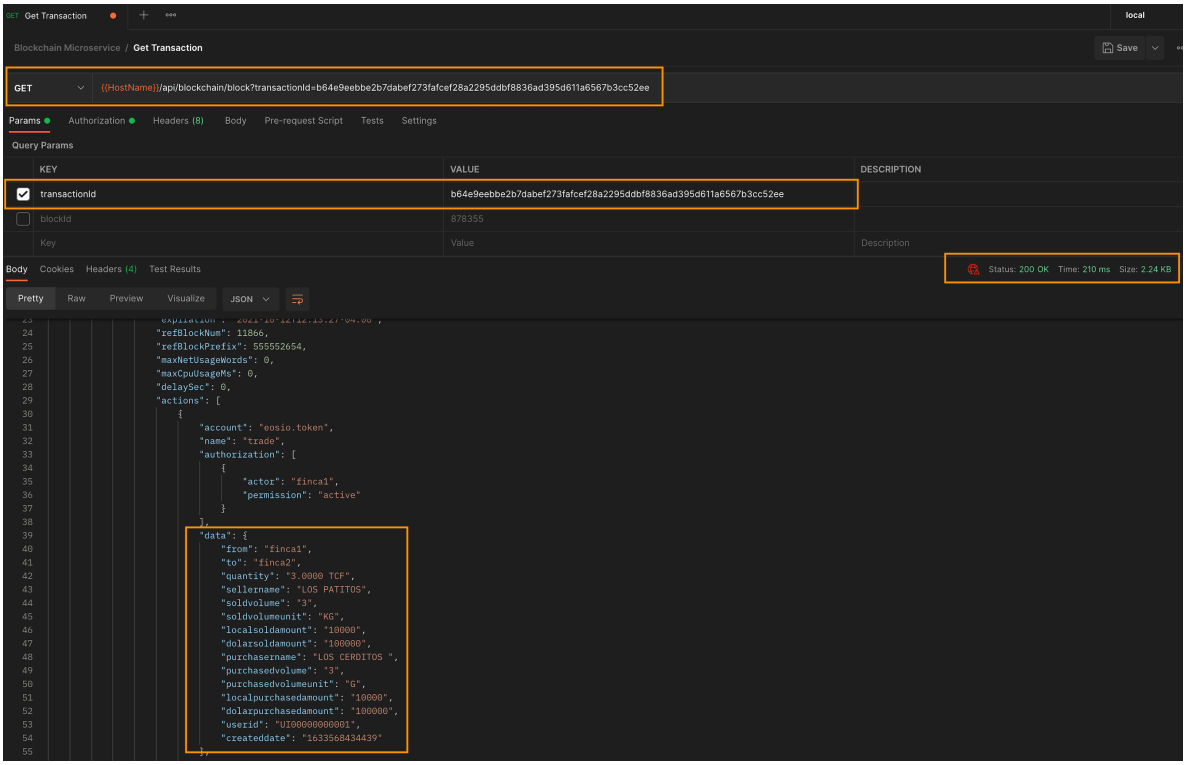


Figura 55 Ejecución desde *Postman* para obtener un bloque

FUENTE: Elaboración propia

Para esta historia de usuario se llevaron a cabo las siguientes pruebas unitarias:

1. *GetBlock_WhenIsBadRequest_ReturnApiException:* Esta prueba unitaria garantiza que el API responda con un mensaje *Bad Request* y un código de 400 cuando la información enviada no contiene un id de bloque o un id de transacción, de otra manera no se puede devolver la información almacenada en el servicio de *Blockchain*.

```
[Test]
[TestCase(null, null, 400, "Missing parameters")]
[TestCase("KJDFGJH56KJH3453", "3344", 400, "Only one parameter should be defined")]
public void GetBlock_WhenIsBadRequest_ReturnApiException(string transactionId, string blockId, int expectedStatusCode, string expectedMessage)
{
    //Arrange

    //Act
    ApiException ex = Assert.ThrowsAsync<ApiException>(() => blockchainController.GetBlockAsync(transactionId, blockId));

    //Assert
    Assert.IsNotNull((HttpStatusCode)ex.Code, "Missing status code");
    Assert.AreEqual((HttpStatusCode)ex.Code, (HttpStatusCode)expectedStatusCode, "Unexpected status code");
    Assert.AreEqual(ex.Message, expectedMessage, "Unexpected message");

}
```

Figura 56 Prueba unitaria GetBlock_WhenIsBadRequest_ReturnApiException

FUENTE: Elaboración propia

2. *GetBlock_WhenIsValidRequest_ReturnOk:* Esta prueba unitaria garantiza que el API responda con un mensaje *OK* y un código de 200 cuando la información enviada puede ser procesada y la información puede ser retornada sin inconveniente.

```
[Test]
[TestCase(null, "12345", 200)]
[TestCase("KJDFGJH56KJH3453", null, 200)]
public async Task GetBlock_WhenIsValidRequest_ReturnOk(string transactionId, string blockId, int expectedStatusCode)
{
    //Arrange

    //Act
    var result = await blockchainController.GetBlockAsync(transactionId, blockId) as OkObjectResult;

    //Assert
    Assert.IsNotNull((HttpStatusCode)result.StatusCode, "Missing status code");
    Assert.AreEqual((HttpStatusCode)result.StatusCode, (HttpStatusCode)expectedStatusCode, "Unexpected status code");

}
```

Figura 57 Prueba unitaria GetBlock_WhenIsValidRequest_ReturnOk

FUENTE: Elaboración propia

Finalmente, se puede observar en la siguiente imagen la ejecución de las dos pruebas descritas anteriormente:

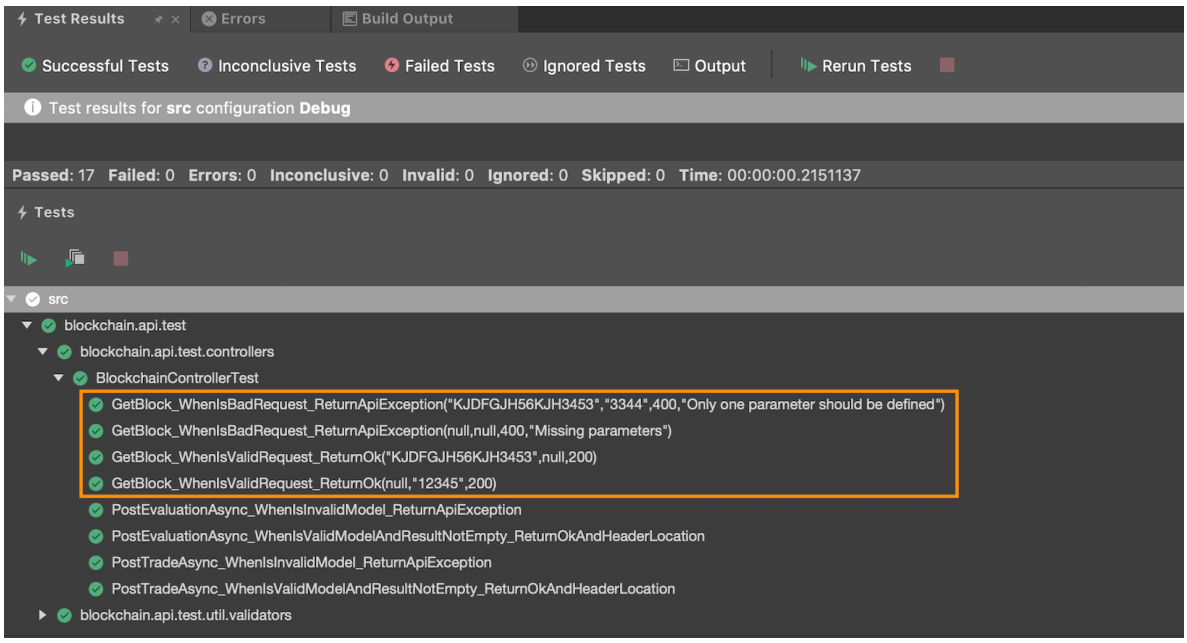


Figura 58 Ejecución exitosa de las pruebas unitarias

FUENTE: Elaboración propia

5.4.4.2 Validar unidades de medida permitidos por el microservicio

Tal como se definió en la sección 5.2.2 y 5.2.3, se programaron validaciones para que los valores fueran requeridos y para ciertos campos se validan las unidades de peso. A continuación, se muestra la definición de los valores permitidos y un caso práctico de respuesta del API cuando los valores no son válidos.

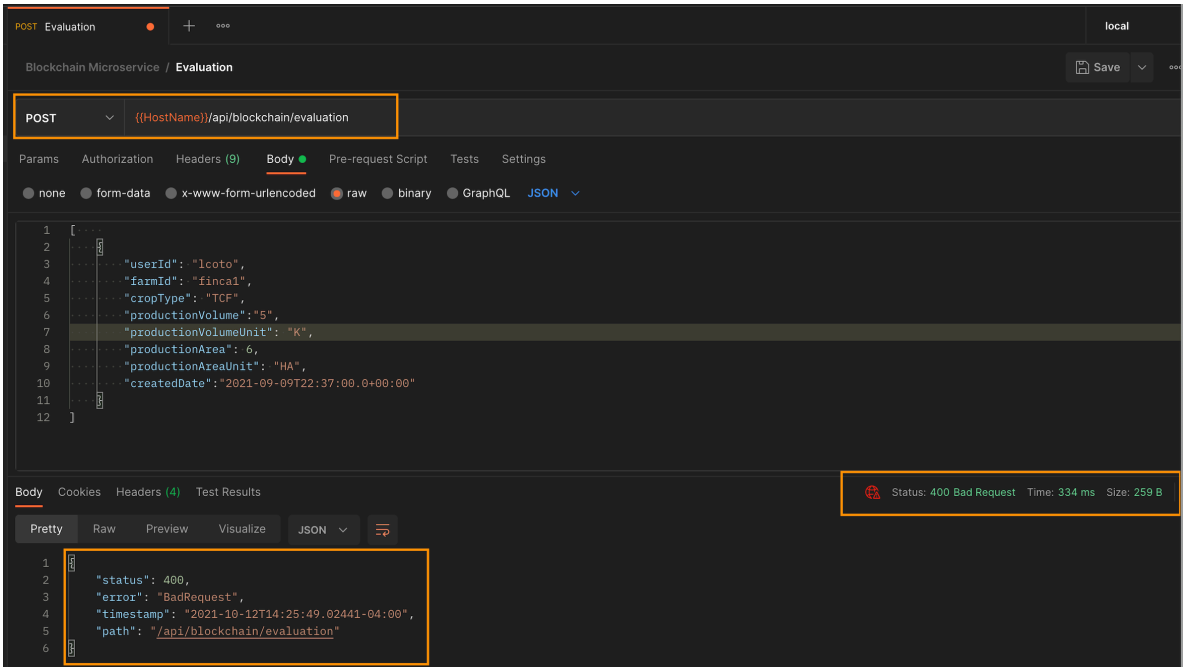


Figura 59 Prueba de ejecución con datos inválidos

FUENTE: Elaboración propia

Para esta historia de usuario se desarrolló la siguiente prueba unitaria:

1. *WeightUnitsAttribute_WhenIsCalle_ReturnApiException:* Esta prueba unitaria garantiza que la regla de validación de unidades de medida sea persistente y no afecte la lógica de registro de transacciones de compra y venta.

```
[Test]
[TestCase("T", true)]
[TestCase("kg", true)]
[TestCase("g", true)]
[TestCase("TON", true)]
[TestCase("LB", true)]
[TestCase("OZ", true)]
[TestCase("o", false)]
[TestCase("k", false)]
[TestCase("", false)]
public void WeightUnitsAttribute_WhenIsCalle_ReturnApiException(string weightUnitName, bool expectedResult)
{
    //Arrange
    var attrib = new WeightUnitsAttribute();
    //Act
    var result = attrib.IsValid(weightUnitName);

    //Assert
    Assert.AreEqual(expectedResult, result, "Unexpected validation result");
}
```

Figura 60 Prueba unitaria WeightUnitsAttribute_WhenIsCalle_ReturnApiException

FUENTE: Elaboración propia

Finalmente, se puede observar en la siguiente imagen la ejecución de la prueba descrita anteriormente:

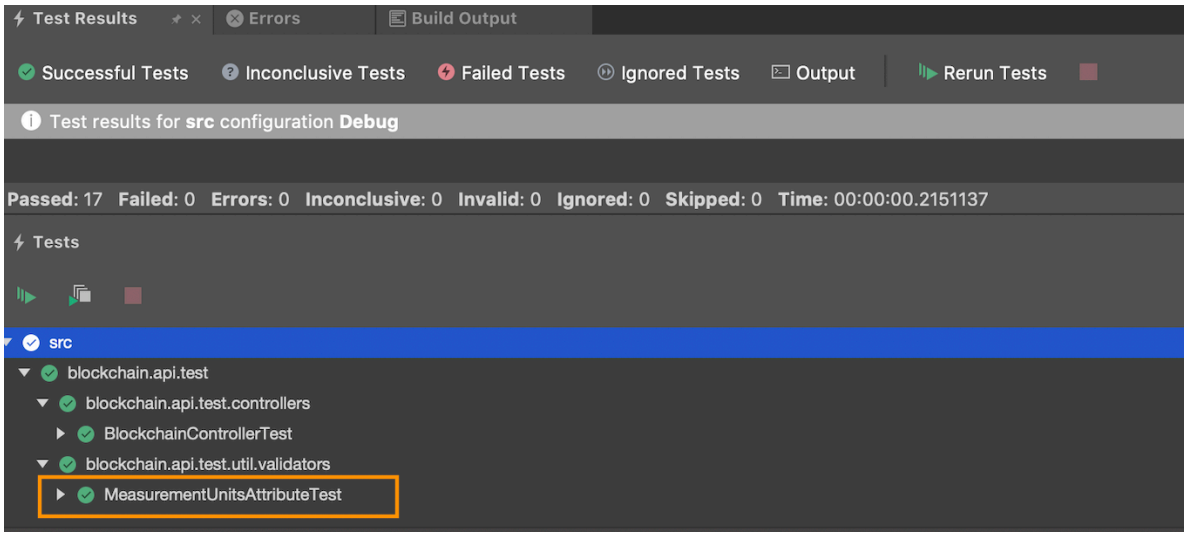


Figura 61 Ejecución exitosa de la prueba unitaria

FUENTE: Elaboración propia

5.5 Implementación de plan piloto

Esta sección tiene como objetivo implementar un plan piloto de lanzamiento de la solución, que permita hacer uso del microservicio para efectos de pruebas y mantenimientos.

Lo que se desarrollará en este apartado estará sustentado las historias de usuario con ID 9 y 10 definidas en la sección 5.3.5, donde se definen configuraciones a nivel de infraestructura de la RAS utilizando *Microsoft Azure,* es importante mencionar que estas dos historias de usuario se desarrollaron durante el sprint # 5 que comprendió del 12 al 19 de octubre de 2021.

Adicionalmente, se contemplará la explicación a los miembros del equipo de TI de la RAS, con el objetivo que puedan dar mantenimiento de la solución.

5.5.1 Capacitación técnica y documentación

Para explicar las principales características técnicas que contienen los componentes que conforman la solución se crearon archivos README.md.

Adicionalmente estos archivos describen procesos importantes para la compilación y ejecución del código en ambientes de desarrollo. Con esta información los futuros desarrolladores podrán desarrollar las diferentes mejoras y mantenimientos de la solución.

Tal como se muestra en la siguiente imagen, se crearon 4 archivos README.md:

- El primero corresponde a la documentación requerida para el *backend*, en esta sección del repositorio se encuentra el código fuente del microservicio desarrollado.
- El segundo archivo corresponde a las especificaciones del *frontend,* que por el momento no se esta utilizando ya que dentro del alcance del proyecto no había trabajo para este componente.
- El tercer archivo corresponde a las instrucciones requeridas para el correcto funcionamiento de EOS y el *Smart-Contract* implementado.
- Y por último se encuentra un archivo en general, que brinda una visión general de como está constituida la solución, sus componentes y como interactúan entre si.

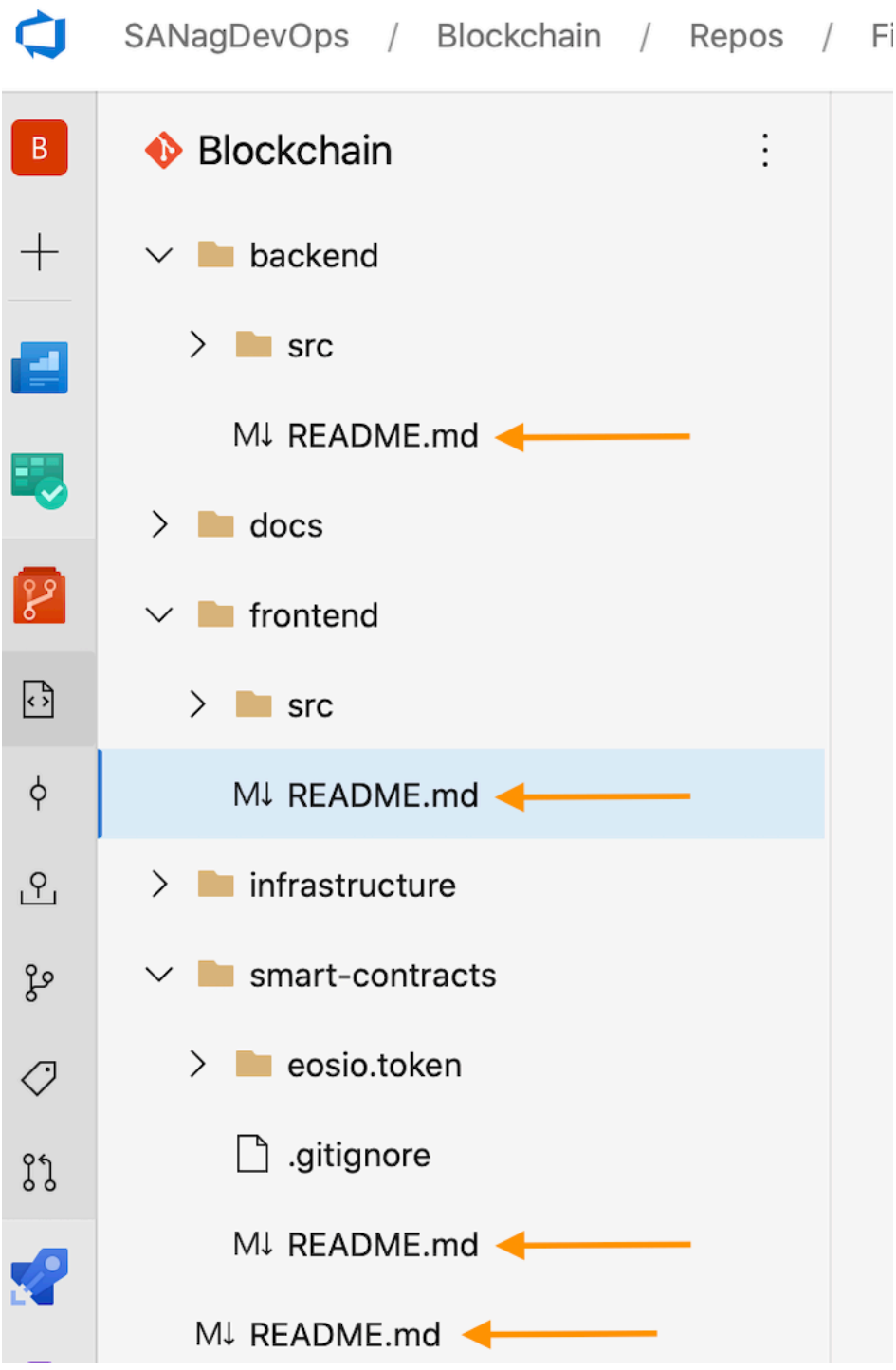


Figura 62 Archivos Readme.md

FUENTE: Elaboración propia

Es importante mencionar que estos archivos fueron revisados según (Jiménez & Rodríguez, 2021), esta sesión permitió entender el contenido de cada uno de estos archivos y al mismo tiempo se ejecutaron las instrucciones utilizando la máquina de uno de los miembros del equipo de la RAS que asistió a la sesión.

5.5.2 Lanzamiento de la solución en Microsoft Azure

Como parte de uno de los alcances de este proyecto de investigación, se llevaron a cabo una serie de configuraciones y levantamiento de servicios en la nube de *Microsoft Azure* que permitieron la ejecución exitosa de la solución. A continuación, se describirán dichas configuraciones:

5.5.2.1 Pipeline automatizado para la publicación del microservicio

A continuación, se presenta una imagen con la configuración establecida en Azure DevOps, la cual se nombró Blockchain.Microservice.CI.

Adicionalmente en la imagen se muestran los pasos habilitados para generar el artefacto que posteriormente se publicara en el servicio de aplicación.

- *Restore:* En este paso el *pipeline* descargará y hará la restauración de los paquetes *NuGet* utilizados por la aplicación.
- *Build:* Habiendo restaurado las dependencias de la solución, lo siguiente que se lleva a cabo es la compilación del código para asegurar que no haya problemas de ejecución.

- *Test:* En este paso se ejecutan las pruebas unitarias definidas y programadas para la solución.
- *Publish:* Se generan las librerías finales y archivos requeridos que serán publicados en el servicio de aplicación de Azure.

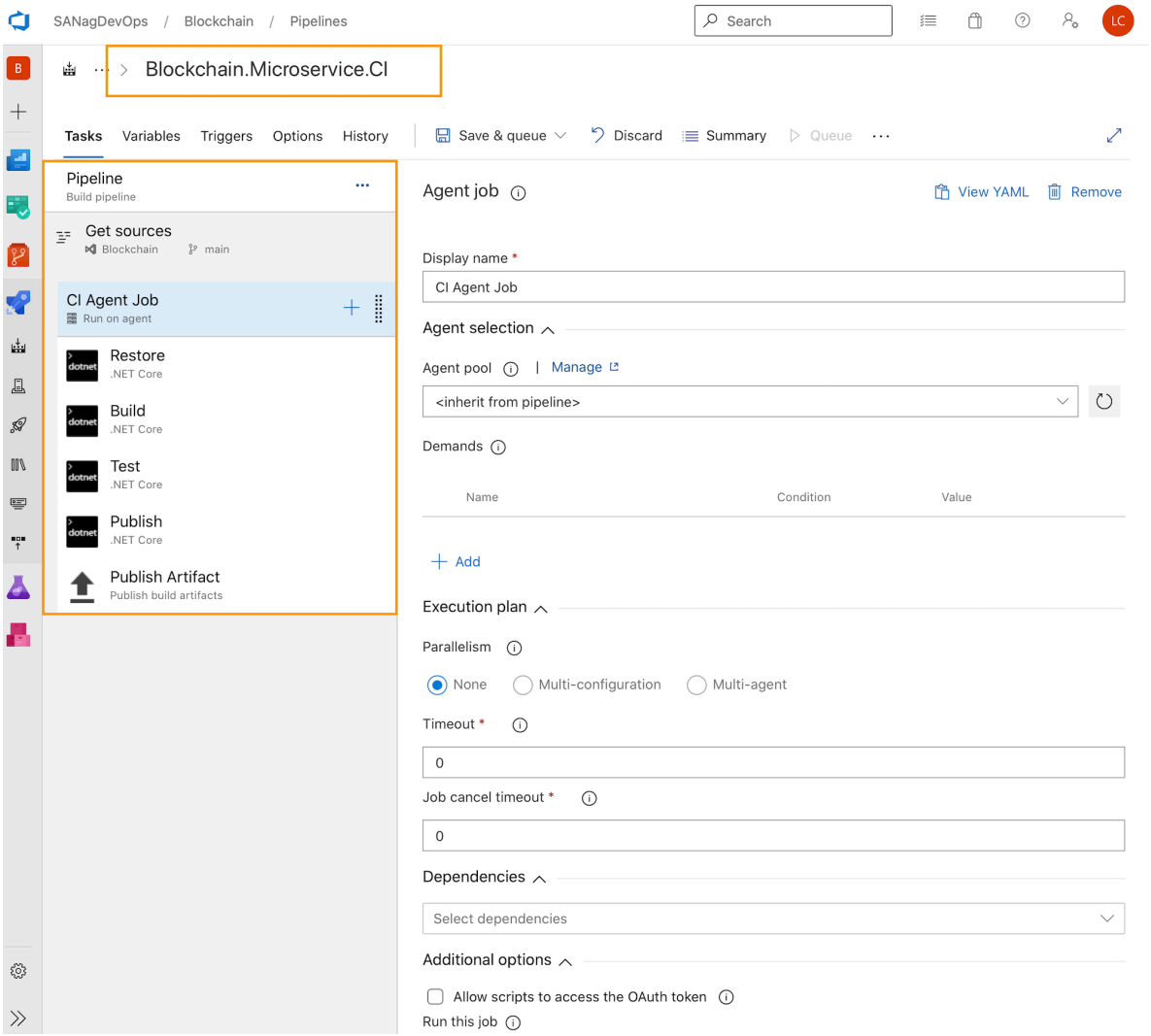


Figura 63 Configuración del pipeline de la solución en Azure DevOps

FUENTE: Elaboración propia

A continuación, se muestra la bitácora de ejecución del pipeline con la versión final del código fuente.

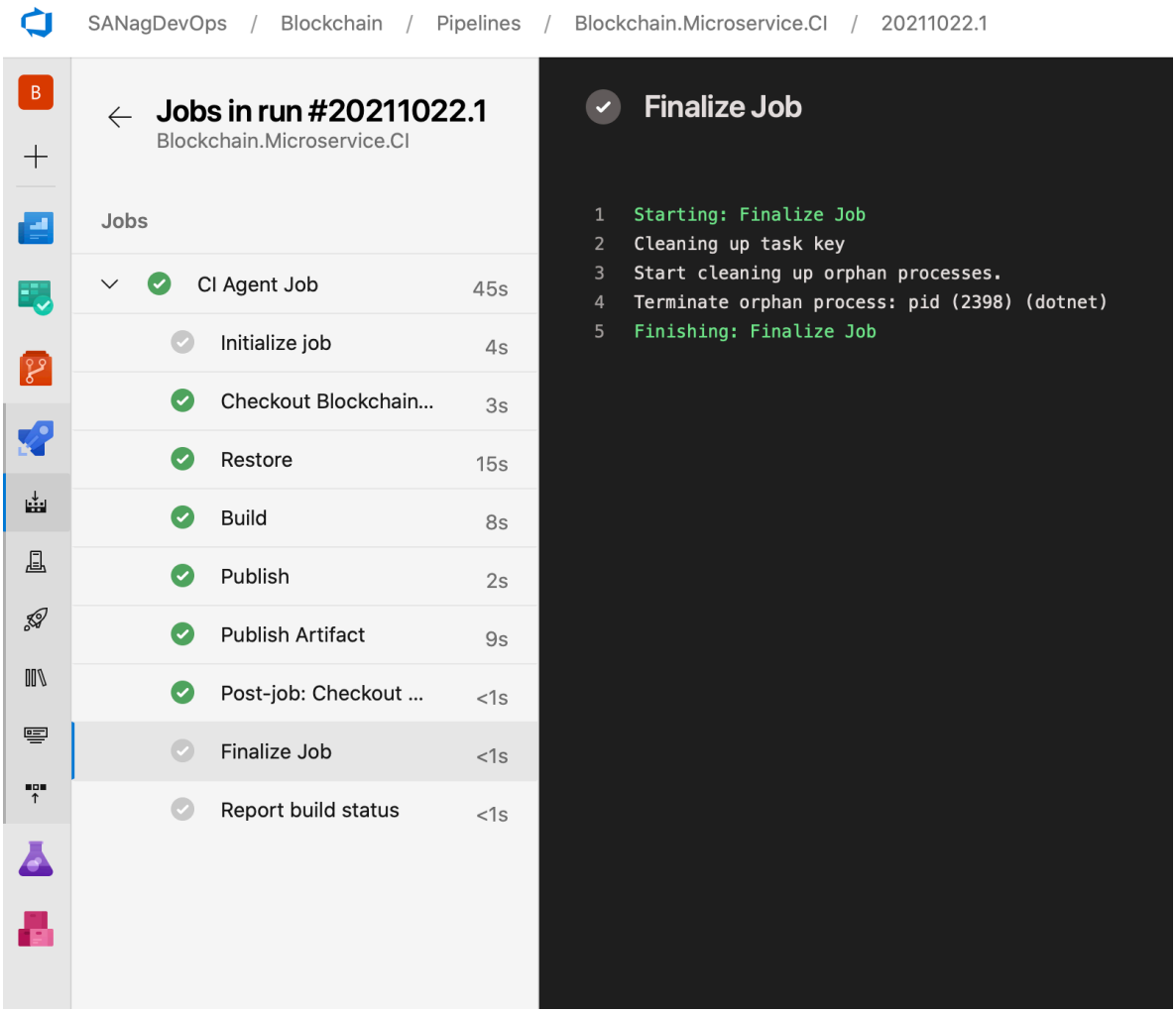


Figura 64 Prueba de ejecución del pipeline

FUENTE: Elaboración propia

Una vez generados los archivos requeridos en el último paso del pipeline y habiéndose completado este proceso, lo siguiente que se configuró es el proceso de *Release* en Azure DevOps, el cual tiene como objetivo tomar dichos archivos finales y publicarlos en el servicio de aplicaciones de Azure.

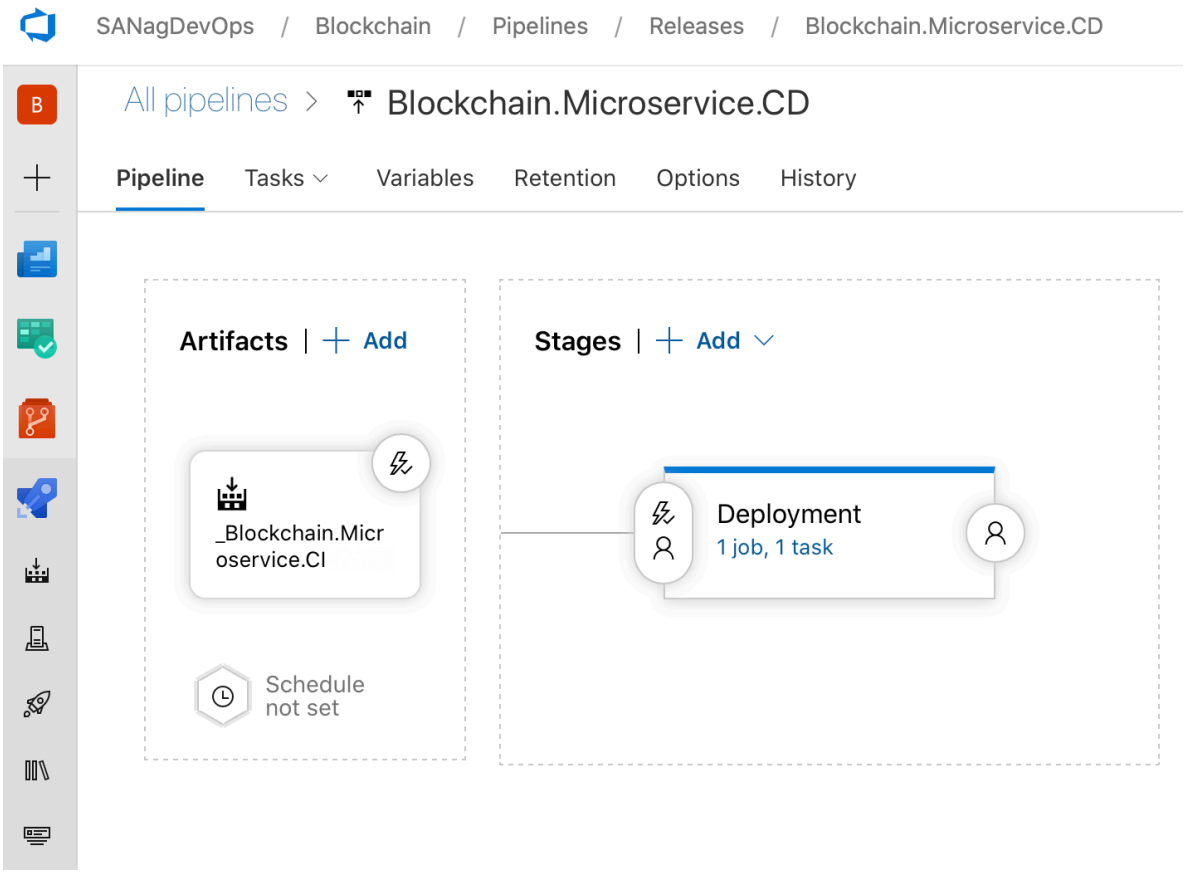


Figura 65 Configuración de los Releases en Azure DevOps

FUENTE: Elaboración propia

A continuación, se ilustra la última ejecución del proceso completada de manera exitosa.

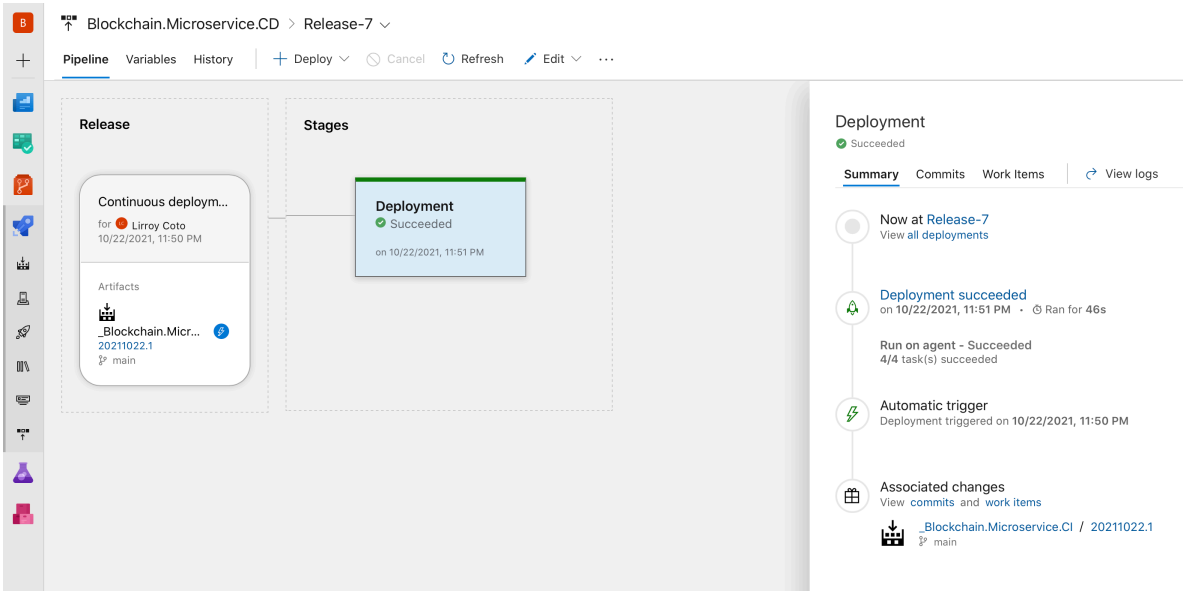


Figura 66 Bitácora de última ejecución del *Release*

FUENTE: Elaboración propia

5.5.2.2 Servicio de aplicación en la nube

Siendo el alcance un plan piloto, se procedió a la configuración del servicio con la categoría B1 del servicio el cual no tiene costo.

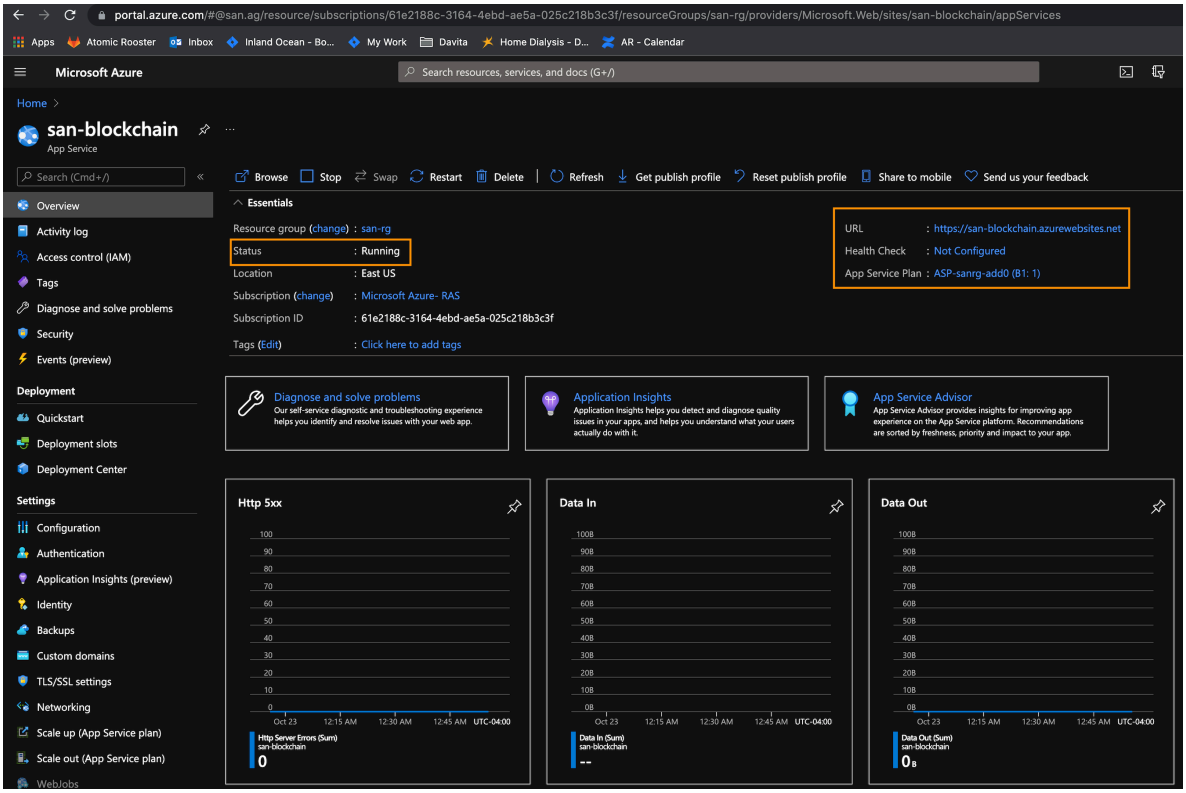


Figura 67 Pantalla de configuración del servicio de aplicación

FUENTE: Elaboración propia

Adicionalmente, se configuro Application Insights para poder revisar las bitácoras de ejecución y monitorear el correcto funcionamiento de la aplicación.

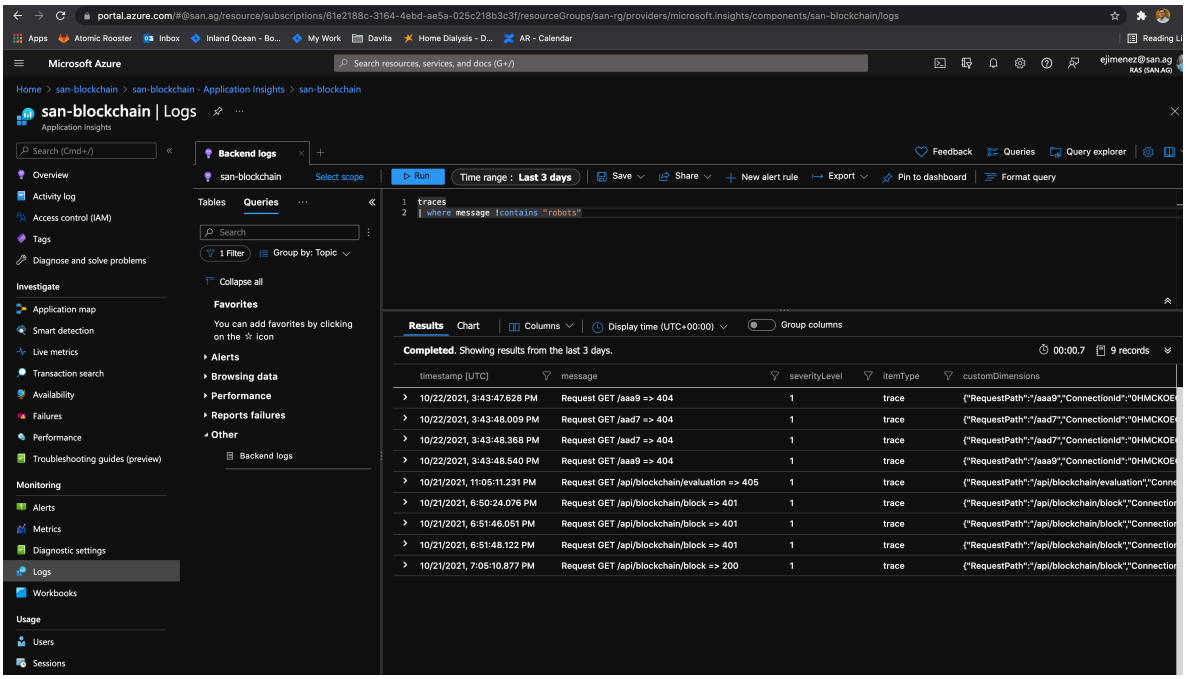


Figura 68 Pantalla de monitoreo y bitácoras

FUENTE: Elaboración propia

Por último, pero no menos importante se muestra una prueba del microservicio siendo accedido desde el navegador con la url establecida por Microsoft Azure.

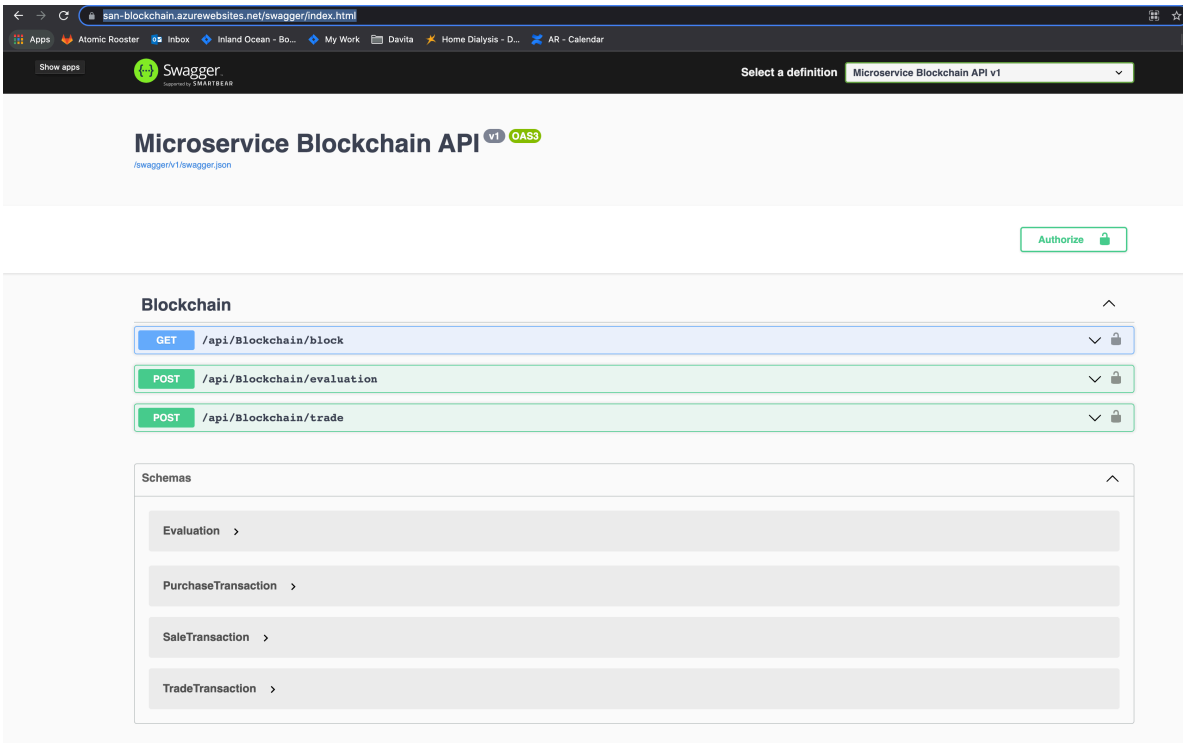


Figura 69 Microservicio de Blockchain publicado y habilitado

FUENTE: Elaboración propia

5.5.2.3 Máquina virtual para EOS.io

Para ejecutar una red privada de EOS y el servicio de Blockchain, se habilitó una máquina virtual en Linux, que expone de manera segura las tres herramientas principales que incluye EOS.

A continuación, se muestra una prueba de ejecución del servicio de EOS en dicha máquina virtual, siendo accedida mediante SSH desde una terminal remota.

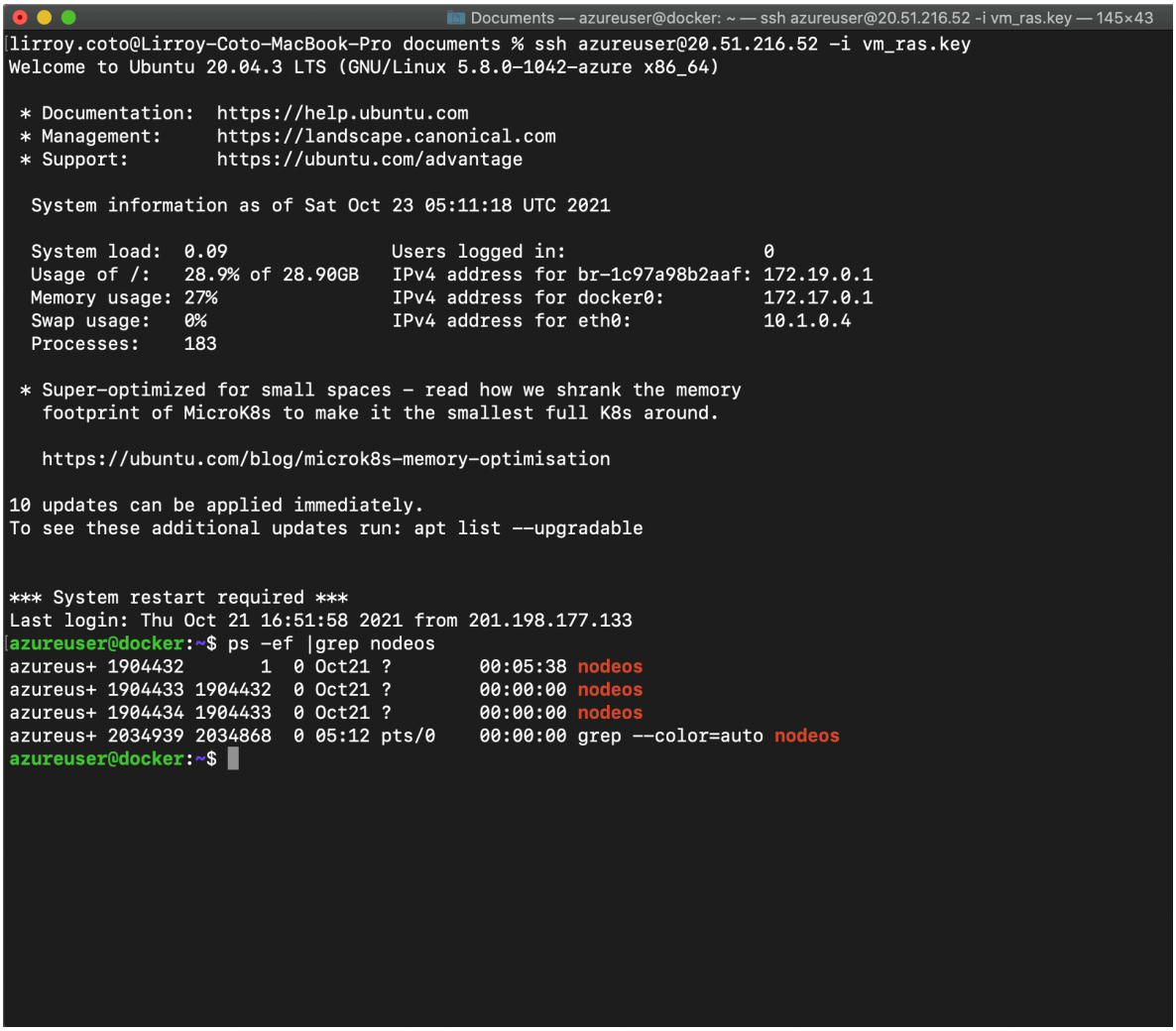


Figura 70 Conexión remota a máquina virtual y servicio de Nodes en ejecución

FUENTE: Elaboración propia

Adicionalmente, se puede observar a continuación la ejecución de un comando de la herramienta de Cleos que permite ver la configuración activa del Blockchain.

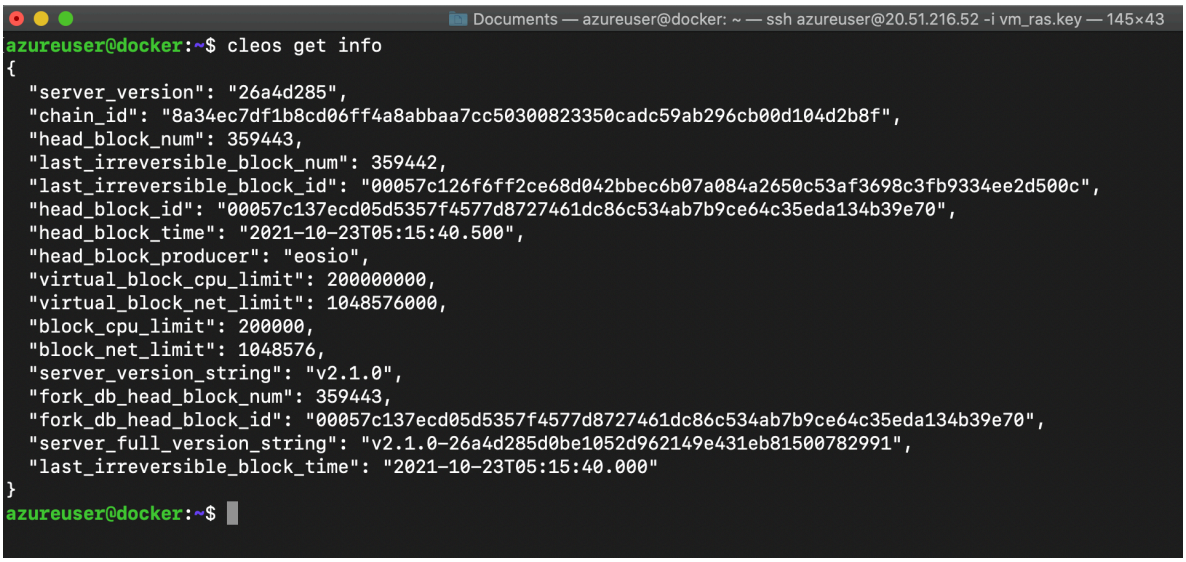


Figura 71 Configuración de Nodes

FUENTE: Elaboración propia

CAPÍTULO VI: CONCLUSIONES Y RECOMENDACIONES

Como conclusión a todos los procesos de este proyecto: Servicio de trazabilidad y transparencia en cadenas de suministro y productores agrícolas. Y habiendo establecido una solución para el problema planteado, a continuación, se realiza un análisis final de los objetivos establecidos.

6.1 Investigación de casos de éxito del uso de *Blockchain*

Gracias a la investigación de casos de éxito aplicando la tecnología de Blockchain, se pudo respaldar, reforzar e inclusive tomar referencias de lecciones aprendidas de dichos artículos, que permitieran el desarrollo de la solución usando Blockchain como eje principal.

Adicionalmente, se pudo evidenciar la importancia de contar con tecnologías confiables que garanticen la legitimidad de la información, ya que muchas decisiones a nivel de negocio y la vida cotidiana giran alrededor de los datos almacenados.

Con la ayuda de esta investigación, se pudo reafirmar que Blockchain ofrece un nivel superior de escritura de información, y mediante los procesos de firma digital que incorpora esta tecnología, con lo cual y sumado a los beneficios de trazabilidad y legitimad de la información afirmaron la elección de la tecnología para resolver el problema en investigación.

Siendo *Blockchain* una tecnología que apunta a un crecimiento exponencial a nivel mundial se recomienda al equipo de la RAS, participar activamente en foros oficiales y confiables referentes a esta tecnología, para mantenerse a la vanguardia, y de

esta forma prever cualquier toma de decisión referente a la utilización de la tecnología.

6.2 Definición de datos requeridos para la transacción de evaluación, venta y compra agrícola

Mediante un cuestionario y una entrevista, aplicada a diferentes muestras de la población de la RAS, se lograron identificar las historias de usuario del proyecto, y estas a su vez, permitieron delimitar mas detalladamente el alcance del proyecto y la funcionalidad esperada por la RAS.

Adicionalmente, se logró establecer una estructura para la redacción de historias de usuario, que representaron los requerimientos funcionales y no funcionales de la solución.

Por otro lado, para poder llevar a cabo el control y seguimiento de las historias de usuario completadas, se estableció una pizarra de trabajo mediante Azure DevOps y se establecieron sprints de trabajo para dar visibilidad del avance del proyecto en el aspecto técnico, todos los martes se fijó la revisión del sprint con el equipo técnico de la RAS.

Se recomienda al equipo de la RAS continuar con el uso de las metodologías ágiles de desarrollo para realizar cualquier ajuste, cambio o mejora a la solución, esto con el fin de entregar constantemente nuevas versiones de la aplicación que cumpla con los requerimientos del negocio y garantizar la evolución del software.

6.3 Diagrama de arquitectura de la solución

Con la creación del diagrama de arquitectura de la solución, se logró identificar los diferentes componentes de software y hardware que debían ser configurados y programados respectivamente, además de la interacción de dichos componentes entre sí.

Con la ayuda del equipo de la RAS, y tomando en consideración aspectos importantes como presupuesto y seguridad, se crearon los recursos definidos en el diagrama de arquitectura para la creación de un servicio de aplicación donde se publicó el microservicio de Blockchain.

También se habilitó una máquina virtual, donde se configuró EOS.io y sus herramientas: Nodeos, Cleos y Keosd. Esto le brindó al equipo de la RAS una red privada de Blockchain donde se almacenan sus bloques de información.

Adicionalmente, con la ayuda del diagrama de arquitectura, se logró determinar la forma en la que el software iba a interactuar con los servicios disponibles en Microsoft Azure.

Siendo la infraestructura un elemento fundamental para el éxito de este proyecto, a continuación, se enlistan una serie de recomendaciones que pueden ser consideras a futuro para robustecer la infraestructura que soporta la solución del microservicio de *Blockchain*:

- Evaluar la posibilidad de crear un plan de respaldo para la máquina virtual donde se ejecuta el servicio de EOS.
- Evaluar la implementación de un certificado de seguridad para que el servicio de Nodeos corra mediante el protocolo Https.
- Habilitar que la máquina virtual permita únicamente el tráfico proveniente del servicio de aplicación de Azure.
- Capacitar al equipo técnico en el desarrollo de Smart-Contracts, los cuales abstraen la lógica de interacción directa con Blockchain.
- Constante monitoreo de la aplicación mediante *Application Insights*, para poder mejorar el rendimiento o tomar acción sobre posibles mejoras.
- Velar por el correcto establecimiento de recursos de hardware para garantizar el funcionamiento de los servicios del software.

6.4 Programación del API REST

Con la programación de un API REST basado en una arquitectura de microservicio, se logró abstraer la lógica de negocio de la RAS, y luego escribir dicha lógica en bloques de Blockchain mediante EOS y su Smart-Contract *eosio.token*.

Adicionalmente, se lograron configurar los servicios de bitácoras de la solución, esto le permitirá al equipo técnico de la RAS monitorear el uso del microservicio, y podrán de una manera detallada dar seguimiento a errores, excepciones o eventos importantes que suceden en el microservicio.

Otro punto a concluir, es que se logró el establecimiento de una documentación detallada del API haciendo uso de las especificaciones de OpenApi y de Swagger, con esta documentación el equipo de la RAS podrá fácilmente identificar la forma en que sus aplicaciones internas deben hacer uso del microservicio.

En el aspecto de seguridad, se logró implementar Okta como servicio de autenticación para el microservicio, lo que significa que solo peticiones autorizadas y administradas por el equipo de la RAS podrán acceder al recurso de Blockchain, esto brinda a la solución una capa de seguridad para evitar cualquier uso descontrolado del microservicio.

Finalmente, para garantizar el correcto funcionamiento de la lógica abstraída en el microservicio, se establecieron pruebas unitarias que velan por la integridad del código y la lógica de negocio aplicada.

A continuación, se enlistan las recomendaciones al equipo de la RAS para potencializar el funcionamiento del API:

- Aplicación de roles para hacer uso de los puntos de acceso del microservicio de Blockchain.
- Evaluar la implementación de pruebas de integración adicionales a las ya definidas como pruebas unitarias, para velar por el correcto funcionamiento de componentes periféricos.
- Establecer patrones y estandarizaciones para los desarrolladores que tengan a cargo modificaciones o mejoras de la aplicación a futuro.

- No omitir la creación de pruebas unitarias relacionados a las mejoras de la solución o mantenimientos de esta.
- Apalancamiento en técnicas de manejo de versiones en un API y utilización del cache para maximizar el rendimiento y facilitar el mantenimiento de este.

6.5 Plan piloto

Haciendo uso de Microsoft Azure y Azure DevOps, se completó un plan piloto de publicación de la solución y una transición de conocimiento técnico al equipo de TI de la RAS.

Mediante Azure DevOps, se logró desarrollar un proceso de publicación de la aplicación, por lo tanto, cada vez que se envíe un cambio, mejora o corrección a la rama principal del código fuente de la solución, automáticamente se ejecutará un proceso que revisa, compila, prueba y publica el microservicio en la infraestructura de la RAS haciendo uso de Microsoft Azure.

Adicionalmente, se coordinó una sesión con el equipo técnico de la RAS, para revisar a detalle la estructura de la solución, capacitar técnicamente al equipo y mostrar el funcionamiento de la documentación técnica haciendo uso de los archivos *Readme.md* situados a lo largo de la estructura del repositorio de código

Sumado a las conclusiones anteriores, a continuación, se enlistan una serie de recomendaciones para el equipo de la RAS bajo el criterio de plan piloto:

- Segregación de ramas para el control de versiones y ambientes de la aplicación, por ejemplo, crear un ambiente para QA y otro para UAT.
- Establecimiento de reglas, aprobaciones y puntos de control para aceptar cambios o nuevas versiones del código de la solución.
- Mantener actualizados los archivos README.md que contienen información técnica necesaria para la ejecución y compilación de los componentes de la solución.
- Velar por la constante capacitación al negocio, para que los miembros no técnicos de la organización comprendan los alcances de la solución.

BIBLIOGRAFÍA

Error! Hyperlink reference not valid.

GLOSARIO

- .Net Core: es la plataforma de desarrollo de Microsoft más moderna, de código fuente abierto, multiplataforma y de alto rendimiento para la creación de todo tipo de aplicaciones.
- Automatización: Convertir ciertos movimientos en movimientos automáticos o indeliberados.
- C#: es una evolución que **Microsoft** realizó de este lenguaje, tomando lo mejor de los lenguajes C y C++, y ha continuado añadiéndole funcionalidades, tomando de otros lenguajes, como java, algo de su sintaxis evolucionada.
- Dashboard: es una herramienta de gestión de la información que monitoriza, analiza y muestra de manera visual los indicadores clave de desempeño (KPI), métricas y datos fundamentales para hacer un seguimiento del estado de una empresa, un departamento, una campaña o un proceso específico.
- Deliverables: Entregables.
- DevOps: El término DevOps, que es una combinación de los términos ingleses *development* (desarrollo) y *operations* (operaciones), designa la unión de personas, procesos y tecnología para ofrecer valor a los clientes de forma constante.
- Framework: es un marco o esquema de trabajo generalmente utilizado por programadores para realizar el desarrollo de software.

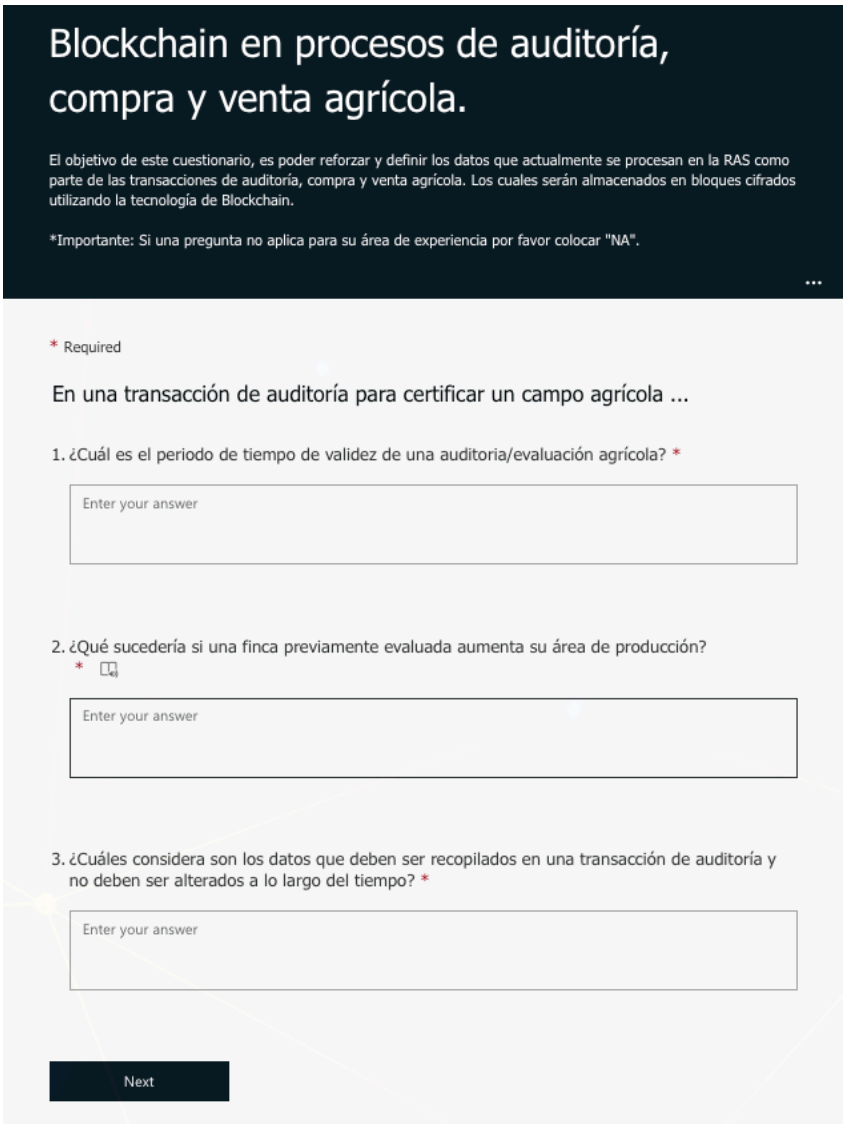
- Frontend: Es la parte de un sitio web que interactúa con los usuarios, por eso decimos que está del lado del cliente.
- HTTP: Abreviatura de la forma inglesa *Hypertext Transfer Protocol*, 'protocolo de transferencia de hipertextos', que se utiliza en algunas direcciones de internet.
- IoT: Agrupación e interconexión de dispositivos y objetos a través de una red (bien sea privada o Internet, la red de redes), dónde todos ellos podrían ser visibles e interaccionar.
- JSON: Corresponde a las siglas *JavaScript Object Notation* o Notación de Objetos de JavaScript, es un formato ligero de intercambio de datos, que resulta sencillo de leer y escribir para los programadores y simple de interpretar y generar para las máquinas.
- Kilovatio: es la unidad que se utiliza para medir la potencia eléctrica
- Monolítico: Son aquellos sistemas en los que su centro es un grupo de estructuras fijas, las cuales funcionan entre sí.
- NuGet: es un complemento para Visual Studio para instalar y gestionar librerías de terceros de una manera automatizada.
- NUnit: es un framework de unit testing (ya sea pruebas unitarias, de integración o de cualquier otro tipo)
- PHP: Acrónimo recursivo de PHP: *Hypertext Preprocessor*, es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML.

- Plataforma: es un sistema que sirve como base para hacer funcionar determinados módulos de *hardware* o de *software* con los que es compatible.
- QA: Su significado en ingles *Quality Assurance,* es la manera de prevenir errores y defectos en productos manufacturados y evitando problemas a la hora de entregar productos o servicios a los clientes.
- RAS: Red de Agricultura Sostenible.
- SDK: Un kit de desarrollo de software (SDK) es un conjunto de herramientas que ofrece generalmente el fabricante de una plataforma de hardware, un sistema operativo (SO) o un lenguaje de programación.
- SSH: SSH o *Secure Shell*, es un protocolo de administración remota que le permite a los usuarios controlar y modificar sus servidores remotos a través de Internet a través de un mecanismo de autenticación.
- Token (Blockchain): una unidad de valor que una organización crea para gobernar su modelo de negocio y dar más poder a sus usuarios para interactuar con sus productos, al tiempo que facilita la distribución y reparto de beneficios entre todos sus accionistas
- Token (Seguridad): Un token de seguridad (también llamado llave digital o llave electrónica) es un dispositivo físico utilizado para acceder a un recurso restringido electrónicamente. El token se utiliza como complemento o en lugar de una contraseña.

- UAT: Significa "Prueba de aceptación del usuario". UAT es un proceso diseñado para ayudar a garantizar que los productos cumplan con las expectativas del usuario cuando sean lanzados
- Versión: gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración de este.
- Visual Studio: es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para Windows y macOS.
- XML: Este acrónimo significa *Extensible Markup Language*, que es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos.
- Ihub de la RAS: Nombre que recibe la plataforma en la cual se registran los usuarios de la RAS y se llevan a cabo el proceso de las evaluaciones.

ANEXOS

7.1 Cuestionario enviado al equipo de la RAS



Blockchain en procesos de auditoría, compra y venta agrícola.

El objetivo de este cuestionario, es poder reforzar y definir los datos que actualmente se procesan en la RAS como parte de las transacciones de auditoría, compra y venta agrícola. Los cuales serán almacenados en bloques cifrados utilizando la tecnología de Blockchain.

*Importante: Si una pregunta no aplica para su área de experiencia por favor colocar "NA".

...

* Required

En una transacción de auditoría para certificar un campo agrícola ...

1. ¿Cuál es el periodo de tiempo de validez de una auditoria/evaluación agrícola? *

Enter your answer

2. ¿Qué sucedería si una finca previamente evaluada aumenta su área de producción? *

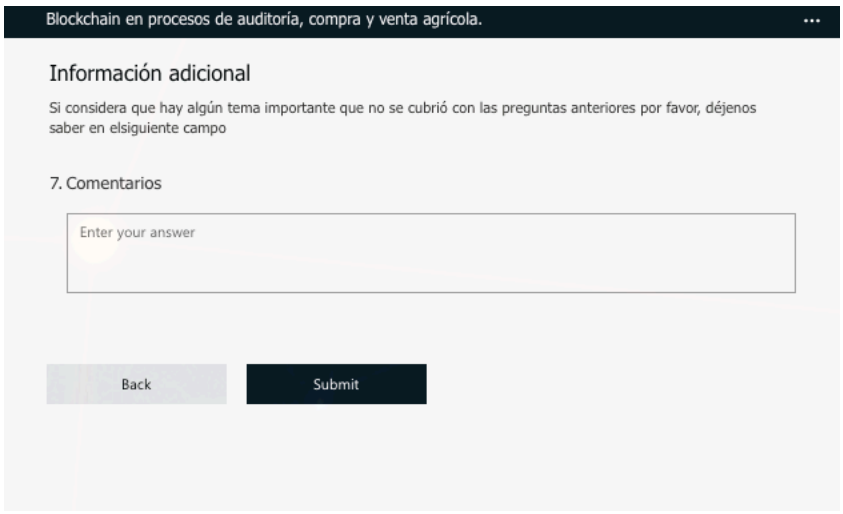
Enter your answer

3. ¿Cuáles considera son los datos que deben ser recopilados en una transacción de auditoría y no deben ser alterados a lo largo del tiempo? *

Enter your answer

Next


Blockchain en procesos de auditoría, compra y venta agrícola.
* Required
En una transacción de compra y venta agrícola ...
4. Dado que pueden existir compradores que tramitan la mercancía en diferentes unidades de medida, es posible manejar un estándar en la transacción? Ejemplo. Solo utilizar toneladas *
Enter your answer
5. Dado que pueden existir compradores/vendedores que tramitan en moneda local y otros en moneda dólar, es posible manejar un estándar en la transacción? Ejemplo. Solo utilizar dólares *
Enter your answer
6. ¿Cuáles considera son los datos que deben ser recopilados en las transacciones de ventas y compras, y no deben ser alterados a lo largo del tiempo? *
Enter your answer
Back
Next


Blockchain en procesos de auditoría, compra y venta agrícola.
Información adicional
Si considera que hay algún tema importante que no se cubrió con las preguntas anteriores por favor, déjenos saber en elsiguiente campo
7. Comentarios
Enter your answer
Back
Submit

7.2 Carta de aceptación de requerimientos

Martes, 19 de octubre de 2021

Departamento de Tecnología de Información y Comunicación

Red de Agricultores Sostenibles Costa Rica, S.A.

Estimados,

El objetivo de la presente es para confirmar la aceptación de los requerimientos técnicos definidos mediante la entrevista realizada el 27 de agosto de 2021 y el cuestionario enviado el 01 de septiembre de 2021 a los miembros de la RAS mediante el formulario de Google.

Para mas detalle de los requerimientos, se puede ver el anexo a esta carta, donde se enlista cada uno de ellos.

Atentamente,

Lirroy Coto

Estudiante

Jose Rodriguez

Firmado digitalmente por Jose Rodriguez
Fecha: 2021.10.19 18:07:48 -06'00'

Jose Rodríguez

Gerente de TI

7.3 Preguntas realizadas durante la entrevista

Documento de análisis de requerimientos y especificaciones técnicas

A continuación se presentan una lista de preguntas que permitirá determinar la situación técnica actual de la RAS, y permitirá a sus vez poder definir los requerimientos cubiertos en el alcance del proyecto.

Preguntas relacionadas a transacciones de evaluación, compra y venta agrícola

1. ¿Cuáles datos se recopilan actualmente durante una evaluación, compra y venta agrícola, que requieren ser almacenados en un bloque de Blockchain?
2. ¿Cuáles validaciones de datos se aplican actualmente a los datos recopilados en una transacción de evaluación, compra y venta agrícola?
3. ¿Cuáles datos de auditoria de sistemas se están registrando actualmente que se quieren incluir en el bloque de Blockchain?
4. ¿Cuáles unidades de medida utiliza la RAS para el registro de las cantidades de producción?
5. Existe la posibilidad de estandarizar las unidades de medida a una única unidad?

6. ¿Cuál es el objetivo de calcular saldos de los productores agrícolas?
7. ¿Cómo se calculan los saldos de los productores actualmente?

Preguntas relacionadas a situación tecnológica actual de la empresa

1. Actualmente como publican sus aplicaciones, cuentan con infraestructura en la nube o en sitio?
2. ¿Cómo llevan a cabo el proceso de autenticación de los usuarios para el uso de sus aplicaciones?
3. ¿Qué herramienta utilizan para velar por la ciclo de vida del software?
4. ¿Existe algún servicio de Blockchain implementado actualmente?
5. ¿Cuáles son las tecnologías estándar que utilizan para el desarrollo de sus aplicaciones?
6. ¿Cuentan con algún documento de reglas para el desarrollo o programación de nuevo software?
7. ¿Cuentan con algún repositorio de código y documentación para incluir más software?
8. ¿Cuentan con alguna política que prevenga el uso de código abierto?
9. ¿Cuáles ambientes aparte de producción manejan para la revisión y pruebas de nuevos entregables?