UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

FACULTAD DE INGENIERÍA

REINGENIERÍA DE UN SISTEMA DE COMUNICACIÓN CON DISPOSITIVOS MÓVILES GPS DE UNA EMPRESA DE SEGURIDAD Y LOGÍSTICA

REPORTE DE APLICACIÓN DE CONOCIMIENTOS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

RICARDO GARCÍA CEJUDO

TUTOR:

DR. MARCELO ROMERO HUERTAS

FACULTAD DE INGENIERÍA
UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

Toluca, México, octubre 2021

Resumen

El presente documento muestra el trabajo realizado para la reingeniería y optimización de una aplicación web cuyo propósito es la comunicación correcta y efectiva con dispositivos móviles GPS. La reingeniería consistió en diseñar y crear una nueva versión del sistema existente con la finalidad de generar más valor reduciendo tiempos de comunicación entre dispositivos GPS, así como poder comunicarse con distintos proveedores de GPS, donde cada uno tiene su propia trama de comunicación que por motivos de seguridad de la empresa no fue permitido compartirla.

Para realizar la reingeniería del sistema se empleó la metodología de SCRUM, una metodología ágil donde se realizaron entregas continuas al ambiente de producción. Este reporte documenta cómo se ejecutaron las diferentes fases dentro del proceso de desarrollo.

Se inicia describiendo la problemática que tenía la empresa con el sistema anterior, así como las nuevas necesidades del negocio, que en conjunto fueron la base para el planteamiento de las alternativas de solución y selección de la solución llevada a cabo.

En la toma de requerimientos se realizó un modelado de negocio, donde se consideró el valor de cada uno de ellos con el objetivo de asignar una prioridad y planificar correctamente los Sprints o iteraciones dentro de nuestra metodología de SCRUM.

En la etapa de análisis y diseño se incluyen diagramas de flujo, arquitectura planteada, así como el modelado UML de los requerimientos, se definen los estándares de programación, el patrón de diseño y el conjunto de tecnologías a utilizar.

Para la fase de pruebas del sistema se realizó un diseño de diferentes tipos de pruebas con la finalidad de asegurar que el producto cumpliera los estándares de calidad de la empresa y asegurar la funcionalidad continua de la implementación.

Índice

Tablas

Figuras

Introducción

Hoy en día las empresas apuestan más por la tecnología que hace algunos años, debido a esto los sistemas de información han tomado mucha mayor importancia y relevancia, puesto que ayudan a mantener o mejorar la posición competitiva de una compañía, influyendo en el comportamiento o incluso en la orientación del negocio de ésta. El contar con sistemas robustos, seguros y adaptables al cambio es fundamental para dicho propósito.

Todo sistema surge con el objetivo de cubrir una necesidad de negocio, sin embargo, sin el apoyo de una metodología de desarrollo de software sólida y el encaminamiento correcto, dicho sistema puede no cubrir la necesidad y por el contrario ofrecer más vulnerabilidades o deficiencias.

En este documento se habla sobre la aplicación web de una empresa de seguridad en el autotransporte de carga que cuenta con un centro de monitoreo que trabaja todos los días del año y que está monitoreando distintos servicios en recorrido a largo del territorio mexicano, para dicho seguimiento se instala un dispositivo GPS en cada transporte vehicular y éstos se encuentran comunicados con dicho sistema, se tienen conectados miles de dispositivos reportando mensajería constantemente, cada mensaje incluye información del transporte como es ubicación (latitud, longitud, altitud), velocidad, orientación y una serie de alertas que pueden generar eventos de reacción, como es la pulsación de un botón de pánico ante una situación de peligro, entre otros.

La empresa cuenta con distintos proveedores de dispositivos GPS, en el que cada proveedor tiene su propia forma de comunicación para el envío de mensajes o tramas, por lo que anteriormente se tenía un sistema por cada proveedor, esto volvía complicado el mantenimiento ante nuevas solicitudes de negocio o bien actualización de firmwares de dispositivos GPS, ya que el cambio solicitado tenía que aplicarse a cada uno de estos sistemas.

Debido a que el negocio de la compañía es de seguridad en autotransporte de carga, se vuelve importante el tiempo de reacción ante eventos o situaciones de riesgo de un vehículo, como puede ser una falla mecánica, un accidente automotriz, una desviación de ruta o un robo a mano armada, esto aunado a que cada vehículo lleva consigo un valor monetario alto en la carga y un compromiso de fecha de entrega la criticidad se eleva considerablemente. El tiempo en que se tenía reportando cada GPS era de 1 minuto en movimiento y 5 minutos cuando el vehículo está detenido, esto debido al pobre rendimiento del sistema, ya que el reducir el tiempo de reporteo se aumenta el número de mensajes a procesar y esto ralentizaba el sistema, limitando los canales de comunicación y todos los procesos asociados se veían afectados. Ante esta situación no se podía reducir el tiempo de reporteo de los GPS por lo que al presentarse una situación de peligro podría tomar hasta 5 minutos para que el centro de monitoreo de la empresa se percatara del evento.

La complejidad que presenta el sistema para mejorar el tiempo de respuesta es alta, ya que no se tiene sólo un sistema, si no uno por cada proveedor de GPS, ya que anteriormente consideraban que al tener un protocolo distinto de comunicación se debería crear una aplicación específica para esta, además de que en caso de tener un nuevo proveedor se tiene que desarrollar un nuevo sistema que cubra la lógica de comunicación, sumado a que se tiene la obligación de dar una respuesta rápida y oportuna ante cualquier riesgo que presente el transporte en su recorrido, hacen que la empresa se vea en la necesidad de hacer cambios importantes para su sistema clave de negocio.

El presente proyecto de aplicación de conocimientos muestra la reingeniería del sistema mencionado, dicho proceso consiste en generar una nueva versión de éste con el objetivo de que esta nueva versión sea flexible, pueda soportar múltiples proveedores, mantenible, el hacer un cambio a un proveedor previo o agregar uno nuevo sea lo más transparente posible, así como optimizar el rendimiento de la

plataforma para ofrecer una respuesta más rápida ante las situaciones de peligro mencionadas.

La ingeniería de software tomó un valor importante para el éxito de este proyecto, en el cual se buscó generar software de calidad, aumentando el rendimiento de comunicación y reduciendo costos de implementación, así como tiempo ante nuevas implementaciones, además de generar la documentación necesaria que forme una base inicial del sistema a partir de la cual se pueda dar continuidad en los mantenimientos y evolución futura del mismo [4, 24, 8, 25].

En la etapa de análisis y diseño se hizo uso del lenguaje unificado de modelado, UML (Unified Modeling Language), siendo este un lenguaje estándar que permite comunicar claramente requerimientos, arquitecturas y diseños. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados [1]. Por otro lado, se utilizó la metodología ágil de SCRUM, el cual es un marco de trabajo para desarrollo ágil de software en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo y obtener el mejor resultado posible de los proyectos [2].

En la etapa de implementación del proyecto se usó Node.js como plataforma de desarrollo multiplataforma, es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, por ejemplo, servidores web [3].

Como resultado de este proyecto, se implantó un nuevo sistema que reúne las nuevas especificaciones y necesidades, logrando optimizar los tiempos de reporteo al tener mensajes de cada dispositivo GPS a 4 segundos en movimiento y 1 minuto detenidos,

cabe mencionar que esta limitación de tiempo depende directamente de cada proveedor de GPS y no del aplicativo, ya que con ciertos proveedores se realizaron pruebas de estrés donde se tuvieron resultados de comunicación en tiempo real con GPS que sí lo soportaban, es decir el envío y recepción de mensajes se realizaba en milisegundos, sin embargo con otros proveedores no fue posible esto, por lo que la empresa optó por usar el tiempo de 4 segundos como estándar. Finalmente, se crea un sólo proyecto para todos los proveedores de GPS existentes y se crea un nuevo mensaje de GPS estándar que ayuda a la integración de nuevos proveedores de manera rápida y efectiva, logrando que el centro de monitoreo de la empresa se alerte ante situaciones de riesgo casi en tiempo real.

Objetivo general del reporte de aplicación de conocimientos

Describir la reingeniería realizada a un sistema web que funge como servidor TCP para la comunicación directa bidireccional entre dispositivos móviles GPS y el aplicativo, usando UML para la maquetación, SCRUM como metodología de desarrollo y Node.js como plataforma de desarrollo.

Alcances y limitaciones

La funcionalidad a considerar de este sistema quedó determinada por la existente en el sistema anterior al momento de iniciar con el proyecto, incorporando todas aquellas nuevas mejoras de rendimiento sin que éstas afectaran las reglas de negocio existentes. El diseño de software considera una arquitectura completamente escalable, resiliente, mantenible y sobre todo con rendimiento óptimo, reduciendo costos de implementación y tiempos ante la integración de nuevos proveedores de dispositivos GPS.

Directriz del reporte de aplicación de conocimientos

Los conocimientos empleados para la implementación del sistema de este reporte están relacionados al área de ingeniería de software y de forma específica para el desarrollo de aplicaciones web en plataforma de desarrollo Node.js con lenguaje de

programación JavaScript; por lo que es recomendable que el lector tenga conocimientos básicos en metodologías para el desarrollo de sistemas y en lenguajes de programación. Por razones de confidencialidad, no se describen detalles técnicos ni de negocio que puedan revelar identidades y representen riesgos para la integridad de la empresa propietaria del sistema en cuestión.

Organización del documento

El contenido de este reporte de aplicación de conocimientos está conformado por tres capítulos estructurados de la siguiente forma:

El Capítulo I. Antecedentes, explica la problemática que presenta el sistema actual y describe las necesidades que el negocio tiene, planteando entonces las alternativas de solución. Así mismo aborda un reconocimiento de los procesos del negocio sobre la funcionalidad completa y establece los principales conceptos que conforman criterios a seguir para el desarrollo de una aplicación escalable. El capítulo finaliza con la definición de funcionalidad dentro del alcance del proyecto.

El Capítulo II. Diseño e implementación, describe cómo se llevó a cabo el proceso de toma de requerimientos y el análisis y diseño UML que se realizó a partir de dichos requerimientos, se contempla además un diseño arquitectónico de software, así como un diseño de funcionalidad. También en este capítulo se menciona la etapa de construcción de los elementos de software que satisfacen tanto los requerimientos como el diseño.

El Capítulo III. Pruebas e implementación, documenta los diferentes tipos de pruebas realizadas para garantizar la calidad del producto resultante. Así mismo se comenta sobre la implementación del sistema y puesta en marcha en un ambiente productivo.
Finalmente, en las conclusiones se describen los resultados y beneficios que se obtuvieron con el desarrollo de este proyecto, la experiencia desarrollando un sistema como éste y el valor que un sistema puede aportar a una empresa.

Capítulo I. Antecedentes

Este capítulo describe la finalidad del sistema, la problemática encontrada, el análisis preliminar en función a la problemática y la visión y alcance del proyecto.

1.1 Análisis preliminar

Una empresa dedicada al negocio de seguridad de vehículos que busca prevenir, frustrar o recuperar robos de mercancía, instala dispositivos GPS en transportes de carga como puede ser, tráiler, camioneta, camión de 3 y 1/2 toneladas, o autos particulares. Los dispositivos se instalan para emitir datos como la ubicación, velocidad, orientación del vehículo, eventos como botones de pánico, donde el transporte tiene un botón escondido en el vehículo, de tal forma que el chofer sea el único que sabe dónde se encuentra, con la finalidad de que ante una situación de peligro sea pulsado y llegue la notificación al centro de monitoreo de la empresa, el cuál tratará de enviar un custodio al lugar con el objetivo de prevenir un robo de mercancía o bien un asalto a mano armada.

Como se puede observar el tiempo de reacción ante un evento toma mucha importancia, ya que la vida de personas está en riesgo, además de que se puede perder una carga con altos valores monetarios.

El sistema que tenían en comunicación directa con los GPS presentaba lentitud ya que no soportaba más de 700 GPS conectados al mismo puerto por un mal manejo de memoria y conexión de red TCP. De igual forma, al existir distintos proveedores de GPS y al tener cada uno su propio protocolo de comunicación se tenía un sistema por cada proveedor que la empresa integraba. Esto provocaba que ante un cambio en la lógica de negocio tenían que modificar cada uno de estos sistemas, realizar pruebas pertinentes a todos los sistemas y en seguida liberarlos a producción.

Por otro lado, cada sistema puede enviar mensajes o comandos directamente a cada GPS para activar por ejemplo un paro de motor gradual en caso de robo con el

objetivo de que el transporte no pueda moverse una vez se detecte que ha sido siniestrado. Con la funcionalidad del sistema mencionado, la lógica de negocio indicaba que, al tener una nueva solicitud de comando, se tenía que ir a buscar a la base de datos a qué sistema estaba conectado un GPS, validar el puerto de conexión, obtener el comando que debe enviar y tratar de enviarlo, sin embargo, por la lentitud previamente descrita, el comando tenía un retraso en el envío de hasta 2 minutos o bien muchas veces ni siquiera llegaba dicho comando lo que provocaba un menor número de recuperaciones en eventos de robo.

El sistema fue desarrollado en el año 2010 con el objetivo de satisfacer las necesidades de la empresa, sin embargo, derivado del crecimiento exponencial gracias al éxito de la organización con los buenos números de recuperación y frustración de robos que habían obtenido se aumentó el número de GPS, así como el número de proveedores de estos. Lo que provocó que el sistema se viera estresado y los tiempos de respuesta ante una situación de riesgo se vieron afectados y el número de eventos recuperados o frustrados fue en decremento.

1.1.1 Planteamiento del problema

Como resultado de la incorporación de más GPS y más proveedores se empieza un sin fin de problemas que hace que la empresa tenga que dar un giro completo y aumentar la prioridad por subsanar las deficiencias del sistema actual, donde se describen las problemáticas al momento de analizar los sistemas:

a) **Versiones diferentes entre cada sistema**

Se tenían 4 sistemas, uno para cada proveedor de GPS, sin embargo, la lógica de negocio interna de cada uno no era la misma, puesto que al tener más sistemas se volvía más difícil el control de cambios entre ellos, logrando que no se tuvieran los mismos cambios.

b) Complejidad para incorporar un nuevo proveedor de GPS

Al tener un crecimiento exponencial en el negocio la empresa siempre estaba a la escucha de integraciones con nuevos proveedores, sin embargo, estas oportunidades se perdían ya que resultaba casi imposible el integrar un nuevo proveedor debido a la complejidad que presentaba el crear un nuevo sistema, desde cero, crear la comunicación con el proveedor y luego realizar las pruebas de integración, extendiendo en demasía los tiempos de desarrollo.

c) Límite de conexiones concurrentes

Al conectar más dispositivos GPS se observó que el sistema no soportaba más de 700 conexiones simultáneas ya que al aumentar este número de conexiones el sistema empezaba a ralentizarse o bien caer en errores inesperados, haciendo inestable la solución y sobre todo el negocio de la empresa.

d) Pobre rendimiento

Debido a la forma en que se desarrollaba cada sistema, se tenía un rendimiento muy bajo comparado con las nuevas expectativas de la empresa. Sobre todo, porque se menciona hay vidas en riesgo y pérdidas millonarias de mercancía, con los tiempos que se estaban obteniendo resultaba complicado el recuperar vehículos reportados como siniestrados o bien el llegar a tiempo a lugar a frustrar un robo.

e) Herencia de problemas

El desarrollo de la primera versión del sistema fue llevado a cabo con la ausencia de una metodología formal, lo que derivó a tener una baja calidad en el sistema, así como la falta de elementos de control para la operación de éste, problema que aumentó cuando se tuvo la necesidad de agregar más GPS y más proveedores de éstos. Donde se hacía una clonación del proyecto base, con las carencias antes mencionadas y se implementaba el protocolo de comunicación del nuevo proveedor.

Dentro de las principales carencias de la aplicación se encuentran errores de diseño, requerimientos y documentación, siendo las principales:

- **Mal manejo de memoria.** Al ser un servidor web con conexiones concurrentes de múltiples dispositivos al cerrarse una conexión, deben liberarse todos los recursos asociados a ésta ya que de otra forma se estarán teniendo fugas de memoria lo que provoque cierres repentinos de la aplicación o bien trabajar de manera lenta.
- **Manejo de errores deficiente.** A pesar de que el sistema escribe un log de errores; muchas de las excepciones dentro de la aplicación no son controladas lo que provoca un funcionamiento inadecuado o cierres repentinos, por lo que es difícil rastrear dichos errores parar tratar d encontrar la raíz de los errores.
- **Funcionalidad acoplada.** Los diferentes módulos que componen la aplicación están muy acoplados entre sí y no es posible que estos puedan ser utilizados de forma independiente.
- **Mayor esfuerzo para soporte y mantenimiento de la aplicación.** Conforme se clonaron los sistemas por incorporación de nuevos proveedores, el esfuerzo requerido para llevar a cabo funciones de soporte y mantenimiento al sistema se fue multiplicando, ocasionando que el personal asignado a dichas actividades sea insuficiente para la ejecución de éstas. Esta situación por lo general impacta en los tiempos de atención y por consecuencia inconformidades en los clientes de la empresa.

1.1.2 Necesidades de negocio

Para proponer una solución adecuada al problema planteado y tener una visión completa del proyecto, dentro del análisis preliminar se identificaron las necesidades de negocio, mismas que a continuación se listan:

- Tener un sistema que permita reducir el tiempo de implementación de un nuevo proveedor de GPS.

- Contar con un sistema que incluya toda la funcionalidad existente entre las diferentes instancias actuales.
- Tener la posibilidad de configurar la funcionalidad de acuerdo con las necesidades de cada proveedor de GPS.
- Hacer más flexible el desarrollo y mantenimiento del sistema que posibilite agregar nuevas funcionalidades de valor agregado para todos los clientes.
- Contar con un sistema más estable y de alta calidad.
- Reducir los tiempos y costos de desarrollo, así como de mantenimiento.
- Automatizar procesos manuales para mitigar los errores operativos.
- Aumentar el número de GPS conectados de manera concurrente.
- Reducir el tiempo de reporteo de los GPS para tener una visión en tiempo real de eventos.
- Envío de comandos a GPS con éxito
- Persistencia de mensajes sin afectar rendimiento de procesos de comunicación

1.1.3 Alternativas de solución

Entendiendo la problemática y considerando las necesidades que tiene el negocio se plantearon dos alternativas de solución, la primera fue considerar una aplicación existente en la cuál se tuviera la certeza de que cubría más de las reglas de negocio requeridas por el usuario final y agregar la estandarización de mensajes, así como el resto de protocolos de comunicación de proveedores y la segunda fue realizar una reingeniería y crear la aplicación desde cero, a continuación se listan ventajas y desventajas de estas opciones:

1. **Tomar la versión más completa y agregar la funcionalidad específica de cada proveedor para generar una sola versión.**
    a. Ventajas
        i. Menor tiempo.
        ii. Menor costo.
    b. Desventajas
        i. No se puede hacer explícitamente multi proveedor.

ii. No considera agregar nuevas funcionalidades.
iii. No considera depurar una gran cantidad de errores.
iv. No considera la mejora de rendimiento.

2. **Diseñar y crear una nueva aplicación altamente escalable, mantenible y parametrizable.**
    a. Ventajas
        i. Permite la parametrización y configuración de funcionalidad y reglas de negocio.
        ii. Arquitectura que permita aprovechar nuevas tecnologías que tengan un mejor rendimiento.
        iii. Aplicación multi proveedor con una lógica de negocio centralizada.
        iv. Aumento de rendimiento.
    b. Desventajas
        i. Mayor tiempo.
        ii. Mayor costo.

1.1.4 Alternativa de solución seleccionada y justificación

Actualmente el sistema es considerado un producto clave para la empresa, pues es la entrada de datos de los vehículos en recorrido y de este sistema parte mucha lógica de negocio posterior a cada uno de los mensajes reportados por los dispositivos GPS.

Por lo anterior, para mantener a la organización en una posición favorable con miras al crecimiento que tienen proyectado en los siguientes años, se llevó a cabo la segunda alternativa de solución porque ésta nos permite:

- **Contar con una solución flexible.** En el cual a través de la parametrización se pueda asignar o limitar la funcionalidad, realizar cambios en las reglas de negocio, incorporar nuevos proveedores, configuración de tiempos y respuestas.

- **Estandarizar un mensaje de GPS.** Con este nuevo sistema se tiene un nuevo mensaje estándar de GPS, donde cada propiedad del mensaje se llame igual independientemente del proveedor del que provenga la trama.
- **Una sola versión.** Se tendrá una sola versión completa y actualizada; que priorice el rendimiento para una reacción oportuna ante situaciones adversas.
- **Altos estándares de calidad.** Se tendrá una arquitectura resiliente, escalable, mantenible, además de realizar las pruebas pertinentes para asegurar el rendimiento, así como conocer los alcances de este nuevo sistema con el fin de prever errores de funcionamiento, o bien funcionamiento inadecuado. Esto con el objetivo de cubrir con los 15 factores que el modelo de Calidad de Software Deutsch and Willis describe: Correctividad, Fiabilidad, Eficiencia, Integridad, Usabilidad, Mantenibilidad, Flexibilidad, Portabilidad, Reutilización, Interoperabilidad, Verificabilidad, Capacidad de Expansión, Seguridad, Manejabilidad y Vigencia [25].

1.2 Visión y Alcance del proyecto

1.2.1 Alcance

Una vez seleccionada y aceptada la alternativa de solución se definió el alcance del proyecto, el cual es el siguiente:

- **Análisis preliminar.** El objetivo de este análisis es contar con un entendimiento adecuado del problema, tener una visión más clara de las necesidades de negocio y generar un inventario de funcionalidad existente en cada instancia del aplicativo. Por otro lado, se llevó a cabo en paralelo con la definición de alcance y fue fundamental para completar éste. En esta etapa se hizo una introspección de lo que hacían los sistemas y la forma en que estaban desarrollados, identificar puntos de fallo, vulnerabilidades, así como posibles mejoras.
- **Definición de arquitectura y estándares tecnológicos.** En esta definición se obtuvo un conjunto de tecnologías englobadas dentro del concepto de Arquitectura orientada a servicios (SOA), mediante las cuales se diseñó y

construyó la aplicación. Otras de las arquitecturas contempladas fueron Reingeniería Web o iWeb y Arquitectura Cliente-Servidor, sin embargo fueron descartadas porque no cumplían con objetivos como escalabilidad, flexibilidad y mantenible [25].

- **Definición de requerimientos, análisis y diseño.** Especificaciones del nuevo sistema que incluye la nueva arquitectura con la funcionalidad actual. En esta etapa se crearon Historias de usuario a través de los requerimientos levantados con la finalidad de tener claro el volumen y complejidad del sistema.
- **Construcción y código fuente.** Implementación de la solución seleccionada. Descrito en el Capítulo 2 y 3 del documento.
- **Diseño y elaboración de casos de prueba.** Al ser un sistema sin interacción con usuarios se realizaron pruebas de integración a componentes lógicos, así como pruebas de estrés, además de la creación de matrices de prueba para buscar la aceptación por parte de los usuarios y el negocio.
- **Implantación del sistema en ambiente productivo.** Puesta en marcha del sistema en ambiente productivo, así como la monitorización automática del mismo. En esta etapa fue importante tener reuniones con el equipo de Soporte ya que al ser una operación de 24/7 los 365 días del año, era importante capacitar al equipo en cómo identificar anomalías o afectaciones del sistema a la operación de la empresa.
- **Manual técnico del sistema.** Documento que describa cómo está constituido el sistema y que sirva para capacitar a próximos desarrolladores.

1.2.2 Aspectos fuera de alcance

Quedó fuera del alcance del presente proyecto:

- Funcionalidad que no forma parte del sistema actual.
- Mantenimiento a sistemas previos.

1.2.3 Funcionalidad desarrollada

La funcionalidad o procesos considerados en el alcance, así como la forma de distribución dentro de Sprints se describe de forma general en la Tabla 1:

Tabla 1. *Funcionalidad desarrollada por módulo*

Módulo	**Funcionalidad**
Sprint 1 - Creación de plantilla de proyecto y gestión de control	Se crea el proyecto para la solución, así como el repositorio de código para iniciar con el desarrollo. Análisis para optimizar tiempos y recursos, así como soportar múltiples GPS por puerto.
Sprint 2 - Estandarización de mensajes de proveedores GPS	Se debe crear la estructura del mensaje estándar de GPS para que las tramas de cada proveedor se puedan representar con este mensaje. De igual forma se debe estandarizar el mensaje de envío de comandos a dispositivos GPS con la finalidad de poder activar o desactivar sensores de este. Por otro lado, se define la opción para poder reducir o aumentar el tiempo entre reportes de cada GPS.
Sprint 3 - Implementación de comunicación de proveedor Suntech	Posibilidad de recibir mensajes de GPS Suntech y obtener como resultado un mensaje de GPS estándar con los datos necesarios para continuar con los procesos de lógica de negocio, este proveedor posee actualmente más del 70% de la operación, por lo que cuenta con mayor prioridad para iniciar la implementación.
Sprint 4 - Implementación de comunicación de proveedor Cellocator	Posibilidad de recibir mensajes de GPS Cellocator y obtener como resultado un mensaje de GPS estándar con los datos necesarios para continuar con los procesos de lógica de negocio, este proveedor posee alrededor del 20% de la operación, por tal motivo la implementación se realiza en este punto.
Sprint 5 - Implementación	Posibilidad de recibir mensajes de GPS Calamp y obtener

de comunicación de proveedor Calamp	como resultado un mensaje de GPS estándar con los datos necesarios para continuar con los procesos de lógica de negocio, este proveedor posee alrededor del 10% de la operación, por tal motivo la implementación se realiza en este punto.
Sprint 6 - Desarrollo de envío y recepción de comandos GPS	Se implementa el envío de comandos a dispositivos GPS de cada proveedor con la finalidad de poder activar o desactivar sensores de este.
Sprint 7 - Realizar pruebas UAT y de aceptación	Creación de pruebas unitarias y de integración a componentes con el fin de asegurar el correcto funcionamiento del sistema. Pruebas UAT y de aceptación por parte del cliente. Notificar de manera automática sobre alertas en el funcionamiento de la plataforma, así como contar con archivos de logs que ayuden a dictaminar posibles errores.
Configuraciones generales	El sistema puede ser multi instancia lo que mejore el rendimiento y aumente el número de GPS conectados de manera concurrente, en donde pueda configurarse cuántas instancias son necesarias, los puertos de comunicación y la salida de los mensajes de dispositivos GPS.

Capítulo II. Análisis, Diseño e implementación

En este capítulo se describe la especificación de requerimientos, el análisis y diseño, y la implementación del diseño.

2.1 Definición de requerimientos

En el levantamiento de requerimientos se detalla cada uno de los requerimientos identificados como parte del proceso de análisis y desarrollo de requerimientos que servirán de base para el diseño, pruebas e implementación del producto.

El objetivo de la fase de definición de requerimientos es especificar lo que el sistema deberá de hacer, es una descripción completa del sistema que se va a desarrollar en una redacción clara de modo que no se pueda mal interpretar [4].

Para la definición de requerimientos se tomó como referencia la documentación generada en la etapa del análisis preliminar, donde se definió la problemática a resolver, el alcance y directrices para tener en cuenta.

El proceso consistió en analizar de manera detallada cada una de las necesidades observadas sin perder el enfoque de definir una aplicación flexible y altamente configurable, así como los criterios a seguir para lograr un óptimo rendimiento. De igual forma se analizó el sistema actual para no cometer las malas prácticas que ya se venían arrastrando.

2.1.1 Especificación de requerimientos

Los requerimientos del sistema fueron documentados a través de historias de usuario, siguiendo la buena práctica de la metodología de SCRUM, en la Tabla 2 y Tabla 3 se puede observar la forma de documentar cada una de las funcionalidades, así como los criterios de aceptación para cada historia de usuario.

Tabla 2. *Historia de usuario HU-G-012*

Historia de usuario	[HU-G-012] Estandarización de mensajes de ubicación
Liberación	Sprint 2
Descripción	
Como desarrollador de software, requiero homologar un mensaje donde, independientemente del proveedor del GPS que envíe el mensaje se tenga un sólo tipo de mensaje de ubicación, con la finalidad de simplificar la lógica de negocio de los sistemas subsecuentes y éstos sólo analicen un tipo de mensaje.	
Antecedentes	
Hoy en día existen 4 tipos de mensaje y cada mensaje es distinto.	
Criterios de aceptación	
1. El mensaje debe considerar un campo para identificar el tipo de proveedor que envía el mensaje. 2. El mensaje debe considerar un campo para identificar el GPS que está enviando su ubicación o reporte de evento. 3. El mensaje debe considerar un campo para identificar el Evento del mensaje como puede ser de ubicación o de alerta. 4. El mensaje debe considerar uno o varios campos que describan la ubicación del mensaje, es decir información como latitud, longitud y altitud. 5. El mensaje debe considerar un campo que indique la velocidad en kilómetros por hora. 6. El mensaje debe considerar un campo de orientación del vehículo, de 0 a 359 grados. 7. El mensaje debe considerar un campo que indique el porcentaje de batería con el que cuenta el GPS. 8. El mensaje debe considerar un campo de fecha en formato UTC. 9. El mensaje debe considerar un campo de temperatura con valor de referencia en grados Celsius.	

10. El mensaje debe considerar una bandera para identificar si el mensaje es en tiempo real o de memoria del GPS.
11. Mantener el contenido del mensaje original para futuras referencias.

Tabla 3. *Historia de usuario HU-G-013*

Historia de usuario	[HU-G-013] Estandarización de mensajes de comando
Liberación	Sprint 2
Descripción	
Como desarrollador de software, requiero que exista un sólo tipo de comando que reciba el sistema y lo envíe al GPS, con la finalidad de estandarizar el mensaje de tipo de comando y realizar de manera más eficiente el proceso de envío de comandos.	
Antecedentes	
Hoy en día existen 4 tipos de mensaje de comando y cada mensaje es distinto.	
Criterios de aceptación	
1. El mensaje debe contener un identificador único del comando que sirva para actualizar el estatus del comando. 2. El mensaje debe contener el identificador del GPS para determinar a qué GPS se enviará dicho comando. 3. El mensaje debe agregar el identificador del Proveedor de GPS, para determinar cómo enviar el comando. 4. El mensaje debe contener el identificador del tipo de comando a enviar. 5. El mensaje debe considerar el nombre de usuario que realiza el envío del comando. 6. El mensaje debe contener un identificador del estatus en el que se encuentra la ejecución del comando. 7. El mensaje debe considerar un tiempo máximo para la ejecución del comando. 8. Es necesario que el mensaje pueda considerar parámetros extra dependiendo del protocolo de comunicación que contenga el GPS.	

Al tener un protocolo de comunicación por cada proveedor distinto de GPS, el resultado final son mensajes distintos, parte del análisis que se hizo en este proyecto constó en validar el mensaje final de cada proveedor, donde se pudo observar que a pesar de que cada mensaje era diferente se podría estandarizar en un sólo tipo de mensaje con el objetivo de agregar un campo más a cada mensaje donde se pudiera indicar a qué proveedor pertenece, de esta forma surgió la necesidad de tener un catálogo de proveedores de GPS. En la Tabla 4 siguiente se listan los campos para una entidad de proveedor.

Tabla 4. *Catálogo de proveedores*

Campo	**Tipo**	**Descripción**
Id	Número	Identificador del proveedor asignado de manera automática al momento de crear al proveedor.
Name	Cadena	Nombre del proveedor asociado.
Encoding	Cadena	Formato en el que envía la trama del mensaje, por ejemplo, UTF-8, ASCII, entre otros.
Status	Booleano	Estatus de activo o inactivo para conocer si es necesario llevar a cabo una comunicación con GPS de este proveedor.

Por otro lado, durante el análisis de las tramas se pudo destacar que cada proveedor a su vez maneja un listado de eventos que pueden enviar ante diversas situaciones, como puede ser al momento de abrir una puerta del transporte, al abrir la caja del transporte, al pulsar un botón de pánico, entre otros. Por lo que de aquí surgió la necesidad de contar con un catálogo de eventos genéricos, ver Tabla 5 y un catálogo de eventos por proveedor, ver Tabla 6, con el fin de registrar el número de evento o alerta que se envía la trama y poder relacionarla con un evento genérico.

Tabla 5. *Catálogo de eventos genéricos*

Campo	**Tipo**	**Descripción**
Id	Número	Identificador del evento.
Name	Cadena	Nombre del evento. Por ejemplo “Botón de pánico”, “Puerta Abierta”, “Puerta Cerrada”, entre otros.
Status	Booleano	Estatus del evento, activo o inactivo.

Tabla 6. *Relación de eventos genéricos y eventos del proveedor*

Campo	**Tipo**	**Descripción**
ProviderId	Número	Identificador del proveedor al que pertenece la asociación del evento genérico y la del evento del proveedor.
EventId	Número	Identificador del evento genérico.
ProviderEventId	Número	Identificador del evento exclusivo del proveedor.
Description	Cadena	Descripción del evento asociado al proveedor.
Status	Booleano	Estatus de la relación de eventos, activo o inactivo.

Una vez se extrajo la relación de eventos genéricos y eventos de proveedor se inició el proceso de estandarización del mensaje de GPS, que consistió en normalizar las propiedades de cada mensaje, como es latitud, longitud, velocidad, orientación, fecha, entre otras. Donde se tiene por ejemplo que la propiedad de fecha un proveedor la envía en tiempo central, otro en horario del pacífico o en el horario local de México. Para los eventos un proveedor envía un 100 si es un botón de pánico y otro proveedor manda 911 para indicar el mismo evento, la latitud un proveedor puede enviarla en radianes mientras que otros en grados, en este punto se entiende en qué consistió dicho proceso de estandarización y se puede ver que se tienen especificaciones distintas en cada protocolo de comunicación.

Buscando lograr que el resultado final sea obtener un sólo tipo de mensaje, se realizó el proceso de estandarización y unificación de propiedades, contemplando las propiedades estrictamente necesarias para seguir con la lógica del negocio en los sistemas que necesitan de un mensaje de GPS, de tal forma que los sistemas subsecuentes puedan tomar dicho mensaje y realizar los métodos de negocio definidos, al finalizarlo se pudo obtener un mensaje con las propiedades que se observan en la Tabla 7.

Tabla 7. *Mensaje estándar de GPS*

Campo	**Tipo**	**Descripción**
ProviderId	Número	Identificador del proveedor que está enviando el mensaje.
DeviceId	Cadena	Identificador que envía cada proveedor para diferenciar los mensajes de un GPS en particular.
EventId	Número	Identificador del evento / razón por la que se envía un mensaje del GPS.
Latitude	Decimal	Latitud de la ubicación GPS.
Longitude	Decimal	Longitud de la ubicación GPS.
Altitude	Decimal	Altitud de la ubicación GPS.
Speed	Decimal	Velocidad de la ubicación GPS.
Odometer	Decimal	Kilometraje recorrido por una unidad GPS, en metros.
Degrees	Decimal	Orientación de la ubicación GPS, de 0 a 359 grados.
Satellite	Entero	Número de satélites de donde se está obteniendo la ubicación GPS.
Temperature	Decimal	Temperatura a la que se encuentra el dispositivo.
Battery	Decimal	Porcentaje de batería del dispositivo, de 0 a 100%.
GpsDate	Fecha	Fecha en UTC de la ubicación GPS.
RawData	Cadena	Cadena con la información en crudo de lo que está enviando el proveedor de GPS en su mensaje.

2.2 Análisis y Diseño

El objetivo del análisis y diseño es mostrar cómo se realizará el sistema en la fase de implementación, contiene descripciones de cómo los objetos de un diseño de clases colaboran para desempeñar casos de uso.

La presente fase tuvo como finalidad realizar un análisis para generar el diseño de software del sistema, también dentro de esta fase se llevaron a cabo la definición de; diseño de clases, interfaces de lógica y de negocio, conjunto de tecnologías, estándares, arquitectura y base de datos, que forman parte de un documento de arquitectura y en conjunto con el diseño de software fueron una base sólida en la construcción del sistema.

En la elaboración del análisis y diseño se consideraron los elementos previos a esta fase: análisis preliminar y especificación de requerimientos de negocio; y se mantuvo en mente el enfoque de diseñar una aplicación flexible y altamente parametrizable; para lo cual en la fase previa fue realizado un arduo trabajo y en la presente fase se continuó y completó dicho trabajo en lo referente a diseño.

2.2.1 Análisis y diseño de software

Una arquitectura de software se ocupa del diseño e implementación de la estructura de software a alto nivel. Es el resultado de ensamblar un cierto número de elementos de arquitectura en algunas formas bien definidas para satisfacer la mayor funcionalidad y requerimientos como el desempeño del sistema, así como algunos otros requerimientos no funcionales tales como confiabilidad, escalabilidad, portabilidad y disponibilidad [5].

Para crear el diagrama de una arquitectura de software, usamos un modelo compuesto de múltiples vistas o perspectivas con el propósito de dirigir arquitecturas grandes y complejas. El modelo utilizado en el proyecto es el modelo de vistas de arquitectura "4+1" creado por el ingeniero de software Philippe Kruchten, donde

múltiples vistas concurrentes pueden utilizarse para describir la arquitectura de software de un sistema complejo; formado de cinco vistas principales, ver Figura 1 [6]:

1. **La vista lógica.** La cual es el modelo de objeto del diseño. (cuando un método de diseño orientado a objetos es usado).
2. **La vista de proceso.** El cual captura la concurrencia y aspectos de sincronización del diseño.
3. **La vista física.** La cual describe los mapeos del software dentro del hardware y refleja sus aspectos distribuidos.
4. **La vista de desarrollo.** La cual describe la organización estática del software en su ambiente de desarrollo.
5. **Escenarios.** Grupo de casos de uso. Por cada vista se define la serie de elementos a utilizar (componentes, contenedores y conectores), conectando la arquitectura de acuerdo con la vista de alguno de los requerimientos.

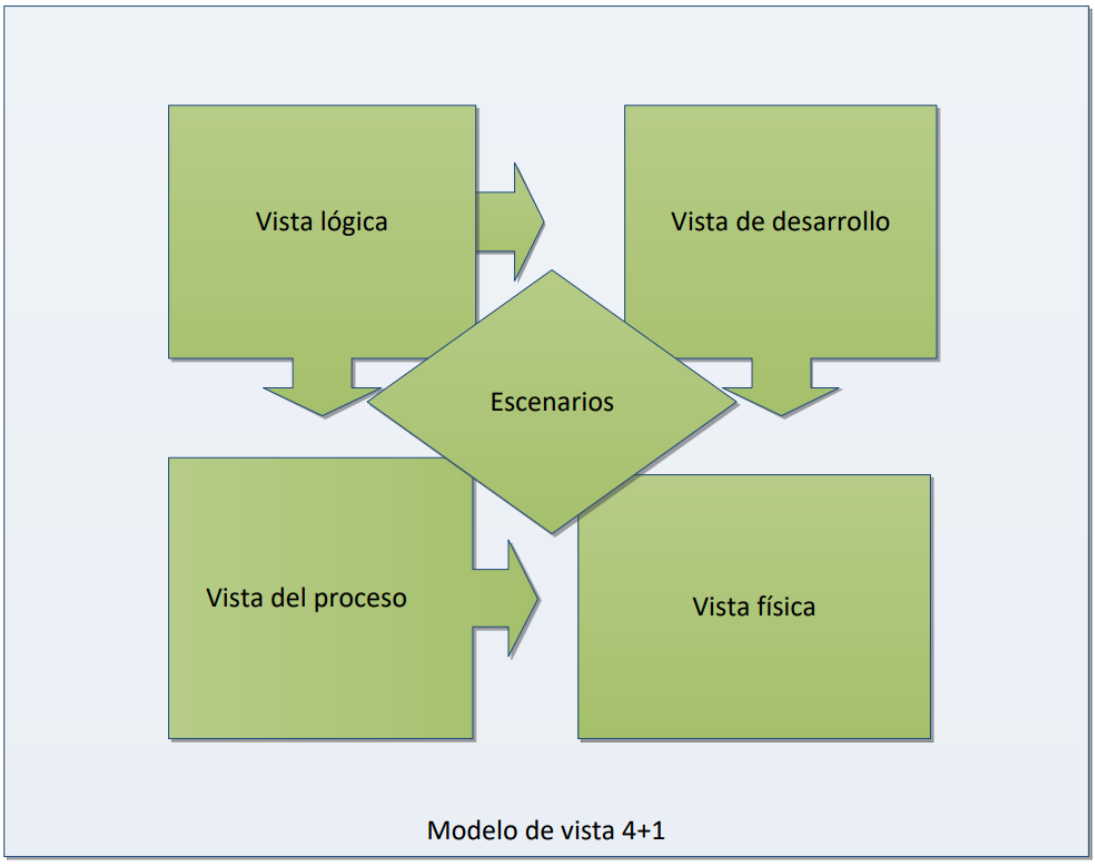

Figura 1. *Modelo de vistas de arquitectura 4+1*

2.2.1.1 Vista lógica

La arquitectura lógica principalmente soporta los requerimientos funcionales, lo que el sistema debe proveer en términos de servicios a sus usuarios. El sistema es descompuesto en una serie de abstracciones clave, tomadas del dominio del problema, en la forma de objetos o clases de objetos, ellos explotan los principios de la abstracción, encapsulación y herencia.

Uno de los objetivos del proyecto es contar con documentación que ayude al personal técnico del área de desarrollo; siendo importante que la documentación sea útil, por lo que en esta vista se utiliza el enfoque Rational/Booch para representar la arquitectura lógica, por medio de diagramas de clase y diagramas de secuencia que aporten valor en el entendimiento y construcción del sistema [4, 7].

2.2.1.1.1 Diagrama de clases

Las definiciones de las clases y métodos de clases fueron diseñadas en base a la documentación de requerimientos. El diagrama de clases se enfocó en ser ya una propuesta de solución para llevar a cabo la estandarización y reutilización de procesos para la lógica de negocio, se proponen paquetes basados en la nomenclatura sugerida, ver Figura 2.

Las definiciones de los métodos de clases fueron diseñadas con base en la documentación de requerimientos y propuesta de diseño sin acoplamiento, con el fin de asegurar la correcta integración de los distintos componentes.

El diseño propuso el desacoplamiento entre la lógica de lectura de un mensaje GPS y la funcionalidad posterior a ésta, con la finalidad de que al agregar un nuevo proveedor el impacto no se vea reflejado en la lógica existente.

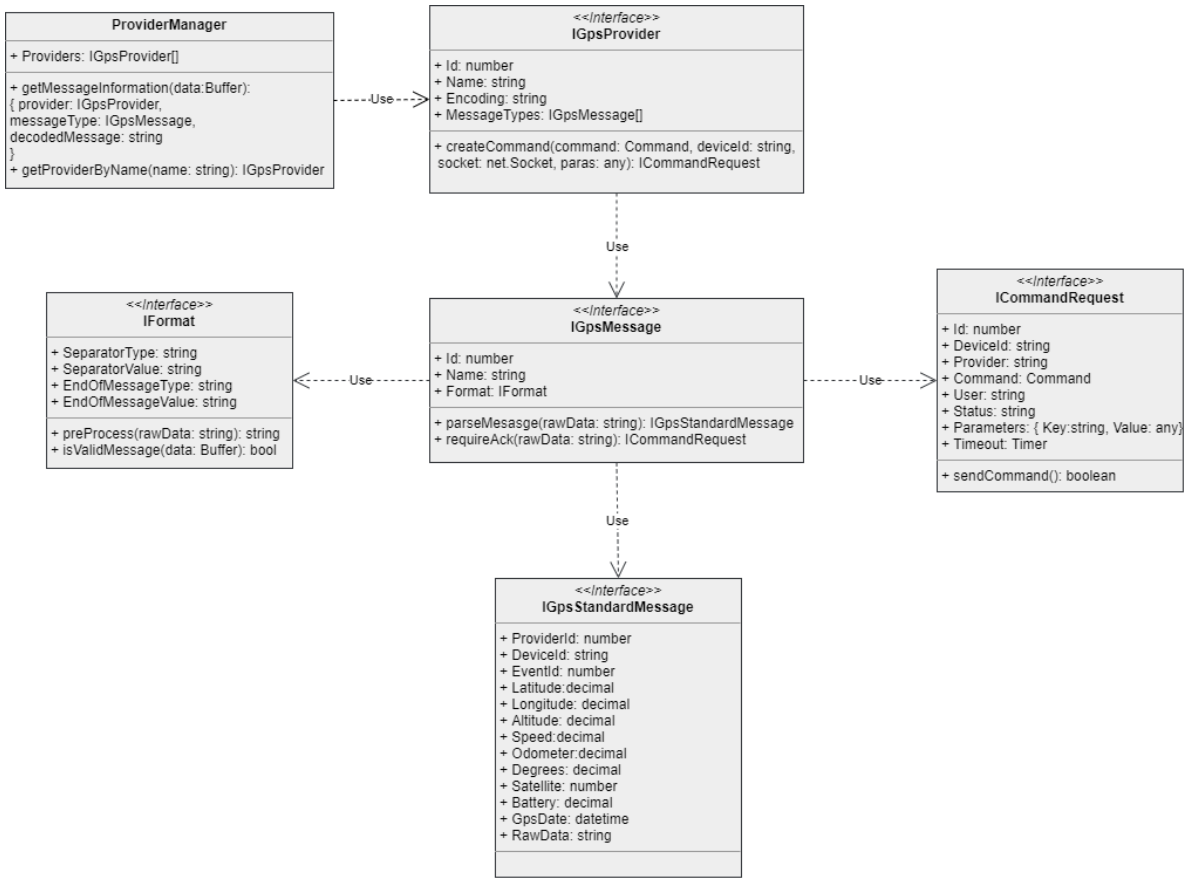

Figura 2. *Diagrama de clases*

Diagrama de secuencia

Los diagramas de secuencia fueron construidos con base a la documentación de requerimientos. En estos diagramas se muestra la interacción de objetos de la aplicación, muchos de estos definidos en el diagrama de clases, ver Figura 3 [4, 7].

2.2.1.2 Vista del proceso

En esta vista representamos los flujos de trabajo paso a paso de negocio y operacionales de los componentes que conforman el sistema. Los diagramas UML que representan la vista del proceso incluyen los diagramas de actividad que son los diagramas realizados para el presente proyecto [7, 8].

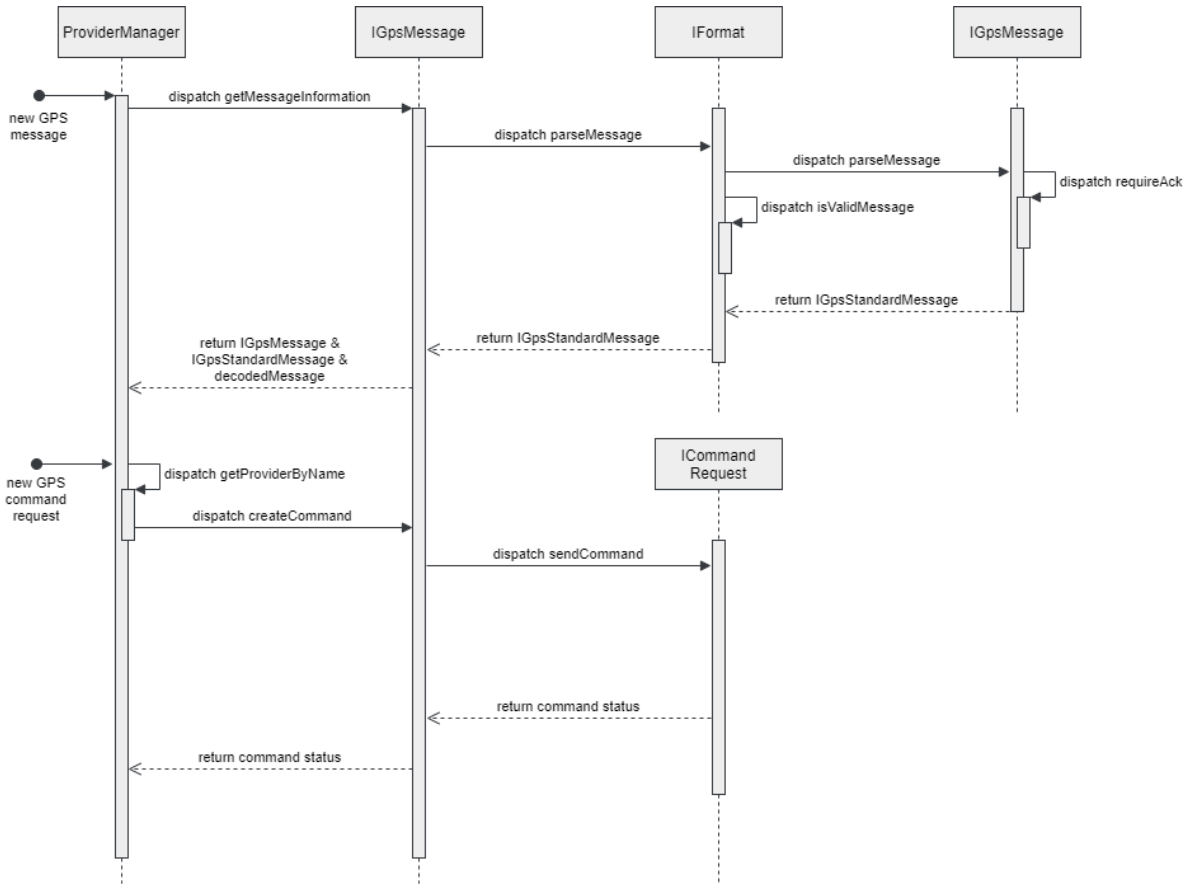

Figura 3. *Diagrama de secuencia*

Un proceso es un grupo de tareas que forman una unidad ejecutable, los cuales representan el nivel sobre el cual el proceso de arquitectura puede ser tácticamente

controlado (es decir, iniciado, recuperado, reconfigurado y apagado). El proceso de arquitectura puede ser descrito a diferentes niveles de abstracción, cada nivel enfocándose en diferentes asuntos. Sin que importe el nivel de abstracción del procedimiento, el diagrama de actividades UML se utiliza para representar detalles de éste. En el nivel de análisis, los diagramas de actividades deben usarse sólo donde la funcionalidad sea relativamente compleja, ver Figura 4 y Figura 5**Error! Reference source not found.** [8].

2.2.1.3 Vista de desarrollo

La arquitectura del desarrollo se enfoca en la organización de módulos de software en el mismo ambiente de desarrollo de software. El software es empacado en pequeños fragmentos, librerías de programas, o subsistemas que pueden ser desarrollados por uno o un conjunto de desarrolladores.

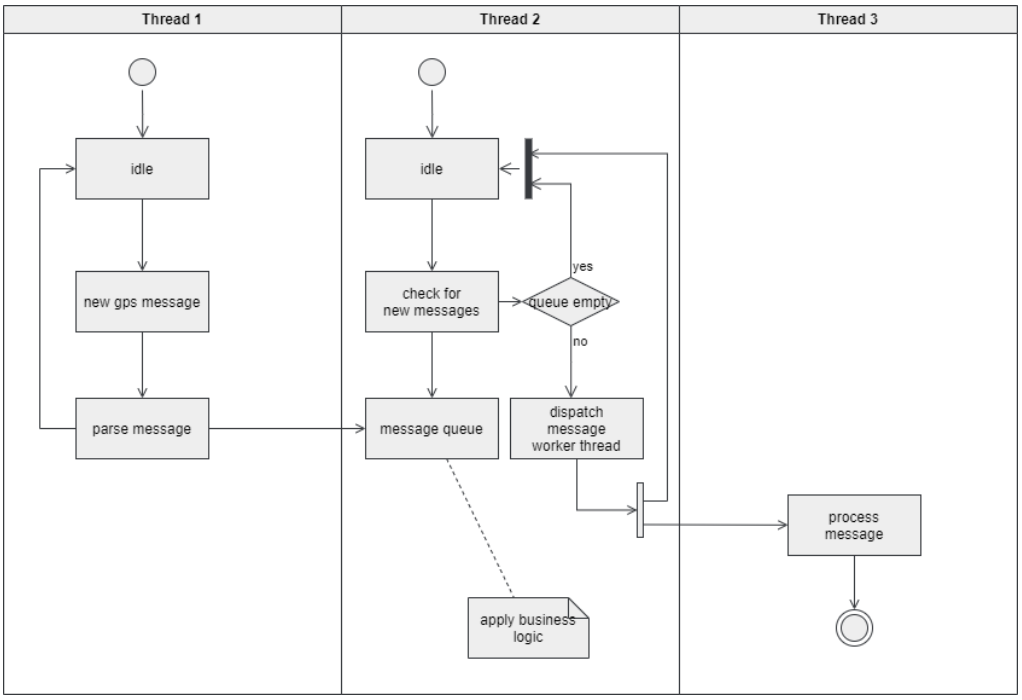

Figura 4. *Diagrama de actividad, nuevo mensaje de GPS*

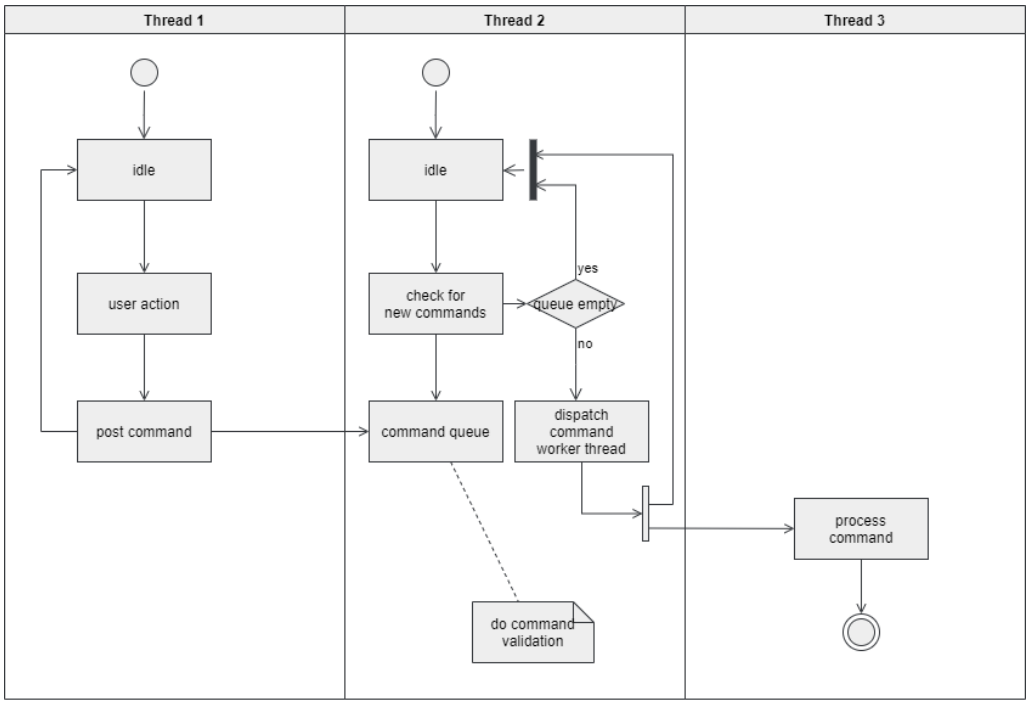

Figura 5. *Diagrama de actividad, envío de comandos*

La vista de desarrollo sirve como la base para el alojamiento de los requerimientos, para la distribución del trabajo a los equipos (o incluso para la organización del equipo), para evaluar el costo y planificación, para monitorear el progreso del proyecto, para razonar acerca de la reutilización del software, portabilidad y seguridad. Es lo básico para establecer una línea de producto.

Esta vista también es conocida como la vista de implementación. Los diagramas UML usados para representar la vista de desarrollo incluyen los diagramas de paquetes y componentes [5].

Diagrama de paquetes

Este diagrama de paquetes muestra la organización a nivel componentes de software que se obtuvo de los artefactos de desarrollo que se implementaron en la etapa de construcción, cabe mencionar que la nomenclatura especificada se encuentra

documentada en el apartado nomenclatura de nombrado de paquetes (sección 2.2.3.1.1 Estándares de programación).

Diagrama de componentes

Un componente representa una parte de un sistema modular, desplegable y reemplazable, que encapsula la implementación y expone un conjunto de interfaces, pueden ser un ejecutable o código fuente. Los elementos en UML para este fin son los diagramas de componentes (ver Figura 6) [4, 7].

2.2.1.4 Vista física

En esta vista se describió el diagrama de despliegue (ver Figura 7) que sirve para representar la distribución física de los componentes de software que conforman el sistema. Estos componentes son:

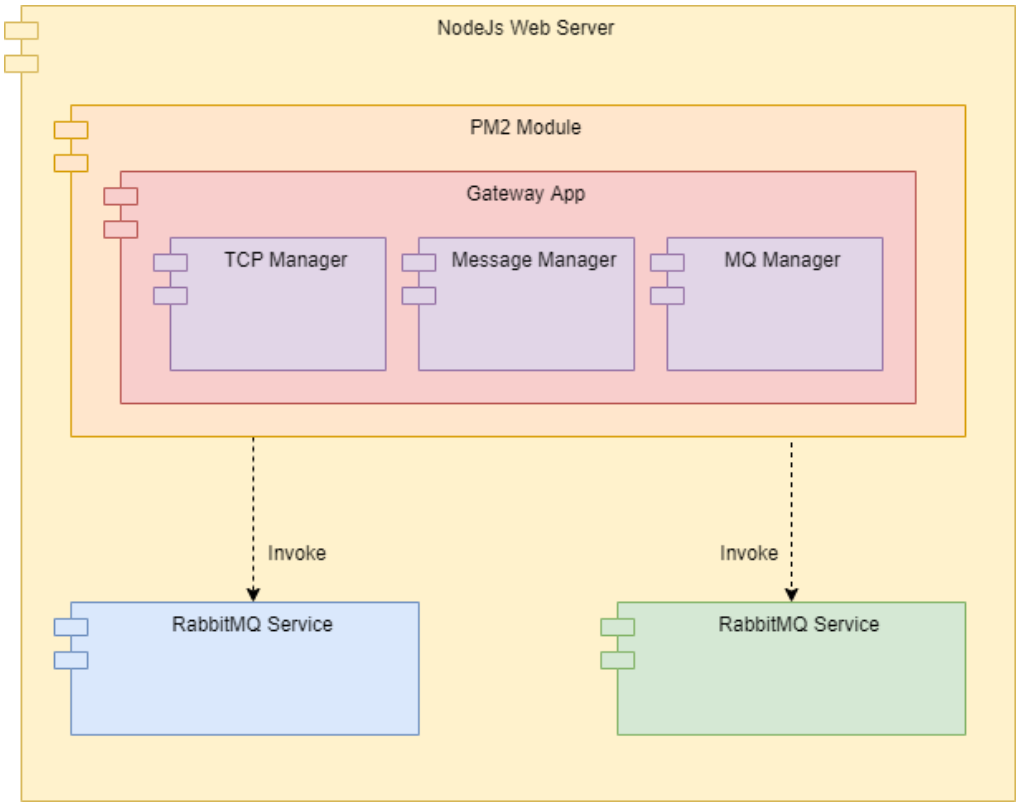

Figura 6. *Diagrama de componentes*

- **PM2.** Es un manejador de procesos para Node.js, PM2 o Process Manager 2, es una herramienta de código abierto, que ayuda a los desarrolladores y DevOps a gestionar aplicaciones en ambientes productivos para mantenerlos en línea 24/7 [9, 10].
- **RabbitMQ Server.** Es un software de negociación de mensajes de código abierto que funciona como un middleware de mensajería. Implementa el estándar Advanced Message Queuing Protocol (AMQP). El servidor RabbitMQ está escrito en Erlang y utiliza el framework Open Telecom Platform (OTP) para construir sus capacidades de ejecución distribuida y conmutación ante errores [12].
- **TCP/IP Server.** Es la identificación del grupo de protocolos de red que hacen posible la transferencia de datos en redes, entre equipos informáticos e internet. El modelo TCP/IP permite un intercambio de datos fiable dentro de una red, definiendo los pasos a seguir desde que se envían los datos (en paquetes) hasta que son recibidos [11].
- **Redis.** Es un motor de base de datos en memoria, basado en el almacenamiento en tablas de hashes (clave/valor) pero que opcionalmente puede ser usada como una base de datos durable o persistente. Otra funcionalidad de Redis es la comunicación entre procesos (Inter-Process Communication, IPC) [13].
- **Aplicación.** Es el aplicativo web desarrollado para cumplir con los requerimientos.
- **Mocha.** Mocha es un marco de pruebas de JavaScript para los programas Node.js, que ofrece soporte de navegador, pruebas asincrónicas, informes de cobertura de prueba y el uso de cualquier biblioteca de aserciones [14].
- **Chai.** Chai es una biblioteca de aserciones para el Node.js y el navegador que se puede combinar de manera transparente con cualquier marco de pruebas de JavaScript [15].

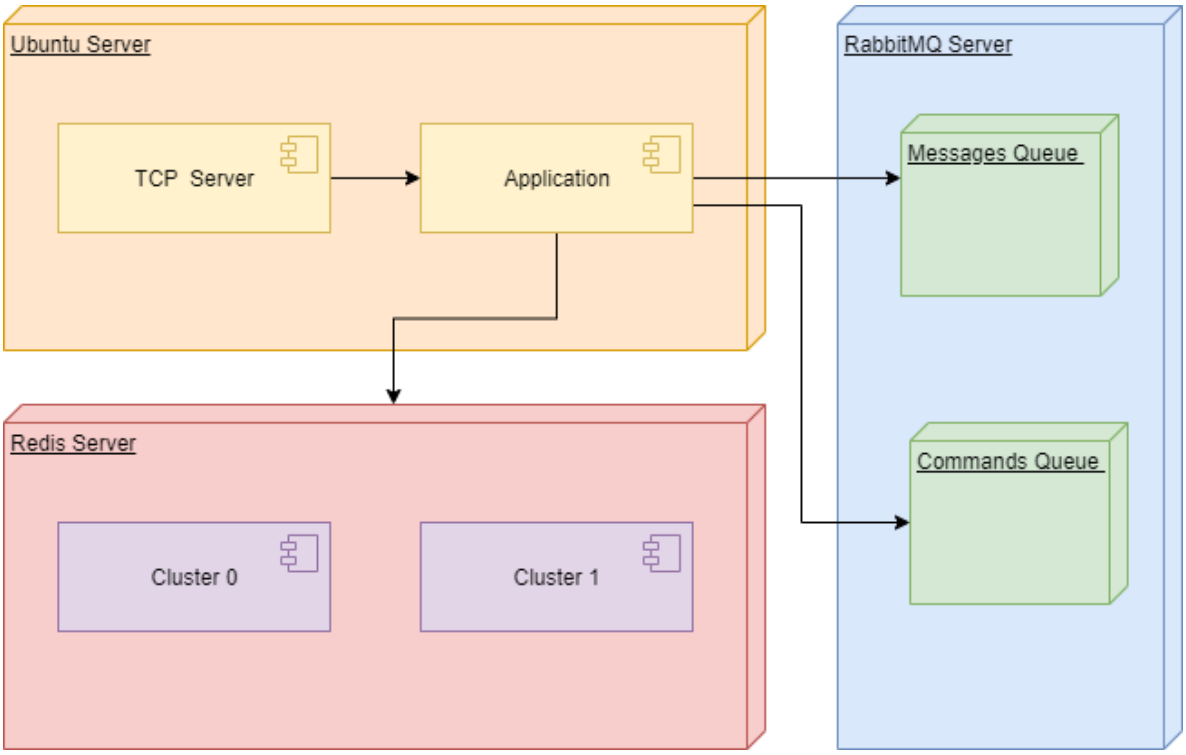

Figura 7. *Diagrama de despliegue*

2.2.1.5 Escenarios

Los escenarios describen secuencias de interacciones entre objetos y entre procesos, esta vista también es conocida como vista de casos de uso y en UML esta vista es representada por los diagramas de casos de uso.

2.2.2 Análisis y diseño de usabilidad

Para este tipo de aplicación, al ser un sistema que sólo tiene comunicación entre dispositivos GPS y los deposita en una cola, donde un proceso posterior los tome y realice los cálculos pertinentes, no fue necesario crear una interfaz gráfica.

2.2.3 Estándares y tecnologías

2.2.3.1 Estándares

Los estándares de codificación son importantes en el desarrollo de software para asegurar la buena calidad, legibilidad, integración, confiabilidad, seguridad y

mantenimiento de código. Con el objetivo de cumplir con lo mencionado para el presente proyecto fueron definidos los estándares sobre los cuales se realiza la construcción del sistema.

Estándares de programación

Considerando que el framework sobre el cuál se construyó el sistema es Node.js, los estándares de codificación definidos fueron los que se establecen en el documento *ECMAScript® 2020 Language Specification,* publicados en el sitio de ECMA International [16, 17]. Por otro lado, se consideró usar TypeScript para un mejor control del proyecto debido a su definición de tipados, ya que es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases. Además, extiende la sintaxis de JavaScript, por tanto, cualquier código JavaScript existente funciona sin problemas. Está pensado para grandes proyectos, los cuales a través de un compilador de TypeScript se traducen a código JavaScript original [18].

Especificaciones de seguridad

En temas de seguridad la empresa se rige bajo los estándares que dicta PCI DSS y cuenta actualmente con una certificación de PCI; por lo que el sistema tuvo que cumplir con los lineamientos de seguridad que PCI marca. Tratándose de una aplicación web los estándares de seguridad considerados fueron los que se estipulan en la lista del top ten de la página de OWASP. Este top ten representa un amplio consenso acerca de los problemas más críticos en cuanto a seguridad de las aplicaciones web, los miembros del proyecto incluyen una variedad de expertos en seguridad de todo el mundo, quienes han compartido sus conocimientos y habilidades para producir dicha lista [19, 20].

2.2.3.2 Tecnologías

Esta sección muestra la arquitectura de los elementos de software que fueron definidos para la construcción y ejecución del sistema, mencionando para cada uno de estos elementos su objetivo, importancia e interacción con los otros elementos.

Estos elementos fueron definidos en base a la tecnología que es utilizada actualmente por la empresa.

- **Node.js.** Contrasta con el modelo de concurrencia más común de hoy en día, en el que se emplean hilos del Sistema Operativo. Las redes basadas en hilos son relativamente ineficientes y muy difíciles de usar. Además, los usuarios de Node.js están libres de preocuparse por el bloqueo del proceso, ya que no existe. Casi ninguna función en Node.js realiza I/O directamente, por lo que el proceso nunca se bloquea. Por ello, es muy propicio desarrollar sistemas escalables en Node.js (Tabla 8).

Tabla 8. *Node.js*

Plataforma	**Versión**	**Interacción con otros componentes**
Linux 64bits	Ubuntu 18.10	PM2, TCP/IP Server

- **PM2.** Es un administrador de procesos de producción para aplicaciones Node.js con un balanceador de carga incorporado. Le permite mantener las aplicaciones activas para siempre, recargarlas sin tiempo de inactividad y facilitar las tareas comunes de administración del sistema (Tabla 9).

Tabla 9. *PM2*

Plataforma	**Versión**	**Interacción con otros componentes**
Linux 64bits	PM2 4.5.6	Node.js, TCP/IP Server

- **RabbitMQ Server.** RabbitMQ es liviano y fácil de implementar en las instalaciones y en la nube. Admite múltiples protocolos de mensajería. RabbitMQ se puede implementar en configuraciones distribuidas y federadas para cumplir con los requisitos de alta disponibilidad y gran escala (Tabla 10).

Tabla 10. *RabbitMQ Server*

Plataforma	**Versión**	**Interacción con otros componentes**

Linux 64bits	RabbitMQ 3.8.16	Node.js, PM2

- **TCP/IP Server.** El módulo de red proporciona una API de red asíncrona para crear servidores TCP o IPC basados en flujo (Tabla 11).

Tabla 11. *TCP/IP Server*

Plataforma	**Versión**	**Interacción con otros componentes**
Linux 64bits	Net 16.2.0	Node.js, PM2

- **Redis.** Redis es un almacén de estructura de datos en memoria de código abierto, que se utiliza como base de datos, caché y agente de mensajes. Redis proporciona estructuras de datos como cadenas, hashes, listas, conjuntos, conjuntos ordenados con consultas de rango, mapas de bits, índices geoespaciales y flujos (Tabla 12).

Tabla 12. *Redis*

Plataforma	**Versión**	**Interacción con otros componentes**
Linux 64bits	Redis 6.0.6	Node.js, PM2

- **Aplicación.** Es el aplicativo web desarrollado para cumplir con los requerimientos del sistema y de cada módulo que lo componen. Está basada en Java y J2EE siguiendo buenas prácticas de programación y calidad del código; compuesta de una serie de frameworks de desarrollo que le permiten ser una aplicación robusta, extensible y mantenible (Tabla 13).

Tabla 13. *Aplicación*

Plataforma	**Versión**	**Interacción con otros componentes**
Linux 64bits	NA	Visual Studio Code

- **Visual Studio Code.** Visual Studio Code es un editor de código redefinido y optimizado para crear y depurar aplicaciones web y en la nube modernas (Tabla 14).

Tabla 14. *Visual Studio Code*

Plataforma	**Versión**	**Interacción con otros componentes**
Windows 10 64bits	VS Code 1.56.2	Aplicación

- **Git.** Git es un sistema de control de versiones distribuido de código abierto y gratuito diseñado para manejar todo, desde proyectos pequeños a muy grandes con velocidad y eficiencia (Tabla 15).

Tabla 15. *Git*

Plataforma	**Versión**	**Interacción con otros componentes**
Windows 10 64bits	Git 2.31.1	Visual Studio Code

2.2.3.3 Modelado de mensaje GPS

Para el control, envío y recepción de paquetes y comandos se crearon dos tipos de mensaje, donde se incluyeron sólo las propiedades necesarias ambos en formato JavaScript Object Notation (JSON). JSON es un formato ligero de intercambio de datos. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - diciembre 1999. Está constituido por dos estructuras [21]:

- **Una colección de pares de nombre/valor.** En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- **Una lista ordenada de valores.** En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

A continuación, se muestran ambos mensajes donde se describe el uso de cada propiedad:

- **Mensaje Estándar de ubicación.** Mensaje utilizado para extraer las propiedades del GPS de paquete de ubicación, estos mensajes son enviados por los dispositivos cada 12 segundos y son utilizados para realizar más cálculos una vez se depositen en la cola de mensajes de RabbitMQ (ver Figura 8).

- **Mensaje Estándar de comando.** Mensaje utilizado para el envío de comandos hacia un GPS, como obtener ubicación, paro de motor, activar motor, encender luces, apagar luces, entre otros. Los comandos son enviados desde una plataforma web, los cuáles se insertan en una cola de RabbitMQ y la aplicación los lee, los interpreta y posteriormente realiza el envío al GPS (ver Figura 10).

```
export interface IGpsStandardMessage {
    ProviderId?: number;
    DeviceId?: string;
    EventId?: string;
    Latitude?: string;
    Longitude?: string;
    Altitude?: string;
    Speed?: string;
    KmOdometer?: string;
    Degrees?: string;
    Satellite?: string;
    Temperature?: string;
    Battery?: string;
    SignalValid?: string;
    GpsDate?: string;
    RawData?: string;
}
```

Figura 8. *Estructura de mensaje de ubicación*

Las propiedades del mensaje de ubicación se describen quedando de la forma siguiente:

- **ProviderId.** ID del fabricante del GPS, dependiendo del fabricante se obtiene el ID y se selecciona.
- **DeviceId.** ID del GPS que está enviando el mensaje, cada dispositivo y fabricante tiene su estándar.
- **EventId.** ID del evento que envía el proveedor, cada fabricante tiene un tipo de trama donde puede enviar un número o una cadena indicando que ese mensaje es referente a una ubicación, una alerta como botón de pánico presionado, puerta de transporte abierta, entre otras.
- **Latitude.** Latitud de la ubicación del GPS donde se encuentra reportando.
- **Longitude.** Longitud de la ubicación del GPS donde se encuentra reportando.
- **Altitude.** Altitud de la ubicación del GPS donde se encuentra reportando.
- **Speed.** Velocidad de la ubicación del GPS donde se encuentra reportando.
- **KmOdometer.** Kilometraje recorrido desde el día 1 en la instalación del GPS y la fecha actual.
- **Degrees.** Orientación del GPS en grados, de 0 a 359 grados.
- **Satellite.** Número de satélites a los que está conectado el GPS para generar la ubicación.
- **Temperature.** Temperatura actual de la ubicación del GPS donde se encuentra reportando.
- **Battery.** Si es un GPS portátil indica el nivel de batería con el que cuenta el GPS, en caso de un GPS fijo indica el voltaje que recibe directamente del transporte donde se encuentra instalado.
- **SignalValid.** Bandera que ayuda a identificar si el mensaje que se está recibiendo está surgiendo de la memoria del GPS o bien es un mensaje en tiempo real.
- **GpsDate.** Fecha en UTC del dispositivo GPS en la cual se realizó el mensaje enviado.

- **RawData.** Propiedad donde se almacena el mensaje original del GPS para posterior revisión o auditoría.

Un ejemplo de mensaje de ubicación se muestra en la Figura 9.


```
{
    "RawData": "4d43475000308b0c0008010101180400000200f00000000005e75000000
0000000000000000b18000402047478b3f52800fa01e8030000000000000000220b000306e5
07e1",
    "EventId": 32,
    "DeviceId": 822064,
    "Latitude": 19.00000012152878,
    "Longitude": -99.00000003011544,
    "Altitude": 10,
    "Speed": 0,
    "KmOdometer": 0,
    "Satellite": 4,
    "Battery": 2.06,
    "Degrees": 0,
    "GpsDate": "2021-06-03T00:11:34.000Z",
    "ProviderId": 0
}
```


Figura 9. *Ejemplo de mensaje estándar de ubicación de GPS*

```
export interface ICommandRequest {
    Id: number;
    DeviceId: string;
    Provider: string;
    Command: Command;
    User: string;
    Status: string;
    Parameters: { Key: string; Value: any }[];
    Timeout: any;
}
```

Figura 10. *Estructura de mensaje de comando*

Las propiedades del mensaje de comando se describen a continuación:

- **Id.** Identificador único del comando, esta propiedad está vacía en un principio puesto que el sistema es quien asigna dicho ID para un mejor seguimiento en el paso de mensajes.

- **DeviceId.** ID del GPS al que se estará enviando el mensaje de comando, cada dispositivo y fabricante tiene su estándar.
- **Provider.** Nombre del proveedor al que pertenece el GPS con la finalidad de que el sistema conozca qué tipo de mensajería soporta el GPS.
- **Command.** Tipo de comando a enviar, este valor se toma de una Enumeración de comandos que se omite por seguridad del proyecto.
- **User.** Nombre de usuario que está enviando el comando para guardar auditoría.
- **Status.** Estatus del mensaje de comando, "Waiting", "Sending", "Sent", "Received", "Executed", "Not Executed" y "Unknown", esta propiedad se usa para saber en qué parte del proceso se encuentra el comando y poder notificar a los involucrados.
- **Parameters.** Dependiendo del tipo de comando puede que se requieran parámetros para enviar dicho comando, como puede ser el modelo específico del GPS de un proveedor, el valor asignar dependiendo de la trama y mensajería de este, entre otros.
- **Timeout.** Temporizador que se utiliza para poner un tiempo máximo de respuesta por parte del GPS, si en este tiempo el GPS no responde que recibe el comando el estatus de dicho comando pasa a "Not Executed"

Un ejemplo de mensaje de comando se muestra en la Figura 11, algunos datos se omiten por seguridad:

```
{
    "Id": "[Id]",
    "DeviceId": "0580000014",
    "Provider": "[Provider]",
    "Command": 4,
    "User": "[User]",
    "Status": "Waiting",
    "Parameters": {},
    "Timeout": "[Timeout]"
}
```

Figura 11. *Ejemplo de mensaje estándar de comando*

Como se puede observar de esta forma los mensajes de ubicación, eventos y comandos fueron estandarizados y depositados en una cola de RabbitMQ donde el resto de los sistemas ya pueden emplear la lógica de negocio que mejor convenga para cada tipo de mensaje, ofreciendo un rendimiento óptimo para este proceso y brindando la posibilidad de registrar más tipos de mensajes y de distintos proveedores.

2.3 Construcción

Esta etapa tuvo como objetivo la generación de código del Sistema, dicha generación se realizó en base a los entregables de las etapas de Definición de Requerimientos y Análisis y Diseño.

Definición de Requerimientos:

- Documentos de Especificación de Requerimientos.
- Documentos de Casos de Uso.
- Documentos del Diccionario de Datos.

Análisis y Diseño:

- Documento de usabilidad.
- Diseño de software.
- Estándares de programación.
- Tecnologías
- Modelado de mensaje GPS

Los diferentes elementos de código construidos para el Sistema quedaron conformados por los siguientes:

- Aplicación web.
- Servicios web.
- Proyecto Node.js de comunicación con RabbitMQ.
- Servicio de Redis para la comunicación entre múltiples procesos de Node.js.

Los diferentes elementos de software construidos son, en su conjunto, una aplicación que puede operar para dar servicio a un volumen alto de dispositivos GPS con base en la configuración de un conjunto amplio de parámetros como fue el objetivo inicial, donde hoy operan más de 8000 dispositivos enviando mensajes cada 12 segundos para GPS portátiles y 4 segundos para GPS fijos.

Como buena práctica se validó la limitante de que un puerto TCP / IP puede atender máximo 2500 conexiones, por lo que actualmente se tienen 4 procesos Node.js en un puerto diferente atendiendo en promedio este número de conexiones aunado a que la operación de los GPS es de 24 horas al día, todos los días del año.

Capítulo III. Pruebas en la solución

En este capítulo se documentan el tipo de pruebas que fueron realizadas al Sistema, la implementación y puesta a punto.

3.1 Pruebas

Las pruebas del software son una función del control de calidad que tiene como objetivo principal detectar errores; siendo la principal tarea del aseguramiento de la calidad de software garantizar que las pruebas se planeen en forma apropiada y que se realicen con eficiencia, de modo que la probabilidad de que logren su principal objetivo sea máxima [8].

Para garantizar la calidad de software del presente proyecto se realizó una planeación de pruebas con el objetivo de definir y detallar cada una de las actividades a realizar

para llevar a cabo el aseguramiento y control de las pruebas y con el fin de cumplir los criterios de aceptación de la empresa, así como cumplir con los requerimientos funcionales y no funcionales definidos.

Este plan de calidad y pruebas estableció las directrices a llevar en el proyecto describiendo de forma clara todas las pruebas que se ejecutaron a lo largo del proyecto para asegurar que el funcionamiento y el comportamiento son los correctos y satisfacen todas las necesidades y requerimientos.

Los diferentes tipos de prueba que se realizaron al proyecto son:

- Pruebas Unitarias
- Pruebas de Integración de Componentes (CIT siglas en inglés)
- Pruebas de Integración de Sistema (SIT siglas en inglés)
- Pruebas de Rendimiento
- Pruebas de Seguridad.

3.1.1 Pruebas unitarias

Permiten verificar la funcionalidad y estructura de cada componente desarrollado, individualmente del sistema.

El alcance de estas pruebas fue para el desarrollo de la aplicación web; por lo que todo el código desarrollado está enfocado a pruebas mediante el uso de los frameworks Mocha y Chai. Estas pruebas fueron ejecutadas y validadas por el desarrollador al momento de completar la construcción de cada componente [14, 15].

Dentro de la arquitectura de elementos de software (en ambiente de desarrollo) se usó Visual Studio Code, para la ejecución y validación de las pruebas unitarias de forma automática y garantizar que los componentes funcionaran de forma correcta. Uno de los objetivos de las pruebas unitarias fue el tener la certeza de que todo el código esté probado, por lo que a través de estas pruebas y la herramienta de Jira se

pudo obtener un Reporte de Cobertura de Pruebas o Test Coverage Report ver Figura 12, obteniendo un resultado de 92% de código probado de manera aislada, donde el 8% restante se debe a que son escenarios de pruebas donde se requiere la interacción con los usuarios como puede ser el envío de comandos y tiempos de espera al recibir el estatus de ejecución del comando.

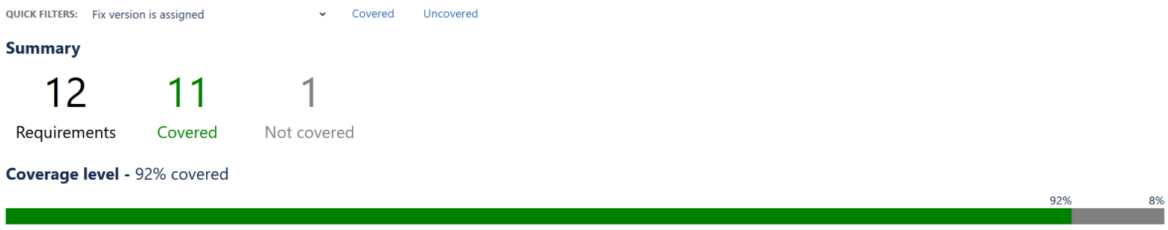

Figura 12. *Test coverage report*

3.1.2 Pruebas CIT, SIT, de Rendimiento y UAT

Mediante estos tipos de prueba se validó uno de los principales objetivos del Sistema, el ser una aplicación altamente configurable y parametrizable, además de contar con un rendimiento excesivamente alto y que a través de esta parametrización y configuración se permitiera el manejo de diferentes proveedores y GPS simultáneos con información particular para cada uno de ellos y con un gran número de GPS conectados.

3.1.2.1 Pruebas CIT

Estas pruebas fueron realizadas por el área de calidad y mediante estas pruebas se verificó el correcto ensamblaje entre los distintos elementos o componentes de software que componen el sistema desarrollado: Aplicación web, Implementación de Redis para el envío de comandos y comunicación interprocesos, Colas de RabbitMQ y Sistemas relacionados con el fin de garantizar que su operación integrada fuera correcta.

Dentro de estas pruebas se prestó especial atención en el cumplimiento al correcto funcionamiento de los componentes mencionados en el párrafo anterior cuando se

tienen diferentes GPS operando y enviando mensajería de manera simultánea; el arreglo probado se conformó como se indica a continuación:

- Una instancia, un proceso para recibir conexiones de GPS a un sólo puerto.
- Una instancia, dos procesos para recibir conexiones de GPS para dos puertos distintos.
- Una instancia, tres procesos para recibir conexiones de GPS para tres puertos distintos.

3.1.2.2 Pruebas SIT

Estas pruebas fueron realizadas por el área de calidad y mediante estas se verificó que el sistema cumple con la funcionalidad, necesidades y con los requerimientos definidos por la empresa. Se verificó el correcto funcionamiento del sistema completo incluyendo casos de prueba que buscaron las fallas del sistema.

3.1.2.3 Pruebas de Rendimiento

Mediante estas pruebas se validó el correcto comportamiento del sistema cuando existe una carga concurrente y procesa un volumen alto de datos, estas pruebas incluyeron las pruebas de Carga y de Estrés.

- **Pruebas de Estrés**: a través de estas pruebas se verificó que el sistema funcionara apropiadamente y sin errores, bajo condiciones de un número máximo de GPS conectados y múltiples proveedores desempeñando diferentes transacciones.
- **Pruebas de Volumen**: mediante éstas se verificó el tiempo de respuesta del sistema para procesar mensajes de distintos GPS simulados, bajo diferentes condiciones de carga, enfocándose a la extracción y carga de información y de forma concurrente de un número máximo de GPS conectados y múltiples proveedores. Para la realización de estas pruebas se utilizó la herramienta GSM GPRS Unit Simulator del proveedor de GPS llamado Cellocator [25]. Esta

herramienta brinda la posibilidad de simular hasta 10,000 GPS, dependiendo de las características de la máquina donde corre, de tal forma que fue útil hacer pruebas con 10,000 GPS ya que se pudo observar que un puerto TCP, por la forma en que se estaba manejando la mensajería, soportaba hasta 2,500 GPS simultáneos, de tal forma que se estipularon 2,000 a 2,500 GPS por puerto, actualmente corren 4 procesos en el entorno de producción soportando hasta 10,000 GPS, sin embargo se tienen únicamente alrededor de 6,000 GPS, ocupando un 60% de la capacidad y con la ventaja de poder agregar tantos procesos sean necesarios para agregar más puertos y abastecer un crecimiento que se pueda dar conforme avance el tiempo.

3.1.2.4 Pruebas UAT

Una vez que las pruebas de CIT, SIT y de Rendimiento concluyeron de forma exitosa se realizaron las pruebas de aceptación de usuario. En esta fase se realizó la validación por el área final/experta del negocio con la finalidad de obtener la confirmación en el cumplimiento de las necesidades establecidas por la empresa y/o especificación de requerimientos correspondiente. El objetivo fue emular las condiciones de uso y comportamiento del producto en condiciones reales, las pruebas realizadas no se centraron en la identificación de problemas simples o estéticos, sino en poner a prueba la operación y estabilidad del producto.

Para el seguimiento de estas pruebas, así como obtener la aceptación por parte del usuario final y del negocio se realizó un documento de matriz de pruebas donde el usuario debía llenar datos como Fecha de prueba, Resultado obtenido y en dado caso de encontrar algún inconveniente, Hallazgos encontrados de tal forma que se obtuviera un estatus para cada una de las pruebas ejecutadas así como su visto bueno para liberación al ambiente de producción. En la Tabla 16 se puede observar el tipo de formato utilizado para llevar a cabo el proceso descrito.

Tabla 16. *Matriz de prueba caso CP-G-027*

Caso de prueba	CP-G-027 - Envío de comando "Paro de

	motor” a GPS
Usuarios asignados	Monitorista y área operativa
Ambiente de pruebas	UAT
Funcionalidad a probar	Envío, recepción y ejecución del comando “Paro de motor”
Resultado esperado	El usuario envía la solicitud de paro de motor y el vehículo se detendrá de manera segura reduciendo la velocidad del mismo.
Pasos a seguir	
1. Mantener comunicación continua con usuario operativo. 2. Ingresar a plataforma con usuario y contraseña proporcionados. 3. Buscar y seleccionar el GPS “Test Device G05”. 4. Ir a la barra de comandos, seleccionar el comando de “Paro de motor”. 5. Dar clic en “Enviar comando”. 6. Acceder credenciales de supervisor proporcionadas. 7. La plataforma deberá mostrar un mensaje de “Comando ejecutado de manera exitosa”. 8. Comprobar con usuario operativo que el transporte inicie proceso de detención seguro. 9. Validar que el transporte no pueda encenderse una vez termine proceso de detención seguro.	
Resultado obtenido	En espera | En ejecución | Satisfactorio | con Hallazgos
Fecha de prueba	[Ingresar fecha de prueba.]
Hallazgos encontrados	[Listar los hallazgos encontrados durante la realización de la prueba.]

3.1.3 Pruebas de seguridad

La seguridad del software es una actividad del aseguramiento del software que se centra en la identificación y evaluación de los peligros potenciales que podrían afectarlo negativamente y que podrían ocasionar que falle todo el sistema. Si los peligros se identifican al principio del proceso del software, las características de su diseño se especifican de modo que los eliminen o controlen [8].

Mediante las pruebas de seguridad es posible detectar el nivel de seguridad interna y externa de los sistemas de información de una empresa, determinando el grado de acceso que tendría un ataque [8].

Para el Sistema se realizaron pruebas de penetración externas para evaluar los controles de seguridad y obtener posibles huecos y vulnerabilidades que pudieran afectar la operación y tener un impacto en la seguridad de la información para la confidencialidad, integridad y disponibilidad en la aplicación web [19].

Se realizó la evaluación de la seguridad de la aplicación web mediante herramientas especializadas de penetración y con procedimientos manuales que permitieron evidenciar algunos huecos de seguridad existentes.

3.1.3.1 Metodología utilizada

Se utilizó como referencia para el ataque controlado la metodología definida en el Open Source Security Testing Methodology Manual (OSSTMM) en su versión 3.0, del Institute for Security and Open Methodologies (ISECOM). Dicha metodología dicta los lineamientos para la clasificación de las pruebas, definición de los vectores de ataque y secuencia de actividades, con el objetivo de asegurar que los resultados sean consistentes y provean la mayor información para realizar las acciones de seguimiento ver **Error! Reference source not found.** [23].

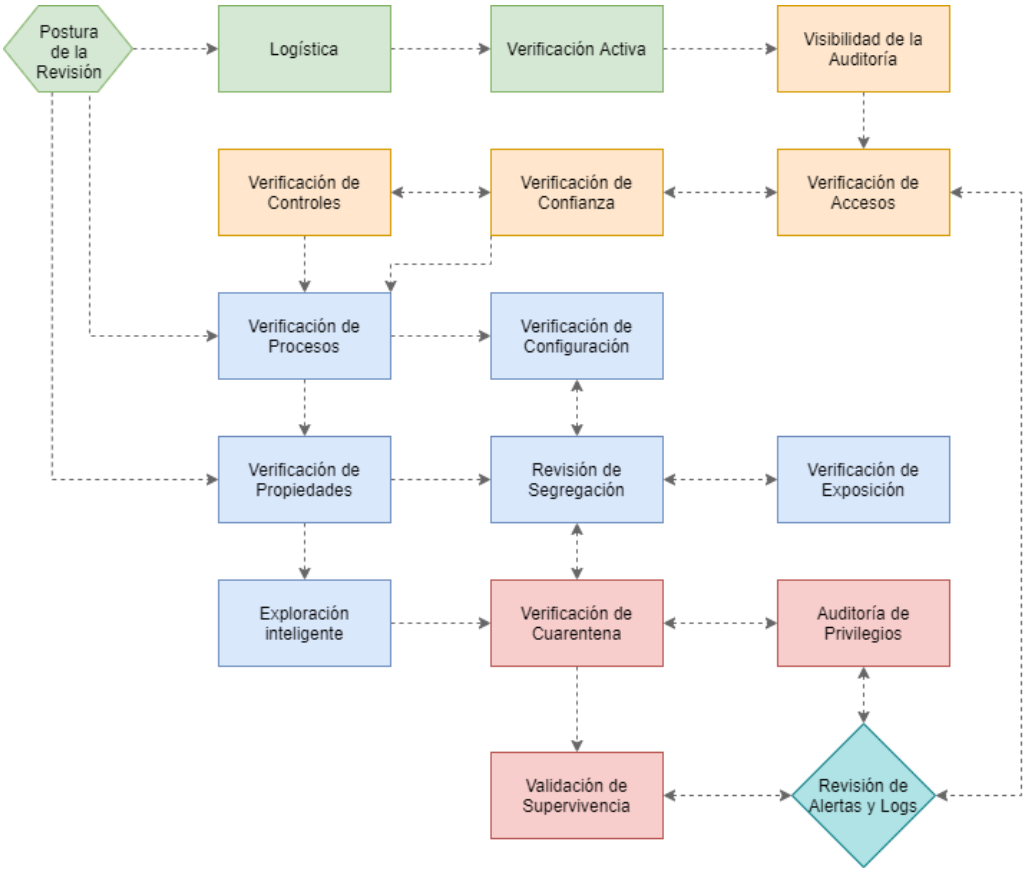

Figura 13. *Diagrama de metodología OSSTMM*

3.1.3.2 Actividades realizadas

Para las pruebas se realizaron las siguientes actividades:

- Reconocimiento e identificación de los activos.
- Enumeración de servicios y puertos abiertos.
- Valoración e identificación de vulnerabilidades.
- Verificación de vulnerabilidades.
- Pruebas de Penetración.
    - Aplicaciones Web: las definidas por el TOP 10 de OWASP que incluyen:
        - A1 – Inyección.
        - A2 – Pérdida de Autenticación y Gestión de Sesiones.
        - A3 – Secuencia de Comandos en Sitios Cruzados (XSS).
        - A4 – Referencia Directa Insegura a Objetos.
        - A5 – Configuración de Seguridad Incorrecta.

        - A6 – Exposición de Datos Sensibles.
        - A7 – Ausencia de Control de Acceso a las Funciones.
        - A8 – Falsificación de Peticiones en Sitios Cruzados (CSRF).
        - A9 – Uso de Componentes con Vulnerabilidades Conocidas.
        - A10 – Redirecciones y reenvíos no validados.
    - Revisión en la capa de Sistema Operativo y en la de Aplicativo para cada servicio.
        - Brechas de seguridad por tipo de servicio.
        - Criticidad de cada vulnerabilidad encontrada.
    - Verificación del nivel de exposición por cada vulnerabilidad explotada.

3.1.3.3 Herramientas utilizadas

Para la realización de las pruebas de seguridad se utilizaron herramientas especializadas listadas a continuación:

- Análisis de Vulnerabilidades:
    - Nexpose
    - Nmap
    - OpenVAS
    - Acunetix
    - W3AF
- Pruebas de Penetración:
    - Metasploit
    - Armitage
    - Kali 2.0
    - Parrot Security

3.1.3.4 Reporte técnico

Como resultado de las pruebas se generó un reporte técnico de resultados con todos los hallazgos encontrados durante la ejecución de las pruebas, así como todos los

huecos de seguridad encontrados y su respectivo procedimiento de remediación sugerido.

Las vulnerabilidades encontradas fueron:

- **Denial of Service Attack (DoS).** También conocido como ataque de denegación de servicio, es un ataque que causa que un servicio o recurso sea inaccesible a los distintos usuarios.

  Solución: Al ser una aplicación TCP, el sistema expone puertos donde los GPS pueden conectarse y enviar su mensajería, sin embargo, cualquier cliente TCP que conozca los puertos puede realizar una conexión y crear "conexiones tontas" por lo que para solucionar este punto se agregó una validación para identificar el tipo de mensajería que está enviando, si la mensajería no corresponde a una trama conocida de un GPS / proveedor la conexión se rechaza y no permite mantener la conexión activa. Dando espacio para más conexiones de GPS que sí deban enviar mensajería.

- **TCP Syn Flood.** Este ataque es un tipo de Denegación de Servicio que consiste en mantener una conexión activa y tratar de consumir recursos en el servidor objetivo, de igual forma envía peticiones de manera simultánea y rápida de tal forma que el servidor no pueda responder a todas provocando una saturación de red.

  Solución: Para esta vulnerabilidad se agregó un temporizador a manera de timeout, de tal forma que, si un cliente conectado al servicio TCP no envía peticiones durante 3 minutos, la conexión se cierra y espera a que se realice una nueva conexión y envíe mensajería adecuada.

Del reporte se pudo determinar que no se encontraron más vulnerabilidades, concluyendo que la aplicación es altamente segura; además de que las

vulnerabilidades encontradas no requirieron un esfuerzo mayor y fueron de fácil remediación.

3.2 Implantación

Esta fase representa el último eslabón en la metodología utilizada para desarrollar el sistema, en el presente proyecto esta etapa consistió en la instalación del producto en un ambiente de producción y puesta a punto del Sistema.

3.2.1 Instalación

Esta actividad consistió básicamente en la instalación de los elementos de software que conforman al producto desarrollado en la infraestructura de un ambiente productivo y validar que todos los elementos instalados funcionen de forma correcta.

Por motivos de seguridad de la infraestructura, la instalación del Sistema en ambiente productivo fue realizada por el equipo de Soporte Técnico de la empresa y fue llevada a cabo mediante la documentación técnica que se generó para dicho fin, mencionada en la fase de construcción.
Los elementos de software se instalaron de forma correcta conforme a la arquitectura propuesta en la fase de diseño.

3.2.2 Puesta a punto

La puesta a punto del Sistema consistió en la configuración del Sistema para la operación de los distintos proveedores de GPS.

Las actividades realizadas fueron:

- Dar de alta los proveedores actuales con su configuración de mensajería
- Asignar eventos, alertas y comandos para cada proveedor de GPS
- Configuración de parámetros locales al proveedor
- Creación de perfiles de GPS
- Creación de políticas de seguridad

- Creación de perfiles de seguridad
- Configuración de notificaciones

Una vez que se tuvo la configuración del sistema se realizó una prueba de concepto con GPS productivos de forma controlada, la cual fue completamente satisfactoria, los GPS enviaban y recibían sin complicaciones los mensajes, ningún mensaje se pierde, la operación no se ve afectada si un nodo de la aplicación no está corriendo, ya que la arquitectura orientada a servicios puede crear un nuevo nodo en caso de falla y la operación se restablece, prácticamente, de manera inmediata.

El área de soporte quedó encargada de la supervisión del sistema, así como la notificación de anomalías e incidentes que pudieran afectar la operación del negocio, los primeros días también fue importante estar monitoreando de manera continua y constante que el sistema no tuviera ningún tipo de reducción en el rendimiento ya que con el aumento de dispositivos GPS era importante validar que las pruebas de estrés habían sido las suficientes para emular la carga de trabajo del ambiente de producción.

3.2.3 Resultados obtenidos

Es importante destacar los resultados obtenidos durante el proceso de pruebas así como lo fue durante la obtención de métricas durante los primeros 3 meses, los cuales se muestran en la Tabla 17.

Tabla 17. *Resultados obtenidos*

Métrica	Anterior	Actual	Conclusión
Tiempo de visualización de GPS	60 segundos para visualizar la unidad cuando se encuentra en movimiento.	4 segundos para visualizar la unidad cuando se encuentra en movimiento.	Se incrementa 15 veces el tiempo de visualización de la unidad en movimiento.

Integración de nuevo proveedores GPS	30 días para integrar un nuevo protocolo de comunicación.	3 días para integrar un nuevo protocolo de comunicación.	Se reduce 10 veces el tiempo de integración de nuevos proveedores.
Detección de situaciones de riesgo	75 segundos en el que un monitorista recibe una alerta y determina que es una situación de riesgo.	15 segundos en que el monitorista recibe una alerta e identifica la situación de riesgo.	Se reduce 5 veces el tiempo de detección de situaciones de riesgo.
Número de GPS reportando a plataforma	Más de 2,000 GPS reportando al sistema de manera concurrente.	Más de 6,000 GPS reportando al sistema de manera concurrente.	Incremento de 3 veces el número de GPS reportando de manera concurrente.

Conclusiones

A lo largo del desarrollo del presente reporte se hace notable la importancia que toma la ingeniería de software en el desarrollo de un producto de calidad; ayudándonos a establecer una metodología bien definida y llevar de forma ordenada el desarrollo de software durante todo el ciclo de vida de éste.

Mediante la metodología utilizada se pudo realizar un modelado de negocio que permitió conocer los diferentes procesos y flujos de negocio y con base en ellos establecer conceptos fundamentales que fueron clave en las subsecuentes fases del desarrollo. El tener claridad en los objetivos, así como una constante aportación de mejora de procesos y propuestas tanto de diseño como tecnológicas hicieron posible concluir con éxito el proyecto.

Finalmente, el resultado del presente proyecto fue la generación e implantación de un producto de software de alta calidad que cumplió con los objetivos de contar con una sola aplicación web que reuniera las diferentes funcionalidades que están distribuidas en las distintas instancias por cada proveedor de GPS de la aplicación actual, ser una aplicación completamente configurable y contar con la documentación correspondiente para el área técnica.

La flexibilidad que ofrece la aplicación resultante permite incorporar cualquier número de proveedores y GPS sólo con darlos de alta en el sistema y asignarles la funcionalidad que éstos adquieran, como son eventos, alertas y comandos.

La fácil adaptación del sistema para personalizar el negocio fue fundamental, además de que el rendimiento fue el necesario, puesto que sabemos que en un negocio de 24/7 en una empresa de seguridad donde el tiempo de respuesta para la recepción de un mensaje de ubicación del GPS o el envío de comando para el mismo es realmente importante, en el que podríamos estar hablando que el demorar más de 3 segundos en responder ante una situación de riesgo puede representar el riesgo de una vida.
Por otra parte, este sistema cambió la visión de la empresa en donde estaban acostumbrados a que los sistemas por la cantidad de información “deben ser lentos” lo cual quedó altamente demostrado, que una buena implementación con los recursos necesarios de un sistema, hace que el rendimiento de este sea tan ágil como sea requerido.

Trabajo futuro

La implantación del sistema fue realizada para contemplar diversos tipos de proveedores de GPS, distintas tramas de mensajería, un elevado rendimiento para responder de manera inmediata a situaciones de riesgo, entre otros. Sin embargo, algo que no se cubrió en esta planeación fue el implementar Integración Continua y Despliegue Continuo (CI/CD por sus siglas en inglés) el cual es un proceso que pretender estar implementando nuevas funcionalidades e integrándolas de manera

automática, probando y detectando las fallas cuanto antes, asegurando que el software pueda ser liberado en cualquier momento y de forma confiable.

Una vez realizada esta mejora al sistema, quedan pendientes las mejoras a sistemas que funcionan alrededor de este, ya que se puede decir que esta implementación es el núcleo del negocio, pero es importante considerar el implementar el consumo de mensajes de colas de RabbitMQ para que cada sistema consecuente aplique las reglas de negocio pertinente para cada necesidad de la empresa o de un cliente en específico.

Al cambiar la manera de pensar de la empresa de que los sistemas no tienen por qué ser lentos por la cantidad de información que procesan, es de suma importancia mantener siempre el enfoque y rendimiento de la aplicación ante el incremento de dispositivos GPS, por lo que se planea un monitoreo continuo para evitar cualquier incidente en el ambiente productivo. Esta funcionalidad será incorporada en la aplicación como servicios de auto-reporte mediante interfaces gráficas que indiquen el número de GPS conectados por puerto, tiempo de respuesta constante entre GPS y servidor o aplicación y se generarán dashboards interactivos para explorar los datos analíticamente.

Competencias y aprendizajes adquiridos

El llevar a cabo este proyecto de reingeniería me ha permitido fortalecer una vez más el aprendizaje adquirido durante mi carrera profesional, así como poner en práctica mi experiencia en el desarrollo de sistemas para dar solución a problemas de gran complejidad y generar sistemas de alta calidad.

Este proyecto también me aportó aprendizaje con relación a las consideraciones que se deben tener en cuenta para el desarrollo de aplicaciones web seguras, además de los tipos de pruebas que deben realizarse y herramientas utilizadas para validar que se cumpla con los requisitos de seguridad.

Es importante destacar que el perfil obtenido a partir de la finalización de los estudios profesionales fue clave para el desarrollo del sistema, puesto que incorporó un punto de vista analítico, pero a la vez lo suficientemente creativo para que al momento de brindar una solución al problema planteado en un principio fuera lo bastante robusto para soportar las necesidades de este.

Para finalizar, en el ámbito profesional el desarrollo de la Reingeniería y optimización en el sistema de comunicación con dispositivos móviles GPS representó un reto importante, ya que el estar a cargo de un proyecto y programarlo indica una carga mayor de trabajo y se refleja en el tiempo de conclusión, sin embargo, se considera que se pudo cumplir con cada una de las metas comprometidas al inicio del proyecto.

Referencias

[1] Lenguaje unificado de modelado, consultado en abril de 2021 disponible en: https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado

[2] Scrum (Desarrollo de software), consultado en abril de 2021 disponible en: https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)

[3] Node.js, consultado en abril de 2021 disponible en: https://nodejs.org/es/

[4] Craig, L. (2003), UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado, 2nd ed., Pearson Prentice Hall, Madrid, España.

[5] Bass L., Clements P., Kazman R. (2013), Software Architecture in Practice, 3rd ed., Pearson Education, Boston, US.

[6] Kruchten P., (1995), “Architectural Blueprints—The “4+1” View Model of Software Architecture”, consultado en abril de 2021 disponible en: http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf

[7] Booch G., James R., Jacobson I. (2010), El lenguaje unificado de Modelado: Aprenda UML directamente de sus creadores, 2nd ed., Pearson Prentice Hall, Madrid, España.

[8] Pressman, R. (2005), Ingeniería del software: Un enfoque práctico, 6th ed., McGraw-Hill, Madrid, España.

[9] PM2 – Advanced, Production Process Manager for Node.js, consultado en abril de 2021, disponible en: https://pm2.keymetrics.io

[10] PM2 (Software), consultado en abril de 2021, disponible en: https://en.wikipedia.org/wiki/PM2_(software)

[11] Qué es TCP/IP y características principales, consultado en abril de 2021, disponible en https://openwebinars.net/blog/que-es-tcpip/

[12] Messaging that just works – RabbitMQ, consultado en abril de 2021, disponible en https://www.rabbitmq.com

[13] Redis, consultado en abril de 2021, disponible en: https://redis.io

[14] Mocha – the fun, simple, flexible JavaScript framework, consultado en abril de 2021, disponible en: https://mochajs.org

[15] Chai, consultado en abril de 2021, disponible en: https://www.chaijs.com

[16] ECMA-262 - ECMA International, consultado en junio de 2021, disponible en: https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

[17] ECMAScript® 2020 Language Specification, consultado en junio de 2021, disponible en: https://www.ecma-international.org/wp-content/uploads/ECMA-262.pdf

[18] TypeScript: Typed JavaScript at Any Scale, consultado en junio de 2021, disponible en: https://www.typescriptlang.org/

[19] OWASP Top Ten Web Application Security Risks | OWASP, consultado en junio de 2021, disponible en: www.owasp.org/index.php/Top_10_2013-Top_10

[20] PCI Security Standards Council, “PCI DSS”, consultado en junio de 2021, disponible en: https://es.pcisecuritystandards.org/minisite/en/pci-dss-v3-0.php

[21] JSON, cosultado en junio de 2021, disponible en: https://www.json.org/json-es.html

[22] Cellocator – “Welcome to Cellocator”, consultado en junio de 2021, disponible en: https://www.cellocator.com/

[23] ISECOM “OSSTMM 3 – The Open-Source Security Testing Methodology Manual”, consultado en junio de 2021, disponible en: http://isecom.org/OSSTMM.3.pdf

[24] Weitzenfeld, A. (2005), Ingenieria de Software Orientada a Objetos con UML, 1st ed., Thomsom, D.F., Mexico.11

[25] Galin, D. (2004), Software Quality Assurance: From theory to implementation, Pearson Addison-Wesley