UNIVERSITAT JAUME I

Grado en Ingeniería Informática

Trabajo Final de Grado

Integración de servicios Microsoft Dynamics NAV - Movildata

Autor:
Dionisio García García

Supervisor:
Belén Collado López
Tutor académico:
Pablo Boronat Pérez

Fecha de lectura: 27 de Junio de 2018
Curso académico 2017/2018

Resumen

El presente documento recoge los detalles de la integración de servicios realizada durante la estancia en prácticas por el alumno Dionisio García García. El principal objetivo del proyecto es dotar a una configuración del ERP Microsoft Dynamics NAV, orientada al sector del transporte, de un mayor grado de interacción con las flotas de vehículos que se precisa gestionar, accediendo a la información requerida en el momento justo en la que esta se crea. La integración de un conjunto de servicios web, ofertados por la empresa Movildata, permitirán mejorar el conjunto de herramientas que actualmente usan los operadores de tráfico.

Como resultado del proyecto se obtiene el desarrollo de unos objetos propios del ERP Microsoft Dynamics NAV, capacitados para poder realizar las oportunas peticiones a los servicios web de interés, además de un trabajo de integración de dichos objetos en el propio ERP. El trabajo realizado permitirá automatizar la gestión de las órdenes de trabajo en curso de un planificador de tráfico, mediante la recepción de datos, fruto de la interacción con una aplicación Android del lado de los transportistas.

El desarrollo del presente proyecto ha sido realizado en la sede de la Asociación Empresarial Castellonense de Transporte de Mercancías (ACTM). También se ha contado con la colaboración de Sanmartí, una empresa de transporte de mercancías con años de experiencia en el sector, la cual además de servir de referente, ha contribuido de forma activa en la labor de validación del software desarrollado.

Palabras clave

Microsoft Dynamics NAV, Movildata, Integración, Servicios Web

Keywords

Microsoft Dynamics NAV, Movildata, Integration, Web Services

Índice general

1. Introducción **7**

1.1. Contexto y motivación del proyecto 7

1.2. La Empresa 8

1.3. Objetivos del proyecto 9

1.4. Estructura de la memoria 11

2. Descripción del proyecto **13**

2.1. Descripción general 13

2.2. Herramientas y tecnologías 16

2.2.1. Microsoft Dynamics NAV 16

2.2.2. Postman 17

2.2.3. Sublime Text 17

2.2.4. SDKs Movildata 17

2.2.5. MagicDraw UML Editor 19

2.2.6. API Google Maps 19

3. Planificación del proyecto **21**

3.1. Metodología 21

3.2. Planificación 22

3.3. Estimación de recursos y costes del proyecto 24

3.4. Seguimiento del proyecto 26

4. Análisis y diseño del sistema **29**

4.1. Análisis del sistema 29

4.2. Diseño de la arquitectura del sistema 33

5. Implementación y pruebas **35**

5.1. Detalles de implementación 35

5.1.1. Primeras pruebas de conectividad 35

5.1.2. Desarrollo del *CodeUnit* "50006 - Funciones Movildata" 43

5.1.3. Desarrollo del *CodeUnit* "50007 - Funciones Navifleet" 50

5.1.4. Desarrollo del *CodeUnit* "Timer" 65

5.1.5. Desarrollo del *CodeUnit* "50008 - Funciones Movilflota" 66

5.1.6. Integración de servicios con Google Maps 70

5.2. Verificación y validación 70

6. Conclusiones **71**

6.1. Conclusión en el ámbito formativo 71

6.2. Conclusión en el ámbito profesional 71

6.3. Conclusión en el ámbito personal 72

6.4. Futuras extensiones del proyecto 73

Bibliografia **76**

A. Modelo de entrevista para los operadores de tráfico **79**

B. Resumen batería de entrevistas **83**

C. Propuesta de automatización del patrón de mensajería **87**

D. Última toma de requisitos **93**

Índice de figuras **98**

Índice de cuadros **99**

Capítulo 1

Introducción

Índice

1.1. Contexto y motivación del proyecto 7
1.2. La Empresa 8
1.3. Objetivos del proyecto 9
1.4. Estructura de la memoria 11

1.1. Contexto y motivación del proyecto

La propuesta del presente proyecto formativo ofertado a la Universidad Jaume I para los alumnos del Grado de Ingeniería Informática, surge de una necesidad real de una determinada empresa afiliada a la Asociación Empresarial Castellonense de Transporte de Mercancías (ACTM). La necesidad que presenta el asociado gira en torno a conseguir un nivel superior de gestión y control de la información sobre la flota de vehículos que posee. Se requiere pues, automatizar un conjunto de mecanismos de monitorización e implantar nuevos servicios que doten al personal al cargo del seguimiento de los conductores, de un mayor control sobre la información que se va generando. Para poder alcanzar este nivel de automatismo se precisará de la comunicación del ERP Microsoft Dynamics Nav con un sistema externo de seguimiento y control de flotas, propiedad de la empresa murciana Movildata.

El nivel extra de integración propuesto tiene como núcleo el uso de un conjunto de llamadas a servicios web sobre los servidores de Movildata, los cuales están continuamente registrando datos provenientes de los terminales móviles con los que están equipados los vehículos. Dichas llamadas a los servicios web permiten obtener, entre otros datos, el acceso al posicionamiento exacto de un vehículo, envío y recepción de mensajería de texto, obtención de kilometrajes, parámetros relativos al tacógrafo, además de una detallada gestión de órdenes de trabajo y expedientes de viaje para los vehículos de una determinada compañía.

1.2. La Empresa

La Asociación Empresarial Castellonense de Transporte de Mercancías (ACTM), con sede en Castellón de la Plana, más concretamente en la Ciudad del Transporte, es una entidad sin ánimo de lucro que agrupa a empresarios del transporte de mercancías por carretera de toda la provincia de Castellón.

Fundada el 7 de junio de 1977, en la actualidad, la asociación cuenta con alrededor de 160 socios los cuales son autónomos, pequeñas y medianas empresas y agencias.

La asociación forma parte de CETM (Confederación Española de Transportes de Mercancías), la cual agrupa a empresarios del ámbito del transporte de mercancías de todo el territorio nacional, siendo esta la principal organización nacional del sector y la más destacada en Europa gracias al número de afiliados y vehículos.

Entre las principales actividades desarrolladas por la ACTM por sus diferentes departamentos, destacamos:

- Desarrollo de actividades de formación dirigidas a los profesionales de la conducción.
- Servicios internos y externos de informática, tanto en instalaciones como en materia de aplicaciones informáticas comunes o especificas del transporte.
- Servicios de tramitación administrativa y de gestión tanto para la ACTM como para los asociados.
- Servicios de tramitación administrativa frente a los distintos órganos de la administración pública.

Ampliando el detalle de los servicios de informática que presta la ACTM, destacar que el departamento de informática proporciona a sus afiliados asesoramiento informático tanto a nivel de redes como de usuario, además de prestar soporte con distintas herramientas informáticas.

En lo que al presente proyecto respecta, hacer especial mención al desarrollo, implantación y mantenimiento de una aplicación a modo de vertical de transporte, implantada en el ERP Microsoft Dynamics NAV. Esta configuración del ERP permite la gestión administrativa, contable y fiscal de la actividad propia de una empresa del sector del transporte, bien en la propia sede o a través de servidores residentes en la asociación.

1.3. Objetivos del proyecto

El objetivo principal del proyecto es poder ofrecer una solución informática real, la cual permita una mayor interacción del ERP Microsoft Dynamics NAV con los servicios de gestión y control de flota propios de la empresa Movildata. Tras la finalización de la estancia en prácticas y en base a los resultados obtenidos, se tomará la decisión de habilitar el trabajo de integración realizado en el vertical del transporte. De forma análoga se barajará la posibilidad de adquirir un determinado número de dispositivos de tipo tableta, dotados del sistema operativo Android. Esta configuración de dispositivos permitirá poder optar a trabajar con la aplicación móvil MFLEET, cuyo aspecto se presenta en la figura 1.1. Con desarrollo perteneciente a Movildata, MFLEET se presenta como una aplicación atractiva y de sencillo uso con la que realizar las principales labores de seguimiento de vehículos del lado del conductor.

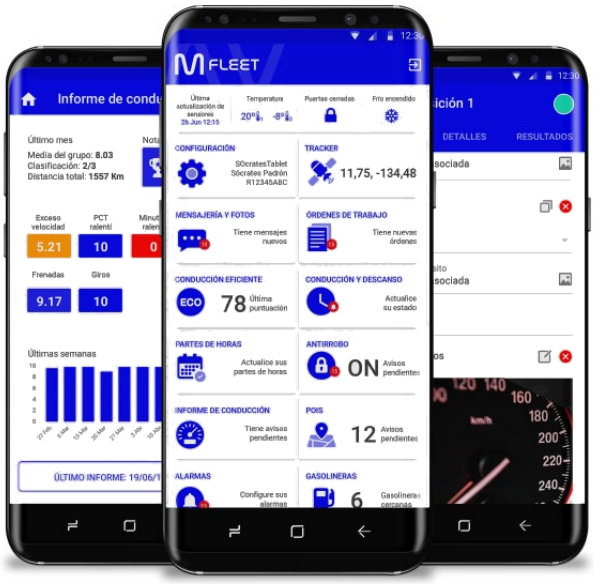

Figura 1.1: Aspecto de la aplicación MFLEET.

A rasgos generales se precisa de un desarrollo de código sobre un conjunto de objetos y estructuras del ERP Microsoft Dynamics NAV que permitan la recepción, transformación y almacenaje de los items de información requeridos en cada caso.

La enumeración de las actividades incluidas en los objetivos del proyecto se presentan a continuación:

- Estudio de la propuesta de proyecto.
  - Estudio del escenario de la propuesta de proyecto.
  - Estudio de la documentación de Movildata.
  - Estudio del manual de referencia de Microsoft Dynamics Nav.
  - Visita y toma de requisitos al cliente asociado de ACTM, Sanmartí.

- Diseño e implementación.
    - Pruebas de conectividad de Microsoft Dynamics Nav contra la API de Movildata
    - Creación de funciones de interpretación de estructuras de JSON y SOAP-XML programadas en lenguaje C/AL.
    - Diseño del modelado de estructuras de datos que permitirán alojar los datos fruto de las llamadas a los servicios web de Movildata.
    - Programación formal del conjunto de objetos, propiedades y disparadores en lenguaje C/AL en el entorno de desarrollo de Dynamics Nav para la solicitud y recogida de datos, procedentes del acceso a los servicios web de Movildata.
- Test, habilitación de la implantación y validación con el usuario.
    - Test de integración de sistemas.
    - Habilitación de los objetos programados en la base de datos del cliente.
    - Generación de documentación a modo de procedimientos de usuario.
    - Visita al cliente, verificación y validación del producto desarrollado.

A fin de poder delimitar de forma más precisa el alcance del proyecto, se procede a enumerar una serie de objetivos que quedarían fuera del mismo. El presente proyecto no es un trabajo de creación y habilitación de módulos de venta o gestión de recursos empresariales como tal, propios de la intensificación de Grado, Sistemas de la Información. Por tanto, el objetivo de este trabajo no ha sido aprender la herramienta Microsoft Dynamics NAV en profundidad y desde la perspectiva de un alumno con las competencias de la asignatura EI1046, Sistema de Información Integrados. Podría resumirse el presente proyecto como un desarrollo puro de software que permite comunicar dos sistemas mediante el uso de servicios web. Nociones de contabilidad, generación de informes, composición de menús, gestión de usuarios o incluso, gestión de recursos que no guarden relación con la información que se rescata de las llamadas a servicios web a Movildata, quedan fuera de los objetivos del proyecto.

El logro del total de objetivos viene condicionado en gran medida por el tiempo del que se dispone durante la estancia en prácticas para un alumno del Grado de Ingeniería Informática (300 horas presenciales). Teniendo en cuenta el punto del que se parte; con un gran nivel de desconocimiento sobre los ERPs y teniendo que realizar un estudio en profundidad de la documentación de Movildata, además de otros aspectos relativos, se requiere de un planteamiento realista acorde a los recursos de los que se dispone. El mejor de los planteamientos posibles pasa por lograr una integración base que permita realizar la labor de conexionado anteriormente citada, con vistas a ser ampliado y/o modificado en un futuro si así se requiere. Este enfoque tiene como objetivo ir generando un producto de forma incremental, logrando desarrollar la mayor cantidad de funcionalidad asociada a los requerimientos planteados por las necesidades del cliente.

La finalización del periodo de la estancia en prácticas debe proporcionar una documentación que permita aplicar cambios y mejoras a la integración, así como describir el funcionamiento de cada una de las funcionalidades desarrolladas. Tanto el departamento de informática de ACTM, como los usuarios finales, deben poder encontrar en esta documentación una guía de referencia para poder hacer uso de las distintas partes de la integración.

1.4. Estructura de la memoria

A continuación se resume brevemente la temática que aborda cada capítulo, además de presentar el contenido de los anexos adjuntos al presente trabajo:

Capítulo 1: El primero de los capítulos que presenta este trabajo, introduce al lector en el contexto del proyecto, qué motivaciones lo han originado y qué objetivos ambiciona la consecución del mismo; además de una breve descripción sobre la sede de la empresa donde se ha realizado la estancia en prácticas.

Capítulo 2: Una definición detallada del proyecto se recoge en este capítulo. Las herramientas más representativas que han sido usadas durante la estancia tienen una breve descripción en el presente capítulo.

Capítulo 3: Aspectos relacionados con la metodología escogida para la realización del proyecto, una comparativa con la planificación inicial y la final, además de las medidas de seguimiento para la consecución de la planificación son abordados en el capítulo. También es incluido un apartado de costes y estimación de recursos.

Capítulo 4: El total de labores de análisis llevadas a cabo y la presentación formal de los requisitos que debe cumplir el desarrollo de integración de sistemas son recogidos en este capítulo. Un apartado de diseño de arquitectura permitirá al lector comprender qué forma ha tomado la integración desarrollada.

Capítulo 5: La implementación realizada y su problemática asociada a la fase de desarrollo se recogen en este capítulo. Una combinación de esquemas, texto y código pretenden que el lector pueda reproducir parcial o totalmente el trabajo realizado.

Capítulo 6: Una conclusión final desde diferentes puntos de vista: profesional, personal y formativo, intentan hacer una valoración global que reúna el mayor número de aspectos con los que se ha trabajado durante la estancia en prácticas.

Anexo A: El primero de los anexos presenta la plantilla que sirve para realizar la toma de requisitos a los operadores de tráfico. Su cometido es servir de guía para poder entrevistar de forma presencial y personalizada a cada uno de los operadores.

Anexo B: El segundo anexo recoge el total de peticiones de mejora, fruto de la toma de requisitos realizada sobre el conjunto de operadores de tráfico. Este documento filtra y muestra únicamente los aspectos relativos a la integración de servicios con Movildata.

Anexo C: El tercer anexo contiene un documento que presenta el trabajo de análisis sobre la automatización del patrón de interacción entre operadores de tráfico y transportistas.

Anexo D: Un último anexo muestra el documento que recoge la última batería de refinamientos y requisitos recogidos en una reunión en la sede del cliente.

Capítulo 2

Descripción del proyecto

Índice

2.1. Descripción general 13
2.2. Herramientas y tecnologías 16
2.2.1. Microsoft Dynamics NAV 16
2.2.2. Postman 17
2.2.3. Sublime Text 17
2.2.4. SDKs Movildata 17
2.2.5. MagicDraw UML Editor 19
2.2.6. API Google Maps 19

2.1. Descripción general

La integración de servicios sobre la que se ha trabajado, cubre una serie de aspectos de mejora para un vertical del transporte desarrollado sobre la base del ERP Microsoft Dynamics NAV. Definiremos a un vertical del transporte como una aplicación informática que suele tener como base un ERP y que ofrece una configuración de uso que permite dar soporte a las necesidades de gestión de recursos de las empresas de transporte de mercancías. Los principales usuarios del vertical son aquellos asociados de ACTM quienes, al darse de alta, cuentan con la posibilidad de hacer uso de esta refinada y optimizada solución.

De entre todos los asociados de ACTM, nos centraremos en Sanmartí, una empresa de transportes de mercancías, con una gran trayectoria en el sector y quien nos presta soporte para poder realizar el proyecto, así como nos sirve de referencia a fin de poder dar forma a la integración, aportando su experiencia acumulada.

Realmente el proyecto nace de la necesidad que Sanmartí tiene de mejorar el sistema de control y gestión de flotas, compuesta hasta la fecha por una combinación entre el vertical mencionado, una aplicación web de gestión de flotas de la empresa Movildata y varias consultas telefónicas a los propios conductores al cargo de los distintos servicios de transporte.

Los principales usuarios de esta configuración de herramientas de seguimiento de flotas, son un equipo de operadores de tráfico. A cada operador de tráfico se le asigna un ámbito territorial y en consecuencia ha de velar por que un conjunto de conductores asignados lleven a cabo sus servicios de la manera más eficiente y segura posible.

Un servicio de transporte que ha de ser gestionado por un operador de tráfico podría resumirse en una secuencia de pasos rutinaria que se describe a continuación:

- Creación de un pedido de venta por parte de un cliente que desea recibir una cantidad determinada de mercancía en una fecha concreta.
- Asignación de un recurso en forma de vehículo para efectuar el servicio de transporte.
- Control y seguimiento de la información del pedido de venta en el planificador de tráfico.

Con respecto al planificador de tráfico citado, se trata de una herramienta que dispone el vertical del transporte, capaz de ofrecer el detalle sobre un conjunto de pedidos de venta en curso. Común para todos los operadores de tráfico, el planificador es el lugar donde van variando los diferentes estados para un pedido de venta. Esta variación de estados hasta la fecha se realiza de forma manual con la introducción de unos determinados campos de información. La figura 2.1 muestra el aspecto del planificador de tráfico.

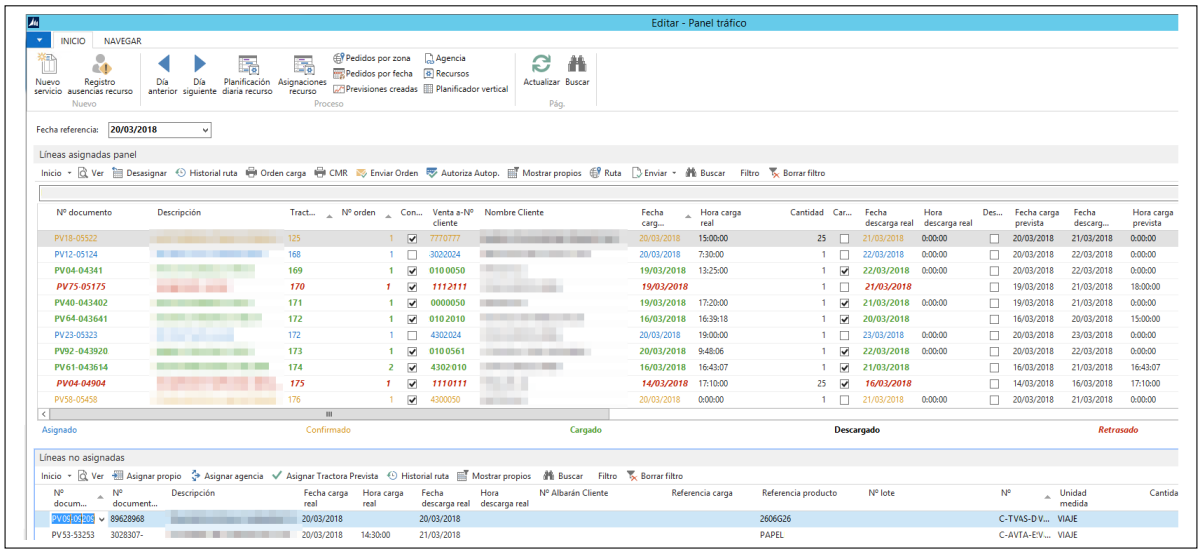

Figura 2.1: Planificador de tráfico.

Según Sanmartí, la configuración de herramientas anteriormente citada requiere de un nuevo planteamiento a fin de poder automatizar una serie de tareas realizadas manualmente y con bastante frecuencia por parte de los operadores de tráfico. El nuevo planteamiento pasa por integrar algunos de los servicios web que la empresa Movildata oferta.

Actualmente Sanmartí cuenta con un nivel de integración con los servicios de Movildata que permite la comunicación por mensajería de forma bidireccional entre operadores de tráfico y conductores. Esta comunicación puede realizarse mediante el uso de unos terminales marca

Garmin montados en los vehículos en el lado de los transportistas, y una aplicación web, propietaria de Movildata que permite el seguimiento por carretera mediante mapas y además llevar a cabo las labores de mensajería citadas entre otras muchas funciones.

La aplicación web anteriormente citada presenta a priori una serie de inconvenientes reportados directamente por los operadores de tráfico. Destacamos el tener que encontrar un mensaje concreto de entre una gran cantidad de mensajes, fruto de la múltiples interacciones con los conductores. La aplicación muestra en un único canal de mensajería, compartido por todos los operadores, el total de mensajes enviados.

Los mensajes recibidos por los conductores contienen información de especial interés para el seguimiento de los pedidos de venta, además de cubrir aspectos complementarios a los servicios de trasporte que realizan. Destacamos confirmaciones de inicio o finalización de actividad, datos de kilometrajes y carga de mercancía entre otros. Esta información posteriormente será incluida en la facturación que finalmente se haga llegar al cliente que solicitó el servicio de transporte. La figura 2.2 muestra el aspecto de la aplicación web que gestionan los operadores de tráfico.

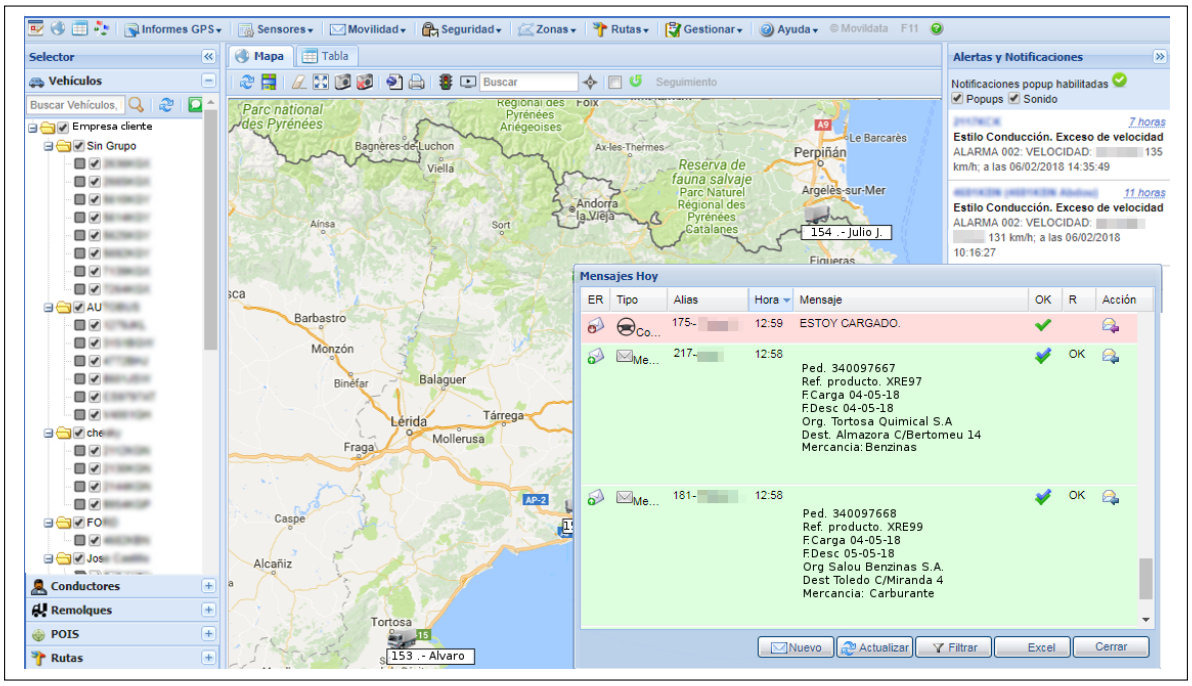

Figura 2.2: Aplicación web Movilflota.

Aquella información que no se puede recoger a través del canal de mensajería, debido a eventuales fallos en la cobertura del servicio, precisa por lo general de un seguimiento telefónico. La inmediatez de realizar una llamada telefónica permite obtener información veraz al instante. La información que no es recogida por el canal de mensajería o por vía telefónica, se recibe en la sede en forma de documentos impresos, donde los propios profesionales del transporte cumplimentan una serie de campos de información. Esta última forma de recoger la información es quizás más propensa a contener errores, pues en ocasiones, los profesionales del transporte realizan las anotaciones de datos a destiempo con la actividad que realizan. Para cuando esta información llega a la sede, el servicio de transporte ya ha finalizado y han de detectarse entonces, posibles errores y aplicar las oportunas correcciones en cada caso.

2.2. Herramientas y tecnologías

El total de las herramientas usadas durante la consecución del proyecto serán descritas a continuación.

2.2.1. Microsoft Dynamics NAV

En su versión 2015 este ERP proporciona la base necesaria, a partir de la cual, el vertical de transporte ha sido desarrollado.

Dynamics NAV es una solución de gestión de recursos empresariales de gran éxito comercializada por Microsoft. Su principal sector de mercado son las pequeñas y medianas empresas de todo el mundo. Su aspecto atractivo, experiencia de usuario similar al uso de los sistemas operativos Windows y su gran nivel de adaptabilidad, hacen que este ERP se posicione entre las opciones más valoradas a la hora de escoger un ERP para una empresa en la actualidad.

Muchas de las empresas que ya cuentan con una solución Dynamics NAV se hallan migrando sus instalaciones locales a versiones con servicio en la nube, llamadas Dynamics 365. Las versiones *cloud* ofrecen la gestión de la información de forma cómoda, alojando los datos en los servidores dedicados de Microsoft. Dynamics 365 se puede adquirir junto a otros paquetes de software de Microsoft, formando planes adaptables de pago por uso.

Referente al desarrollo del presente proyecto, se ha procedido a realizar una instalación básica en el computador personal del alumno. La versión elegida para realizar la instalación ha sido la 2016. Esta versión de Microsoft Dynamics NAV ha resultado ser compatible en el ámbito del código desarrollado con la versión 2015, aportando un nivel notorio de comodidad en el editor de código de desarrollo de objetos. La instalación en el computador personal ha sido posible gracias a la activación de una licencia de estudiante proporcionada por el profesorado de la asignatura EI1046, Sistema de Información Integrados.

El objetivo de la instalación ha sido recrear un proceso de instalación real de la herramienta, además de dotar al computador personal del alumno de una configuración de herramientas de desarrollo compatible con el del vertical del transporte de ACTM, y poder llevar a cabo un sinfín de pruebas y ejercicios en un entorno seguro y confiable.

El fundamento de trabajo de Microsoft Dynamics NAV es el desarrollo mediante el Client/Server Integrated Development Environment (C/SIDE) de una serie de objetos propios del ERP. El desarrollo de estos objetos permite configurar la aplicación hacia las necesidades particulares que pueda tener una organización. La aplicación ya cuenta con varios módulos y una funcionalidad base, nativa del estándar de Microsoft, la cual es más que suficiente para poder atender las necesidades de gestión de recursos empresariales básicas.

2.2.2. Postman

Postman [8] es una completa suite de pruebas enfocada al desarrollo de APIs. Gracias a una interfaz de sencillo uso y el excelente nivel de configuración sobre el tipo de parámetros con los que se puede trabajar, permite la gestión de llamadas a servicios web de una forma rápida y fácil. Postman cuenta además con un registro de las llamadas efectuadas, guardando para cada caso, el conjunto de parámetros involucrados junto a los cuerpos de los mensajes, tornándose una herramienta altamente productiva.

Postman permite su uso sin coste alguno para pequeños proyectos. Adherirse a planes de uso de la herramienta con carácter profesional es posible. Estos planes de uso aportan un uso compartido entre usuarios, además de características avanzadas y total asistencia a la organización que las contrate.

Para el presente proyecto se ha trabajado principalmente con el protocolo "Single Object Access Protocol" (SOAP) y el archiconocido formato ligero de texto "Java Script Object Notation" (JSON). Postman ha sido perfectamente capaz de dar soporte para devolver cada uno de los resultados para cada una de las llamadas a servicios web y permitir así el poder desarrollar código de forma más ágil en Dynamics NAV.

2.2.3. Sublime Text

En paralelo al editor que proporciona el entorno de desarrollo de Microsoft Dynamics NAV y como principal herramienta de edición de texto plano, se ha usado Sublime Text [9]. Este editor de textos permite la edición de código de forma fácil y atractiva. Un amplio abanico de configuraciones permiten que este editor satisfaga las necesidades de los desarrolladores más exigentes, aportando un efecto visual muy de moda.

2.2.4. SDKs Movildata

Los "Software Development Kit" (SDKs) de Movildata son tres documentos en línea a modo de soporte para desarrolladores e integradores de sistemas. Cada uno de los SDK agrupa un conjunto de especificaciones y características que permiten programar los servicios web ofertados por Movildata. El objetivo común de estos SDK es la gestión y control de flotas de vehículos por carretera, bien sea para repartidores de paquetería, comerciales o transportistas de mercancías en general.

La figura 2.3 muestra para cada SDK su ámbito de acción sobre el total de servicios posibles a integrar. A continuación se detallarán los servicios que incluye cada documento para poder ser integrados con otros sistemas de la información.

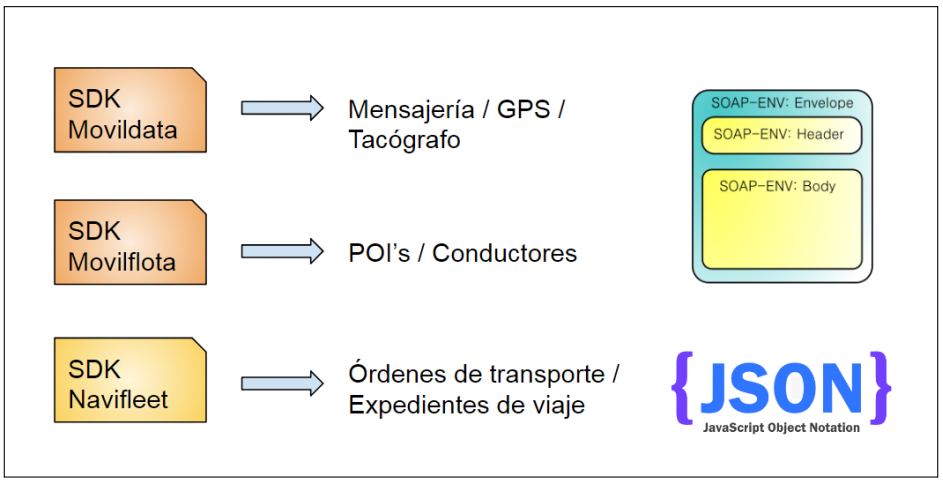

Figura 2.3: SDKs de Movildata.

SDK Movildata

El SDK de Movildata reúne una gran cantidad de servicios web de tipo SOAP-XML que permiten al integrador dotar al sistema ERP de la siguiente funcionalidad:

- Recepción de alarmas producidas por los conductores
- Obtención de kilometrajes.
- Obtención de datos registrados por el FMS/CAN-BUS.
- Obtención de la última localización registrada.
- Obtención de datos del GPS.
- Obtención de datos del tacógrafo.
- Gestión de gastos generados por los conductores.
- Gestión de mensajería simple.

Puede accederse al SDK de Movildata a través de la siguiente referencia bibliográfica [**?**].

SDK Movilflota

El SDK de Movilflota propone del mismo modo un conjunto de servicios web que requieren del protocolo SOAP-XML para la gestión de los POIs (Puntos de interés). Entre los servicios web proporcionados por el SDK destacar los siguientes:

- Gestión de conductores.
- Gestión en la asignación de conductores y vehículos.
- Adición de POIs.
- Borrado de POIs.
- Actualización de POIs.
- Obtención de los tipos de POIs.
- Obtención del total de POIs dados de alta.

SDK Navifleet

Este último SDK posee una corta lista de llamadas a servicios web en formato JSON. Los servicios web en este caso, permiten la gestión de órdenes de trabajo con un claro enfoque al sector del transporte de mercancías. El objetivo que propone este SDK es poder operar con la aplicación Android MFLEET del lado de los conductores. Este SDK proporciona una configuración base muy rica en la estructuración de la información para realizar la labor de seguimiento de flotas de vehículos. Entre los servicios web proporcionados por el SDK destacar los siguientes:

- Envío de órdenes de trabajo.
- Actualización de órdenes de trabajo.
- Cancelación de órdenes de trabajo.
- Obtención de eventos para una orden de trabajo

Indicar que no ha sido posible publicar en el presente documento el acceso a los SDKs mediante una referencia bibliográfica. También aquellas referencias a las URLs incluidas en los fragmentos de código ilustrativos, expuestos en apartados posteriores, han sido ocultados. Esto ha sido así por petición expresa de la empresa propietaria Movildata. Para cualquier información o aspecto relacionado con la documentación que ha permitido llevar a cabo las diferentes tareas de integración de servicios se recomienda realizar la consulta pertinente directamente a Movildata [6].

2.2.5. MagicDraw UML Editor

MagicDraw [7] es la herramienta CASE de modelado UML por excelencia. Especialmente diseñada para equipos de analistas de software, arquitectos de software, ingenieros y programadores en general. Esta suite es capaz de satisfacer las más variadas exigencias con respecto al modelado y representación de sistemas de la información. La herramienta cuenta con la posibilidad de trabajar con una gran cantidad de diagramas para poder asistir en la programación orientada a objetos. Además destacar que MagicDraw cuenta con la posibilidad de modelar esquemas de bases de datos, realizar estudios de ingeniería inversa sobre código y realizar análisis de aplicaciones web con la extensión UWE.

MagicDraw ha sido usado para la confección de un diagrama de secuencia que permita describir el patrón de interacción entre conductor y operador de tráfico. De igual modo, se ha usado para la elaboración del diagrama de casos de uso que recoge la mayor parte de la funcionalidad a desarrollar requerida por el cliente.

2.2.6. API Google Maps

Desde Junio de 2005, Google lanzó para uso libre de desarrolladores la posibilidad de poder dotar a sus aplicaciones de geolocalización haciendo uso de la API de Google Maps [1]. De este modo se cuenta con la posibilidad de poder realizar gráficos con posicionamiento preciso de

ubicaciones de objetos sobre mapas. Esta API de sencillo uso, contribuye en la construcción de más interesantes y atractivas aplicaciones.

El trabajo realizado en base a esta API incluye la apertura de mapas en un navegador externo al ERP a fin de poder posicionar vehículos y realizar cálculos de distancias.

Capítulo 3

Planificación del proyecto

Índice

3.1. Metodología 21
3.2. Planificación 22
3.3. Estimación de recursos y costes del proyecto 24
3.4. Seguimiento del proyecto 26

3.1. Metodología

El presente proyecto ha sido desarrollado siguiendo unos principios concretos propios de las metodologías ágiles. Sin embargo no se ha seguido ninguna metodología de forma rigurosa. La principal razón ha sido el desconocimiento del entorno de trabajo y de las personas que iban a tomar parte en este proyecto de forma directa o indirecta, así como la forma de acontecer los sucesos. A continuación se comentarán aquellos aspectos que han hecho que este proyecto haya adoptado una metodología de trabajo muy próxima a las definidas como ágiles, con fundamento en una serie de principios que seguro recordarán de forma no casual a otras metodologías de renombre.

El principio que ha primado sobre el resto ha sido sin duda la entrega frecuente de software funcional con aporte de valor significativo al cliente. Para el caso, el rol de cliente lo ha asumido la empresa de transportes que ha colaborado con el proyecto, Transportes Sanmartí.

Las entregas de software en forma de objetos de Microsoft Dynamics NAV han sido habilitadas en la base de datos del cliente, obteniendo así funcionalidad al instante para evaluar. Tras la presentación de la funcionalidad desarrollada se procede siempre a validar de forma objetiva el desarrollo conjuntamente con el cliente, obteniendo en cada caso la retroalimentación necesaria para realizar los posteriores refinamientos.

Un segundo principio sobre el que se ha trabajado ha sido el de crear software única y exclusivamente bajo demanda. En un ambiente en el que la definición de requisitos es propensa

al cambio, como ha sido el caso del presente proyecto, prima atender de forma concisa aquella funcionalidad que se ha acordado desarrollar. Este principio no ha sido inconveniente para estar continuamente abiertos a posibles cambios que afecten directamente en la especificación de requisitos inicial o el poder dar cabida a nuevos requerimientos por parte del cliente.

Por ser un proyecto desarrollado por dos personas, en varios momentos de codificación se ha empleado la técnica de programación por pares, más conocida como "Pair programming". Esto ha sido posible gracias a una estupenda coordinación entre supervisora y alumno, donde ha primado en todo momento el volcado de conocimiento mutuo.

Hay que destacar pues, que el ámbito de desarrollo, se ha visto influenciado por un espíritu de mejora continua, conocido en la jerga del "lean manufacturing" por Kaizen, el cual ha estado presente durante todas las etapas del proyecto.

Con respecto a la codificación, esta se ha llevado a cabo siguiendo las recomendaciones propias de un desarrollo mantenible, modular y siguiendo las convenciones de codificación propuestas por la comunidad de desarrolladores de Dynamics NAV. De esta forma, futuras intervenciones en la integración serán posibles en el menor coste de tiempo y esfuerzo. El lenguaje predominante en los desarrollos efectuados ha sido el C/AL. Este lenguaje de manejo de datos para el lado de desarrollo de Microsoft Dynamics NAV hereda la filosofía y fundamento de PASCAL, siendo un lenguaje de programación estructurado y fuertemente tipado. En C/AL el código se estructura en funciones o procedimientos facilitando así el uso de la programación estructurada. El tipo de datos para las variables debe ser declarado con antelación para que estas puedan ser usadas. C/AL requiere del uso de ciertas librerías externas .NET, las cuales podemos importar cómodamente y que resultan familiares en el entorno de desarrollo de Dynamics NAV.

3.2. Planificación

En cuanto a la planificación del proyecto cabe destacar que en un primer momento se había realizado una planificación del tiempo haciendo uso de modelos de gestión predictiva de proyectos. Sin embargo la realidad ha sido otra muy distinta. La naturaleza de los acontecimientos y cómo estos han ido sucediendo en la línea temporal correspondiente a la estancia en prácticas, ha sido bastante irregular, por lo que el modelo predictivo de gestión inicialmente propuesto no ha sido posible cumplirlo. En la figura 2.3 se presenta el "Work Breakdown Structure" (WBS), incluido en la Propuesta Técnica realizada para este proyecto durante la primera quincena de la estancia.

La planificación que finalmente se ha llevado a cabo podría estructurarse en cuatro bloques claramente diferenciables entre sí. Cada uno de estos bloques se compone de un periodo de formación, toma de requisitos, desarrollo de código y verificación final con el cliente. La repartición de las 300 horas de trabajo se realiza de un modo más o menos equitativo con los cuatro bloques. El resumen de las acciones llevadas a cabo en cada uno de los bloques se describe a continuación:

Fase 1. Formación y puesta en escena.

La primera de las labores es la presentación del personal, instalaciones y equipos con los que se va a trabajar. Posteriormente se especifican los objetivos que engloban el proyecto de integración de servicios Dynamics NAV - Movildata. El proyecto ya había sido presupuestado por una consultoría informática y estimado su desarrollo en tiempo y coste económico.

Durante la primera fase se llevan a cabo tareas de formación con respecto al entorno de desarrollo de Microsoft Dynamics NAV. El estudio parcial del manual de Navision 4.0 ha supuesto el punto de partida en la iniciación con el ERP. Varias sesiones de trabajo en las que la supervisora ha atendido los requisitos de desarrollo de varios de sus asociados han servido para dar una idea de la forma de trabajar con Microsoft Dynamics NAV en un entorno laboral real.

La información relativa a los SDKs de Movildata empieza a ser estudiada con detenimiento, pues con la extracción de datos efectuada y el análisis de la información que se recopile de los requisitos del cliente, ha de poder hallarse una simbiosis entre el sistema de mensajería propuesto por Movildata y el vertical del transporte que gestionan los asociados de ACTM.

En esta primera fase se incluye también una toma de requisitos, efectuada a los operadores de tráfico de la empresa Transportes Sanmartí, quienes nos hacen llegar su punto de vista con respecto al planificador de trafico incluido en el vertical del transporte de ACTM y nos comentan ciertos aspectos a mejorar sobre su día a día.

Fase 2. Desarrollo del *CodeUnit* Movildata.

Podemos decir que la segunda fase se centra en la elaboración de código sobre los servicios web del SDK de Movilflota. Un conjunto de llamadas HTTP de tipo POST siguiendo el protocolo de comunicación SOAP-XML ofrecen una funcionalidad que permite recepcionar información en tiempo real, información sobre el tacógrafo, GPS, alertas o mensajería simple. Las primeras pruebas de conectividad con Postman recogen los primeros resultados. La elaboración del *CodeUnit* "50006 Funciones Movildata" contiene una selección con las llamadas a servicios web que tienen un mayor aporte de valor para el cliente.

Fase 3. Desarrollo del *CodeUnit* Navifleet.

El estudio a fondo del SDK Navifleet permite la elaboración del *CodeUnit* "50007 Funciones Navifleet", el cual permite el envío y cancelación de órdenes de trabajo con el formato propio de los pedidos de venta del cliente. Mediante llamadas también de tipo HTTP POST, esta vez con el envío de parámetros en formato JSON, se desarrolla un objeto que permite alcanzar un nivel de integración de servicios altamente eficaz.

Fase 4. Desarrollo del *CodeUnit* Movilflota y finalización de proyecto.

Un estudio del SDK de Movilflota aporta el punto de conocimiento necesario para poder traer los POIs dados de alta en la aplicación web de gestión de Movildata por los operadores de tráfico. También podemos realizar las distintas operaciones CRUD (Creación, lectura, actualización y borrado) relativas a los POIs. Una labor de realización de la demo que se presentará en ante el tribunal universitario es elaborado de igual modo durante esta fase.

3.3. Estimación de recursos y costes del proyecto

Se procederá a enumerar los aspectos de infraestructura y de material informático empleados así como el desglose de las horas de mano de obra a nivel programador junior. Las estimaciones son meras aproximaciones que pretenden poner de manifiesto el conocimiento sobre la existencia de costes sobre el material utilizado. Las cantidades monetarias son en algunos casos orientativas y pueden variar sustancialmente en el momento de la lectura del presente documento.

La cantidad de horas en las que está estimado el proyecto es fija y son 300 horas de trabajo efectivo práctico en la empresa que propone el proyecto formativo.

Para acondicionar un entorno de trabajo se requiere de unas instalaciones las cuales ACTM proporciona junto a unos servicios mínimos de luz, agua y conexión a internet. Para el alumno que va a realizar la estancia en prácticas, existe un equipo configurado expresamente compuesto por una torre de procesamiento, pantalla, ratón y teclado. Su coste no será contabilizado.

Para desarrollos personales se ha usado el computador personal. Este se trata de un modelo actualmente descatalogado de la marca MSI (CX61). El portátil monta un procesador Intel Core i5-4210M @ 2.60GHz. La memoria principal es de 4GB. La memoria secundaria es un disco duro SSD de 240 GB que reemplaza al disco mecánico que viene de serie. Un equipo con esta configuración en el mercado suele rebasar los 500 euros. Adjuntaremos el precio de este equipo como parte del presupuesto.

Una licencia personal para poder hacer uso del editor de textos Sublime Text asciende a 80 dólares, que al cambio supone alrededor de 68 euros (el propietario puede hacer uso de la licencia en varias máquinas distintas).

Las horas de analista de software, las cuales incluyen las labores de toma de requisitos y generación de documentación complementaria, se estiman en 60 euros por hora. Con una carga horaria de alrededor de un cuarto del tiempo, al total habría que añadir 4200 euros brutos en concepto de labores de analista.

Para las horas de programación se presenta una tarifa por hora de 40 euros. Este precio incluye labores de investigación, diseño y codificación de soluciones software. La tarifa presentada hace mención al perfil de programador junior.

El bloque de horas de testing supone alrededor de un quinto del total del tiempo de la estancia. El precio por hora de un testeador de sistemas se estima en 45 euros, por lo que se sumará una cantidad de 2700 euros al presupuesto.

De no contar con la experiencia que la supervisora aporta sobre la herramienta Microsoft Dynamics NAV, quizás se hubiera requerido de un curso de tipo *"Starter"*, que pusiese en contexto al alumno. El curso presentado en la siguiente referencia bibliográfica [2], se oferta un curso de 25 horas para iniciarse en el desarrollo de Dynamics NAV por 894 euros.

Para una instalación en el computador personal bastará con una licencia de Microsoft Windows 7 Pro para poder trabajar con el entorno de desarrollo de Dynamics NAV. Esta tiene un coste en el mercado de 45 euros. Adjuntaremos su coste al total.

De no contar con la licencia de estudiante proporcionada por el profesorado de la asignatura EI1046, Sistema de Información Integrados, La licencia de desarrollador para Microsoft Dynamics NAV podría acarrear sobre 1200 euros de gasto.

Para realizar labores de modelado de sistemas se ha empleado la herramienta MagicDraw. La licencia académica estándar esta valorada en 99 dolares, lo que supone un incremento de 84 euros en el presupuesto.

Se presenta pues en el cuadro 3.1 el total de herramientas anteriormente presentadas junto al coste económico que suponen y la suma total en euros que producen. Nótese que este cuadro esta elaborado en base a una estimación de costes suponiendo el tener que adquirir todas las herramientas tecnológicas involucradas.

Concepto	Cantidad	Precio unitario	Total
MSI i5-4210M + 4GB RAM + 240GB SSD	1	500	500
Licencia Sublime Text	1	68	68
Hora de analista de software	70	60	4200
Hora de programador junior	80	50	4000
Hora de tester de sistemas	45	60	2700
Curso de formación desarrollador MSDN	1	2400	2400
Licencia M.Windows 7 Pro	1	45	45
Licencia desarrollador MSDN	1	894	894
Licencia MagicDraw STD Academic Edition	1	84	84
Total			14891 euros

Cuadro 3.1: Costes en bruto para el proyecto de integración de sistemas

Finalmente se presenta el cuadro 3.2, el cual presupuesta el precio de la mano de obra de programador, analista y tester, junto al equipo personal usado. Este presupuesto correspondería al de una empresa que ya contase con infraestructura y licencias anteriormente citadas.

Concepto	Cantidad	Precio unitario	Total
MSI i5-4210M + 4GB RAM + 240GB SSD	1	500	500
Hora de analista de software	70	60	4200
Hora de programador junior	80	50	4000
Hora de tester de sistemas	45	60	2700
Total			11400 euros

Cuadro 3.2: Costes en bruto para el proyecto de integración de sistemas

3.4. Seguimiento del proyecto

La principal herramienta de seguimiento del proyecto han sido las reuniones informativas con frecuencia diaria, semanal y quincenal. Las reuniones diarias han servido para poder comentar aspectos relativos a la jornada de trabajo, posibles problemas que hayan podido surgir o avances significativos. Las reuniones semanales han tenido como objetivo mantener el ritmo de trabajo requerido para la presente batería de requisitos en la que se está trabajando, comentando cuáles han sido los avances obtenidos y que inconvenientes se han encontrado durante la semana. Las reuniones quincenales tienen como principal cometido, cerrar un ciclo de trabajo en el que se resumen las mejoras obtenidas para el vertical. Finalmente, se prepara la correspondiente documentación y el trabajo es presentado al cliente, quien evalúa el desarrollo realizado.

Los informes quincenales en gran medida han sido otro indicador de seguimiento del proyecto a tener en cuenta. Todos y cada uno de los informes han sido elaborados y liberados puntualmente, recibiendo las oportunas correcciones de parte del tutor. Hasta seis informes quincenales se han elaborado durante la estancia en prácticas.

Un seguimiento de horas en una hoja de LibreOffice Calc ha sido de gran utilidad para contabilizar el balance de horas realizado semanalmente, acumulación mensual y global. Hay que destacar su utilidad para detectar semanas con exceso o déficit de horas de trabajo.

Por tanto, el seguimiento del proyecto ha sido una labor continuada que ha tenido comienzo a la par que el propio proyecto y que lo ha acompañado durante cada una de las distintas fases.

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		**Integración Dinamics Nav - movildata**	**57,45 días**	**mar 06/02/18**	**jue 26/04/18**	
2		**Desarrollo de la propuesta técnica**	**57,45 días**	**mar 06/02/18**	**jue 26/04/18**	
3		**Inicio**	**1 día**	**mar 06/02/18**	**mar 06/02/18**	
4		Definición del proyecto con tutor y supervisor	0,4 días	mar 06/02/18	mar 06/02/18	
5		Definir método de trabajo	0,3 días	mar 06/02/18	mar 06/02/18	4
6		Definir formatos y estándares de trabajo	0,3 días	mar 06/02/18	mar 06/02/18	5
7		**Documentar y planificar el proyecto**	**7 días**	**mar 06/02/18**	**jue 15/02/18**	**5**
8		Revisar contexto y buscar información	3 días	mar 06/02/18	vie 09/02/18	5
9		Identificar alcance y objetivos	4 días	vie 09/02/18	jue 15/02/18	8
10		**Planificar el proyecto**	**4 días**	**jue 15/02/18**	**mié 21/02/18**	**9**
11		Definir tareas y estimar fechas	0,5 días	jue 15/02/18	vie 16/02/18	9
12		Crear diagrama de Gantt	0,5 días	vie 16/02/18	vie 16/02/18	11
13		Documentar la propuesta del proyecto	2 días	vie 16/02/18	mar 20/02/18	12
14		Librar la propuesta técnica	1 día	mar 20/02/18	mié 21/02/18	13
15		**Desarrollo técnico del proyecto**	**2 días**	**mié 21/02/18**	**vie 23/02/18**	**14**
16		**Definir requisitos del proyecto**	**2 días**	**mié 21/02/18**	**vie 23/02/18**	
17		Definir requisitos tecnológicos y de plataforma	2 días	mié 21/02/18	vie 23/02/18	
18		**Análisis**	**2 días**	**vie 23/02/18**	**mar 27/02/18**	**17**
19		Reunión refinamiento de requisitos con	1 día	vie 23/02/18	lun 26/02/18	
20		Verificación y validación de requisitos	1 día	lun 26/02/18	mar 27/02/18	19
21		**Diseño**	**3 días**	**mar 27/02/18**	**vie 02/03/18**	**20**
22		Modelado de datos de la integración	3 días	mar 27/02/18	vie 02/03/18	
23		**Desarrollo de producto**	**11 días**	**vie 02/03/18**	**lun 19/03/18**	**22**
24		Programación accesos webservices	5 días	vie 02/03/18	vie 09/03/18	
25		Programación tablas y formularios de gestión de la información	6 días	vie 09/03/18	lun 19/03/18	24
26		Test de integración de sistemas	11 días	vie 02/03/18	lun 19/03/18	22
27		**Puesta en marcha**	**8 días**	**lun 19/03/18**	**jue 29/03/18**	**26**
28		Habilitación objetos en la BBDD del cliente	2 días	lun 19/03/18	mié 21/03/18	
29		Elaboración de documentación (procedimientos	4 días	mié 21/03/18	mar 27/03/18	28
30		Entrega final y asistencia a los usuarios	2 días	mar 27/03/18	jue 29/03/18	29
31		**Documentación y presentación del TFG**	**19,75 días**	**jue 29/03/18**	**jue 26/04/18**	**30**
32		Redacción de informes quincenales	2 días	jue 29/03/18	lun 02/04/18	
33		Redacción de la memoria técnica	11,75 días	jue 29/03/18	lun 16/04/18	30
34		Entrega de la memória técnica	0 días	lun 16/04/18	lun 16/04/18	33
35		Preparación presentación oral	7 días	lun 16/04/18	mié 25/04/18	34
36		Presentación oral	1 día	mié 25/04/18	jue 26/04/18	35

Figura 3.1: Work Breakdown Structure inicial presentado en la propuesta técnica.

Capítulo 4

Análisis y diseño del sistema

Índice

4.1. Análisis del sistema 29
4.2. Diseño de la arquitectura del sistema 33

4.1. Análisis del sistema

La fase de análisis es quizás la fase que más valor tiene de todo el proyecto. Para poder hacer frente a las reuniones con el cliente y poder realizar una labor de toma de requisitos en condiciones, previamente se ha requerido de un estudio previo de la herramienta Microsoft Dynamics NAV, así como de las posibilidades de los servicios web que oferta Movildata. Además, la supervisora ha introducido al alumno en los sistemas de trabajo y tratado de la información, orientada a los ERP's y a la dinámica de funcionamiento de las empresas del transporte. La justificación de estas labores de formación previas pasa por poder contrastar de forma efectiva la viabilidad de aquella funcionalidad que pueda solicitar el cliente.

La fase de análisis da comienzo, desde el principio, con la presentación de los objetivos del proyecto por parte de la supervisora. Esta presentación junto al estudio de una propuesta previa existente de realización del proyecto, encauzan de algún modo el rumbo para poder empezar a trabajar con la recopilación de información.

Tras un breve período de formación en las herramientas que serán el núcleo de la integración de sistemas, llega el momento de realizar una toma de requisitos formal en la propia sede de la empresa de transportes colaboradora. Esta actividad tiene como objetivo recoger el total de detalles para aquella funcionalidad requerida por el equipo de operadores de tráfico. El modo de proceder consistirá en la extracción de la información mediante una entrevista oral personalizada con el apoyo de una plantilla de entrevista previamente elaborada. Esta plantilla se adjunta en el Anexo A.

Una vez recogida toda la información, esta será estudiada cuidadosamente y contrastada

con los objetivos iniciales del proyecto. Fruto de la comparación, se observa que en esencia, el proyecto contaba inicialmente con una propuesta bien definida, pues la gran mayoría de los requisitos ya constaban como parte de los objetivos del proyecto informático y por tanto no existen inconsistencias.

Tras la labor de toma de requisitos efectuada al equipo de operadores de tráfico, se elabora un documento el cual recoge (ya filtrada) la información que permite definir la batería de requisitos con la que dar comienzo al proyecto. El documento también recoge unos pocos aspectos reportados por los operadores de tráfico que escapan de los objetivos de la integración de servicios. Estos aspectos son recogidos igualmente por si fueran de interés en algún momento para el departamento de informática de ACTM. El documento citado se puede consultar en el Anexo B.

A continuación se presenta la batería de requisitos elaborada a partir del documento que resulta de la toma a los operadores de tráfico:

Introducción automatizada de kilogramos de carga/descarga de mercancía.

Este requisito de vital importancia se centra en la recogida de kilogramos de forma precisa en el momento justo del pesaje del vehículo una vez cargada la mercancía. No sirven aproximaciones o cifras con márgenes de error. El cliente que solicita un servicio de transporte espera una cantidad de mercancía exacta y esta ha de ser correctamente gestionada y facturada a fin de prestar el mejor de los servicios posible.

Recibir alertas de los conductores en el ERP.

Se plantea el poder recepcionar incidencias reportadas por los conductores, guardando relación con los servicios de transporte que realizan. En la actualidad esta labor se realiza o bien a través de telefonía o a través del canal de mensajería de Movildata. El retraso que pueda sufrir un servicio de transporte acarrea pérdidas sustanciales en la empresa, por lo que debe prestarse especial atención a aquellos servicios que presenten alguna anomalía durante su ejecución.

Gestión de mensajería desde el ERP.

Se plantea el poder gestionar la mensajería con los conductores directamente en el planificador de tráfico, sobre los propios pedidos de venta. Entre los principales inconvenientes que lo hacen inviable, destacar la sobrecarga de información en el propio planificador. La aplicación web de Movildata ya ofrece el soporte para gestionar la mensajería de texto simple y por tanto se opta a no realizar duplicidades de funcionalidad.

Control de kilometraje.

Este requisito, con un nivel de importancia similar o superior al de control de pesajes, precisa de una lectura de kilometraje automática de un vehículo tras una operación de carga o descarga de mercancía. Esta información servirá para estimar cuán rentables son los servicios de transporte prestados, así como también podría ser un indicador que permita proponer las pertinentes operaciones de mantenimiento de los vehículos de forma automática. El control de kilometrajes es necesario para poder realizar la liquidación a los transportistas de una serie de incentivos a final de mes.

Automatización de cambios de estado de los pedidos de venta en curso.

En la actualidad, los pedidos de venta varían su estado (asignado, confirmado, cargado, descargado) en base a la cumplimentación de unos campos requeridos en la propia línea de venta sobre el planificador de tráfico, junto a la posterior confirmación manual mediante un checkbox de los operadores de tráfico. El mecanismo que permite la automatización del planificador es pues, la recogida de información de forma automática durante la interacción operador de tráfico y transportista. Este patrón de interacción está descrito con mayor detalle en el Anexo C. De algún modo el presente requisito es una composición de varios de los requisitos presentados anteriormente, estructurando su resultado producido. Por tanto este requisito será clasificado como un Epic (requisito descomponible), el cual será resuelto mediante el uso de un conjunto de servicios web incluidos en el SDK de Navifleet y la aplicación Android MFLEET.

Envío de órdenes de trabajo evitando la apertura de un navegador web.

Antes de realizar operación alguna con respecto a la integración, el envío de órdenes de trabajo se efectuaba mediante una llamaba a un servicio web de Movildata de envío de mensajería a los conductores. La llamada se efectuaba mediante una URL compuesta haciendo uso de la función HYPERLINK. Esta función propia del lenguaje de desarrollo de Microsoft Dynamics NAV permite abrir una URL en un navegador externo. La llamada a la función HYPERLINK originaba la apertura de un navegador externo por cada orden de trabajo enviada con la consecuente molestia visual asociada.

Poder posicionar un vehículo en el mapa desde el ERP.

Se requiere en muchas de las ocasiones poder mostrar la posición actual para un vehículo concreto. Si bien, el gestor web de Movildata ya ofrece un posicionamiento en el mapa para un vehículo, el total de detalles que ofrece un mapa de Google Maps puede ser de gran utilidad en un momento dado para un operador de tráfico.

Mayor integración de los puntos de interés (POIs) con el ERP.

Los POIs no son más que ubicaciones o puntos específicos de interés, los cuales son gestionados por los operadores de tráfico en la aplicación web de Movildata. Un POI está definido generalmente por un título, coordenadas de posición, dirección y descripción. Entre los tipos de POIs que se gestionan encontramos: puntos de carga/descarga de mercancía, lavaderos, talleres, peajes, restaurantes, etc. Se propone una sincronización entre los puntos dados de alta en el gestor de Movildata y la información recogida en la tabla "Orígenes y destinos" del vertical del transporte a fin de poder acabar gestionando el total de POIs desde el vertical.

En esencia aquí se recogen el total de requerimientos sobre los que se fundamenta el trabajo de la integración de servicios propuesta. Algunos de estos requerimientos han sufrido en fases previas al desarrollo o incluso durante el proceso de elaboración de la funcionalidad, cambios de pequeño o mediano impacto sobre la especificación aquí presentada. Cabe recordar que de forma quincenal los resultados fruto del desarrollo efectuado, han sido contrastados con el cliente y ello en muchos casos ha supuesto cambios en la batería de requisitos presentada.

A fin de obtener un mayor nivel de comprensión de la funcionalidad requerida y como guía visual se presenta en la figura 4.1 un Diagrama de Casos de Uso que recoge el total de la funcionalidad requerida. Nótese que el diagrama no recoge la automatización de los cambios de estado para un pedido de venta en el planificador. Como se ha comentado en su descripción, este es un Epic que dará origen al *CodeUnit* "“50007 - Funciones Navifleet”, el cual requiere de un trabajo en profundidad de la documentación del SDK de Navifleet.

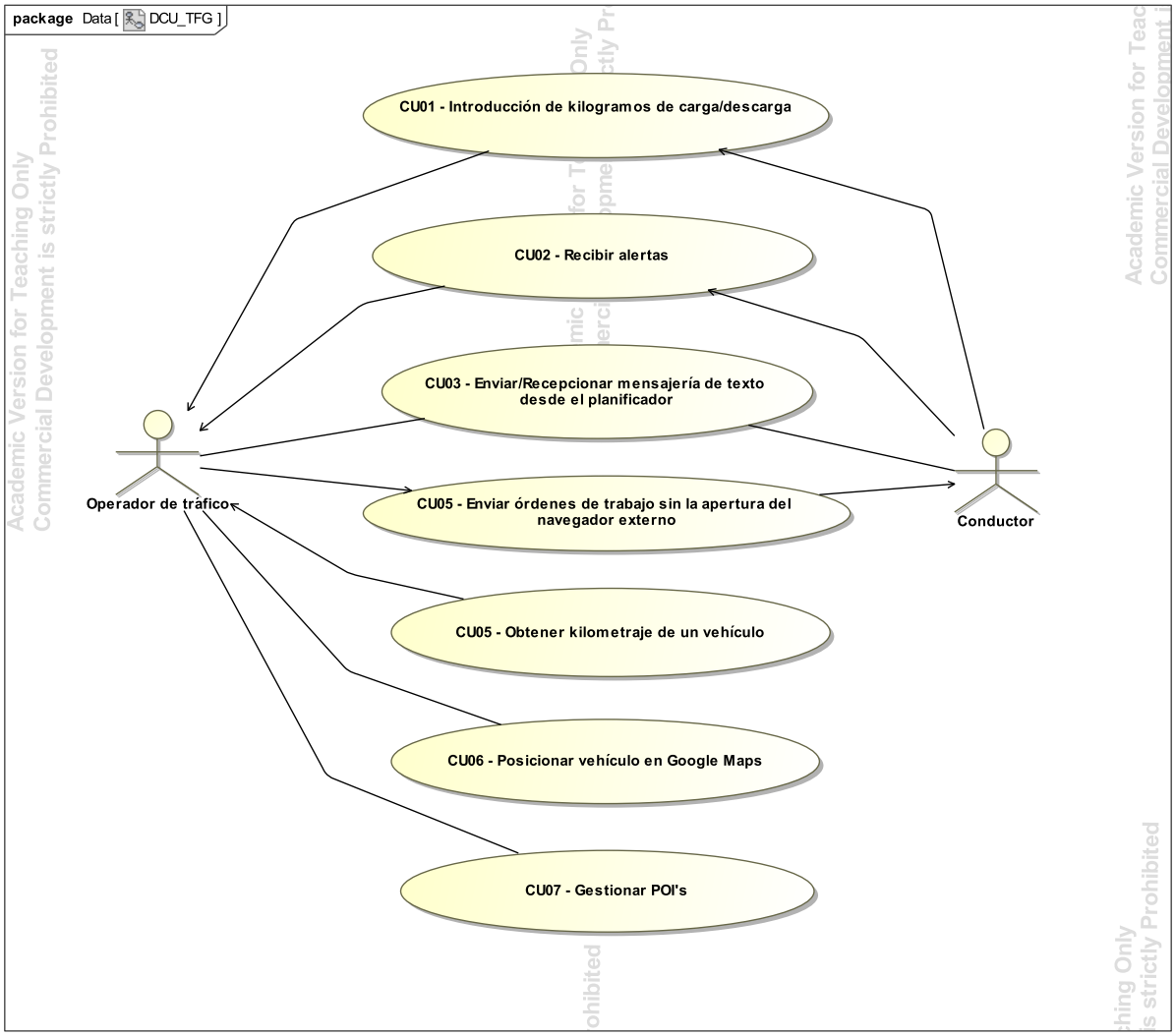

Figura 4.1: Diagrama de casos de uso de la integración de servicios.

El diagrama de casos de uso presenta siete casos de uso, cuya resolución se presentará con el nivel de detalle adecuado en el capítulo “Implementación y desarrollo”. De forma análoga, el problema planteado de la automatización del planificador tendrá también su espacio correspondiente en dicho capítulo, recogiendo los pormenores de la solución escogida para su implantación.

4.2. Diseño de la arquitectura del sistema

El software a desarrollar ha de permitir una interacción de sistemas robusta y de calidad. Microsoft Dynamics NAV tomará el rol de sistema consumidor de servicios web y los servidores de Movildata proporcionarán respuesta a las peticiones de datos que se precisen desde el ERP. En lo referente al desarrollo a realizar, este se basará en la programación de los objetos *CodeUnit* que permitan llevar a cabo las peticiones a los servicios web.

Cada *CodeUnit* tendrá su ámbito en un SDK de los presentados por Movildata para poder dar soporte a un conjunto de servicios distinto. La idea es poder incluir aquellos métodos que permitan acceder a los servicios web que el cliente considere de interés y luego poder explotarlos en el vertical del transporte de ACTM. La arquitectura software que se describe tiene su representación gráfica en la figura 4.2. En la misma puede apreciarse el conexionado de sistemas que debe permitir comunicar operadores de tráfico y transportistas haciendo uso de la aplicación Android MFLEET y el vertical del transporte.

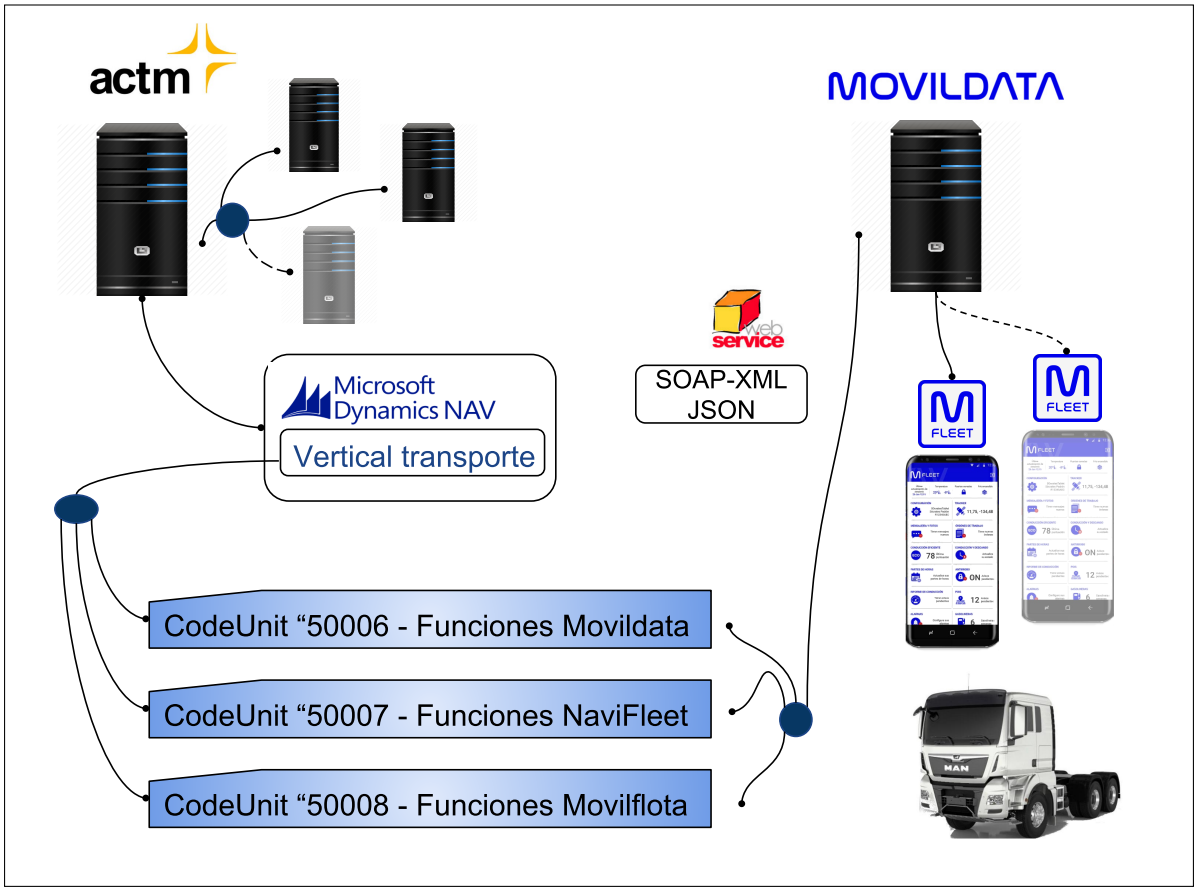

Figura 4.2: Arquitectura de comunicación entre sistemas.

Un resumen de la disposición de elementos presentada en la figura 4.2 puede describirse como un conjunto de servidores con una instalación local de Dynamics NAV, dotada del vertical de transporte, con capacidad de comunicación con los dispositivos configurados con MFLEET a través de los *CodeUnit* encargados de efectuar las llamadas a servicios web.

Con respecto a la composición interna de cada uno de los *CodeUnit*s que toman parte en este desarrollo, indicar que cada uno esta provisto de las URLs, cabeceras HTTP e información necesaria para poder actuar con total autonomía. La filosofía de construcción de estos objetos es proveer de un conjunto de métodos accesibles, sobre los cuales efectuar las llamadas a los servicios web, haciendo uso de otro subgrupo de métodos no accesibles, encargados de armar y ejecutar dichas llamadas. El *CodeUnit* "50006 - Funciones Movildata" y el "50008 - Funciones Movilflota" comparten gran parte de los métodos base que permiten efectuar las llamadas a servicios web, dado que ambos operan con el protocolo SOAP-XML. La figura 4.3 muestra la composición interna de cada *CodeUnit* a desarrollar.

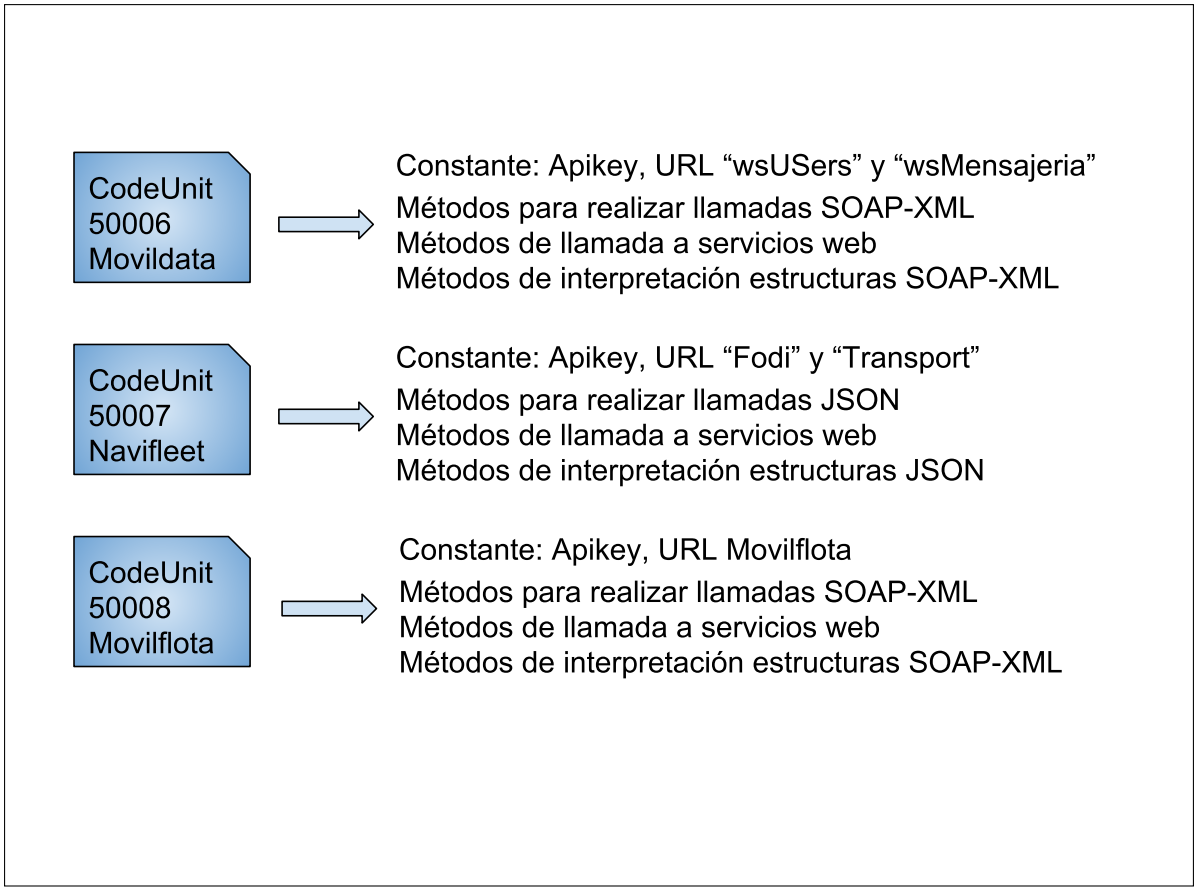

Figura 4.3: Estructura de los *CodeUnits* a desarrollar.

El conjunto de métodos, accesibles para poder realizar las llamadas a cada uno de los servicios web en el ámbito de actuación de cada uno de los *CodeUnits*, variará dependiendo de las necesidades del cliente. La descripción y detalle de estos métodos figura en el siguiente apartado.

Capítulo 5

Implementación y pruebas

Índice

5.1. Detalles de implementación 35
5.1.1. Primeras pruebas de conectividad 35
5.1.2. Desarrollo del *CodeUnit* "50006 - Funciones Movildata" 43
5.1.3. Desarrollo del *CodeUnit* "50007 - Funciones Navifleet" 50
5.1.4. Desarrollo del *CodeUnit* "Timer" 65
5.1.5. Desarrollo del *CodeUnit* "50008 - Funciones Movilflota" 66
5.1.6. Integración de servicios con Google Maps 70
5.2. Verificación y validación 70

5.1. Detalles de implementación

5.1.1. Primeras pruebas de conectividad

Una vez realizadas las labores de estudio previas sobre la documentación para integradores que ofrece la empresa Movildata y antes de efectuar una labor de análisis que recoja el total de requisitos que se van a implantar, se procede a intentar reproducir dos llamadas a servicios web de dos SDKs distintos: Movildata y Navifleet respectivamente.

La reproducción de ambas llamadas es en esencia similar, variando en la forma que los parámetros son enviados y el tipo de cabeceras que componen la llamada HTTP de tipo POST en ambos casos.

Para poder reproducir las llamadas SOAP que propone tanto el SDK de Movildata como el de Movilflota se hace uso de la herramienta Postman, la cual permite de forma fácil realizar la llamada para cada servicio web con la configuración de parámetros indicada en la documentación de Movildata. Postman devuelve para cada configuración de llamada el resultado asociado producido. De forma muy visual pueden evaluarse rápidamente los servicios web.

Para realizar las pruebas se cuenta con las credenciales y un conjunto de datos de la empresa de transportes colaboradora. Las primeras pruebas de conectividad serán realizadas sobre dispositivos reales que montan los vehículos de los transportistas. De este modo, podrá hacerse un seguimiento de las pruebas realizadas y así constatar la efectividad y el buen funcionamiento de los desarrollos.

El primero de los servicios web elegido para ser reproducido con Postman es "PostMessageSimple", un servicio web que incluye el SDK de Movildata. Este servicio permite el envío de un mensaje de texto simple a un conductor con un dispositivo tipo Garmin o de nueva generación denominado Navifleet. Los parámetros de definición del mensaje que se va a enviar con "PostMessageSimple" son:

- Tipo de mensaje (Simple, con confirmación, tipo SI/NO, con respuesta predefinida).
- IMEI del dispositivo gestionado por un conductor.
- Identificador del operador de tráfico que envía la orden.
- Cuerpo del mensaje de texto a enviar.
- ID del mensaje a enviar (Necesario para un posterior seguimiento).
- APIKEY. Clave vinculada a la empresa de transportes que realiza el seguimiento.

Para poder realizar la llamada al servicio descrito, se precisa declarar las siguientes cabeceras HTTP:

- Content Type - text/xml
- SOAPAction - http://nombredominio.com/ws/wsMensajeria/PostMessageSimple

Con esta configuración de parámetros dentro de una estructura XML propia del protocolo SOAP, cumplimentando debidamente los elementos SOAP-Envelope, SOAP-Header y SOAP-Body, se compone el primer envío de orden y este es ejecutado en Postman como muestra la figura 5.1. Para profundizar en la comprensión del protocolo SOAP se recomienda la visita a la siguiente referencia bibliográfica [10].

Como resultado de la llamada efectuada, el servidor de Movildata puede contestar con un código de error o bien indicar que el proceso de envío ha tenido éxito, adjuntando en este caso el nombre del conductor en cuestión al que se le ha enviado el mensaje. El mensaje que se recoge tiene este aspecto:

"Se ha enviado el mensaje al vehículo 345-F. Martínez. No se tiene registros de conectividad, es posible que el Garmin esté desconectado u opción TASK desactivada en sistema."

El mensaje que se recoge de vuelta hace referencia a que posiblemente el conductor al que se le ha enviado el mensaje, monta en su vehículo un dispositivo más moderno que los de la marca Garmin (dotado por ejemplo del sistema Navifleet).

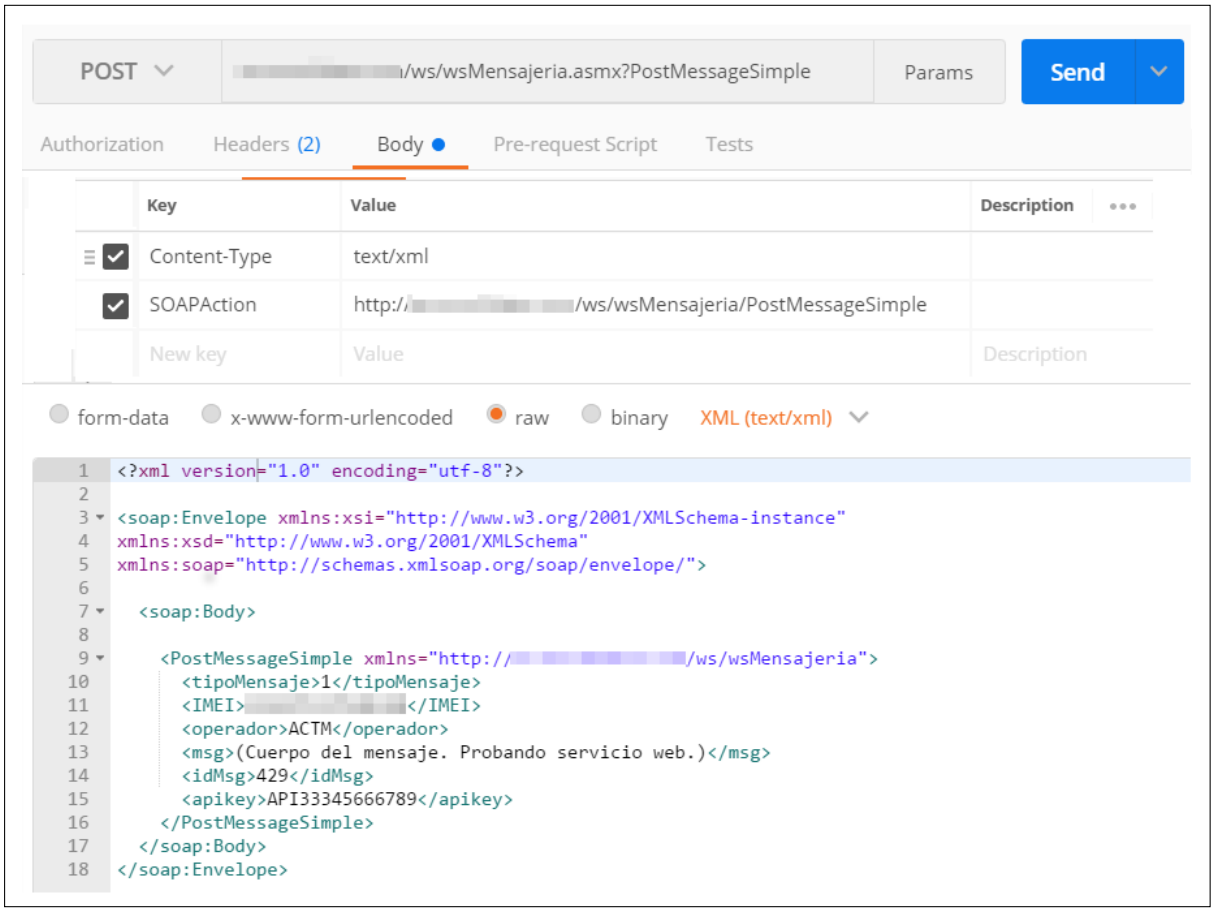

Figura 5.1: PostMessageSimple Postman.

Una vez se tiene claro el planteamiento del servicio web, se procede a codificar la llamada al servicio web "PostMessageSimple" en el ERP Microsoft Dynamics NAV. Para la realización de estas pruebas de conectividad se habilita un *CodeUnit* de prueba. Debido a que el lenguaje C/AL es un lenguaje de manejo de datos, este requerirá de unas librerías externas .NET para poder comunicarse con los servidores de Movildata mediante el uso de los servicios web.

Es en este punto donde se realiza el primero de los aprendizajes importantes en cuanto a la filosofía de trabajo de los ERPs. La tendencia a querer desarrollar nuevas funcionalidades desde 0 o la creación de nuevas tablas u objetos sin haber consultado las que ya hay en el estándar, es un error típico que se suele cometer entre programadores no iniciados en el mundo de los ERP.

La estrategia propuesta por la supervisora, la cual nos ahorrará una gran cantidad de tiempo y esfuerzo pasa por realizar un estudio sobre la funcionalidad del estándar de Microsoft con el fin de encontrar algún módulo o desarrollo que incluya interacción con sistemas externos mediante servicios web. De entre todos los *CodeUnits* que componen el estándar de Microsoft Dynamics NAV y los propios del desarrollo del vertical del transporte de ACTM, nos quedamos con la referencia del *CodeUnit* que incluye el proceso automático de envío de los datos de las facturas emitidas y recibidas al "Suministro Inmediato de Información de la Agencia Tributaria", más conocido por las siglas SII.

Este *CodeUnit* nos brinda un interesante punto de partida, dado que en el código fuente que incluye se puede apreciar un patrón de envío de información mediante una estructura de tipo SOAP-XML. Este esquema de trabajo es en gran medida lo que se necesita para poder reproducir las llamadas a servicios web incluidas en los SDKs de Movildata y Movilflota.

La figura 5.2 muestra un fragmento de código capaz de realizar la llamada a "PostMessageSimple" fruto de la adaptación de la solución que presentaba el *CodeUnit* que permite el envío al SII. De este modo ya se cuenta con una solución software del lado de Dynamics NAV, que permite el envío del mensaje simple, obteniendo los mismos resultados tal y como lo hacía la interfaz gráfica Postman. El resultado que devuelve el servidor, tras efectuar la llamada al servicio web, es almacenado en la variable de texto "ResponseTxt". Únicamente necesitaríamos hacer uso de la función "MESSAGE" propia del lenguaje C/AL e imprimirla por pantalla para cerciorarnos de que la llamada al servicio web ha tenido éxito. Esto provocará que en el Cliente Adaptado a Roles (RTC), aparezca un diálogo con el mensaje devuelto por el servidor de Movildata.

Algunas de las variables de tipo DotNet que se requieren para la creación de la llamada al servicio web "'PostMessageSimple", se recogen en el cuadro 5.1.

Nombre	Tipo	Libreria .NET
WebRequest	DotNet	System.Net.WebRequest.'System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'
HttpWebRequest	DotNet	System.Net.HttpWebRequest.'System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'
HttpRequestHeader	DotNet	System.Net.HttpWebRequest.'System,Version=4.0.0.0,Culture=neutral,PublicKeyToken=b77a5c561934e089'
RequestStream	DotNet	System.IO.Stream.'mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'
Encoding	DotNet	System.Text.Encoding.'mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'
ByteArray	DotNet	System.Array.'mscorlib, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'
Uri	DotNet	System.Uri.'System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'
HttpWebResponse	DotNet	System.Net.HttpWebResponse.'System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'
StatusCode	DotNet	System.Net.HttpStatusCode.'System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'

Cuadro 5.1: Variables DotNet asociadas a servicios web.

```
WebServiceUrl := 'http://nombredominio.com/ws/wsMensajeria.asmx?PostMessageSimple';

llamadaSOAP := '<?xml version="1.0" encoding="utf-8"?>'+
 '<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema'+
 'xmlns:xsd="http://www.w3.org/2001/XMLSchema" ' +
 'xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">'+
 '<soap:Body>'+
   '<PostMessageSimple xmlns="http://nombredominio.com/ws/wsMensajeria">'+
     '<tipoMensaje>1</tipoMensaje>'+
     '<IMEI>'+'IM123456'+'</IMEI>'+
     '<operador>'+'MDV'+'</operador>'+
     '<msg>'+'Mensaje de prueba'+'</msg>'+
     '<idMsg>'+'523'+'</idMsg>'+
     '<apikey>'+'APIK123456'+'</apikey>'+
   '</PostMessageSimple>'+
  '</soap:Body>'+
 '</soap:Envelope>';

HttpWebRequest := WebRequest.Create(Uri.Uri(WebServiceUrl));
HttpWebRequest.ContentType := 'text/xml';
HttpWebRequest.Headers.Add('SOAPAction','http://nombredominio.com/ws/wsMensajeria/
    PostMessageSimple');
HttpWebRequest.Method := 'POST';


ByteArray := Encoding.UTF8.GetBytes(llamadaSOAP);
HttpWebRequest.ContentLength := ByteArray.Length;
IF NOT TryCreateRequestStream(HttpWebRequest,RequestStream) THEN BEGIN
ERROR(NoConnectionErr,'');
EXIT;
END;

RequestStream.Write(ByteArray,0,ByteArray.Length);

IF NOT TryGetWebResponse(HttpWebRequest,HttpWebResponse) THEN BEGIN
ERROR(NoResponseErr,'');
EXIT;
END;

StatusCode := HttpWebResponse.StatusCode;
StatusDescription := HttpWebResponse.StatusDescription;
ResponseTxt := ReadHttpResponseAsText(HttpWebResponse);
IF NOT StatusCode.Equals(StatusCode.Accepted)
AND NOT StatusCode.Equals(StatusCode.OK) THEN BEGIN
ERROR(
  STRSUBSTNO(CommunicationErr,StatusDescription),
  ResponseTxt);
EXIT;
END;
```

Figura 5.2: Codificación de “PostMessageSimple” en lenguaje C/AL.

En este punto la prueba de conectividad entre Microsoft Dynamics NAV y los servidores de Movildata ha resultado ser exitosa. A partir de ahora, puede reproducirse el envío de mensajes simples tal y como se está haciendo en el vertical del transporte en la actualidad. Sin embargo esta llamada a "PostMessageSimple" presenta una mejora sustancial con respecto al modo en el que anteriormente se efectuaba. La llamada al servicio web se realizaba mediante una URL que concatenaba todos los parámetros de la llamada "PostMessageSimple", componiendo una orden de trabajo o el envío de algún mensaje compuesto con información complementaria al servicio de transporte a realizar. Esta URL compuesta, posteriormente era ejecutada por la función "HYPERLINK" la cual produce una apertura de un navegador externo y realiza el envío correspondiente. Esto originaba obviamente una molestia justificada al operador de turno, quien tenía que ir cerrando varias de estas ventanas innecesarias ajenas al ERP.

La primera de las mejoras implantables en el vertical del transporte llega con la substitución de la llamada a la función "HIPERLINK", por la llamada al servicio web presentado en la figura 5.1. Este servicio será posible integrarlo junto a otros al finalizar el desarrollo del *CodeUnit* "50006 - Funciones Movildata".

El resto de servicios web que recoge el SDK de Movildata tienen una estructura prácticamente idéntica, requiriendo en cada caso, una configuración de parámetros distinta. También variará la URL de llamada, la cual apuntará a "wsUsers" o "wsMensajería" según esté clasificado el servicio web en cuestión. Finalmente deberemos fijarnos en la cabecera HTTP "SOAPAction" la cual ha de corresponder exactamente con el servicio web al que se quiere efectuar la llamada.

Cuando se precisa hacer una prueba de un determinado servicio web con Postman no es necesario tener que teclear por completo sus estructuras SOAP-XML. Se dispone de una descripción formal de métodos en línea para cada SDK. Accediendo a la descripción de cada servicio web, se presenta el cuerpo de la petición HTTP junto al cuerpo de una respuesta, cuando la llamada SOAP ha tenido éxito. Para realizar una prueba de un servicio web en concreto basta con copiar y pegar la estructura de la llamada y substituir cada etiqueta por el dato correspondiente en el cuerpo del mensaje. La figura 5.3 muestra el detalle que ofrece la definición en línea para el servicio web "PostMessageSimple".

Una vez se tiene claro el patrón de llamada a los servicios web SOAP-XML, queda otra prueba de conectividad que realizar. Esta prueba consiste en realizar sobre los servidores de Movildata una llamada del SDK de Navifleet, cuyos parámetros deben conformar un cuerpo con estructura JSON.

El cambio sustancial con respecto al modo de operar con las llamadas SOAP es que ahora el cuerpo del mensaje tiene un modo de estructurar la información distinta. Este formato de composición de la información es un estándar en la actualidad para el envío de la información por red. Entre las ventajas a destacar está su ligereza. JSON se centra únicamente en el envío de datos, dejando de lado la gran cantidad de metadatos propia del protocolo SOAP-XML. Con respecto a las cabeceras HTTP únicamente se requiere declarar el tipo de contenido que va a enviarse. La URL debe apuntar al destino del servicio web.

La figura 5.4 muestra la codificación que permite la cancelación de una orden de trabajo según las directrices del SDK de Navifleet. Por tanto, se procede a efectuar la llamada del mismo modo que se ha hecho con "PostMessageSimple", adaptando el código a las exigencias de la definición del servicio web. Esta vez no se cuenta con una definición en línea para los

```
POST /wsMensajeria.asmx HTTP/1.1
Host: [redacted]
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://[redacted]/ws/wsMensajeria/PostMessageSimple"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  <soap:Body>
    <PostMessageSimple xmlns="http://[redacted]/ws/wsMensajeria">
      <tipoMensaje>int</tipoMensaje>
      <IMEI>string</IMEI>
      <operador>string</operador>
      <msg>string</msg>
      <idMsg>int</idMsg>
      <apikey>string</apikey>
    </PostMessageSimple>
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema
  <soap:Body>
    <PostMessageSimpleResponse xmlns="http://[redacted]/ws/wsMensajeria">
      <PostMessageSimpleResult>string</PostMessageSimpleResult>
    </PostMessageSimpleResponse>
  </soap:Body>
</soap:Envelope>
```

Figura 5.3: Documentación en línea para “PostMessageSimple”.

servicios web de Navifleet, sin embargo el propio SDK adjunta varios ejemplos de llamadas con distintos conjuntos de datos de prueba. Esto será suficiente para poder ahondar en las posibilidades de configuración que nos brinda el SDK de Navifleet.

El código incluido en la figura 5.4 omite gran parte de la estructura que permite realizar la cancelación de una orden para el sistema de trabajo Navifleet, sin embargo el código que se ha omitido es prácticamente el mismo que el de la figura 5.1. Como se ha anunciado ya, hay gran compartición de código entre los dos tipos de llamadas.

Para la llamada al servicio de cancelación de órdenes de trabajo, si la APIKEY fuera correcta y el código de la orden correspondiera con una orden de trabajo dada de alta en el cliente propietario de la Apikey, tendríamos una estructura JSON anidada indicando que todo ha ido bien. Para cualquier otra casuística, recibiríamos la misma estructura con los correspondientes mensajes de error y un mensaje explicativo indicando el error producido en la operación de cancelación de la orden.

```
WebServiceUrl := 'https://nombredominio.com/md.appmobile.webapi/ERP/FODI_Order';

JsonBody := '{"sAPIKEY":"API334754",'+
'"sAction":"C",'+
'"nIdForm":"417",'+
'"aoOrder":[{"sCode": "PV14-001"}]}';


HttpWebRequest := WebRequest.Create(Uri.Uri(WebServiceUrl));
HttpWebRequest.ContentType := 'application/json';
HttpWebRequest.Method := 'POST';
```

Figura 5.4: Codificación del servicio web de cancelación de órdenes de trabajo.

Figura 5.5: Respuesta del servidor al cancelar la orden "PV14-001".

La figura 5.5 muestra la respuesta que obtendríamos en la herramienta Postman al llamar al servicio web de cancelación de órdenes del SDK de Navifleet. Como puede apreciarse nos es devuelto un mensaje de error, dado que la orden de trabajo no existía para el cliente con la Apikey que intentaba cancelarla.

Es importante cerciorarse de que en ambos casos, trabajando con el estándar SOAP o con el formato de texto ligero JSON, cuando se efectúa una llamada a un servicio web determinado no se recepciona un único dato o conjunto de datos que se requieren, sino que es devuelta una respuesta SOAP o JSON en cada caso. A partir de la respuesta devuelta, será faena del integrador de servicios extraer del mensaje de retorno aquella información de interés. En lo que al proyecto se refiere, se han diseñado varios intérpretes que dan solución localizada a las exigencias de cada uno de los servicios web con los que se han trabajado.

5.1.2. Desarrollo del *CodeUnit* "50006 - Funciones Movildata"

Una vez realizadas las pruebas de conectividad es hora de comenzar la fase de desarrollo de la integración de servicios. En base a los requisitos que atañen al SDK de Movildata se decide elaborar un bloque de código en C/AL en forma de objeto *CodeUnit*. Este ha de permitir de forma ágil habilitar los servicios web seleccionados del SDK de Movildata en Dynamics NAV. Para ello se propone elaborar el *CodeUnit* siguiendo una arquitectura software previamente definida. Una primera línea de métodos marcados como locales y por tanto no accesibles desde el exterior del *CodeUnit*, deberán permitir realizar una llamada SOAP-XML. Por otro lado, existirá una segunda línea de métodos, esta vez accesibles, que permitirán efectuar una llamada a un servicio web en concreto, recibiendo los parámetros necesarios en cada caso y retornando los items de información requeridos. En la figura 5.6 puede verse el ámbito de los métodos descritos.

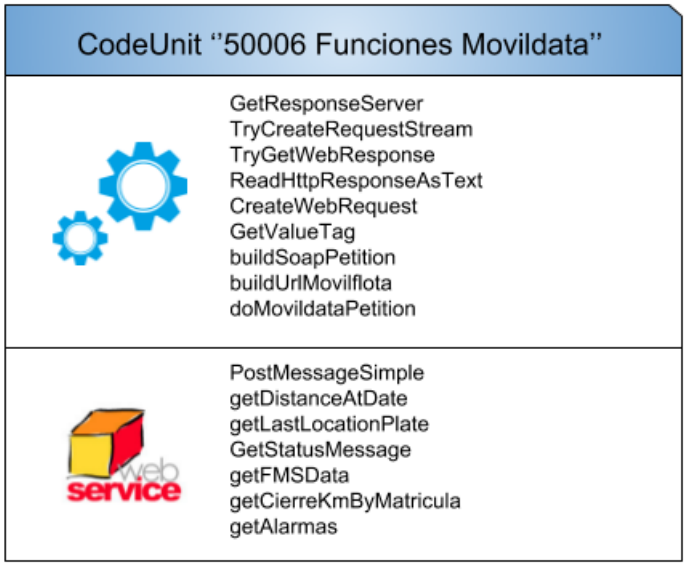

Figura 5.6: Ámbito de los métodos del *CodeUnit* "50006 - Funciones Movildata".

Con la descripción efectuada de la arquitectura software que presentará este objeto *CodeUnit*, el cuadro 5.5 muestra la relación de métodos que contiene el desarrollo efectuado. En el cuadro puede apreciarse detalles como el nombre del método, una breve descripción de su cometido y los parámetros necesarios de invocación. El cuadro presenta en primer lugar los métodos que, combinados entre sí, permitirían la ejecución de una llamada a cualquiera de los servicios web; en las últimas posiciones del cuadro se muestran aquellos que permitirían llamar a un determinado servicio.

Nombre función	Descripción	Parámetros	Desarrollo
GetResponseServer	Retorna un dato concreto (usando una etiqueta como tercer parámetro), o estructura SOAP-XML si se introduce la cadena vacía como tercer parámetro.	HttpWebRequest DotNet Mensaje Text Item Text	Propio
TryCreateRequestStream	Establecer un flujo de envío de datos para una petición HttpWebRequest	HttpWebRequest DotNet RequestStream DotNet	Estándar
TryGetWebResponse	Vuelca en HttpWebResponse, el resultado de la petición HttpWebRequest	HttpWebResponse DotNet HttpWebRequest DotNet	Estándar
ReadHttpResponseAsText	Devuelve el flujo de respuesta intepretado como una cadena de texto.	HttpWebResponse DotNet	Estándar
CreateWebRequest	Prepara una petición HTTP POST, mediante una URL y una acción SOAP.	Url Text Action Text	Propio
FormatDate	Formatea un tipo Date en una cadena de texto con el formato: 'yyyy-mm-dd'	Date Text	Propio
GetValueTag	Extrae un dato contenido en un cuerpo de respuesta con formato SOAP-XML.	String Text Tag Text	Propio
ShowMarkerOnMap	Muestra un mapa de Google Maps con un pin indicando la posición exacta de un item.	LatLang Text	Propio
buildSoapPetition	Genera un cuerpo de llamada SOAP-XML de forma dinámica en base a un array de parámetros.	vector (Dim 20) Text	Propio
buildUrlMovilflota	Compone el Url necesario, en base al tipo de servicio web requerido.	typeCalling Text method Text	Propio
replaceAll	Reemplaza una cadena de texto contenido dentro de otra cadena de texto para todas las apariciones.	string Text substr Text substrNew Text	Propio
doMovildataPetition	Realiza la llamada a un servicio web, recibiendo un vector de parámetros que definirán la llamada a servicios web.	vector (Dim 20) Text	Propio
PostMessageSimple	Realiza un envio de mensaje simple a un conductor determinado.	tipoMensaje Integer IMEI Text operador Text msg text idMsg Integer	Propio
getDistanceAtDate	Obtención del kilometraje de un determinado vehículo en una fecha concreta.	matricula Text fecha Date	Propio
getLastLocationPlate	Obtención de la última posición registrada (latitud y longitud) para un vehículo.	Plate text	Propio
GetStatusMessage	Obtención del estado para un mensaje enviado a un conductor. (Recoge la respuesta si existe)	idMen Integer	Propio
getFMSData	Obtención de los datos generados por el FMS-CANBUS para aquellos vehículos con el servicio en activo.	desde Date hasta Date matricula Text	Propio
getCierreKmByMatricula	Obtención de kilometraje realizado entre fechas para un vehículo determinado.	desde Date hasta Date matricula Text	Propio
getAlarmas	Obtención de las alarmas producidas en la flota de vehículos del cliente.	desde Date hasta Date	Propio

Cuadro 5.2: Funciones incluidas en el *CodeUnit* "50006 - Funciones Movildata"

Para tratar de comprender la filosofía de trabajo del *CodeUnit* "50006" la figura 5.7 presenta el flujo de llamada de métodos que requeriría la ejecución del servicio web "PostMessageSimple". De forma análoga, la llamada a los demás servicios web originan el mismo patrón de flujo de llamadas a métodos descrito, variando obviamente el primero de ellos: el método llamador.

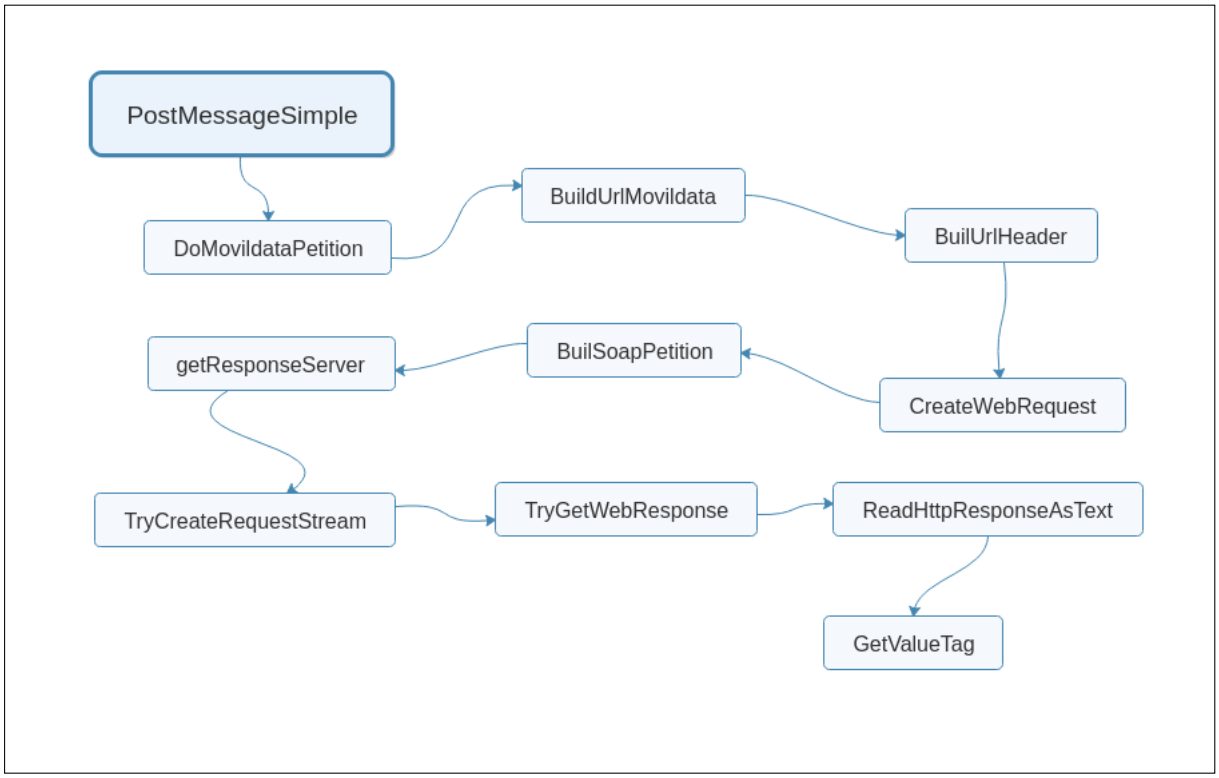

Figura 5.7: Flujo de llamadas a métodos producido por "PostMessageSimple".

Los métodos encargados de hacer la llamada a un determinado servicio web concreto tienen en común el uso de un vector de cadenas de texto global, el cual componen de un modo parecido, variando únicamente en los parámetros característicos de cada servicio. La comprensión de la estrategia del vector compuesto es vital para poder comprender cómo construir las llamadas SOAP-XML de forma dinámica. La figura 5.8 muestra la estructura del vector de cadenas citado. Un aspecto que puede resultar curioso es la numeración de las componentes del vector. En el entorno de desarrollo de Dynamics NAV los vectores comienzan en la posición 1 y no en 0.

La primera posición aloja el nombre del servicio web tal y como lo describe el SDK en línea. La segunda componente indica, mediante un carácter 'U' o 'M', si el servicio web pertenece al grupo de servicios "wsUsers" o "wsMensajeria" respectivamente. La tercera posición del vector alojará la etiqueta asociada al valor que se desea rescatar de la estructura de retorno SOAP-XML.

El resto de componentes del vector son parejas consecutivas que se corresponden con la clave y valor de un parámetro, permitiendo la definición de una serie de parámetros asociados al servicio web en cuestión. Una vez se tiene el vector construido, este servirá como estructura de referencia para componer un cuerpo de una petición SOAP con el método "buildSoapPetition".

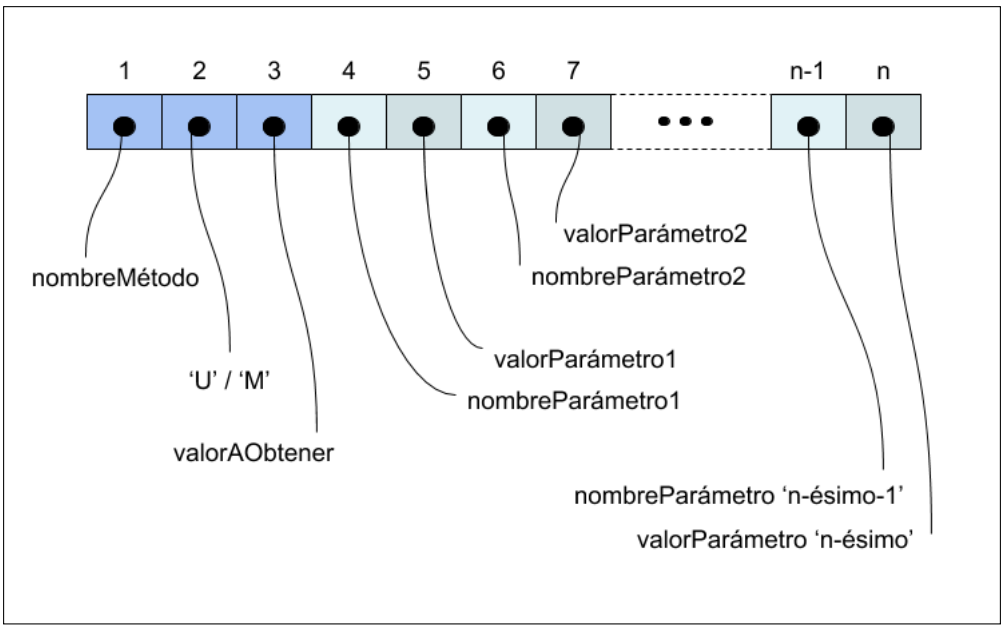

Figura 5.8: Estructura del vector de llamada SOAP.

Sobre los métodos que componen el presente *CodeUnit*, se va a proceder a describir los detalles de aquellos que, debido a su complejidad, puede que no sea suficiente la información que se recoge en la descripción que se incluye para cada uno de ellos en el cuadro 5.5.

GetResponseServer.

Los parámetros de llamada para este método son: el objeto DotNet HttpWebRequest, el cuerpo del mensaje a enviar y el item que se desea rescatar del cuerpo de respuesta SOAP que devolverá el servidor.

La manera de operar del método es preparar el envío del mensaje transformándolo en un array de Bytes para su posterior envío mediante un flujo de envío de datos. La siguiente acción es recoger con la apertura de un flujo de datos de respuesta el resultado de la llamada SOAP. El objeto de respuesta registra el estado de la operación. Dependiendo del código de estado, se lanza una excepción indicando que no ha tenido éxito la llamada al servidor, o se continúa con la interpretación del resultado. Si se ha declarado en el tercer parámetro un valor distinto de la cadena vacía, el método "GetValueTag" tratará de extraer el valor asociado al tercer parámetro indicado, de lo contrario, se devolverá la estructura de la respuesta SOAP al completo.

buildSoapPetition.

Este método construye un cuerpo SOAP-XML a partir de los valores que contiene el vector de cadenas anteriormente descrito, configurando una llamada a un servicio web concreto. El código que contiene la figura 5.9 amplía el nivel de detalle del mecanismo, que permite la construcción dinámica de un cuerpo SOAP. Como valor de retorno, el método devuelve el cuerpo construido en forma de cadena de texto.

doMovildataPetition.

En primer lugar este método llama a "CreateWebRequest", el cual recibe la URL y la cabecera de la acción SOAP a ejecutar. En segundo lugar se crea el cuerpo del mensaje gracias a una llamada a "buildSoapPetition". Finalmente se retorna el resultado de la llamada al método "GetResponseServer".

CreateWebRequest.

La secuencia de pasos en este caso es: inicializar un objeto DotNet llamado "HttpWebRequest" con la URL del servicio web, indicar que se efectuará una operación de tipo POST, declarar el contenido de tipo texto/xml y añadir la cabecera propia de la acción SOAP.

```
 soapRequestBody:= '<?xml version="1.0" encoding="utf-8"?>'+
'<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" '+
'xmlns:xsd="http://www.w3.org/2001/XMLSchema" '+
' xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">'+
'<soap:Body>'+'<'+FORMAT(vector[1]);

IF vector[2] = 'M' THEN
  soapMessage := soapMessage + mensajeria
ELSE
  soapMessage := soapMessage + users;

soapMessage := soapMessage +'<apikey>'+APIKEY+'</apikey>';
i := 4;

//Composicion de los parametros

WHILE vector[i] <> '' DO BEGIN
  soapMessage := soapMessage + '<'+FORMAT(vector[i])+'>'+
  FORMAT(vector[i+1])+'</'+FORMAT(vector[i])+'>';
  i:= i+2;
END;

soapMessage:=soapMessage+'</'+FORMAT(vector[1])+'>'+
'</soap:Body>'+
'</soap:Envelope>';
```

Figura 5.9: Construcción dinámica de una llamada SOAP.

Nótese que el ámbito de las variables DotNet que permiten el envío de servicios web, son globales con respecto al *CodeUnit* al igual que el vector de cadenas anteriormente descrito. Este detalle pretende ser de ayuda en la comprensión de la descomposición del patrón de llamada a servicios web descrito en la sección 5.1.1.

En esencia la configuración de métodos que permiten efectuar una llamada a servicios web descrita para este *CodeUnit* servirá más adelante para la confección del *CodeUnit* "50008 - Funciones Movilflota". El trabajo con el protocolo SOAP es prácticamente el mismo y la estrategia de utilizar dos capas de métodos que descomponen del patrón de llamada presentado, será la misma. Esta filosofía de descomposición del patrón de llamada a servicios web basado en dos líneas de métodos, también la implementará el *CodeUnit* "50007 - Funciones Navifleet", sin embargo el trabajo efectuado variará con respecto a la naturaleza del formato de datos JSON.

"PostMessageSimple" ha sido el ejemplo vehicular que ha permitido describir el trabajo sobre uno de los métodos accesibles que permiten realizar la llamada a un servicio web del SDK de Movildata. Sin embargo el desarrollo del presente *CodeUnit* incluye hasta siete métodos de acceso a servicios que han sido incorporados en base a los requisitos planteados por el cliente.

En resumen los métodos del *CodeUnit* "50006 - Funciones Movildata" que permiten la llamada a los servicios web escogidos del SDK de Movildata son los siguientes:

PostMessageSimple. El envío de un mensaje simple de texto a un conductor es posible mediante la llamada a este servicio web. Reemplazará el envío de órdenes de trabajo e información complementaria, el cual se hacía mediante URLs compuestas y la llamada a la función "HYPERLINK". Los mensajes enviados pueden ser rastreados por medio del id declarado en el momento de la definición de envío de mensaje. Se podrá escoger entre enviar el modelo simple de envío o cualquiera de los descritos en el SDK, variando el tipo mediante la definición de parámetros en el momento del envío de mensaje.

getDistanceAtDate. Aportando la matrícula de un vehículo y una fecha, el servicio web retorna la distancia recorrida en kilómetros registrada para esa fecha.

getLastLocationPlate. Aportando la matrícula de un vehículo, este servicio web retorna la última localización registrada. Concretamente devuelve una concatenación de latitud y longitud separadas por una coma en formato de cadena de texto.

GetStatusMessage. Para un mensaje enviado es posible realizar labores de rastreamiento. Si un conductor recepciona el mensaje en su dispositivo, este se registra y "GetStatusMessage" devuelve esa información a través del id del mensaje enviado. De igual modo, si el conductor lo lee o responde al mensaje, el servicio web registrará estas acciones además de adjuntar la respuesta (en caso de existir).

getFMSData. Algunos vehículos cuentan con el dispositivo de control de flotas conexionado a la interfaz de sistemas de gestión de flotas (FMS). Esto permite acceder a información interna del vehículo, la cual por lo general únicamente está disponible en el sistema de bus interno del vehículo. Este servicio web devuelve un conjunto de datos para un vehículo entre dos fechas. Sobre los datos que es capaz de retornar "getFMSData" destacamos kilometrajes, control de combustible en el tanque y revoluciones por minuto.

getCierreKMByMatricula. Este servicio web permite averiguar a través de la matrícula de un vehículo y dos fechas el kilometraje total efectuado entre ellas.

getAlarmas. Devuelve un conjunto de datos correspondiente al total de alarmas producidas por una flota de vehículos entre dos fechas pasadas como parámetro. Los tipos de alarmas que se registrarán variarán en función de los tipos de alarmas dadas de alta para hacer seguimiento.

Como curiosidad, comentar que este *CodeUnit* incluye una función, "ShowMarkerOnMap", que permite mostrar en un mapa de Google Maps un Marker indicando una posición concreta. Este método tiene sentido llamarlo tras haber obtenido la latitud y longitud para un vehículo (funcionalidad propia de "getLastLocationPlate") o invocando la función con la geolocalización propia de un POI. La forma en la que este método realiza la acción descrita, se debe a la composición de una URL tal y como lo específica la guía para desarrolladores de la API de Google Maps [1].

5.1.3. Desarrollo del *CodeUnit* "50007 - Funciones Navifleet"

La motivación del desarrollo del *CodeUnit* "50007 - Funciones Navifleet" surge de la necesidad de poder automatizar la recepción de un conjunto de datos característicos asociados a un servicio de transporte. Esta necesidad viene recogida explícitamente en la batería de requisitos propuesta por el cliente y requiere pues de una solución robusta que permita además, poder actualizar los estados para un pedido de venta de forma semi-automática en el planificador de tráfico.

Durante las fases de análisis del proyecto se ha estudiado con detenimiento la mejor de las soluciones posibles en base a la toma de requisitos practicada sobre el equipo de operadores de tráfico de la empresa de transporte colaboradora. Los anexos B y C recogen parte de ese estudio realizado. En lo referente a los documentos citados, debe extraerse la importancia de la automatización de un patrón de interacción básico existente entre un operador de tráfico y un transportista. Cabe destacar que para la propuesta de automatización del planificador de tráfico descrita al final del anexo C, esta es finalmente desechada. El planteamiento descrito en la propuesta ha sido descartado básicamente por no explotar en profundidad la funcionalidad que ya implanta de por sí el sistema de gestión de órdenes de Navifleet. La propuesta se basaba en una configuración estática de envío de mensajes simples a los conductores, más su posterior recepción e interpretación del lado de los operadores de tráfico. Del anexo citado se ha de extraer el concepto abstracto del patrón de interacción que solicitan los operadores de tráfico, tal y como muestra la figura 5.10. Como ya se ha anunciado, la solución a la automatización solicitada por los operadores de tráfico vendrá dada por la combinación del uso de un conjunto de servicios web del SDK de Navifleet y la aplicación Android MFLEET.

Tras profundizar en el estudio del SDK de Navifleet y consultar al personal técnico de Movildata las posibilidades que esta herramienta ofrece, se procede a realizar una prospección de los servicios incluidos en el documento técnico.

Para ello, se procede a la instalación de la aplicación Android MFLEET en los terminales móviles del personal informático de ACTM. MFLEET puede encontrarse fácilmente en Google Play. El siguiente paso sería la configuración de la aplicación móvil para poder interactuar con los servicios web de Navifleet. El primero de los datos requeridos es la introducción por parte del conductor de la matrícula del vehículo a su cargo. En este punto la aplicación ya reconoce al conductor. Para acabar de autenticarse, el conductor debe introducir un código PIN asociado a su cuenta de usuario, asegurando de este modo la autenticidad de la persona. Si los datos introducidos son los correctos, el sistema MFLEET estaría preparado para recepcionar órdenes de trabajo y hacer uso de la funcionalidad que brindan los distintos módulos que componen la aplicación. La figura 5.11 muestra el aspecto de la aplicación, siempre y cuando esta cuente con la activación de los módulos, los cuales pueden variar en función de los servicios contratados en Movildata.

De entre los módulos de la aplicación MFLEET destacamos:

- Configuración. Aspectos de autenticación, información de la propia aplicación y parámetros de configuración.
- Mensajería y fotos. Recoge la mensajería simple, aportando la posibilidad al conductor de

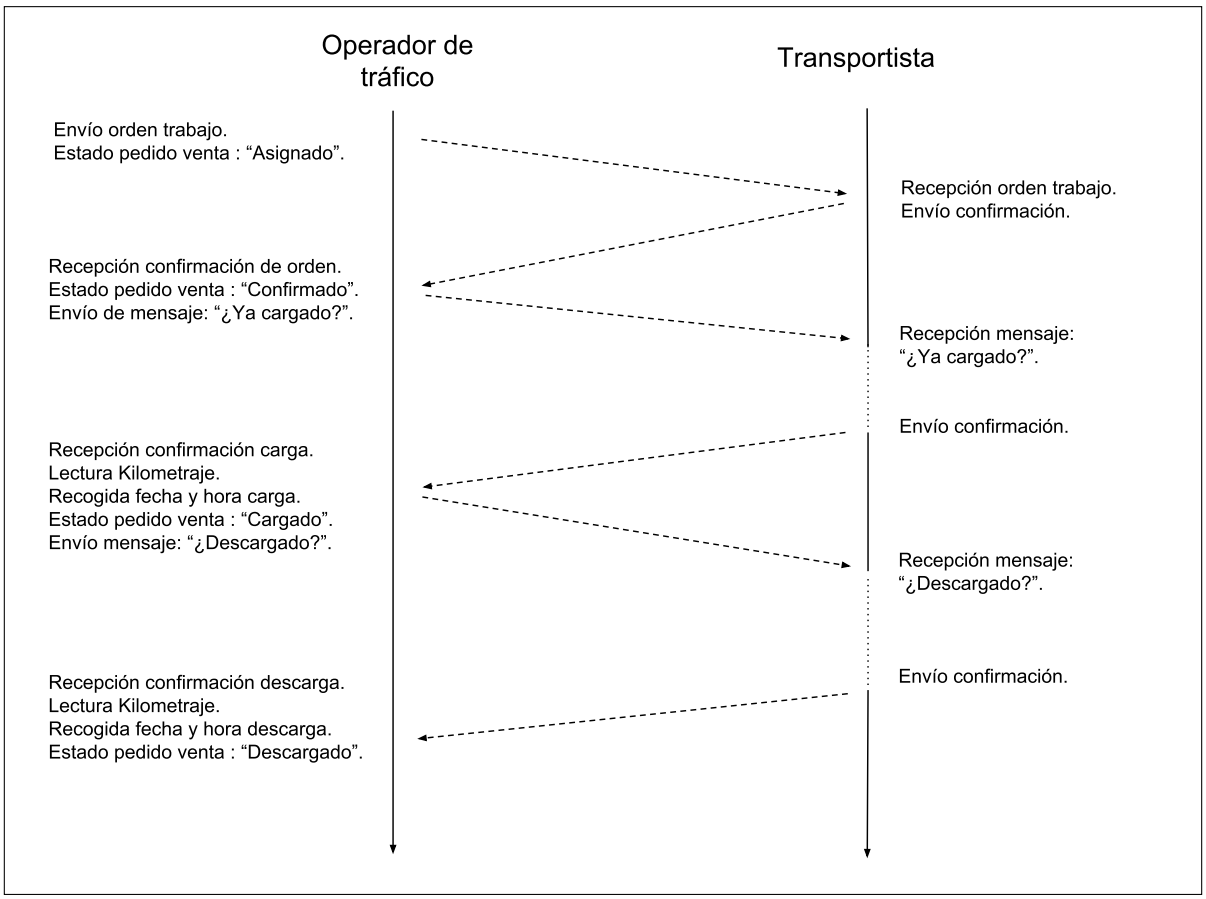

Figura 5.10: Patrón de interacción operador de tráfico - transportista.

responder a los mensajes que recibe.

- Órdenes de trabajo. Módulo en fase experimental.
- Informe de conducción. Datos relativos al estilo de conducción.
- POIs. Ofrece el listado total de POIs dados de alta en la empresa para la que trabaja el conductor dado de alta en la aplicación.
- Alarmas. Configuración de alarmas por parte del conductor.
- Gasolineras. Ofrece información sobre el precio de carburante para las gasolineras más próximas a las coordenadas actuales del vehículo.
- Worklog. Ofrece a los conductores redactar partes de trabajo.
- Transportes. Gestión de las órdenes de trabajo a realizar.

El trabajo de desarrollo del *CodeUnit* “50007 Funciones Navifleet” opera directamente sobre el módulo de ”Transportes”. Este ha sido diseñado para la gestión del flujo de órdenes de trabajo. A través de él, se puede iniciar una orden de trabajo en cualquier instante. Este módulo también

Figura 5.11: Configuración de módulos base en MFLEET.

permite ver el total de órdenes de trabajo encoladas y poder retomar la actividad para aquella orden en curso. La figura 5.12 muestra el aspecto del módulo de Transportes.

El proceso de seguimiento para una orden de trabajo comienza con el envío, por parte de un operador de tráfico, de una orden a un transportista en concreto. Esto requiere de una llamada al servicio web correspondiente, de envío de órdenes del SDK de Navifleet. Una vez enviada la orden, el conductor la recibe en un breve espacio de tiempo y, de no existir otra orden en curso, el conductor ya podría validar el inicio de actividad del servicio. Mediante una serie de pantallas de interacción elegantes y de sencillo uso, la aplicación va guiando al conductor durante las distintas fases que componen un servicio. En paralelo se dispone de un formulario de datos preconfigurado y optimizado para dar soporte a las exigencias de los servicios de transporte a realizar, permitiendo la introducción de la información asociada a una operación de carga o descarga incluida dentro de la propia orden de trabajo. Entre los parámetros que pueden recogerse mediante este formulario, destacamos:

- Tipo de tarea. (Carga/ Descarga).
- Código de tarea.
- Lugar de destino.
- Cliente asociado a la tarea.
- Fecha prevista de realización.

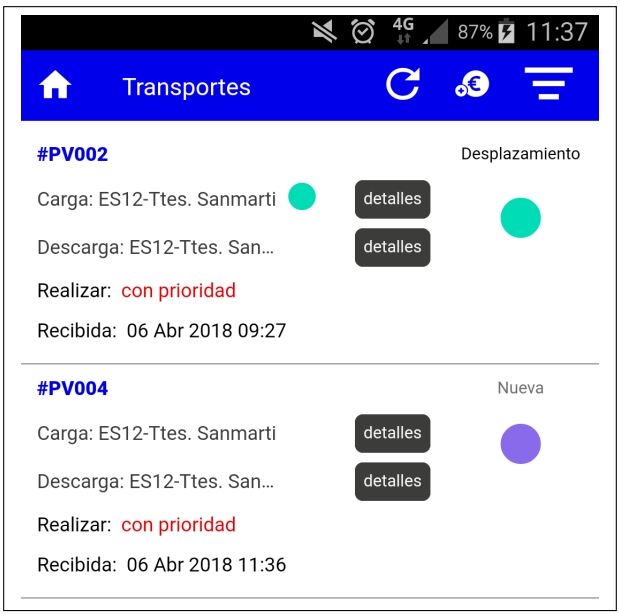

Figura 5.12: Modulo de transportes en MFLEET.

- Ruta asociada.
- Descripción.
- Número de bultos.
- Número de kilogramos de mercancía.
- Asociación de hasta dos documentos.
- Comentarios.
- Firma del cliente.

Las figuras 5.13 y 5.14 presentan el total de pantallas de interacción para un transportista, el cual gestiona una orden de trabajo compuesta por una operación de carga y una posterior operación de descarga de mercancía. Cabe señalar que las posibilidades de configuración de envío de una orden de trabajo son muy elevadas y este ejemplo solo explota una parte de la potencia del servicio web detallado en el SDK de Navifleet de envío de órdenes. Como detalle, destacar que el conductor no puede validar el estado de carga o descarga del vehículo si no aporta antes la cifra de kilogramos de mercancía gestionada, así como el número de bultos asociado. Existe la posibilidad de declarar un supuesto de kilogramos de carga o descarga asociados a una tarea en el momento del envío de la orden, pudiendo el conductor rectificar estas cifras acorde a la cantidad de mercancía gestionada. Como se ha indicado y se puede apreciar en las imágenes citadas, esta información junto a otros detalles, se recogen en un formulario asociado a la tarea que se está realizando (sea este de carga o descarga). Destacar también que, una vez el conductor valida el estado de descarga la orden de trabajo se da por concluida y por tanto desaparece de la cola de órdenes de trabajo, pudiendo atenderse la siguiente orden si esta existiese.

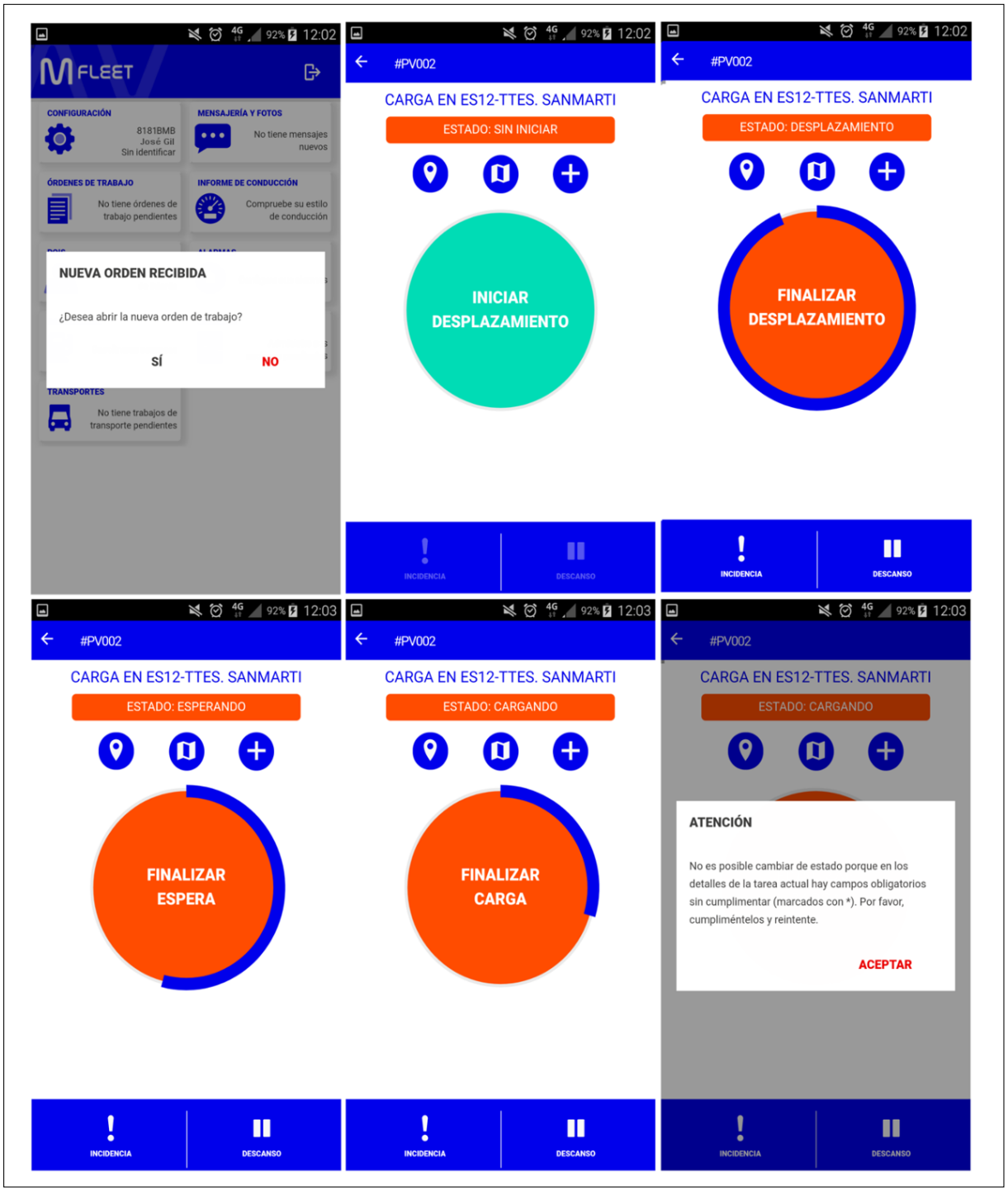

Figura 5.13: Ventanas para una orden de trabajo en MFLEET (1/2).

Otros aspectos como la notificación de incidencias, la declaración de descansos o incluso la declaración de gastos son posibles durante la interacción con una orden de trabajo. Esta información puede rescatarse siempre asociada a una orden de trabajo, a excepción de los gastos reportados por el conductor, los cuales deberán ser rescatados mediante un conjunto de servicios web pertenecientes al SDK de Movildata. La figura 5.15 muestra el aspecto de las

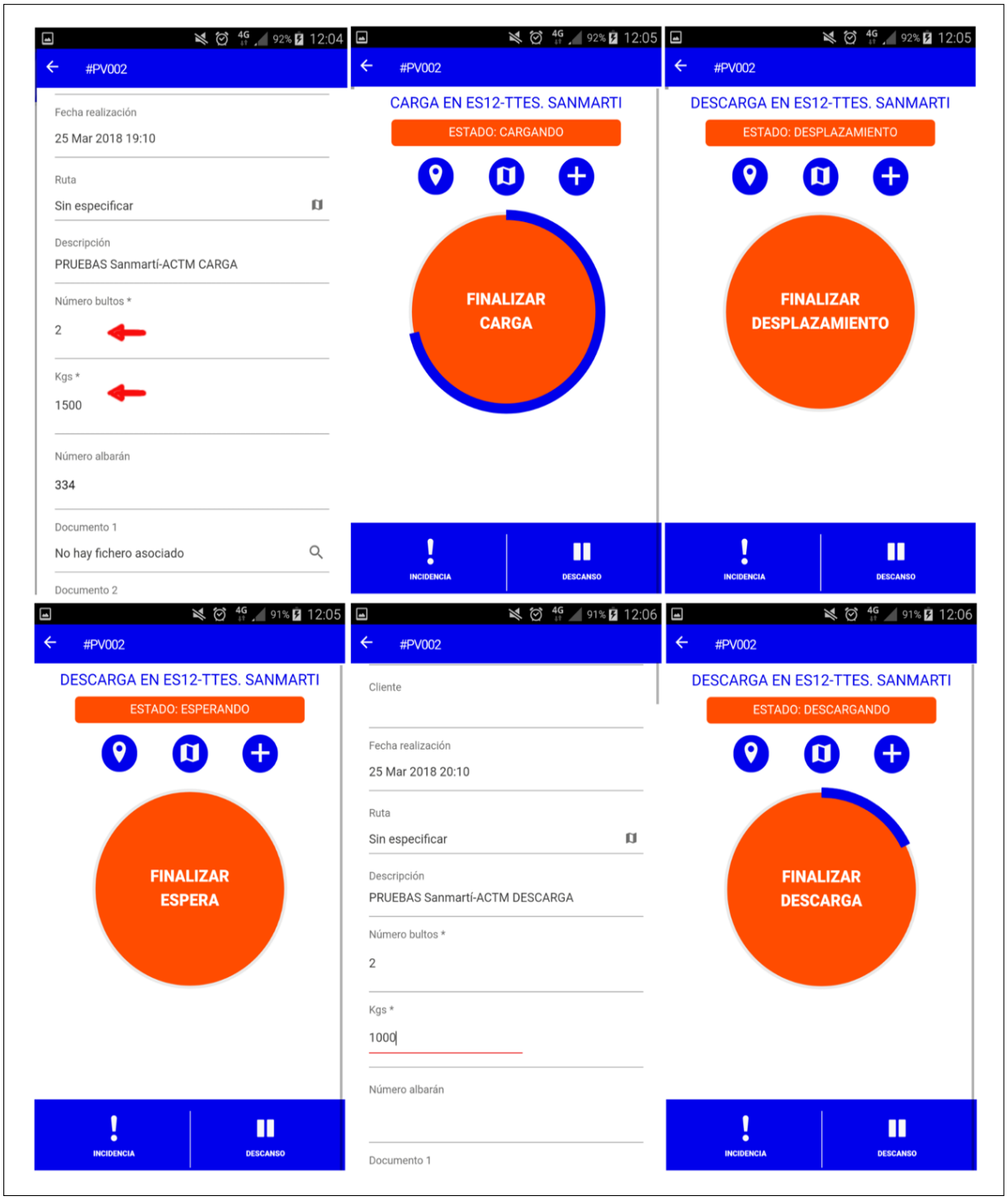

Figura 5.14: Ventanas para una orden de trabajo en MFLEET (2/2).

ventanas de interacción para los aspectos complementarios anteriormente citados.

El servicio web que permite realizar un envío de orden está descrito con todo detalle en el SDK de Navifleet. El envío de órdenes de trabajo es un servicio web que se fundamenta en la

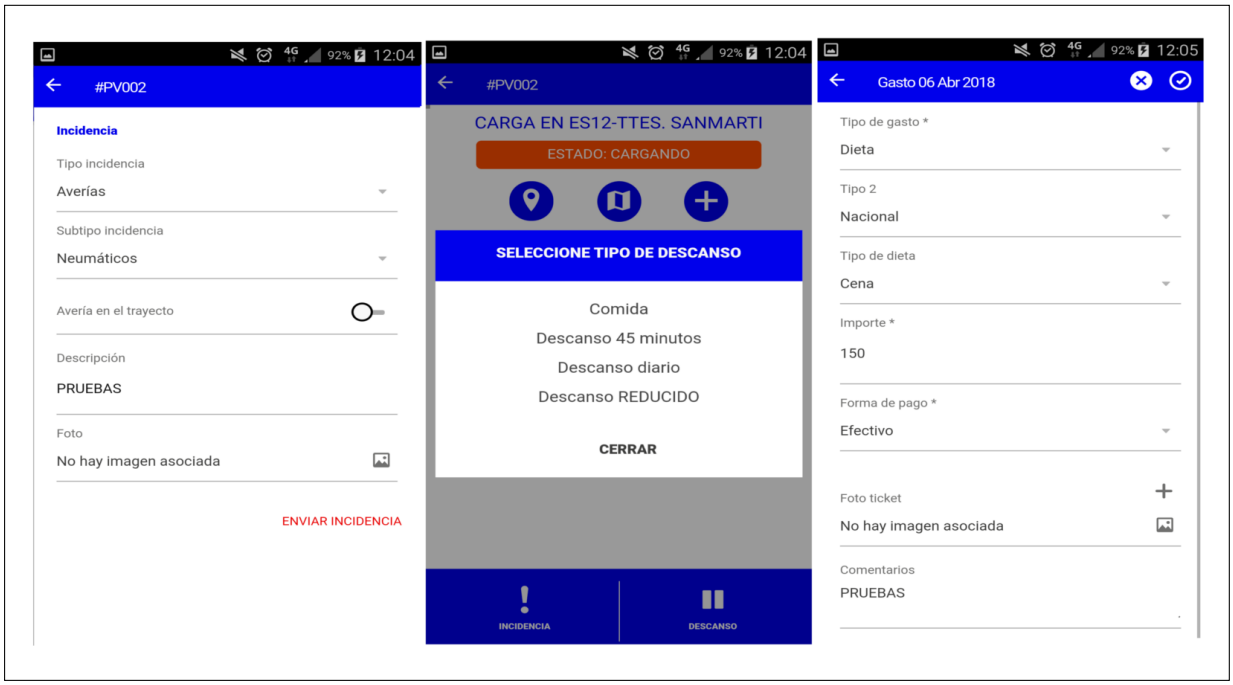

Figura 5.15: Gestión de Incidencias, descansos y gastos en MFLEET.

composición de items de información que permiten definir un conjunto de tareas asociadas a una orden de trabajo. Para poder realizar esta composición, el SDK indica una serie de parámetros de carácter obligatorio y otros de libre configuración. El concepto de orden de trabajo en el sistema MFLEET engloba la consecución de un número determinado de tareas de carga o descarga de mercancía. La configuración mínima para un envío de orden de trabajo requiere la información relativa a la orden a enviar, los datos del conductor objetivo y al menos una operación de carga o descarga asociada a la orden. Esta configuración base ya permite realizar un envío de orden y recepcionarlo en un terminal previamente configurado con MFLEET. Los cuadros 5.3 y 5.4 recogen el total de parámetros con los que se ha trabajado en el presente *CodeUnit* para la confección de órdenes de trabajo. Es importante señalar que hay muchos más, pues las posibilidades de configuración para la composición de órdenes son muy amplias. Los cuadros citados marcan aquellos parámetros obligatorios en la composición de una orden de trabajo con un asterisco.

Nombre	Observaciones
sAPIKEY*	Apikey vinculada al cliente (Proporcionada por Movildata).
nIdForm*	Identificador de tipo de orden (Para todas las órdenes igual).
sAction*	Caracter 'I' (Insertar orden).
aoOrder*	Listado de órdenes a insertar
- sCode*	Código único de orden en el ERP (Visible para el conductor).
- dExp	Fecha y hora UTC de realización de la orden. (Su ausencia indica ASAP).
- sIDVehicle	Matrícula del vehículo que realiza el servicio.
- nAddrFormat	'3' o '1' Dependiendo si se usan IDs de POIs de Movildata o se usa el formato 'lat—lag—texto'
- aoBlock*	Listado de tareas de carga o descarga a realizar

Cuadro 5.3: Parámetros de definición de una orden de trabajo.

Nombre	Observaciones
– nIdBlock*	Identificador del bloque de datos (Valor fijo '436').
– sCodeTask*	Código de la tarea (Único dentro de la orden, no visible para el conductor).
– sCodeType	Tipo de tarea. (TR.TT.CAR/DES, carga o descarga)
– dExpTask	Fecha y hora UTC de realización de la orden.
– sAddrInfo*	Descripción de la dirección de la tarea (según formato indicado en la orden).
– sCustomer	Descripción del cliente para la tarea.
– nIdRoute	Identificador de la ruta en Movilflota para la tarea.
– fPackages	Número de bultos relativos a la tarea.
– fKgs	Kilogramos de carga relativos a la tarea.
– sTask	Observaciones relativas a la tarea.

Cuadro 5.4: Parámetros de definición de una tarea incluida en una orden de trabajo.

Para el trabajo que nos atañe, se procederá a codificar una solución basada en la estructura de servicio de transporte compuesto por una única carga y una única descarga. La composición del cuerpo de texto JSON que ha producido el envío de la orden de trabajo, la cual ha sido gestionada en el conjunto de pantallas que recogen las figuras 5.13 y 5.14 es el que muestra la figura 5.16.


```
{"sAPIKEY":"API/111222333444555",
"sAction":"I",
"nIdForm":"417",
"aoOrder":
    [{  "sCode": "#PV002",
        "sIDVehicle": "8181BMB",
        "nAddrFormat": "3",
        "aoBlock":[
            {"nIdBlock":"436",
            "sCodeTask":"pt1Carga",
            "sCodeType":"TR.TT.CAR",
            "sAddrInfo": "31061",
            "dExpTask":"2018-03-25 17:10:00",
            "sTask": "PRUEBAS Sanmartí-ACTM CARGA"
            },

            {"nIdBlock":"436",
            "sCodeTask":"pt2Descarga",
            "sCodeType":"TR.TT.DES",
            "sAddrInfo": "31061",
            "dExpTask":"2018-03-25 18:10:00",
            "sTask": "PRUEBAS Sanmartí-ACTM DESCARGA"
            }
        ]
    }]
}
```


Figura 5.16: Cuerpo de mensaje JSON para envío de orden.

Con los detalles descritos hasta ahora ya se tiene la base para poder implementar el método "OrderEnvy" para el presente *CodeUnit* "50007 - Funciones Navifleet". Este método, haciendo uso del cuerpo de texto JSON presentado en la figura 5.16 y aprovechando gran parte del

conjunto de métodos no accesibles encargados de realizar la llamada a los servicios web, descritos para el *CodeUnit* "50006 - Funciones Movildata", nos permitirá realizar el envío de órdenes de trabajo a los conductores dotados con la aplicación MFLEET.

La peculiaridad de "OrderEnvy" es que, tras el envío de una orden de trabajo, se recoge la respuesta del servidor y esta vez se interpretará el posible código de error que se genere. La anidación de unas estructuras condicionales de tipo "CASE TRUE OF", propias del lenguaje C/AL, permitirán del lado de Dynamics NAV clasificar adecuadamente el código de error resultante del envío de orden y así poder mostrar un mensaje de texto significativo al operador de tráfico. La figura 5.17 muestra la codificación que permite llevar a cabo la tarea de clasificación del estado asociado al envío de órdenes de trabajo.

```
  CASE TRUE OF
    status = '0':
        EXIT('Orden enviada con exito al conductor con matricula: '+plateNumber);
    status <> '0':
      BEGIN;
        mensaje := 'Una o mas ordenes presentan algun problema: ';

        CASE TRUE OF
          statusOrder = '-89':
            EXIT( mensaje + 'Alguna fecha de carga o descarga no es valida' );
          statusOrder = '-2':
            EXIT( mensaje + 'El conductor indicado no existe' );
          statusOrder = '-3':
            EXIT( mensaje + 'El vehiculo indicado no existe' );
          statusOrder = '-4':
            EXIT( mensaje + 'Ya existe una orden de trabajo con el mismo codigo');
          statusOrder = '-1':
            BEGIN;
              mensaje := 'Una o mas tareas presentan algun problema: ';
              CASE TRUE OF
                      statusTask = '-99':
                        EXIT( mensaje + 'Error en la fecha de la orden de trabajo' );

                      statusTask <> '0':
                        EXIT( mensaje + 'El POI (Punto de interes) no existe' );
            END;
          END;
        END;
      END;
END;
```

Figura 5.17: Clasificación del estado devuelto por el servidor para "OrderEnvy".

Múltiples pruebas de envío de órdenes se han realizado tanto en los terminales móviles personales como en una tablet que la empresa de transporte Sanmartí nos cedió temporalmente para la realización de pruebas. Este último dispositivo pudo ser dotado de la aplicación MFLEET gracias a la intervención remota del equipo técnico de Movildata. Envíos intencionados de valores erróneos originaban cada uno de los códigos de error que se detalla en el SDK de Navifleet.

El siguiente método que se procede a desarrollar ha de permitir la cancelación de una

orden de trabajo enviada. La estructura JSON que permite accionar este servicio web ha sido anteriormente presentada en la figura 5.4. Debido a su sencillez, no se entrará en más detalles. El resultado de la cancelación de la orden de trabajo también se recoge a fin de poder clasificar el posible código de error que se produzca y poder así devolver el correspondiente mensaje explicativo del mismo modo que con el método "OrderEnvy".

Llegados a este punto y con un último método que desarrollar para el presente *CodeUnit*, se precisa de una solución software que permita interpretar las estructuras JSON que nos envíe de vuelta el servidor.

Para poder clasificar los códigos de error de "OrderEnvy" y "cancelOrder" ya se ha usado esta solución. La solución que incluye este *CodeUnit* se basa en la combinación de tres métodos que permiten interpretar de forma muy cómoda una estructura JSON. Esta solución hace especial énfasis en la sencillez de uso al combinar estos tres métodos y poder así extraer un determinado item de información.

Los métodos a los que se hace mención son:

GetValueJSON. Este método permite extraer un valor determinado de un bloque simple JSON, pasando como parámetros el bloque y la clave del valor que buscamos.

GetSectionJSON. La labor de este método es poder posicionarse sobre uno de los tres arrays propios de la trazabilidad de una orden de trabajo. Los arrays son "aoPeriod", "aoTask" y "aoIncident" (Más adelante serán descritos en profundidad). El método recibe como parámetro el cuerpo de texto en formato JSON, tal y como lo devuelve el servidor de Movildata, y finalmente retorna el array seleccionado con todo su contenido.

GetBlockJSON. El tercer método que permite el trabajo con JSON se centra en proporcionar un bloque JSON en concreto. Dentro de cada uno de los arrays que componen la trazabilidad de una orden pueden existir múltiples bloques. Poder posicionarse sobre un bloque en concreto, permitirá posteriormente con "GetValueJSON" acceder al valor deseado. "GetBlockJSON" recibe como parámetros el contenido de un array JSON, una clave característica y su valor asociado conocido.

Es importante señalar que si estos métodos no hallan el valor o las estructuras JSON que se desean encontrar, estos retornan en tal caso la cadena 'N/A' (Not available).

Para dar soporte a clasificar el posible código de error que pudiera generarse con el envío o cancelación de una orden de trabajo, basta con combinar "GetValueJSON" y "GetBlockJSON".

El último de los métodos que queda por presentar para el presente *CodeUnit* es "getStatusOrder". Este método permite de forma totalmente sistemática evaluar una orden de trabajo con una estructura compuesta por una única tarea de carga y otra de descarga. La estrategia de este método es hacer uso de los tres métodos desarrollados para dar soporte a la interpretación de las estructuras JSON, de forma que se puedan rescatar una serie de datos de interés del cliente. Fruto de la interpretación de la orden de trabajo se obtendrá una cadena de texto compuesta por los datos escogidos, separados por una barra vertical, los cuales serán rescatados del lado del ERP y posteriormente cumplimentarán los distintos campos de un pedido de venta.

Con respecto a los datos que se van a extraer de una orden de trabajo, estos han de permitir poder automatizar el cambio de estados del planificador de tráfico además de satisfacer las necesidades de información para un pedido de venta. Entre los datos que "getStatusOrder" ha de poder recoger para una orden de trabajo, desatacamos:

- Confirmación de inicio de actividad.
- Kilometraje del vehículo en el momento de carga o descarga de mercancía.
- Kilogramos de materia cargada o descargada.
- Fechas de carga o descarga.
- Incidencias registradas.

Esta estructura de datos presentada en forma de cadena concatenada con separadores de tipo barra vertical permitirá una extracción de los datos y su posterior manipulación e introducción en el ERP. La figura 5.18 muestra dicha estructura. Recordemos pues que, para cada una de las componentes de esta cadena compuesta, sus valores por defecto serán siempre "N/A"; esto indicará que el dato que se busca en la enésima posición todavía no está disponible. En definitiva, esta cadena compuesta será en esencia el conjunto de datos que permitirá automatizar el planificador.

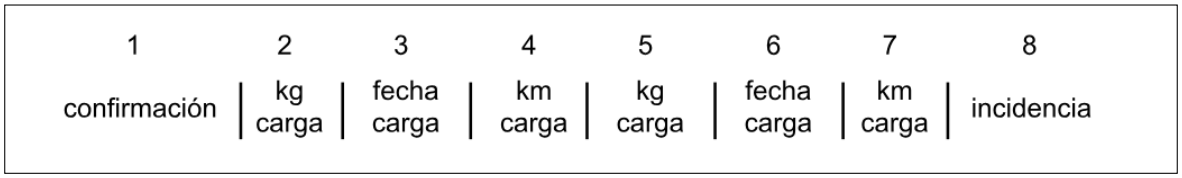

Figura 5.18: Estructura de la cadena de texto compuesta.

La figura 5.19 muestra la trazabilidad para la orden de trabajo creada durante la descripción del método "OrderEnvy". Para hacer uso de este servicio web, se requiere una URL distinta a la que se especifica para los servicios web de creación y cancelación de órdenes de trabajo. Es de vital importancia cerciorarse que la estructura JSON incluida en la figura, es el resultado final de que el conductor haya validado todas las fases de las distintas operaciones de carga y descarga mediante la aplicación MFLEET. Por cada operación de carga o descarga, el conductor ha de validar los estados de desplazamiento, espera y carga/descarga en cada caso. Puesto que puede solicitarse la trazabilidad de una orden de trabajo en cualquier momento desde que esta es enviada al conductor, los servidores de Movildata muestran la trazabilidad de datos existente hasta el momento. Puede ocurrir que, en un intervalo de tiempo de cinco minutos aproximadamente, no tengamos el detalle de las últimas acciones acontecidas debido al tiempo de refresco programado en los servidores de Movildata. En ningún caso este retraso en la obtención de la información supone molestia alguna, pues siempre se acaban recepcionando los datos correctamente.

Vamos a describir la estructura JSON que viene de vuelta al solicitar la trazabilidad de una orden. En primer lugar, si la orden para la que pedimos la trazabilidad existe y pertenece al cliente propietario de la Apikey usada, obtendremos una estructura base JSON. Esta estructura,

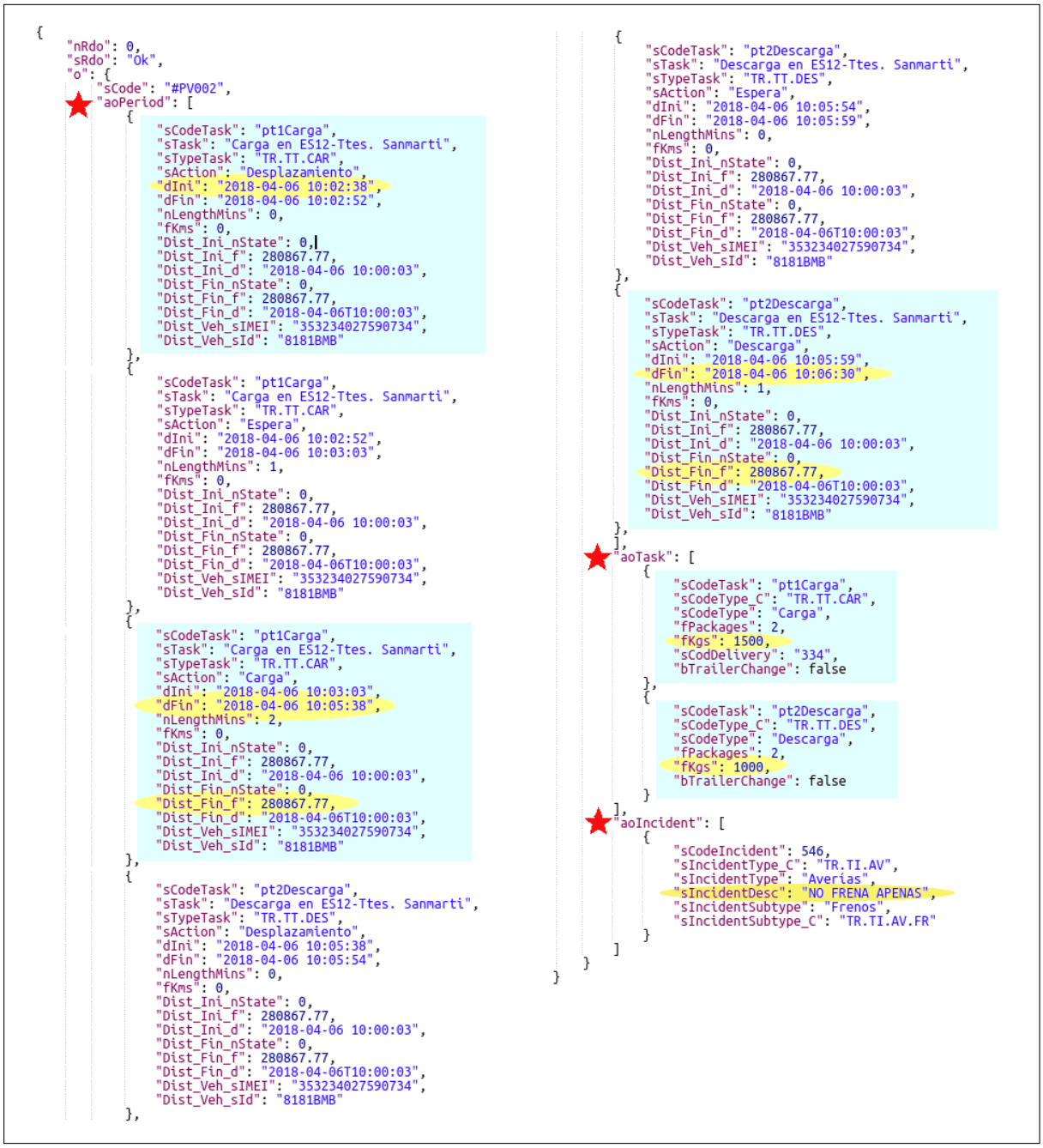

Figura 5.19: Trazabilidad para la orden de trabajo “PV002”.

incluso para cuando todavía el conductor no ha iniciado su actividad, se compone de tres arrays. Estos arrays, los cuales ya han sido citados anteriormente, son: “aoPeriod”, “aoTask” y “aoIncident”. El cometido de estos arrays es mostrar la información disponible hasta el momento para una orden de trabajo. “aoPeriod” muestra una secuencia de tareas finalizadas incluyendo kilometrajes y fechas entre otros datos de interés. Entre las tareas que detalla “aoPeriod” encontramos: inicios de actividad, esperas, cargas, descargas y descansos. “aoTask” recoge los datos introducidos por el conductor en cada formulario correspondiente a las tareas de carga o descarga incluidas en la orden de trabajo. “aoIncident” recogerá aquellas incidencias reportadas

por el conductor, asociadas al tiempo de trabajo de una orden en cuestión. Una combinación de los métodos "GetValueJSON", "GetBlockJSON" y "GetSectionJSON" presentados anteriormente permitirán acceder de forma unívoca a los datos de una orden, si estos existen. La figura 5.19 refleja el ámbito de acción sobre el que los tres métodos de interpretación han de operar en base a la estructura JSON que nos devuelve el servidor de Movildata.

Por poner un ejemplo: si quisiésemos acceder a la fecha de la operación de carga incluida en la orden de trabajo, deberíamos posicionarnos sobre el array que recoge el total de tareas acontecidas ("aoPeriod") con el método "GetSectionJSON", extraer el bloque correspondiente a la tarea de carga con el método "GetBlockJSON" y finalmente acceder al dato en concreto con "GetValueJSON". La sencillez de uso de estos métodos permite en un futuro rescatar diferentes items de información sin realizar grandes operaciones de mantenimiento en el código.

Más adelante se verá que esta estrategia de interpretación de la información sigue totalmente los principios de mantenibilidad y reusabilidad de código, puesto que permitirá de forma fácil componer una segunda solución para la lectura de la trazabilidad para órdenes con múltiples puntos de carga y un único punto de descarga. Esta aproximación será descrita como una línea de trabajo futura, para la cual se presenta una estrategia resolutiva primeriza.

Volviendo con el ejemplo de la orden de trabajo para la que hemos estudiado su trazabilidad, la cadena compuesta que retornará "getStatusOrder", fruto de haber extraído cada uno de los campos requeridos tiene el siguiente aspecto:

"OK—1500—2018-04-06 10:05:38—280867.77—1000—2018-04-06 10:06:30—280867.77—NO FRENA APENAS"

Veamos como se interpretará esta cadena a fin de automatizar los distintos estados para un pedido de venta. La primera de las componentes permitirá variar el estado de un pedido de venta a "confirmado" dado que se ha registrado la fecha de inicio de actividad. Las tres siguientes componentes, además de cumplimentar los respectivos campos de información, permitirán variar de nuevo el estado del pedido de venta a "cargado". Finalmente, las tres siguientes componentes permitirán dar por descargado un pedido de venta una vez hayan sido recepcionados los datos. Nótese que una incidencia podría originarse justo a continuación del inicio de la actividad para un transportista. Dicha incidencia podrá ser recogida y, en el caso del desarrollo efectuado sobre el vertical del transporte, se habilitará un botón asociado a un pedido de venta indicando la existencia de anomalías para ese servicio de transporte en concreto.

El poco tiempo transcurrido entre las operaciones de carga y descarga para la cadena compuesta presentada, es debido a una validación de estados sobre el terminal de prueba configurado con la aplicación Android MFLEET. La no variación del kilometraje para el vehículo sobre el que se ha tomado como referencia para el envío de orden, indica que dicho vehículo no ha registrado movimiento alguno durante el corto tiempo de realización de la prueba con MFLEET.

La figura 5.20 muestra el algoritmo codificado en C/AL que permite la obtención de la cadena compuesta de datos para una orden de trabajo concreta. Destacar la inicialización de las componentes a "N/A" (Not available) en símbolo de ausencia de valor y la estrategia de avance del algoritmo para evaluar la trazabilidad, en base a la existencia o no de la información esperada.

```
confirm := 'N/A';
kgCarga := 'N/A';
kgDesc := 'N/A';
feCarga := 'N/A';
feDesc := 'N/A';
kmCarga := 'N/A';
kmDescarga := 'N/A';
incidencias:= 'N/A';

txtMensaje := '{sAPIKEY:" '+APIKEY_X+'",'+
              'sAction:" '+'ID",'+
              'sCode:" '+orderId+'"}';

txtResponse := EnviarPeticionServidorJSON(URL_TRANSPORT, txtMensaje);


value := GetValueJSON(GetBlockJSON(txtResponse,'sAction','Desplazamiento'),'dIni');

IF value = 'N/A' THEN
  EXIT(confirm+'|'+kgCarga+'|'+feCarga+'|'+kmCarga+'|' + kgDesc+'|'+feDesc+'|'+
    kmDescarga+'|'+incidencias)
ELSE
  confirm := value;
  incidencias := GetValueJSON(GetSectionJSON(txtResponse,'aoIncident'),'sIncidentDesc');

  value := GetBlockJSON(txtResponse,'sAction','Carga');

  IF value = 'N/A' THEN
    EXIT(confirm+'|'+kgCarga+'|'+feCarga+'|'+kmCarga+'|' + kgDesc+'|'+feDesc+'|'+
    kmDescarga+'|'+incidencias)
  ELSE BEGIN
    kgCarga := GetValueJSON(GetBlockJSON(txtResponse,'sCodeType','Carga'),'fKgs');
    feCarga := GetValueJSON(GetBlockJSON(txtResponse,'sAction','Carga'),'dFin');
    kmCarga := GetValueJSON(GetBlockJSON(txtResponse,'sAction','Carga'),'Dist_Fin_f');

    value := GetBlockJSON(txtResponse,'sAction','Descarga');

    IF value = 'N/A' THEN
      EXIT(confirm+'|'+kgCarga+'|'+feCarga+'|'+kmCarga+'|' + kgDesc+'|'+feDesc+'|'+
    kmDescarga+'|'+incidencias)
    ELSE BEGIN
      kgDesc := GetValueJSON(GetBlockJSON(txtResponse,'sCodeType','Descarga'),'fKgs');
      feDesc := GetValueJSON(GetBlockJSON(txtResponse,'sAction','Descarga'),'dFin');
      kmDescarga := GetValueJSON(GetBlockJSON(txtResponse,'sAction','Descarga'),'
    Dist_Fin_f');

    END;
  END;
EXIT(confirm+'|'+kgCarga+'|'+feCarga+'|'+kmCarga+'|' + kgDesc+'|'+feDesc+'|'+kmDescarga+
    '|'+incidencias);
```

Figura 5.20: Codificación “getStatusOrder”.

Finalmente, se muestra en el cuadro 5.5 la relación final de métodos incluidos en el *CodeUnit* "50007 - Funciones Navifleet".

Nombre función	**Descripción**	**Parámetros**	**Desarrollo**
GetResponseServerJSON	Retorna una respuesta con estructura JSON.	HttpWebRequest DotNet Mensaje Text	Propio
TryCreateRequestStream	Establecer un flujo de envío de datos para una petición HttpWebRequest	HttpWebRequest DotNet RequestStream DotNet	Estándar
TryGetWebResponse	Vuelca en HttpWebResponse, el resultado de la petición HttpWebRequest	HttpWebResponse DotNet HttpWebRequest DotNet	Estándar
ReadHttpResponseAsText	Devuelve el flujo de respuesta intepretado como una cadena de texto.	HttpWebResponse DotNet	Estándar
CreateWebRequest	Prepara una petición HTTP POST, mediante una URL y una acción SOAP.	Url Text Action Text	Propio
FormatDate	Formatea un tipo Date en una cadena de texto con el formato: 'yyyy-mm-dd'	Date Text	Propio
GetValueTag	Extrae un dato contenido en un cuerpo de respuesta con formato SOAP-XML.	String Text Tag Text	Propio
EnviarPeticionServidorJSON	Realiza la llamada a un servicio web, recibiendo la url y el mensaje en formato JSON.	url Text text Text	Propio
GetValueJSON	Retorna el valor asociado a una detrminada clave dentro de un bloque simple JSON.	string Text tag Text	Propio
GetSectionJSON	Retorna el array seleccionado incluido en la trazabilidad de una orden de trabajo.	string Text sectionName Text	Propio
GetBlockJSON	Obtiene un bloque JSON simple a partir de indicar una clave y valor características.	string Text blockKey Text blockValue Text	Propio
GetStatusOrder	Obtiene una cadena compuesta con la extracción de datos de interés del cliente. (Recoge la respuesta si existe)	orderId Text	Propio
OrderEnvy	Permite enviar una orden de trabajo a un transportista.	OrderId Date plateNumber text ...	Propio
cancelOrder	Permite cancelar una orden dada de alta en la empresa cliente de Movildata	oderId Text	Propio

Cuadro 5.5: Funciones incluidas en el *CodeUnit* "50007 - Funciones Navifleet".

Adicionalmente queda comentar la configuración para las órdenes de trabajo finalmente programada. Ésta controla algunos items de información más que los descritos anteriormente. Por ejemplo, se pueden recepcionar documentos enviados con la aplicación MFLEET y se registra el posicionamiento de los vehículos para cada operación de carga o descarga. Para añadir nuevos items de información a controlar, únicamente sería necesario ampliar el número de componentes de la cadena de texto compuesta y hacer uso de aquellos servicios web que se requieran.

5.1.4. Desarrollo del *CodeUnit* "Timer"

Para poder automatizar el cambio de estados del planificador de tráfico, se requiere de un proceso en Microsoft Dynamics NAV que, de forma continuada cada cierto tiempo, revise la trazabilidad de las órdenes de trabajo que todavía no hayan alcanzado el estado de "descargado" en el planificador de tráfico. Además, se requiere que dicha solución software permita su inicialización y parada a voluntad del personal de tráfico o bien poder inicializar su actividad cuando el primer operador de tráfico introduzca sus credenciales en el ERP. La condición de parada en este segundo caso sería que ningún operador de tráfico tenga su cuenta de trabajo en activo en Dynamics NAV.

La elección de la estrategia de paro y arranque es indiferente y cualquiera podría ser válida en un principio. Sin embargo se valorará el iniciar el menor número de procesos posible en el momento de arranque del sistema o de habilitación del entorno de trabajo para un usuario. La segunda estrategia tiene de atractivo la comodidad que supone el arranque del proceso sin el control por parte de un humano. La ausencia de paros o marchas del proceso de forma accidental o involuntaria también sería otro punto a favor para la segunda estrategia.

La complejidad de este proceso reside en emular el "Timer" que permitirá, tras cada cierto tiempo, revisar los nuevos datos que han entrado para cada orden. Tras descartar formas de trabajo de antiguas versiones de Dynamics NAV, donde se contaba con triggers del tipo "On-Timer", donde podía escribirse código y ser ejecutado de forma repetida con un espaciamiento en el tiempo, se procede a buscar otras soluciones compatibles con la versión base sobre la que está desarrollado el vertical del transporte de ACTM. Microsoft parece ofrecer una solución en forma de *Add-in*, véase [5], la cual no ha acabado de dar los resultados esperados. También se ha recurrido a consultar blogs frecuentados por profesionales de Dynamics NAV como es el ejemplo de [4]. Tras varias horas de investigación al respecto hallamos una solución que parece adaptarse a nuestro problema. Jonathan Archer desde una entrada al blog de Microsoft presenta una solución al problema del temporizador bastante intuitiva. Como puede observarse en la entrada de Jonathan en el blog comunitario de la herramienta Microsoft Dynamics NAV, [3], se presenta un ejemplo de código que permite realizar escrituras de líneas de texto a un archivo con un espaciamiento en el tiempo de 3 segundos. Este código permite la realización de este cometido gracias al uso de una variable del marco de trabajo .NET. La variable requiere de ser declarada como global y en sus propiedades indicar "RunOnClient = No" y "WithEvents = Yes". Esta última propiedad hará que se habiliten dos triggers, "Elapsed" y "Disposed". "Elapsed" será el trigger que nos permitirá programar la ejecución de código tras el vencimiento de un quantum indicado.

Con esta aproximación ya se puede elaborar un *CodeUnit* que del lado del ERP, cada cinco minutos por ejemplo, realice una llamada a "gestStatusOrder" para cada una de las órdenes de trabajo que siguen activas en el planificador de tráfico y poder así siguiendo con la labor de rescate de la información restante.

Para cada orden de trabajo enviada al transportista de turno, la forma de actualizar su estado en el planificador de tráfico será:

- Si no se tiene constancia de inicio de actividad, el pedido de venta seguirá con el estado "asignado".

- Si la primera componente de la cadena compuesta tiene información, se tiene constancia de inicio de actividad y por tanto el pedido de venta cambiará su estado a "confirmado".

- Si la cadena compuesta trae información de la operación de carga, el pedido de venta actualizará sus campos de información y cambiará su estado a "cargado".

- Si la cadena compuesta trae información de la operación de descarga, el pedido de venta actualizará sus campos de información y cambiará su estado a "descargado".

Alcanzado el estado de descargado, este pedido de venta podría desaparecer del planificador de tráfico donde aparecen todos los pedidos de venta activos y pasar a una ventana de gestión de pedidos finalizados. Sin embargo se ha de evitar el efecto de "teletransporte" visual que esto produce. Sería recomendable que el operador de tráfico confirmara la finalización de una orden de trabajo a fin de sentir un mayor control sobre las acciones que suceden en el planificador.

Con respecto a las incidencias, sería recomendable marcar de forma visual un pedido de venta con un color distinto si esta sufre algún tipo de retraso por una avería, algún tipo de retención en la carretera o cualquier otro motivo reportado por el conductor. Para el caso, Dynamics NAV cuenta con una gama de colores disponible muy pobre, la cual prácticamente ya está siendo usada. Por el momento, un pedido de venta que presenta una incidencia muestra un botón asociado mediante el que se obtiene la causa de la incidencia reportada por el conductor.

5.1.5. Desarrollo del *CodeUnit* "50008 - Funciones Movilflota"

El modo de elaborar este *CodeUnit* guarda bastante relación en el *CodeUnit* "50006 Funciones Movildata". En este *CodeUnit*, también se hará uso de las estructuras que permiten realizar las llamadas a servicios web con base con el protocolo SOAP.

En relación con la gestión de POIs se ha solicitado a la empresa Movildata unas funciones a incluir en el SDK de Movilflota. Estas funciones, orientadas a poder realizar las operaciones de acceso, adición, borrado y actualización, permiten del lado del ERP gestionar los POIs de una organización. Para poder dar soporte a la integración de POIs para la empresa de transportes colaboradora se precisa de una migración de POIs desde la aplicación web de gestión de flotas. La empresa cuenta con más de un millar de puntos de interés que deberían ser traídos al ERP para asegurar una consistencia con el total de POIs dados de alta antes de empezar a gestionar nuevos desde el vertical. Hasta la fecha los operadores de tráfico han dado de alta los POIs a través de la herramienta web de gestión de flotas Movilflota. Pulsando directamente sobre los iconos de los vehículos que tienen en el mapa, ellos directamente ya pueden dar de alta un punto de interés.

El total de métodos incluidos en el *CodeUnit* "50008 - Funciones Movilflota" vienen descritos en el cuadro 5.6.

"getTypePois" trae de vuelta los tipos de agrupaciones para POIs dados de alta en una empresa. Un POI puede hacer referencia a un restaurante, gasolinera, centro de lavado, punto de carga, punto de descarga, etc...

Nombre función	Descripción	Parámetros	Desarrollo
GetResponseServer	Retorna un dato concreto (usando una etiqueta como tercer parámetro) o estructura SOAP-XML, si se introduce la cadena vacía como tercer parámetro.	HttpWebRequest DotNet Mensaje Text Item Text	Propio
TryCreateRequestStream	Establecer un flujo de envío de datos para una petición HttpWebRequest	HttpWebRequest DotNet RequestStream DotNet	Estándar
TryGetWebResponse	Vuelca en HttpWebResponse, el resultado de la petición HttpWebRequest	HttpWebResponse DotNet HttpWebRequest DotNet	Estándar
ReadHttpResponseAsText	Devuelve el flujo de respuesta intepretado como una cadena de texto.	HttpWebResponse DotNet	Estándar
CreateWebRequest	Prepara una petición HTTP POST, mediante una URL y una acción SOAP.	Url Text Action Text	Propio
FormatDate	Formatea un tipo Date en una cadena de texto con el formato: 'yyyy-mm-dd'	Date Text	Propio
GetValueTag	Extrae un dato contenido en un cuerpo de respuesta con formato SOAP-XML.	String Text Tag Text	Propio
buildSoapPetition	Genera un cuerpo de llamada SOAP-XML de forma dinámica en base a un array de parámetros.	vector (Dim 20) Text	Propio
replaceAll	Reemplaza una cadena de texto contenido dentro de otra cadena de texto para todas las apariciones.	string Text substr Text substrNew Text	Propio
doMovildataPetition	Realiza la llamada a un servicio web, recibiendo un vector de parámetros que definirán la llamada a servicios web.	vector (Dim 20) Text	Propio
addPoi	Permite la creación de un nuevo POI.	tipoPoi Text lat Text lon Text name text description Text radio Text clienteId Text	Propio
delPoi	Permite el borrado de un POI dado de alta.	idPoi Text	Propio
getlistPois	Permite recorrer el "Dataset" con todos los POIs dados de alta en una empresa, permitiendo su integración en el ERP.		Propio
getTypePois	Obtención de todos los tipos de POIs dados de alta en la empresa.		Propio
findAllTypes	Itera sobre los tipos de POIs dados de alta en una empresa. (Método auxiliar para "getTypePois")	text Text	Propio
ProcessResponseServer	Navega a través de la estructura XML del "DataSet" que contiene los POIs dados de alta para una empresa. (Método auxiliar para "getListPois")	desde Date hasta Date	Propio

Cuadro 5.6: Funciones incluidas en el *CodeUnit* "50006 - Funciones Movilflota".

Las operaciones CRUD permiten la creación, lectura, actualización y borrado de datos; sin embargo, con la creación y borrado pensamos que basta.

Mediante "addPoi" podemos definir un nuevo punto de interés. De este método señalar que, la respuesta del servidor en caso de que la operación tenga éxito, incluye el ID del POI que posteriormente permitirá su gestión.

El método "delPoi" permitirá para un id de un POI dado de alta en Movildata proceder a su eliminación.

Posiblemente el reto más grande que presenta esta *CodeUnit* reside en migrar todos los POIs dados de alta en la aplicación web de Movilflota a Dynamics Nav. Para ello hacemos uso del servicio web "GetListPois". Este servicio web, de sencilla invocación, retorna una estructura de tipo "Diffgram" / "DataSet" con estructuración de la información en XML. Para poder iterar sobre el total de los POIs y poder incluirlos en la base de datos de Dynamics NAV del cliente, se requiere un objeto .NET que soporte estructuras XML y permita navegar a través de los nodos del "Dataset".

Las variables DotNet requeridas para armar la estructura que permitirá interpretar el "Dataset" que envían los servidores de Movildata al hacer uso del servicio web "GetListPois" son presentadas en el cuadro 5.7.

Nombre	Tipo	Libreria .NET
ResponseXmlDoc	DotNet	System.Xml.XmlDocument.'System.Xml, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'
XmlNodeList	DotNet	System.Xml.XmlNodeList.'System.Xml, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'
XmlNode	DotNet	System.Xml.XmlNode.'System.Xml, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'
XmlAttribute	DotNet	System.Xml.XmlAttribute.'System.Xml, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'
XmlAttributes	DotNet	System.Xml.XmlAttributeCollection.'System.Xml, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089'

Cuadro 5.7: Variables DotNet asociadas a interpretar el "Dataset"

La codificación en lenguaje C/AL que permite interpretar el "Dataset" que traemos de vuelta con "GetListPOIS" se puede observar en la figura 5.21. Nótese que únicamente se muestra la recogida de la respuesta del servidor y la interpretación del "Dataset" mencionado. El proceso de llamada al servicio web se ha omitido debido a que comparte la estructura de los servicios web de tipo SOAP-XML presentados anteriormente. Con respecto a la navegación a través del "Dataset", en el momento en el que se accede al nodo correspondiente a un POI, en ese instante, sus 12 atributos son accesibles y por tanto esta información puede servir para poder dar de alta un POI en la tabla correspondiente en Dynamics NAV para la gestión de los POIs.

```
ResponseXmlDoc:= ResponseXmlDoc.XmlDocument;
ResponseXmlDoc.Load(HttpWebResponse.GetResponseStream);

XmlNode := ResponseXmlDoc.LastChild;

IF XmlNode.HasChildNodes THEN BEGIN
  XmlNode := XmlNode.FirstChild;
  IF XmlNode.HasChildNodes THEN BEGIN
    XmlNode := XmlNode.FirstChild;
    IF XmlNode.HasChildNodes THEN BEGIN
      XmlNode := XmlNode.FirstChild;
      IF XmlNode.HasChildNodes THEN BEGIN
        XmlNode := XmlNode.LastChild;
        IF XmlNode.HasChildNodes THEN BEGIN
          XmlNode := XmlNode.FirstChild;
          IF XmlNode.HasChildNodes THEN BEGIN
            XmlNodeList:= XmlNode.ChildNodes;
            FOR j:=0 TO XmlNodeList.Count-1 DO BEGIN
              XmlNode:= XmlNodeList.Item(j);
              IF XmlNode.ChildNodes.Count = 12 THEN BEGIN
                XmlNodeListInner:=XmlNode.ChildNodes;
                FOR z:=0 TO XmlNodeListInner.Count-1 DO BEGIN
                  XmlNode:= XmlNodeListInner.Item(z);
                  arrParams[z+1]:=FORMAT(XmlNode.InnerText);
                END;
                // Aqui puede accederse al array con los 12 atributos de cada POI
                // MESSAGE(arrParams[1]+arrParams[2]);
              END;
            END;
          END;
        END;
      END;
    END;
  END;
END;
```

Figura 5.21: Interpretación del Dataset “GetListPOIS”.

5.1.6. Integración de servicios con Google Maps

Junto a los requisitos de integración de sistemas, existían peticiones por parte de los operadores de tráfico de poder realizar aperturas de mapas de Google Maps, posicionando vehículos que estuvieran realizando algún servicio de transporte. La solución se ha canalizado realizando la operación de consulta de información para la API de Google Maps a través de URLs compuestas. Esta modalidad de trabajo de la API de Google Maps, permite trabajar cómodamente con búsquedas de direcciones y ubicaciones sin necesidad de tener una "Apikey".

Básicamente se ha trabajado con mapas que permiten el posicionamiento de un vehículo en un determinado momento. Una segunda aplicación ha sido proyectar la distancia a recorrer por un vehículo desde su posición hasta el punto de destino del servicio que está prestando. Calcular la distancia recorrida para un vehículo sin carga hasta la sede es la tercera de las aplicaciones que se le han dado a esta API.

Para efectuar los posicionamientos de los vehículos se ha usado la implantación del servicio web "GetLastLocationPlate" incluido en el *CodeUnit* "50006 - Funciones Movildata", el cual retorna latitud y longitud para un determinado vehículo en el instante de invocación del servicio.

La proyección de la distancia restante hasta destino es una URL compuesta que aúna las coordenadas actuales del vehículo y las del punto de destino al cual se dirige el vehículo.

5.2. Verificación y validación

En cada ciclo de desarrollo efectuado se han realizado labores de verificación y validación, a fin de asegurar la plena satisfacción del cliente.

Con respecto a la parte de verificación, se ha comprobado en todo momento la consecución de los requisitos funcionales y atributos de calidad que componen la batería de requisitos planteados por el cliente. El estudio del grado de consistencia de los requisitos a desarrollar o ya incluso en fases de desarrollo, ha supuesto un filtro de posibles errores en la interpretación de aquella funcionalidad incluida en los objetivos el proyecto.

Técnicas de inspección visual de código de forma sistemática junto al análisis de las representaciones del sistema en forma de diagramas, han permitido detectar errores en fases tempranas, evitando el comprometer el desarrollo de ciertas partes del proyecto en fases más avanzadas.

Hay que destacar las pruebas de aceptación realizadas en la propia sede del cliente. Estas han supuesto un filtraje para aquellos desarrollos que no se han ajustado a las expectativas previstas y que, por tanto, han supuesto generalmente la aplicación de ciertos cambios de enfoque o pequeñas adaptaciones en el desarrollo.

Durante las pruebas de validación con el cliente han sido detectados errores graves, los cuales no pueden aparecer durante el uso del conjunto de herramientas que permiten monitorizar a una flota de vehículos. Cada error señalado ha sido anotado, priorizado y posteriormente subsanado y testeado de nuevo a fin de erradicar los posibles puntos débiles de la integración realizada.

Capítulo 6

Conclusiones

Índice

6.1. Conclusión en el ámbito formativo 71
6.2. Conclusión en el ámbito profesional 71
6.3. Conclusión en el ámbito personal 72
6.4. Futuras extensiones del proyecto 73

6.1. Conclusión en el ámbito formativo

Con la finalización de la estancia en prácticas y la escritura del presente documento, se cierra otro ciclo de aprendizaje más en el ámbito de las asignaturas del Grado en Ingeniería Informática. La EI1054 Prácticas externas y Proyecto Fin de Grado, ha supuesto el primer acercamiento al panorama laboral real que espera al alumno una vez haya finalizado sus estudios de Grado.

El enfoque que se le ha dado al proyecto de integración de sistemas propuesto ha sido sin duda una verdadera experiencia de aprendizaje en todos los sentidos. El primero de los aspectos positivos a destacar ha sido el haber trabajado con el ERP Microsoft Dynamics NAV, adoptando la forma de vertical del transporte. El haber podido comunicar el ERP con el sistema de mensajería de Movildata e integrar una serie de servicios ha sido la excusa perfecta para contar con un punto de inicio en el mundo de los ERPs. Si bien estas herramientas las conocía parcialmente desde un punto de vista de usuario por haber trabajado en varias empresas, no tenía claro su capacidad como herramienta de gestión de bienes empresariales ni el alto nivel de configuración que ofrece, tanto desde el rol usuario como desde el de desarrollador de código.

6.2. Conclusión en el ámbito profesional

La estancia en ACTM me ha parecido un periodo de aprendizaje realmente provechoso desde el punto de vista profesional. La estancia en prácticas ha tenido entidad de proyecto informático

real desde el comienzo. Quiero destacar algunos de los aspectos que han hecho que este proyecto haya sido desarrollado en un ambiente profesional:

- Ambiente de trabajo dinámico con gran sentido del compromiso.
- Obtención de los requisitos, verificación y validación de requisitos con el cliente.
- Trabajo parcial en la sede de los asociados.
- Múltiples consultas realizadas vía email y teléfono al personal de Movildata.
- Obtención de eventos para una orden de trabajo

En cuanto al trato recibido por el personal de cada una de las organizaciones involucradas en la integración, he de admitir que este ha sido excelente en todos los casos. Gracias al nivel de colaboración de todas las partes involucradas, el avance del proyecto ha sido exitoso y ha podido ofrecerse finalmente una solución informática que satisface la mayor parte de los requerimientos planteados desde un principio.

Contar con la experiencia de haber trabajado con los ERPs hace que aumente, en cierto modo, la cantidad de herramientas con las que he tenido contacto durante mi periodo formativo de Grado. Esta enriquecedora experiencia pudiera resultar en una futura línea de trabajo con los ERPs, informatizando algún aspecto en el sector del transporte o satisfaciendo las necesidades del día a día del mundo empresarial.

6.3. Conclusión en el ámbito personal

En lo personal, me gustaría comentar sobre la estancia en prácticas que, pese a ser un bloque formativo más dentro del plan de estudios de Grado, en mi caso sí he tenido la sensación de haber realizado un proyecto informático para un cliente real en un centro de trabajo. Esto ha sido así gracias a la configuración de la realización del trabajo propuesta por la supervisora. Gracias a esto, puedo contar con un acercamiento a lo que el panorama laboral real puede ser en el ámbito de la informática. Por ello y muchos otros aspectos estoy muy contento de haber realizado las prácticas junto a mi supervisora en el proyecto, Belén Collado, quien con altas dosis de positivismo, un gran tacto y siempre promoviendo el buen hacer de las cosas, me ha prestado ayuda y soporte cuando lo he necesitado. De Belén quiero destacar que es una gran profesional con un gran sentido de la responsabilidad, por lo cual la considero todo un referente como personal técnico informático de quien siempre se podría estar aprendiendo cosas nuevas.

Personalmente opino que he tenido mucha suerte al haber realizado la estancia en prácticas en ACTM debido al trato recibido por todos y cada uno de los compañeros de trabajo de la sede. En un ambiente distendido y abierto he podido realizar la estancia sin problema alguno. Por supuesto he contado con un espacio de trabajo tranquilo y dotado de las instalaciones y servicios requeridos básicos para poder realizar todas las tareas que se han planteado en el proyecto formativo.

Quiero agradecer también la labor de tutorización de Pablo Boronat, quien de modo práctico, ha sabido reconducir ciertos aspectos de enfoque sobre la estancia en prácticas y con el que guardo una relación cordial tras haber sido profesor en anteriores asignaturas del Grado. Pablo ha contribuido de forma activa en las correcciones de aquellos documentos liberados en forma de entregas quincenales y borradores del presente documento.

6.4. Futuras extensiones del proyecto

El proyecto plantea muchas posibilidades de mejora o de cambio de enfoque según se quiera ver. A continuación se detallan algunos aspectos que han quedado fuera del proyecto bien por falta de tiempo para su implantación, por escapar de los objetivos planteados inicialmente o por la falta de valor que aportaría en la actualidad a la empresa de transportes colaboradora.

Múltiples cargas y una única descarga

Actualmente, el modelo de trabajo de la empresa de transportes colaboradora presenta una estructura en los pedidos de venta de una única operación de carga de mercancía y una única operación de descarga. La descarga obviamente se efectuará en el punto que haya indicado el cliente que ha requerido del servicio de transporte.

Este modelo estático de trabajo permite realizar de forma sencilla una integración de servicios con la configuración de confección y seguimiento de órdenes de trabajo descrita para el *CodeUnit* "50007 - Funciones Navifleet". Sin embargo, la empresa de transportes cita un posible modo de operar que varía en estructura y para el que no encaja el modelo de trabajo anteriormente citado. La figura 6.1 trata de avanzar la configuración solicitada.

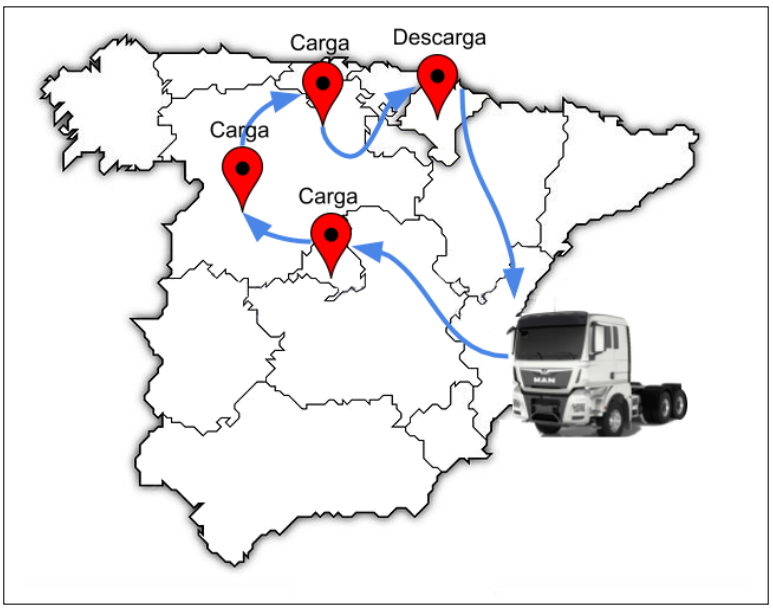

Figura 6.1: Varios puntos de carga, un único punto de descarga.

En un futuro, y de forma ocasional, podrían solicitarse servicios de transporte de mercancías que incluyan múltiples puntos de carga y un único punto de descarga. La complejidad que esto puede esconder puede ser alta. En lo habitual se espera que este servicio pueda asistirlo un único vehículo y por tanto, planificar los puntos de operación para el servicio y confeccionar la orden de trabajo para el conductor. No obstante, el servicio podría tener que ser atendido por varios vehículos debido a distintos motivos. Una posibilidad sería un servicio de transporte de mercancía compuesto por varios vehículos o una asistencia de un segundo vehículo, que sustituya al recurso inicialmente asignado, debido a alguna avería o requerimiento particular.

Por tanto, la integración con MFLEET para los actuales pedidos de venta que gestiona la empresa de transportes ya no nos sirve. Para poder dar soporte a la casuística aquí presentada se requiere de la posibilidad de confeccionar órdenes de trabajo con múltiples puntos de carga y una descarga en base a unas líneas de pedido asociadas a un servicio de transporte, las cuales aporten el detalle suficiente para componer las respectivas órdenes.

Para ello se requiere una construcción dinámica de órdenes de trabajo en base a la definición de datos que puedan contener dichas líneas de pedido. El problema no acaba ahí; la interpretación de las órdenes de trabajo podría variar en esencia y es que se requeriría de una interpretación de la estructura JSON que proporciona el servicio web de obtención de la trazabilidad incluido en el SDK de Navifleet.

Este requisito, planteado en las últimas reuniones mantenidas con el cliente, queda recogido en un documento que expone la última batería de requisitos y refinamientos para la integración. Dicho documento puede consultarse en el anexo D.

Puesto que la configuración de las estructuras y campos que darán soporte al nuevo requisito planteado todavía no se encuentran lo suficientemente maduras en el vertical del transporte, solo queda ofrecer una aproximación codificada que plantee una aproximación a la solución del problema.

La solución se presenta mediante un conjunto de métodos que ampliarían los ya descritos en el *CodeUnit* "50007 - Funciones Navifleet". El primero de los métodos que se presentaría es "OrderEnvy2". La estrategia de funcionamiento de esta segunda versión del envío de órdenes ya ha sido descrita anteriormente. Pasando como parámetro el identificador de un pedido de venta válido en el ERP a OrderEnvy2 este método debe recorrer las líneas de venta asociadas al pedido y componer la respectiva orden junto al conjunto de tareas de carga asociadas y la tarea de descarga final. Como se detalla en la figura 6.2, OrderEnvy requiere la composición de un número no definido de tareas de carga y una única tarea de descarga. La labor de composición de los bloques correspondientes a las tareas a realizar la llevará a cabo el método "generateBlock", componiendo bloques de tareas en formato JSON mediante la recepción directa de la información de cada línea de venta.

Para la interpretación de la trazabilidad de una orden que no sabemos cuántas tareas contiene, necesitamos otra estrategia diferente a la presentada con el método "getStatusOrder". Esta vez se requerirá un recorrido iterativo por el array "aoPeriod", evaluando cada uno de los bloques, extrayendo la información necesaria para poder cumplimentar debidamente las líneas de venta asociadas a la orden de trabajo.

Por tanto, para dar solución a la interpretación de la trazabilidad de esta nueva configuración de órdenes de trabajo se presentan los siguientes métodos:

TraverseBlocks. Este método recorrerá de principio a fin los bloques existentes en el array "aoPeriod" contenidos en la trazabilidad de una orden de trabajo. Tras posi-

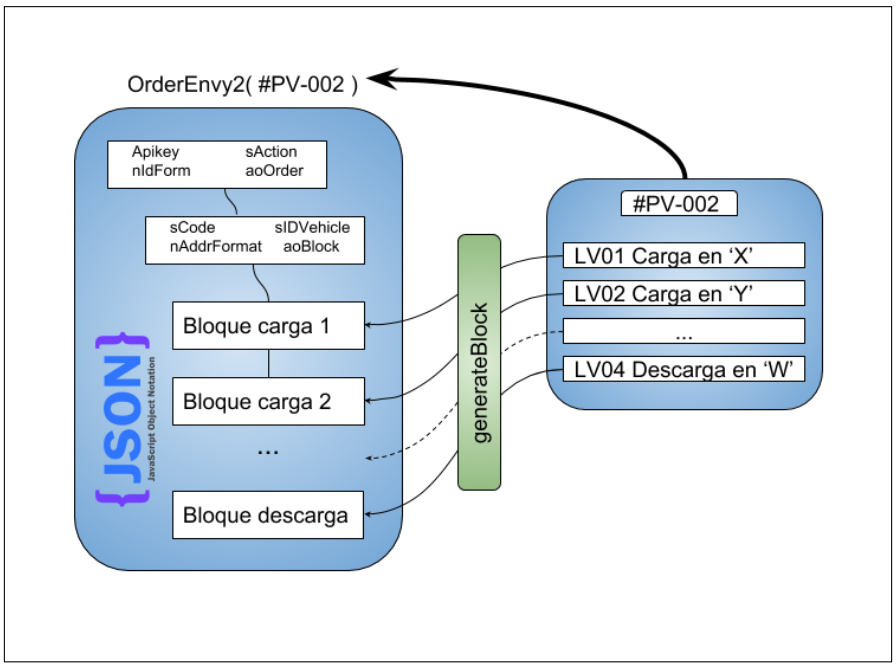

Figura 6.2: Fundamento de trabajo de OrderEnvy2.

cionarse sobre un bloque determinado, este es enviado al método "EvaluateBlock" para la extracción posterior de la información.

EvaluateBlock. La primera de las labores de este método es clasificar el bloque JSON que se recibe como parámetro. Posteriormente, si se trata de un bloque de "desplazamiento", "carga" o "descarga", se extraen los datos correspondientes haciendo uso de los métodos de interpretación de estructuras JSON desarrollados en el *CodeUnit* "50007 - Funciones NaviFleet" actualizando los datos para las líneas de venta directamente sobre el ERP.

Envío múltiple de órdenes de trabajo

Esta característica ha sido detectada durante las fases de pruebas de los servicios que ofrecía el SDK de Navifleet. Estando autenticado en dos dispositivos distintos con las credenciales de dos conductores de la empresa de pruebas, se ha podido realizar en un único envío, el envío de varias órdenes de trabajo dirigidas a ambos conductores. La llegada de las órdenes de trabajo se producía al mismo tiempo y sin incidencias aparentes. La validación de estados de las tareas de carga y descarga se realizaban de forma normal como venía siendo habitual en una configuración de envío para una única orden de trabajo para un único conductor.

Como conclusión se extrae que, con el SDK de Navifleet existe la posibilidad de poder realizar un envío múltiple de órdenes de trabajo si las labores de transporte que se han de realizar se tienen planificadas con la debida antelación y estas están lo suficientemente detalladas. La combinación de este planteamiento, de adaptarse al modo de trabajo de una empresa del transporte, junto a la automatización del planificador, presentarían una configuración de seguimiento de órdenes de trabajo con un alto nivel de automatismo,

precisando únicamente de los pertinentes avisos para cuando suceda alguna anomalía durante el ciclo de vida de un pedido de venta.

Confección de rutas

Esta labor perteneciente a los operadores de tráfico debería considerarse seriamente si realmente se quisiese rentabilizar al máximo los servicios de transporte que se prestan. Existe la posibilidad de elaborar una ruta en el entorno web Movilflota. Esta confección se realiza de forma sencilla a través de la declaración sobre mapa de un conjunto de puntos de paso que conforman una ruta de transporte determinada. La confección de rutas se debe a que los conductores tienen varias formas de llegar a un determinado punto destino. En muchas de las ocasiones se precisa sortear algún tipo de impedimento para el tipo de vehículo que presta el servicio, bien por pretender evitar peajes o bien para lograr desviar a los conductores y así conseguir rutas más eficientes.

Una vez confeccionada una ruta, esta puede ser incluida para la navegación que MFLEET propondrá al conductor que presta el servicio de transporte hasta un punto de destino determinado.

Algunas de las rutas que forzarían a un conductor a realizar una determinada ruta, podrían estar compuestas por unos pocos puntos intermedios y, gracias a este pequeño esfuerzo de elaboración, poder obtener una optimización significativa de los recursos destinados para un servicio de transporte. Esto no siempre será así. Por lo general, las empresas tienen ya definidas una gran cantidad de rutas que realizan, las cuales requerirían de la definición de un gran número de puntos intermedios. Para organizaciones con un sistema de trabajo que incluya una gran cantidad de rutas complejas, la labor de confección de rutas puede suponer un gran esfuerzo para el personal de gestión de tráfico.

La confección de rutas es pues una asignatura pendiente para aquellas empresas que utilicen un sistema de trabajo igual o similar al propuesto por Movildata y por tanto, deseen hacer aún más productivos sus servicios de transporte.

Bibliografía

[1] Google. Google maps platform. https://developers.google.com/maps/documentation/urls/guide. [Consulta: 6 de Junio de 2018].

[2] idese. 8043x c/side solution development in microsoft dynamics nav actualización a 2015. http://www.idese.es/bases-de-datos/cursos-de-dynamics-nav-navision-crm.html?task=view_event&event_id=639. [Consulta: 1 de Junio de 2018].

[3] Jonathan Archer. Time driven actions with .net class timer. https://community.dynamics.com/nav/b/moxie4nav/archive/2014/12/30/time-driven-actions-with-net-class-timer. [Consulta: 16 de Mayo de 2018].

[4] MiBuso. Navision timer in rtc. https://forum.mibuso.com/discussion/53528/navision-timer-in-rtc. [Consulta: 16 de Mayo de 2018].

[5] Microsoft. Using the pingpong add-in. https://docs.microsoft.com/en-us/dynamics-nav/using-the-pingpong-add-in. [Consulta: 16 de Mayo de 2018].

[6] Movildata. Movildata. https://movildata.com/. [Consulta: 7 de Junio de 2018].

[7] NoMagic. Magicdraw. https://www.nomagic.com/products/magicdraw. [Consulta: 6 de Junio de 2018].

[8] Postman. Postman makes api development simple. https://www.getpostman.com/. [Consulta: 6 de Junio de 2018].

[9] SublimeText. A sophisticated text editor for code, markup and prose. https://www.sublimetext.com/. [Consulta: 6 de Junio de 2018].

[10] w3schools. Xml soap. https://www.w3schools.com/xml/xml_soap.asp. [Consulta: 16 de Mayo de 2018].

Anexo A

Modelo de entrevista para los operadores de tráfico

Integración Movildata – Navision

Información habilitada:

- En lista de recursos: ver mapa.
- Ficha de recursos: ver mapa, km actuales.
- Expedientes de viaje: Al elegir la fecha final se calculan los kilómetros finales.
- Planificador de tráfico (Traffic board):
    - Enviar orden (No provoca apertura del navegador web).

Información con posibilidad a ser accedida desde Navision:

- Kilometraje fecha actual.
- Kilometraje entre fechas.
- Envío de mensajes.
- Recepción de mensajes.
- Lectura FMS/CAN-BUS entre dos fechas.
- Última posición conocida para un vehículo (Latitud/Longitud).

Nombre y apellidos del operador de tráfico:

¿Es de utilidad la información habilitada?

¿Ha de aparecer esta en algúna ventana más de Navision, aparte de las mencionados?

¿Hay alguna otra necesidad con respecto a Movildata que se deba incluir dentro de la aplicación de Navision?

Necesidad 1:

Necesidad 2:

Necesidad 3:

¿En qué partes de Navision ha de aparecer esta funcionalidad?

Parte 1:

Parte 2:

Parte 3:

NOTAS:

Anexo B

Resumen batería de entrevistas

Resumen batería de entrevistas realizadas al equipo de control de tráfico de la empresa de transporte Sanmartí

Realizado por Dionisio García García (al315729@uji.es). Jueves 8 de Marzo 2018

El presente documento trata de sintetizar las entrevistas realizadas a los operadores de tráfico de la empresa de transportes de mercancías Sanmartí, sita en Borriol (Castellón).

La **principal motivación** de esta entrevista es recoger el mayor número de propuestas de mejora con respecto al grado de integración que se dispone del ERP Microsoft Dynamics Nav y el servicio de seguimiento y control de flotas de Movildata.

La **obtención de requisitos** se ha llevado a cabo mediante la realización de una serie de entrevistas personalizadas a cada uno de los operadores en su propio puesto de trabajo, recogiendo el mayor número de planteamientos de mejoras con respecto a la labor de seguimiento y control de la flota.

El **perfil del entrevistado** es el de un operador de tráfico que ha de controlar las rutas de un conjunto de conductores que realizan servicios de transporte de mercancías en diferentes ámbitos territoriales, haciendo uso del ERP Dynamics Nav, el cual posee un gestor de pedidos de venta / planificador. Centrándonos en la parte de su trabajo que nos concierne, entre las labores que desempeñan los operadores de tráfico se encuentran:

- Registros de pedidos de clientes en Dynamics Nav..
- Manutención de un listado de recursos asignables. (Tractoras)
- Asignaciones de pedidos de clientes a los recursos en diferentes ámbitos territoriales (cada operador tiene asignado un ámbito territorial concreto).
- Labores de seguimiento de los pedidos. (Mediante el planificador)
- Labores de seguimiento de conductores. (Mensajería Movildata y telefonía)
- Registro de datos de forma manual en el sistema. (Kilometrajes, kilos cargados..)

Actualmente la labor de seguimiento de pedidos, se realiza mediante una interacción en paralelo con el uso del gestor de pedidos de venta / planificador y la aplicación de mensajería con interfaz gráfica de Movildata. Cuando la labor de seguimiento de los conductores por mensajería falla, los operadores recurren a realizar las oportunas llamadas telefónicas. La labor de recogida de información relativa a kilometraje y cantidad de kilos cargada para un servicio de transporte determinado, se realiza de forma manual, a través una conversación telefónica con el conductor. Una segunda opción es la recogida de datos en papel, una vez los conductores finalizan el servicio y vuelven a la sede de Sanmartí.

El total de entrevistas efectuadas sobre el equipo de operadores de tráfico recogen un número de propuestas de mejora considerable. Las propuestas realizadas por los operadores son bastante homogéneas entre sí y se centran básicamente en automatizar ciertos procesos sobre el planificador de Dynamics Nav a fin de evitar el paralelismo existente entre el ERP y el gestor de mensajería que ofrece Movildata mediante su interfaz gráfica web. Las mejoras propuestas están enfocadas a la forma en la que se hace el seguimiento de los conductores, haciendo especial énfasis en la introducción manual y de forma repetitiva a datos relativos a kilometrajes, kilos de carga y estado de los pedidos de venta. Es importante señalar que algunas de las propuestas realizadas por los operadores quedarían fuera del alcance de la integración Dynamics Nav - Movildata, sin embargo estas son recogidas igualmente para un estudio posterior de las mismas.

Una vez realizada la entrevista se recogen los siguientes aspectos:

- **Aspectos relativos** a la integración Dynamics Nav - Movildata:
    - Introducción de la cantidad de kilogramos cargada automáticamente, tras el envío por mensaje del conductor que hace el servicio.
    - Recibir alertas en Dynamics Nav por parte de los conductores.
    - Automatizar el estado para un pedido cuando un chófer haya cargado su mercancía.
    - Poder enviar mensajes desde el planificador a los conductores.
    - Recibir aviso personalizado a cada operador para aquellos conductores que hayan descargado.
    - Incluir en el planificador / gestor de pedidos, el kilometraje restante hasta llegar al punto de destino para un determinado servicio de transporte.
    - Automatizar el marcaje del campo “confirmado” para una orden de trabajo enviada. (Actualmente es marcado manualmente por el operador de tráfico tras leer el correspondiente mensaje enviado por conductor). Esto debería realizarse de forma automática recibiendo una señal por parte del conductor una vez ha recibido la tarea, por ejemplo un “OK”.
    - Control / seguimiento de lectura de mensajes por parte de los conductores.
    - Modificar el envío de mensajes a conductores, evitando que se abra el navegador cada vez.
    - Pulsar sobre un recurso en concreto en la “Matriz de secuencia de recursos” y poder ver el posicionamiento de un vehículo en un mapa de Google Maps.
    - Incluir el total de POIs dados de alta en la aplicación de interfaz gráfica de mensajería de Movildata en el ERP Microsoft Dynamics Nav.

- **Aspectos no relativos** a la integración Dynamics Nav - Movildata:

- Eliminar el modal / diálogo de confirmación: “Enviar comentarios de la ruta” por falta de relevancia. (Incluido en la secuencia de pasos al enviar una orden de trabajo a un conductor a través del planificador).
  Posible solución: *el marcaje de esta opción tiene consecuencias de envio extra de información a un conductor. Estudiar caso.*
- Unificar la información de envío de orden a un conductor. Esto actualmente se realiza en dos pasos, un primer envío de datos mayor y un posterior envío de la dirección concreta del punto de llegada del conductor.
- Sistema de vistas para la información del conjunto de conductores que ha de monitorizar cada operador de tráfico en el planificador sobre el que opera. **Posible solución:** *Uso de filtros de pedidos de venta.*
- Mejorar la legibilidad / grado de interpretación de la “Matriz de secuencia de recursos”.
  - Con colores, resaltando aquellos servicios de transporte que puedan sufrir retrasos. **Problema:** Estaría duplicandose la filosofía del planificador / gestor de pedidos de venta.
  - Replicar la información del servicio que se lleva a cabo para aquellos servicios que conlleven más de un día trabajo y así favorecer la interpretación de la matriz por parte de los operarios.

NOTAS:

- Tal como se tiene ahora el planificador, la comunicación por parte de un conductor de que ha descargado su mercancía conlleva el tener que marcar manualmente por parte del operador la casilla de “descargado” de la línea de venta del planificador, desapareciendo esta del planificador, y pudiendo perder el control sobre el conductor a fin de asignarle la siguiente tarea. Los operarios recurren a la “Matriz de secuencia de recursos” para ver los últimos movimientos para ese conductor en concreto.
- Algunos operadores mencionan que el sistema de mensajería no es infalible, haciendo alusión a problemas de cobertura, con la consecuente molestia de tener que realizar un seguimiento telefónico sobre el conductor que realice un servicio.
- Según los operadores, no todos los conductores realizan la labor de comunicación de estado del servicio haciendo uso de los dispositivos de comunicación que montan los vehículos, ya que no están en la obligación de hacerlo.

Al finalizar la ronda de entrevistas, se recuerda por parte del gerente que la integración de nueva funcionalidad al planificador de Dynamics Nav ha de realizarse sin sobrecargar la pantalla, evitando duplicidades de información y ha de velarse en todo momento por una gestión ligera de datos en los equipos. Se entiende pues, que aquella funcionalidad candidata a ser implantada ha de ser validada por el conjunto de operarios de tráfico y gerencia en todo momento.

Anexo C

Propuesta de automatización del patrón de mensajería

Propuesta de automatización de la interacción operador - conductor

El presente documento trata de poner de manifiesto un estudio sobre la posible implantación de la automatización del patrón de interacción por mensajería existente entre un operador de tráfico y un conductor que realiza un servicio de transporte. Para la realización de este estudio se cuenta con una empresa cuya actividad se centra en el transporte de mercancias por carretera, la cual servirá de modelo.

La presente propuesta surge de la necesidad real de mejorar el conjunto de sistemas de la información que actualmente permiten realizar las labores de seguimiento para un pedido de venta. La gestión de la recepción de la información proveniente del conductor que realiza el servicio en cuestión, es enteramente labor del equipo de operadores de tráfico.

Actualmente se reciben datos desde el canal de mensajería de la aplicación web de Movilflota, por vía telefónica y en forma de documentación escrita una vez esta llega a la sede. Los principales inconvenientes de este modo de proceder, es la tediosa labor que los operadores de tráfico realizan, buscando de forma activa por varios medios de comunicación los items de información requeridos. La información suele llegar con errores y a destiempo, impidiendo que un pedido de venta siga el curso natural de gestión y facturación al cliente final, con las consecuentes perdidas que el retraso supone.

A continuación se presenta el patrón de interacción detectado entre un operador de tráfico y un transportista en su versión teorico-optimizada:

- Un pedido de venta es dado de alta en el ERP Dynamics Nav debido a un encargo de un cliente. El pedido tendrá en este instante el estado de “sin asignar”. Una vez el operador de tráfico selecciona un recurso disponible (Tractora) para realizar el servicio de transporte, casa el pedido de venta con el recurso. En este instante el pedido tiene el estado de “asignado”. Este suceso conlleva la notificación correspondiente al respectivo conductor al cargo de dicho recurso para dar comienzo al servicio. El operador de tráfico realiza entonces un envío de orden de trabajo al condutor.

- El conductor recibe la orden de trabajo con el nivel de detalle suficiente para poder desempeñar el servicio. (Indicaciones, dirección de carga, dirección de descarga, tipo de mercancia...). El conductor debe poner en concocimiento al operador de tráfico que ha recibido correctamente la orden y por tanto que va a proceder a dar comienzo al servicio que se ha de realizar. El conductor notifica al operador que ha recibido la orden.

- El operador de tráfico recibe la confirmación por parte del conductor. En este punto se entiende que el conductor ha comenzado con el servicio. El sistema cambia el estado del pedido de venta a “Confirmado”. El operador envía un mensaje preguntando si se ha cargado la mercancía a transportar, pendiente de confirmar por parte del conductor.

- El conductor, una vez se haya cargado la mercancia en el vehículo, debe contestar al mensaje de confirmación de carga de mercancía con un “SÍ”.

- El operador de tráfico recibe el “Sí” que confirma la operaciónde carga. El sistema de forma automática debe cambiar el estado del pedido de venta a “Cargado”. En este momento se efectua una lectura de kilometraje sobre el vehículo, registrando fecha y hora de carga reales.

- Se enviará un mensaje de forma automática al conductor pidiéndole la introducción de la cantidad de kilogramos cargada.

- El conductor recibe la petición de la introducción de kilogramos de carga. Inmediatamente después de concluir la operación de carga se procederá a introducir la cantidad de kilogramos.

- De forma transparente, se recibe en el sistema la cantidad de kilogramos introducida por el conductor, actualizando el correspondiente campo del pedido.

- De nuevo se enviará un mensaje al conductor para validar el momento de descarga de la mercancía.

- El conductor contesta al mensaje notificando que ha descargado con un “SÍ”.

- El sistema recoge el mensaje del conductor y marca la línea de pedido como “Descargado”. En este punto se realiza una segunda lectura de kilometraje del vehículo, registrándose la fecha y hora de descarga reales.

La figura 1.1 recrea el patrón de interacción descrito anteriormente de forma conceptual:

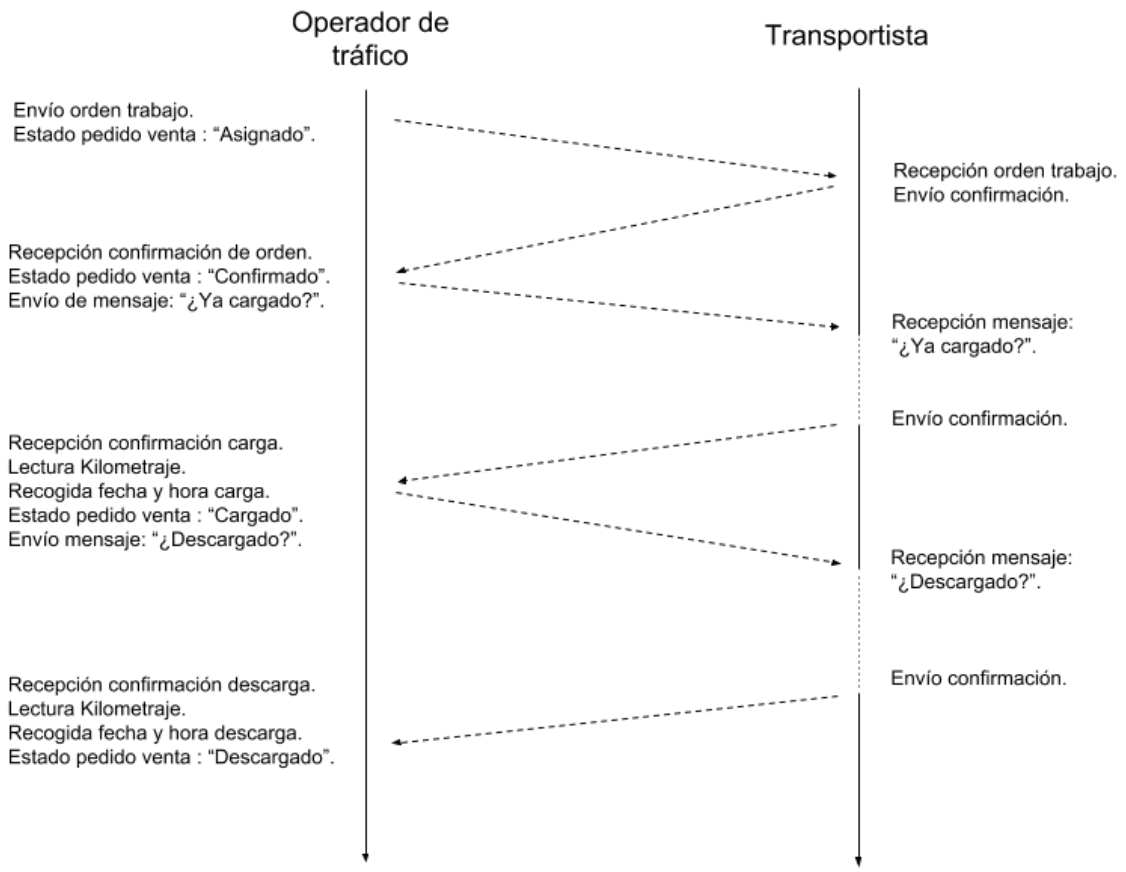

Figura 1.1 - Esquema de mensajería entre operador de tráfico y transportista.

Con la información recopilada sobre el sistema de gestión de flotas de vehículos propuesta por el SDK de Navifleet y la aplicación android MFLEET, se procede a realizar un estudio que permita emular el patrón de interacción descrito. A continuación, en la figura 1.2 se detalla una posible interacción implantable con un nivel de detalle superior.

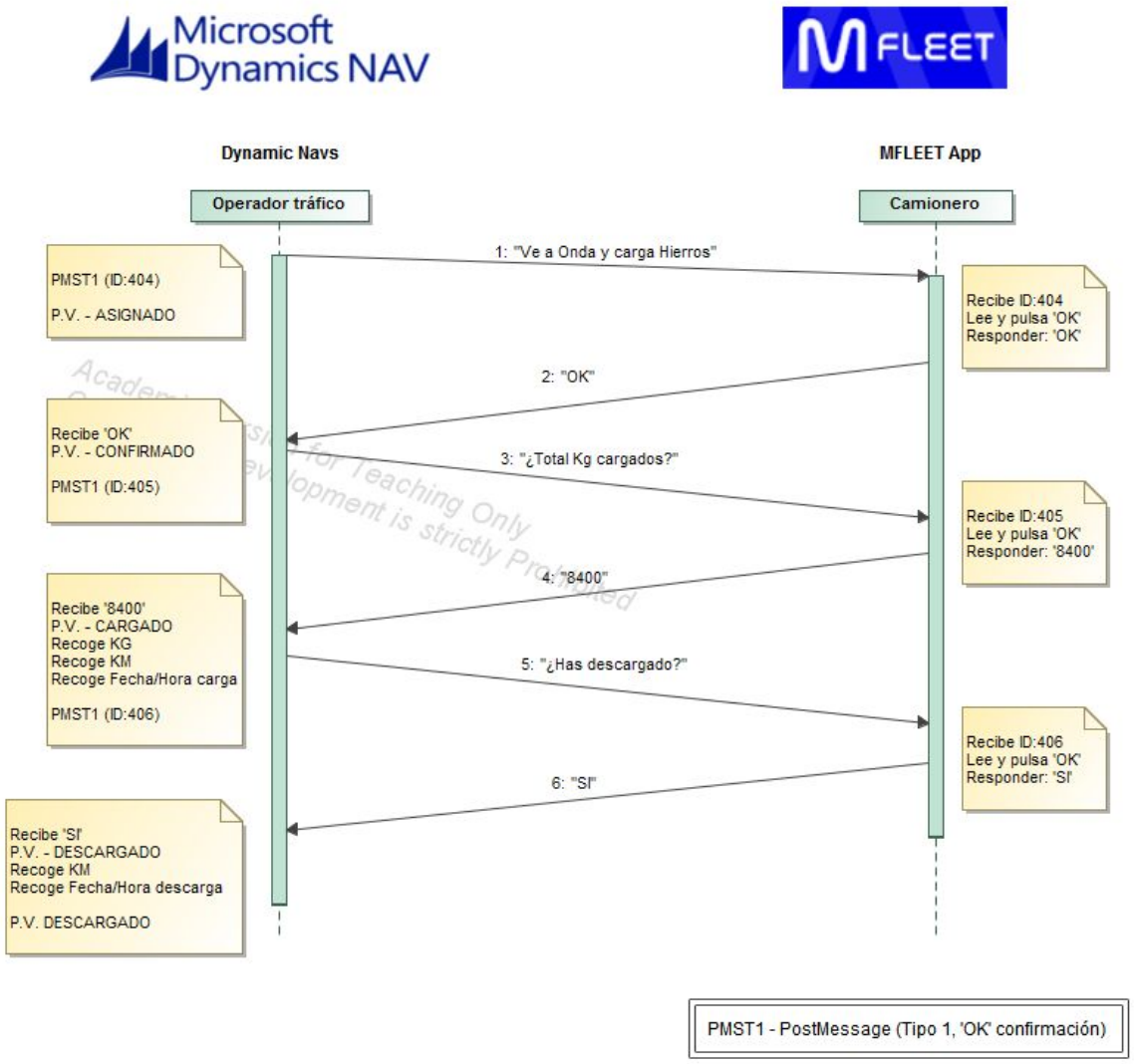

Figura 1.2 - Patrón de interacción implantable en el ERP.

Requisitos para la interacción:

- Dotar a los vehículos del servicio MFLEET (terminal android + APP) para realizar la comunicación.
- Implantación de Juego de llamadas en Microsoft Dynamics Nav más habilitación de objetos y funcionalidad para el procesado de la información.
- Formación de los transportistas (prácticamente nula) sobre el patrón de interacción con los operadores de tráfico. Se requiere una introducción de información precisa.
- Colaboración entre operadores de tráfico y transportistas para la comunicación automatizada.

Posibles problemas de la interacción

- Falta de cobertura en el envío de mensajes.

- No seguir el proceso de envío de datos a través de los dispositivos por parte de los conductores. (posibles causas: olvidos, ser reacio a las tecnologías).

- Introducción de información errónea por parte del conductor de forma accidental. **(**Esto es algo que ya ocurre con la toma de información de forma manual. Nunca se esta exento de cometer errores.)

Ventajas de la implantación de la automatización propuesta

- Recepción de la información de forma automática y al instante.

- Posibilidad real de automatización de cambio de estado para los pedidos de venta. (SinAsignar / Asignado / Cargado / Descargado).

- Datos con un mayor grado de fiabilidad. Se fomenta la toma de datos directamente desde la fuente donde se originan, evitando pasos intermedios o esperas de tiempo.

- Simplificar la labor de envío de datos por parte del transportista. Se trata de una interacción breve y concisa. Requiere el envío de tres mensajes, más el procesado de sus respuestas.

- Simplificación del proceso de recogida de datos por parte del operador de tráfico. Automatización de tareas manuales y repetitivas que pudieran ser una fuente de errores en la información a registrar.

Anexo D

Última toma de requisitos

Última recogida de mejoras Integración Movildata-Dynamics NAV

Fecha de realización: 16/4/2018 **Lugar de encuentro**: Sanmartí

Presentes: Departamento de informática de ACTM, Jefe de Tráfico, Jefe de Administración, Gerencia.

La Integración de servicios web de Movildata con el ERP Microsoft Dynamics NAV llega a sus últimas fases. Actualmente se cuenta con un sistema semi-automático de recogida de información que permite actualizar los estados de los pedidos de venta en el planificador, así como recoger información relativa a los servicios que son realizados por los profesionales del transporte. También son abordadas tareas de seguimiento de vehículos, como lo son posicionamiento en un determinado instante, cálculo de kilometrajes y gestión de puntos de interés (POI's).

Se pretende pues, tras mostrar los resultados del actual estado de la integración, recoger un feeedback que permita refinar el trabajo realizado.

1. Al finalizar una orden de trabajo, estado al cual se llega si se ha descargado la mercancía de un vehículo y recepcionado los correspondientes datos, antes de desaparecer el pedido de venta del planificador, se debe mostrar un mensaje de confirmación al operador de tráfico.
   MOTIVACIÓN: aporte de un mayor nivel de control sobre la información. (Evitar el efecto de desaparición visual de la información).

2. Posibilidad de atender órdenes de trabajo con múltiples puntos de carga y un único punto de descarga.

3. Al cancelar una orden de trabajo enviada a un operario desde el planificador, esta debe reestablecer su estado a "asignado". Actualmente refleja el último estado registrado en la traza de acciones que ha guardado el servidor de Movildata.

4. Incluir en el formulario, la dirección completa del punto de carga o descarga. Incluir número de documento externo. Incluir descripción del material que se transporta.

Incluir refencia del producto. (Ver bloque de información que actualmente envían los operadores de tráfico usando la plataforma web de Movilflota).

5. Colorear un pedido de venta en el planificador si esta registra algún tipo de incidencia reportada por el conductor que realiza dicho servicio de transporte.

6. Incluir latitud y longitud en la recogida de información para una operación de carga o descarga.

7. Recogida de documentos enviados por los transportistas, vinculados a una orden de trabajo, a través de MFLEET.

8. Posibilidad definir rutas en una orden de trabajo.

Índice de figuras

1.1. Aspecto de la aplicación MFLEET. 9

2.1. Planificador de tráfico. 14

2.2. Aplicación web Movilflota. 15

2.3. SDKs de Movildata. 18

3.1. Work Breakdown Structure inicial presentado en la propuesta técnica. 27

4.1. Diagrama de casos de uso de la integración de servicios. 32

4.2. Arquitectura de comunicación entre sistemas. 33

4.3. Estructura de los *CodeUnits* a desarrollar. 34

5.1. PostMessageSimple Postman. 37

5.2. Codificación de "PostMessageSimple" en lenguaje C/AL. 39

5.3. Documentación en línea para "PostMessageSimple". 41

5.4. Codificación del servicio web de cancelación de órdenes de trabajo. 42

5.5. Respuesta del servidor al cancelar la orden "PV14-001". 42

5.6. Ámbito de los métodos del *CodeUnit* "50006 - Funciones Movildata". 43

5.7. Flujo de llamadas a métodos producido por "PostMessageSimple". 45

5.8. Estructura del vector de llamada SOAP. 46

5.9. Construcción dinámica de una llamada SOAP. 48

5.10. Patrón de interacción operador de tráfico - transportista. 51

5.11. Configuración de módulos base en MFLEET. 52

5.12. Modulo de transportes en MFLEET. 53

5.13. Ventanas para una orden de trabajo en MFLEET (1/2). 54

5.14. Ventanas para una orden de trabajo en MFLEET (2/2). 55

5.15. Gestión de Incidencias, descansos y gastos en MFLEET. 56

5.16. Cuerpo de mensaje JSON para envío de orden. 57

5.17. Clasificación del estado devuelto por el servidor para “OrderEnvy”. 58

5.18. Estructura de la cadena de texto compuesta. 60

5.19. Trazabilidad para la orden de trabajo “PV002”. 61

5.20. Codificación “getStatusOrder”. 63

5.21. Interpretación del Dataset “GetListPOIS”. 69

6.1. Varios puntos de carga, un único punto de descarga. 73

6.2. Fundamento de trabajo de OrderEnvy2. 75

Índice de cuadros

3.1. Costes en bruto para el proyecto de integración de sistemas 25

3.2. Costes en bruto para el proyecto de integración de sistemas 25

5.1. Variables DotNet asociadas a servicios web. 38

5.2. Funciones incluidas en el *CodeUnit* "50006 - Funciones Movildata" 44

5.3. Parámetros de definición de una orden de trabajo. 56

5.4. Parámetros de definición de una tarea incluida en una orden de trabajo. 57

5.5. Funciones incluidas en el *CodeUnit* "50007 - Funciones Navifleet". 64

5.6. Funciones incluidas en el *CodeUnit* "50006 - Funciones Movilflota". 67

5.7. Variables DotNet asociadas a interpretar el "Dataset" 68