Universitat de Girona
Escola Politècnica Superior

Projecte/Treball Final de Carrera

Estudi: Eng. Tèc. Informàtica de Gestió. Pla 1993

Títol:

Servei de retirada de vehicles de la via pública del terme municipal de València basat en localització GPS i comunicacions GPRS

Document: Memòria

Alumne: Lluís Gener Estañol

Tutor: Josep Soler i Masó
Departament: Informàtica i Matemàtica Aplicada
Àrea: LSI

Convocatòria: Setembre de 2006

1 – Introducció i objectius 4
1.1 – Empreses implicades 4
1.2 – Sistema de gestió de flotes 5
1.2.1 – Localització geogràfica 6
1.2.2 – Gestió de la informació 6
1.2.3 – Transmissió de dades 6
1.2.4 – Tècniques englobades 6
1.3 – Situació prèvia al projecte 7
1.4 – Objectius del PFC 8
1.5 – Solució proposada 9
1.6 – Implicació en el projecte 10
2 – Tecnologies implicades 11
2.1 – PDA (Personal Digital Assistant) 11
2.2 – Plataforma Micronav.Net Serveis 11
2.3 – Oracle 12
2.4 – Arquitectura del sistema 12
2.4.1 – Arquitectura de servidors 13
2.4.2 – Aplicació client 14
2.4.3 – Elements de mobilitat 16
2.5 – Arquitectura client – servidor 17
2.6 – Sistemes de representació gràfica 18
2.6.1 – Tècniques de representació 20
2.7 – Protocol SOAP 21
2.8 – Serveis Web 21
2.8.1 – Motius per a fer servir serveis web 22
2.9 – Servlets 22
2.9.1 – Cicle de vida d'un Servlet 22
2.10 – Sistema GPS 23
2.10.1 – Càlcul de la posició 24
2.11 – El servei GSM / GPRS 24
3 – Requeriments i especificacions 26
3.1 – Projecte intern 26
3.1.1 – Micronav.Net 26
3.1.2 – Micronav.Net Serveis 26
3.1.3 – Servicleop 27
3.2 – Estat anterior del sistema 28
3.3 – Requeriments del sistema Servicleop 28
3.4 – Fluxogrames i parametrització específica 30
3.4.1 – Flux de serveis 30
3.4.2 – Tipus de serveis i operacions 32
3.4.3 – Paràmetres de serveis 32
3.4.4 – Estats i paràmetres de vehicles 34
4 – Anàlisi del sistema 35
4.1 – Anàlisi general del sistema 35
4.2 – Elements de mobilitat 36
4.2.1 – PDAs 36
4.2.2 – K8s 39
4.3 – Clients del centre de control 39
4.3.1 – Gestió de serveis 40

4.3.2 – Funcionalitat del client centre de control 41
4.4 – Servidors 42
4.4.1 – Servidor GPRS 42
4.4.2 – Servidor de Temps Real 44
4.4.3 – Servidor J2EE 46
4.4.4 – Servidor Oracle 50
5 – Disseny del sistema 52
5.1 – Disseny de base de dades 52
5.1.1 – Administració 53
5.1.2 – Usuaris 54
5.1.3 – Vehicles 56
5.1.4 – Missatgeria 60
5.1.5 – Itineraris 61
5.1.6 – Internacionalització 62
5.1.7 – Gestió de serveis 63
5.2 – Disseny de comunicacions 67
5.2.1 – Flux de comunicacions 67
5.3 – Trames existents 71
5.3.1 – Vehicle / PDA -> Servidor 71
5.3.2 – Servidor -> Vehicle / PDA 77
6 – Implementació 81
6.1 – Entorns, eines i tecnologies de desenvolupament 81
6.2 – Disseny de XMLs per als Servlets 86
6.2.1 – Servlet de Login 86
6.2.2 – Servlet de sol·licituds 88
6.2.3 – Servlet històric de posicions 89
6.2.4 – Servlet històric de serveis (només Micronav.Net Serveis) 91
6.3 – Disseny WSDL per als Webservices 92
6.3.1 – WSRealTime 92
6.3.2 – WSServices 98
7 – Implantació 115
7.1 – Instal·lació 115
7.2 – Fase de proves 116
7.3 – Certificació del sistema 116
7.4 – Cursos de formació 117
7.5 – Manteniment 117
8 – Participació en el PFC 118
8.1 – Abast temporal 118
8.2 – Abast econòmic 120
9 – Conclusions 121
9.1 – Resultats obtinguts 121
9.2 – Perspectives de futur 122
9.3 – Resultats a nivell personal 123
10 – Bibliografia i planes web 124

1 – Introducció i objectius

Knosos és una empresa d'enginyeria de telecomunicacions que durant els darrers anys s'ha dedicat al desenvolupament d'eines d'informació basades en GPS (Global Position System), comunicacions i Internet. Això ha permès oferir solucions tecnològiques adaptades a les necessitats específiques de clients relacionats amb serveis públics i d'emergència, així com a companyies de transport i qualsevol altre que tingui una estreta relació amb el sector logístic o una component de mobilitat important.

El projecte que es presenta ha estat desenvolupat en aquesta empresa per un equip del qual he format part, i tal com veurem en els capítols següents, consisteix en el desenvolupament d'una plataforma que mitjançant localització GPS i comunicacions GPRS permet als encarregats del servei de grua de l' Ajuntament de València comunicar als conductors de les grues que circulen per la ciutat una sèrie de serveis de retirada de vehicles, optimitzant el procés gràcies a la cartografia, les ajudes desenvolupades en el software i a l'emmagatzematge automàtic de totes les accions portades a terme.

1.1 – Empreses implicades

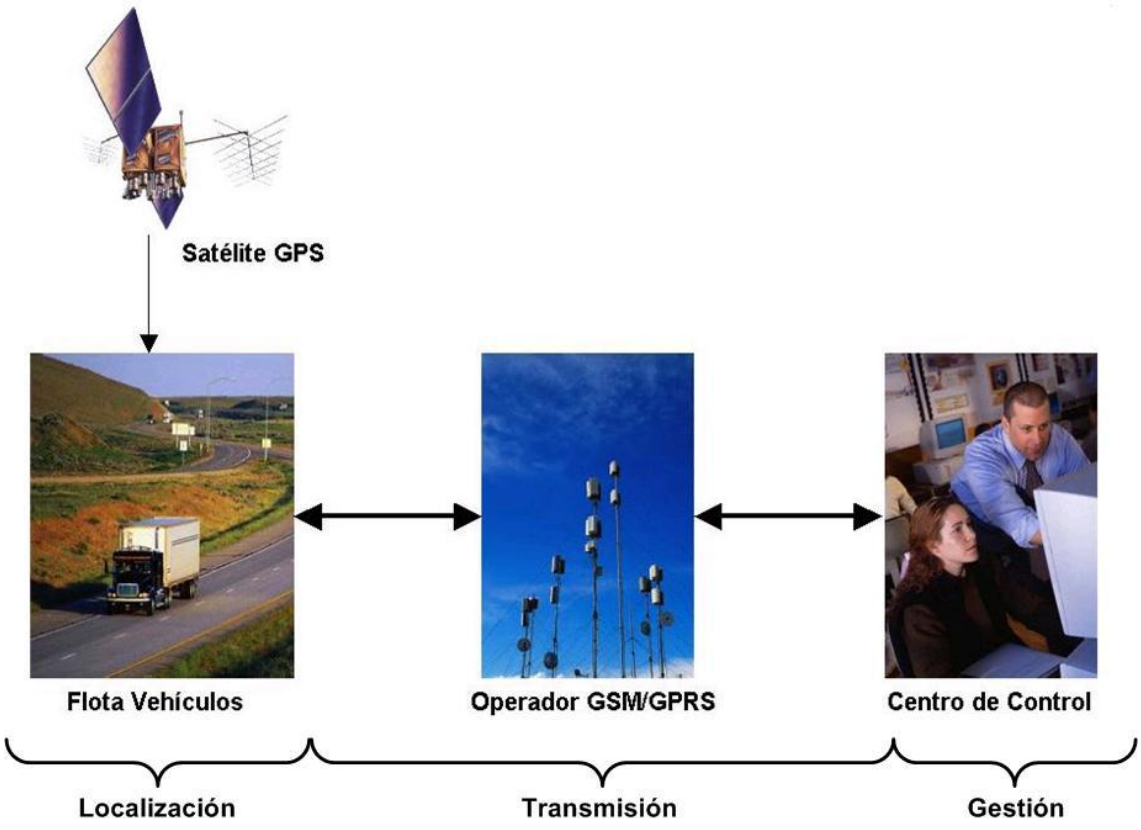
Servicleop és l'empresa a la qual l' Ajuntament de València ha adjudicat la gestió de retirada de vehicles de la via pública durant els 8 propers anys.

Cesser és una empresa escollida per l' Ajuntament de València per a què els ofereixin eines per a l'optimització de recursos a través de l'explotació de dades.

Knosos ha estat l'empresa escollida per Servicleop per dur a terme el desenvolupament del sistema de retirada de vehicles de la via pública per a l' Ajuntament de València, mitjançant l'ús de PDA, localització GPS i comunicacions GPRS que es presenta en aquest PFC.

1.2 – Sistema de gestió de flotes

Per tal d'entendre què entenem per un sistema de gestió de flotes de vehicles, mostrarem un esquema bàsic de tot sistema d'aquest tipus, diferenciant les parts implicades.



1.2.1 – Localització geogràfica

Consisteix en calcular la posició geogràfica d'un vehicle en el nostre cas en un determinat sistema de coordenades. En el cas GPS, la posició ve determinada per la latitud i longitud. Per a tal efecte és necessari que el vehicle disposi d'un receptor GPS, capaç de processar la informació rebuda dels satèl·lits GPS.

1.2.2 – Gestió de la informació

Consisteix en tractar, emmagatzemar, gestionar i explotar la informació procedent dels vehicles. Acostuma a incloure la capacitat de representar sobre cartografia digital les posicions rebudes dels vehicles. Aquesta part del sistema es denomina centre de control, i consta de diverses aplicacions, algunes de les quals són l'objecte d'aquest projecte de final de carrera.

1.2.3 – Transmissió de dades

Es tracta de l' intercanvi de dades entre els vehicles i el centre de control. Els vehicles envien periòdicament la seva informació de localització, així com altre informació, a través de diferents canals de comunicació, com poden ser GSM, TETRA o, com en el cas que ens ocupa, GPRS. El centre de control recull aquestes dades i també és capaç d'enviar informació cap al vehicle, que la rebrà en un terminal de dades (en el nostre cas una PDA).

1.2.4 – Tècniques englobades

Com hem pogut veure, en un sistema de gestió de flotes hi intervenen diverses tècniques:

- Electrònica: apareix en tots els components del sistema, tot i que on té més importància és en els receptors GPS del vehicle, encarregats de processar la informació de posicionament i enviar-la pel canal GPRS, i les PDAs que serviran d'interfície als conductors de la grua per a rebre i enviar informació al centre de control a través també de GPRS.
- Telecomunicacions: Permeten la comunicació entre els vehicles i el centre de control.
- Informàtica: Vital en tots els punts anteriors per a poder gestionar i guardar la informació, així com per a oferir una interfície a l'usuari perquè pugui interactuar amb el sistema.

1.3 – Situació prèvia al projecte

Prèviament a la implantació d'aquest projecte, Servicleop s'encarregava de gestionar el servei de recollida de vehicles mitjançant el canal de veu. Aquests sistema és ineficient ja que des del centre de control no es sap on està i en quin estat es troba cada grua. A més, l'evolució de l'estat dels serveis de grua no queda enregistrat automàticament, cosa que dificulta la gestió.

Després d'un anàlisi, s'arriba a la conclusió que es pot millorar en els aspectes següents:

- Localització: sabent en tot moment on es troba cada grua, és senzill per a un operari del centre de control assignar una retirada a la grua que es trobi en millor situació.
- Estats: sabent com es troba cada grua en cada moment (si ja porta un vehicle enganxat, si està espatllada, etc.) ens estalviem peticions que no podran ser ateses.
- Productivitat: a més dels 2 punts anteriors, també contribueix a la productivitat el fet que tot el procés de comunicació quedi automàticament enregistrat en base de dades i que les interfícies d'usuari tant per a operaris del centre de control

com per a operadors de grua facilitin l'entrada i consulta de dades (a través de llistes, desplegables, i informant-nos dels camps obligatoris, etc.).

1.4 – Objectius del PFC

L'objectiu principal d'aquest PFC és desenvolupar un sistema informàtic per tal d'augmentar l'eficiència del servei de retirada de vehicles de la grua de València.

El sistema a desenvolupar ha de satisfer els requeriments establerts per Servicleop, que són els següents:

Requeriments tecnològics:

- Localització GPS en temps real: cada grua anirà equipada amb un localitzador GPS (K8), que a través de GPRS enviarà periòdicament la seva posició al centre de control. Es disposarà d'una aplicació amb cartografia digital que permetrà veure la posició i l'estat de cada grua en tot moment per a què els operaris del centre de control puguin assignar les retirades amb la major eficiència.
- Serveis: un servei consisteix en la retirada i trasllat a dipòsit d'un vehicle, o en el trasllat entre dipòsits. Els serveis seran enviats als operaris de la grua que els rebran en una PDA a través de GPRS, i seran generats pels usuaris del centre de control que hauran d'introduir certes dades sobre el vehicle a retirar a més d'indicar en cartografia la localització d' aquest. També serà possible que, en la finalització d'un servei, una PDA demani al centre de control a petició del gruer (conductor de la grua) una còpia del servei més recent (per exemple si en el carrer hi havia més vehicles mal estacionats).
- Emmagatzematge: tant les peticions de centre de control, com les accions per part dels gruers, així com les posiciones de les grues, han de quedar registrades automàticament en base de dades, per a la posterior consulta i explotació de dades.
- Explotació de dades: es vol tenir un mecanisme per a exportar les dades des del sistema de Knosos cap al sistema de Cesser, amb la finalitat de generar estadístiques.

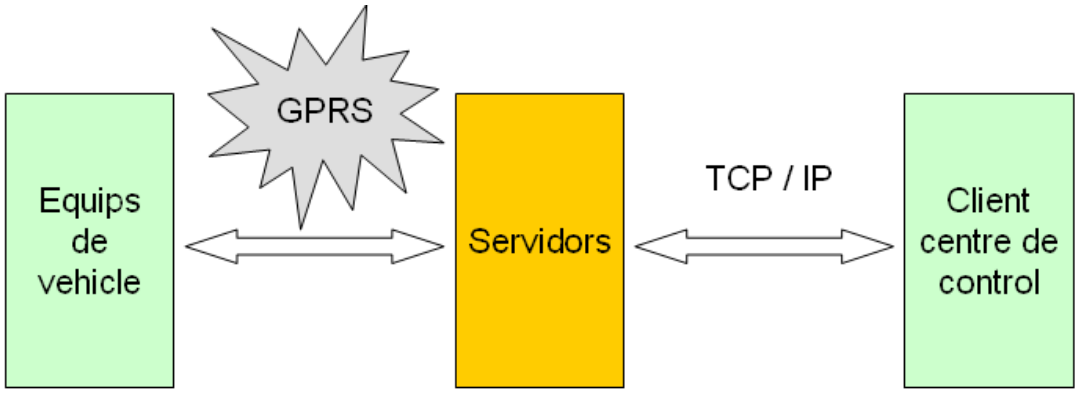
Requeriments procedurals:

- Creació de serveis: entrar la localització i dades d'un vehicle així com del seu destí per tal que la grua el pugui recollir i traslladar.
- Missatgeria: enviar des del centre de control cap a la PDA un missatge de text (com si es tractés d'un SMS).
- Estats: permetre a l'usuari de la PDA indicar l'estat del servei actual, o l'estat operatiu de la grua (si està en un descans, espatllada, etc.).
- Auto-assignació de serveis: una grua ha de ser capaç de demanar sense interacció de centre de control una còpia del seu servei més recent.

1.5 – Solució proposada

En aquest punt es presentarà un esquema de la solució proposada, basada en una estructura amb els següents elements:

- Equips de vehicle
- Client de centre de control
- Servidors
- Comunicacions remotes



Aquest esquema serà detallat i explicat més endavant en aquest PFC.

1.6 – Implicació en el projecte

La meva participació en el conjunt d'aquest projecte ha estat la següent:

- Requeriments i especificacions: he participat en tot el projecte.
- Anàlisi i disseny: he participat en tot el projecte, especialment en l'apartat de servidors.
- Implementació: he participat molt activament en la implementació de les aplicacions servidores, i en ampliació o correcció d'errors de funcionalitats ja existents.
- Implantació i manteniment: he participat en la implantació de les aplicacions de la part servidora en la seva instal·lació, configuració, comunicacions, etc. També he tingut (i tinc) molta participació en la part de manteniment, tant solucionant incidències tècniques com en resolució de dubtes per part del client final.

Aquest projecte ha estat desenvolupat per un equip de 4 persones, format pel director del projecte i 3 desenvolupadors (PDA, Client de centre de control i Servidors), durant uns 4 mesos. En un capítol posterior, veurem la dedicació de l'equip a cadascuna de les tasques comentades.

2 – Tecnologies implicades

En aquest apartat es descriuran les diverses arquitectures i tecnologies de hardware, software i comunicacions utilitzades en aquest projecte.

2.1 – PDA (Personal Digital Assistant)

Al mercat existeixen molts models de PDA, tots ells amb comunicacions GPRS integrades. En el projecte actual s'ha escollit una PDA model Qtek 9090, que es tracta d'un terminal Pocket PC. Aquest fet ofereix l'avantatge de poder desenvolupar aplicacions utilitzant l'entorn de programació Microsoft.Net, cosa que afavoreix el desenvolupament i manteniment de l'aplicació per temes d'optimització de recursos informàtics.

2.2 – Plataforma Micronav.Net Serveis

La plataforma Micronav.Net Serveis (MNNs) de Knosos és part de l'objecte d'aquest projecte, i parteix d'una plataforma de la qual ja disposava Knosos anomenada Micronav.Net (MNN), en explotació des de juny de 2003. Hereda d'aquesta plataforma la gestió de flotes, itineraris i missatgeria, i l'amplia amb funcionalitats orientades a clients de serveis amb l'objectiu d'una màxima parametrització per tal de poder englobar diferents tipus de clients sense tocar ni una línea de programari. Com anirem veien, el PFC actual basa la seva arquitectura en aquesta plataforma, fent-la a la mida del client.

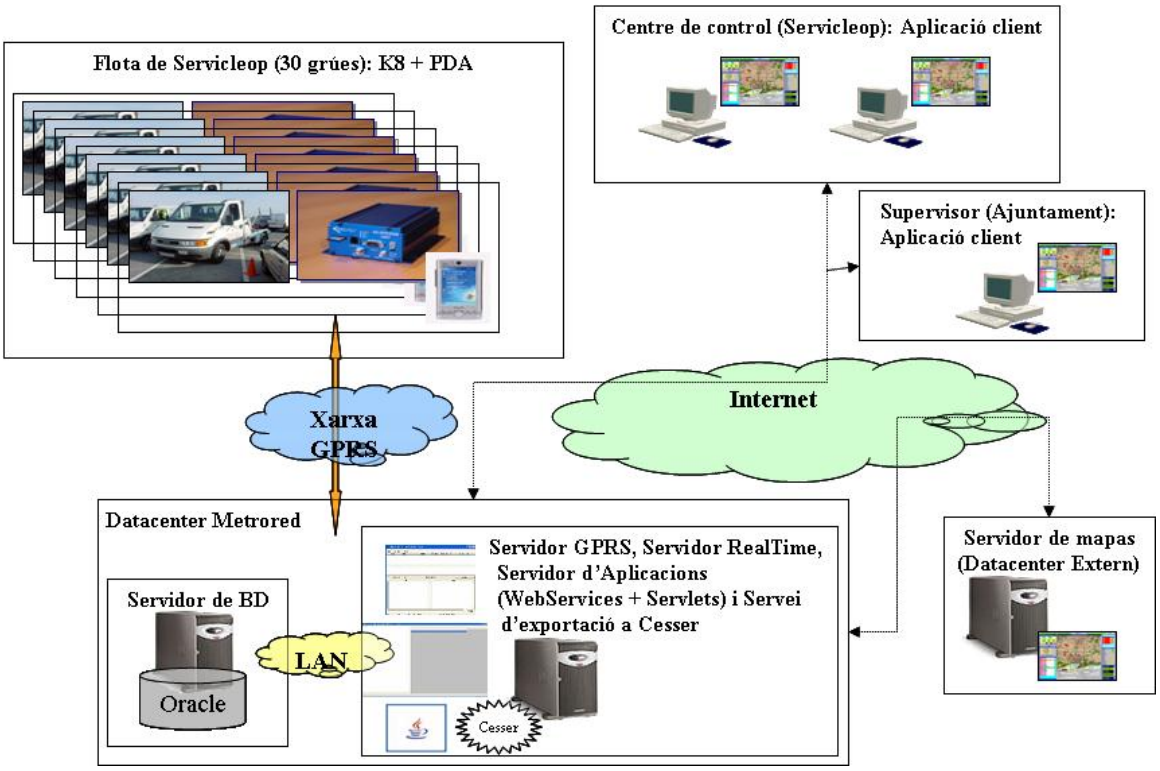
2.3 – Oracle

Oracle és la base de dades escollida per Knosos per a emmagatzemar les dades de la seva plataforma MNNs. Els motius d'aquesta elecció són la seguretat, escalabilitat i la seva potència a nivell procedural, així com el fet que sigui multiplataforma.

A les PDAs instal·larem Oracle Lite, que permetrà l'emmagatzematge en local de dades persistents (com per exemple marques de vehicles). També permet la sincronització d'aquestes dades amb un servidor central, ja que en qualsevol moment poden ser actualitzades (per exemple per a introduir un nou model de cotxe).

2.4 – Arquitectura del sistema

A continuació es presenta i descriu l'arquitectura en la qual consisteix el projecte per tenir una imatge global clara de cadascuna de les parts.



En l'esquema podem identificar els següents elements:

- Flota de Servicleop: Es tracta de 30 grues equipades amb un comunicador GPS/GPRS (K8) que s'encarrega de transmetre les posicions dels vehicles, i amb una PDA que per GPRS es comunicarà amb els usuaris del centre de control.
- Centre de control: A les oficines de Servicleop es disposarà de diverses instàncies de l'aplicació client que permetrà la comunicació amb les PDAs, i la visualització en cartografia de la posició GPS de les grues.
- Supervisor: A l'ajuntament de València hi haurà una altra instància de l'aplicació client, i es connectaran amb un perfil d'usuari que només té permisos de consulta.
- Servidor de mapes: Per a la cartografia, Knosos té un proveïdor anomenat Cercalia, i és un servidor seu que a través d'una petició web ens serveix els mapes per a visualitzar-los en les aplicacions client.
- Datacenter Metrored: Metrored és el proveïdor de serveis informàtics de Servicleop i en un Datacenter seu és on resideixen les diferents màquines i aplicacions que fan la funció de servidor, que són l'objecte d'aquest PFC i detallo en l'apartat següent.

2.4.1 – Arquitectura de servidors

La plataforma servidora d'aquest projecte rep el nom comercial de Micronav.Net Serveis, i és una plataforma dissenyada per a poder oferir servei a múltiples clients de mobilitat amb poc esforç de desenvolupament, gràcies a la seva modularitat i a la seva base de dades altament parametritzable. En el cas de Servicleop apreciem que la plataforma està repartida en dos servidors, un que fa de servidor de base de dades i un altre on resideixen les aplicacions, però cal dir que si es desitja cada aplicació podria estar en un servidor diferent, o estar tot en un sol servidor. Aquesta elecció es fa en funció de l'experiència obtinguda en projectes anteriors i avaluant proves de rendiment.

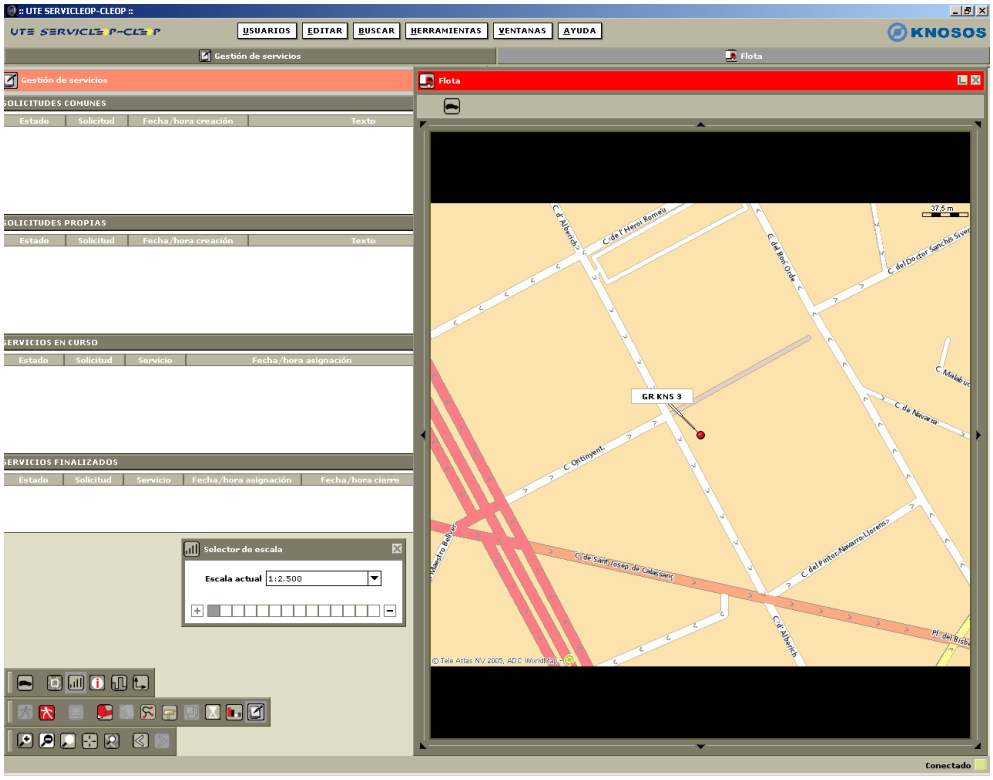
La plataforma servidora de Micronav.Net Serveis consta de:

- Servidor de base de dades: un servidor Oracle ens ofereix la seguretat i la potència necessàries per a poder manejar grans volums de dades eficientment, i la facilitat per a programar tasques de manteniment en el servidor o triggers que descarreguin de feina a les aplicacions.
- Servidor GPRS: és una aplicació que fa d'enllaç entre els elements de mobilitat i el servidor de temps real. Està desenvolupada en Delphi 6.
- Servidor de temps real: és l'encarregat de fer arribar als elements de mobilitat (a través del servidor GPRS) el que desitgin els usuaris del centre de control, i a l' inrevés. Les seves funcions principals són la codificació i descodificació de trames, insercions de posicions GPS i notificacions a clients d' events. És una aplicació desenvolupada en Delphi 6.
- Servidor d'aplicacions: es tracta d'un servidor d'aplicacions Java, JBoss, on resideixen 3 elements:
    - Servlets: Permeten als usuaris del centre de control la descàrrega de grans volums d'informació, com per exemple la configuració de l'usuari a l' entrar al sistema, o històrics.
    - Webservices: Permeten als usuaris del centre de control enviar peticions als elements mòbils, així com la consulta de petits volums d'informació.
    - Cercalia: Es tracta d'un servlet que fa d'enllaç amb el servidor extern de cartografia.

En el cas de Servicleop disposem, a més, del servei d'exportació de dades a Cesser.

2.4.2 – Aplicació client

Micronav.Net Serveis consta d'un client genèric que permet la interacció dels usuaris de centre de control amb els usuaris de mobilitat. En el cas de Servicleop, s'ha modificat lleugerament per a fer-lo més atractiu als usuaris, ja que una interfície particularitzada sempre resulta més atractiva i senzilla que una de genèrica.



Les funcionalitats que ens ofereix l'aplicació client de Servicleop (que podem veure en una captura en la figura anterior) són les següents:

- Visualització en temps real en cartografia de la localització de les grues.
- Visualització en temps real de l'estat dels serveis de les grues, a través de codis de colors.
- Enviament de serveis de retirada o trasllat de vehicles.
- Enviament de missatges.
- Descàrrega d'històrics amb la possibilitat de veure el recorregut en cartografia.
- Altres funcionalitats, com zoom a la cartografia, visualització de punts d'interès...

Altres funcionalitats presents en el client genèric però desactivades en aquesta versió ja que Servicleop no desitjava contractar-les, són per exemple un gestor d'itineraris, o la possibilitat de rebre missatges des dels elements mòbils.

2.4.3 – Elements de mobilitat

Present a cada vehicle (grua en el nostre cas) podrem trobar una K8, un element hardware clau desenvolupat per Knosos. Aquesta s'encarrega de la localització GPS i de transmetre-la al centre de control a través de GPRS. Per a tal efecte la grua portarà instal·lada una antena GPS per a millorar la recepció dels satèl·lits, i la K8 una targeta SIM per a poder utilitzar la xarxa GSM/GPRS.

Cal dir que les K8s poden oferir segons la programació del seu firmware moltes altres funcionalitats, però en el cas que ens ocupa la recepció i transmissió de posicions són la seva tasca clau.

Unitat K8

Antena combinada GPS/GSM

Tarja SIM

A més, cada gruer disposarà d'una PDA Qtek 9090 per a la transmissió d'informació de serveis, gràcies a les seva connectivitat GPRS. Per a tal efecte, la PDA també porta una tarja SIM en el seu interior.

PDA QTEK 9090

2.5 – Arquitectura client – servidor

L'arquitectura client – servidor ofereix un mètode eficient per a utilitzar els recursos de hardware, i és un dels models d'interacció més comuns en aplicacions de xarxa. S'utilitza aquest model per a desenvolupar aplicacions distribuïdes, resultant en processos d'informació independents que poden intercanviar informació, recursos, etc.

S'anomena client el procés que inicia el diàleg entre les parts, sol·licitant alguna cosa. Anomenem servidor al procés encarregat de gestionar les peticions i respondre-les. El servidor és capaç d'atendre concurrentment a diversos clients.

Les funcionalitats típiques d'una aplicació client són:

- Oferir una interfície gràfica amb la qual pugui interactuar l'usuari
- Captura i validació de dades d'entrada
- Generar consultes i visualització d'informes de dades
- Mantenir viva la connexió amb el servidor, enviar i rebre respostes

Els servidors solen tenir les següents característiques:

- Rebre peticions de clients, processar-la i si cal, enviar una resposta
- No acostuma a tenir una interfície gràfica, o en cas de tenir-la sol servir per a la configuració del servidor i visualització de l'activitat
- Centralitzar l'accés a base de dades
- Gestionar, si cal, perifèrics compartits

Per a la comunicació entre clients i servidor és necessària una infraestructura de comunicacions que permeti el transport de dades. La infraestructura pot ser la pròpia màquina, una xarxa local (LAN), o Internet.

Una arquitectura client-servidor ha de garantir:

- El servidor ofereix a tots els clients uns mecanismes d'interacció clars i definits
- El client no requereix saber cap detall sobre la implementació d'aquests mecanismes
- El client no ha de dependre de la localització del servidor, ni la màquina o sistema operatiu on s'executa
- Modificacions en el servidor no han d'implicar canvis, generalment, en els clients. Evidentment hi ha certa flexibilitat, com en cas de noves funcionalitats o correccions d'errors importants

2.6 – Sistemes de representació gràfica

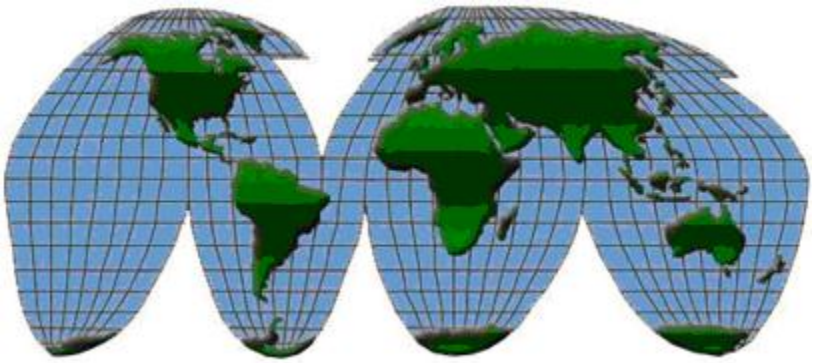
Una de les funcionalitats clau del client de la plataforma de Micronav.Net Serveis és la visualització dels recursos mòbils del sistema sobre cartografia.

Podem definir cartografia com la ciència que estudia la representació plana de la terra, i que s'encarrega d'establir una relació entre els punts de la superfície de l'el·lipsoide i el pla. Aquest procés de transformació s'anomena projecció.

La projecció més utilitzada s'anomena Universal Transversa de Mercator (UTM), basat en el desenvolupament conforme cilíndric de Gauss. Es basa en la projecció de l'el·lipsoide sobre un cilindre amb eix sobre en el pla equatorial i tangent a un meridià origen. Això implica un augment de la deformació a mesura que ens allunyem de l'equador, i per tant es completa amb una projecció polar estereogràfica per regions septentrionals del planeta (coordenades UPS).

A l'el·lipsoide resultant se'l divideix en 60 husos (part d'una superfície de revolució compresa entre dos semiplans que passen per l'eix de la superfície i que estan limitats per ella) iguals, de 6 graus de longitud. Cada hus es divideix en 20 àrees de 6 graus de longitud i 8 de latitud, anomenades zones, que es numeren i quadriculen utilitzant el metre com a unitat de referència.

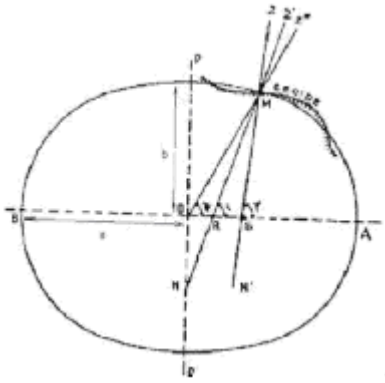
Un cop establerta la projecció, s'adapta segons un *datum* (model matemàtic que permet la representació d'un punt en un mapa).



Representació plana de la superfície terrestre

És sabut que la terra no és una esfera, ni tan sols un cos regular aplanat pels pols. Aquesta irregularitat fa que a cada regió calgui escollir el cos definible matemàticament que més s'adapti a la seva forma. Aquest cos sol tractar-se d'un el·lipsoide, i es diferencien entre sí pels seus radis major i menor i pel seu aplanament.

Cada datum està format per un el·lipsoide i un punt que s'anomena *Fonamental* on la terra i l'el·lipsoide són tangents. D'aquest punt se'n coneixen la latitud i la longitud.



Càlcul del datum d'una regió

En el *Fonamental*, les verticals de l'el·lipsoide i la terra coincideixen. Amb els *datums* definits, ja es pot elaborar la cartografia.

El sistema GPS treballa sempre amb el *datum* WGS84 i el sistema de coordenades UTM.

2.6.1 – Tècniques de representació

Ara que coneixem uns quants fonaments cartogràfics, explicarem diverses maneres de representar-la en una pantalla d'ordinador.

2.6.1.1 – Cartografia raster

Es basa en una imatge que representa un mapa. S'obté a través de l'escaneig d'un mapa cartogràfic normal, o creant una imatge a partir d'un arxiu vectorial. La única informació que ens ofereix aquesta cartografia són píxels, i el nivell de detall que ens pot oferir depèn de la resolució de les imatges.

2.6.1.2 – Cartografia vectorial

Es fa servir per a ser tractada directament per sistemes informàtics. Es basa en el tractament de la imatge com a objectes de dibuix, permetent establir, normalment a través d'una base de dades, relacions entre objectes i coordenades, com per exemple els noms dels carrers o ciutats. És la cartografia més utilitzada en sistemes GIS.

2.6.1.3 – Mapes sinòptics

S'utilitzen freqüentment per a la representació de vehicles que segueixen una ruta preestablerta, com per exemple trens o autobusos. Només mostra la informació d'interès, resultant molt pràctic per saber l'estat del sistema amb un cop d'ull.

Per a la representació dels vehicles es necessita tenir una relació entre coordenades i píxels. La *referenciació* és un procés a partir del qual es generen dues matrius de conversió, una entre coordenades i píxels i l'altra a la inversa.

2.7 – Protocol SOAP

SOAP (Simple Object Access Protocol) és un protocol estàndard creat pel W3C que defineix que els objectes en diferents processos poden comunicar-se intercanviant dades en format XML (Extended Markup Language). SOAP és utilitzat en els serveis web.

L'avantatge de l'ús de XML és, a diferència de DCOM o CORBA que fan servir dades binàries, que els missatges són comprensibles pels humans, de manera que la depuració és més senzilla.

SOAP permet modelar les crides remotes en forma de diversos missatges SOAP que interactuen.

SOAP funciona sobre qualsevol protocol d'Internet, generalment sobre HTTP (l'únic que està homologat pel W3C). Compleix un patró de Capçalera – Implementació de disseny software. Pot incloure una capçalera HEADER, que conté meta-dades sobre seguretat, transaccions, etc., i té un cos BODY on hi ha la informació principal, coneguda com *càrrega útil* (payload). El payload té un esquema XML propi.

S'ha escollit HTTP com a protocol de transport per les seves avantatges, com la facilitat de tractar amb firewalls.

2.8 – Serveis Web

Un servei web és una col·lecció de protocols i estàndards que serveix per a l' intercanvi de dades entre processos. Aplicacions diferents, desenvolupades en llenguatges diversos i executant-se en màquines i sistemes operatius diferents poden comunicar-se sense problema gràcies a ells. Això s'aconsegueix gràcies a la utilització d'estàndards oberts. Les organitzacions OASIS i W3C són les responsables de l'arquitectura i reglamentació dels serveis web.

2.8.1 – Motius per a fer servir serveis web

La principal raó per a utilitzar els serveis web és que es basen en HTTP sobre TCP, i per tant poden funcionar pel port estàndard web que és el 80. Com que moltes empreses disposen de firewalls i complexes regles de filtrat, és molt convenient la utilització d'aquests protocols ja que segurament ja tenen configurada la xarxa per a poder navegar per Internet.

Els serveis web proporcionen un dèbil acoblament entre ell mateix i l'aplicació que l'utilitza. Així, les modificacions de la implementació de cada part no han d'afectar a l'altre. Aquesta flexibilitat és molt important donada la tendència a generar cada vegada aplicacions més grans a partir de petits components distribuïts.

2.9 – Servlets

Els Servlets són petites aplicacions basades en tecnologia Java, que corren sobre un servidor web. Un servlet invocat per un client realitza una acció i retorna un contingut com a resultat.

Es consideren els substituts dels programes i scripts CGI per a servidors web. Són més ràpids, eficients i segurs, i permeten utilitzar tota la potència de J2RE (Java Runtime Environment).

2.9.1 – Cicle de vida d'un Servlet

El cicle de vida d'un Servlet és controlat pel contenidor on està situat. Quan es rep la petició, el contenidor fa les següents accions:

- Si no hi ha instància de Servlet
    - Carregar la classe del Servlet
    - Instanciar-la
    - Inicialitzar-la cridant el mètode Init()
- Invocar el mètode Service, que rep com a paràmetres un objecte Request (que encapsula la petició), i un objecte Response (que encapsula la resposta).

2.10 – Sistema GPS

El sistema GPS (Global Positioning System) va ser desenvolupat pel Departament de Defensa de EEUU, amb fins militars. L'objectiu era proporcionar una estimació de posició, velocitat i temps (PVT) d'un objecte equipat amb un receptor adequat a qualsevol punt del planeta.

L'any 1978 es va posar en òrbita el primer dels actuals 24 satèl·lits, distribuïts en 6 plans diferents, amb una inclinació de 55º respecte l'equador i a uns 22.000 Kms d'alçada.

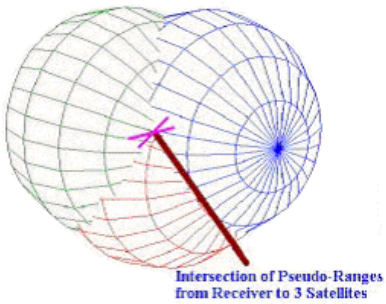
Cap al principi dels 80 es va començar a fer ús civil d'aquest sistema, amb certes limitacions que afectaven bàsicament a la precisió. Es feia servir en navegació aèria, marítima i terrestre, i va suposar un gran avançament en l'organització i estat dels transports i comunicacions a nivell mundial. A dia d'avui, la gratuïtat de les senyals GPS i l'accés lliure per a qualsevol persona o empresa fan que el sistema es faci servir en multitud d'aplicacions i que hagi superat de llarg les expectatives inicials.

El sistema GPS consta de 3 elements:

- Constel·lació de satèl·lits
- Estacions terrestres de monitorització
- Milions de receptors GPS, tant d'ús militar com civil

2.10.1 – Càlcul de la posició

El receptor GPS calcula la seva posició calculant el punt resultant de la intersecció de 3 esferes centrades en els satèl·lits escollits per al càlcul de posició. El radi de les esferes coincideix amb la distància que separa cada satèl·lit del receptor. La intersecció resulta en dos punts, un dels quals és descartat per trobar-se situat a milers de Kms fora de la terra.



La distància és calculada mesurant el temps que triga el receptor en rebre la senyal provinent del satèl·lit. Donada la impossibilitat que el satèl·lit i el receptor tinguin la mateixa base temporal, degut a que els rellotges incorporats als satèl·lits són de Cessi i Rubidi i els dels receptors són rellotges convencionals de Quartz, la distància calculada és incorrecte, i s'anomena pseudo-distància. Per corregir aquests errors de sincronisme és necessària la presència d'un quart satèl·lit.

2.11 – El servei GSM / GPRS

El sistema GSM (Global System for Mobile Communications) és l'estàndard de telefonia mòbil de 2ona generació, que va iniciar comercialment el seu servei al 1992. Permet trucades de veu, missatges de text, etc. amb autentificació d'usuari i seguretat en les comunicacions.

El sistema GPRS (General Packet Radio Service) és considerat com la 2.5 generació de telefonia mòbil (UMTS és la 3era). Implantat a Espanya l'any 2000, no és fins 2002 que es disposa de suficient cobertura nacional i internacional per considerar-lo operatiu a nivell comercial. No requereix grans inversions i reutilitza les infraestructures de GSM, motiu pel qual té la mateixa cobertura que GSM.

GPRS és un servei que permet la transmissió de paquets via ràdio en la telefonia GSM. Un dels seus avantatges és el fet de poder tenir sempre la connexió establerta, podent reservar recursos de ràdio únicament quan ho necessiti per transmetre. GPRS suposa un estalvi de cost important sobre GSM, cosa que va impulsar l'entrada de serveis de dades en el mercat de la telefonia mòbil. GPRS ofereix a més millor qualitat que GSM en quant a disponibilitat, temps de resposta i funcionalitats.

Actualment ja existeix en el mercat el sistema UMTS (Universal Mobile Telecommunications System), que és la 3era generació de telefonia mòbil. Per ara s'ha descartat aquesta tecnologia en el projecte donat l'alt cost dels mòdems i les dades transmeses utilitzant aquest sistema.

3 – Requeriments i especificacions

Al capítol 1 ja hem explicat en una pinzellada quin era l'estat anterior del sistema i què s'esperava d'aquest projecte informàtic. En aquest punt passarem a estendre aquests requeriments.

3.1 – Projecte intern

Per entendre el projecte de Servicleop (del qual tractem en aquest PFC), és necessari entendre el projecte intern de Knosos conegut com a Micronav.Net Serveis (en el qual també he tingut una implicació important), i per entendre aquest cal entendre un altre projecte intern de Knosos conegut com a Micronav.Net.

3.1.1 – Micronav.Net

Micronav.Net neix de la idea d'oferir un servei de gestió de flotes genèric per a clients que no poden o no volen pagar un sistema de gestió propi. Aquest sistema multi-client integra localització en cartografia dels vehicles de les flotes, missatgeria bidireccional, gestió d'itineraris, així com visualització en cartografia de punts d'interès general o descàrrega d'històrics d'activitat. Aquest sistema es troba en explotació des del 2003, i compta amb un gran número de clients que augmenta dia a dia.

3.1.2 – Micronav.Net Serveis

Micronav.Net Serveis (MNNS) apareix com a evolució de Micronav.Net (MNN), també com a projecte intern de l'empresa, al veure's que les funcionalitats bàsiques que oferia aquest podien ser ampliades per arribar a cert sector de clients més exigents que

treballen amb el que anomenem *Serveis*. Aquest fet ha provocat que alguns clients migrin des de MNN cap a MNNS a l' estar interessats en el que el sistema els podia oferir. La meva participació en el projecte de MNNS ha estat important en quant a la part d'implementació i depuració de les noves funcionalitats i de gestió de les base de dades.

S'entén com a servei l'activitat a la que es dedica el client, i que pot ser diferenciada en diverses fases. En el cas de Servicleop, un servei és una retirada de vehicle mal aparcat, o un trasllat entre dipòsits municipals, i aquest servei pot estar pendent, en curs, finalitzat, etc. En un altre client per exemple podria tractar-se d'un transportista de matèries perilloses, i els seus serveis estarien pendents, en càrrega, en descàrrega, finalitzats, etc.

3.1.3 – Servicleop

Servicleop es deslliga de MNNS en quant a que el client decideix que disposarà d'infraestructura pròpia de servidors (és a dir que Knosos ja no els ofereix el servei, sinó que desenvolupa i integra una plataforma basada en MNNS per a ells). A més, es realitza un millor disseny visual del client de centre de control, ja que l'aplicació deixa de ser genèrica per a tot tipus de clients, millorant-se així la interfície d'usuari respecte MNNS.

A part del ja comentat, apareixen dos conceptes nous dels quals no es disposaven a MNNS:

- Per primera vegada a MNNS, l'element de mobilitat que acompanya a la K8 (transmissor GPS de Knosos) no és una K10 (terminal de dades de Knosos que es connecta a la K8 a través del port sèrie) sinó que és una PDA. Aquest fet senzill a simple vista implica la presència de dues targetes de telefonia per a un sol element de mobilitat (un vehicle portarà la K8 que transmetrà la posició GPS i una PDA que s'encarregarà de la gestió dels serveis de grua). Això provoca certes modificacions en el centre de control i base de dades per a tal que

internament les dades intercanviades amb qualsevol dels dos elements siguin tractades com si vinguessin de la mateixa font. La més important d'aquestes modificacions és la gestió d'una nova trama que enviarà la PDA en iniciar una sessió de treball, que indicarà amb quina grua està associada.

- També per primera vegada es desitja que l'element de mobilitat (PDA) pugui iniciar el tràmit d'un servei, encara que per a fer-ho prèviament necessiti tenir ja en memòria dades d'un servei. Això implica la gestió d'una nova trama de comunicació.

3.2 – Estat anterior del sistema

Com ja he comentat anteriorment, abans del projecte actual Servicleop gestionava la grua de València a través del canal de veu. És a dir, un agent localitzava un vehicle mal estacionat, informava a Servicleop, i Servicleop efectuava trucades telefòniques als encarregats de la grua perquè es dirigissin al lloc en qüestió i retiressin el vehicle, depositant-lo on els fos indicat. A més de tot això, després era necessari que el conductor fes un informe del que havia realitzat (si havia trobat o no el vehicle, si les dades informades eren correctes, si l'havia pogut deixar al lloc indicat del dipòsit, etc.) que més tard havia de ser introduït en el seu sistema d'informació.

3.3 – Requeriments del sistema Servicleop

En aquest punt es detallaran les funcionalitats del sistema de Servicleop, que es desenvolupa en aquest PFC, englobant així també les funcionalitats de MNN i de MNNS.

- Gestió de clients: Aquesta funcionalitat ve heredada de MNN, i no és requeriment pel client de Servicleop ja que al tenir infraestructura pròpia, seran els únics clients. Però és bo saber que el sistema permet gestionar diferents

clients definint en base de dades les seves flotes, usuaris, fluxogrames, paràmetres, etc.

- Gestió d'usuaris: Funció que ve des de MNN, encara que s' extén amb nous controls a MNNS. Permet a un client tenir més d'un usuari pel sistema. Aquests usuaris a més disposen d'una sèrie de permisos que els ofereixen o deneguen certes funcionalitats.
- Posicionament en cartografia: Aquesta funcionalitat també s'hereda de MNN. Permet situar en mapes de diferents escales els vehicles de les nostres flotes, podent personalitzar l'estil de cada vehicle.
- Missatgeria bidireccional: Provinent de MNN, a Servicleop es desitja que aquesta sigui només en el sentit de centre de control cap a vehicle. El centre de control té un registre dels missatges que ha enviat i de l'estat d'aquests (rebut, perdut, pendent), i també dels missatges que ha rebut.
- Descàrrega d'històrics: Provinent de MNN, permet veure el recorregut dels vehicles en cartografia segons uns criteris seleccionats. És una de les funcionalitats que més evolucionen a nivell de projecte intern, trobant-se actualment funcionalitats a MNN i MNNS que no són a Servicleop.
- Gestió d'itineraris: Provinent de MNN, no és requerida per Servicleop. Permet a un client establir recorreguts i lligar-los a un espai temporal, definint marges, i comprovar si els vehicles s'ajusten a aquests itineraris.
- Gestió de serveis: Apareix a MNNS, i és el puntal d'aquesta nova plataforma. S'entén com a gestió d'un servei des del moment en que s'introdueixen les dades elementals d'aquest (sol·licitud), fins el moment en què aquesta sol·licitud finalitza. Durant el procés podem veure o no la creació de diversos serveis associats a aquesta sol·licitud, i com aquests van canviant d'estat en pantalla segons uns codis de colors configurables. Els diferents estats dels serveis es defineixen en base de dades en funció d'un diagrama de flux. També es configuren en base de dades els paràmetres que van associats als serveis d'un client específic, així com els diferents tipus de serveis existents.
- Gestió de vehicles: Provinent de MNNS, permet al client saber en quin estat es troba un vehicle d' entre els estats definits en un fluxograma. També podem veure la seva velocitat instantània, rumb, així com veure els serveis que té assignats en aquell moment i els seus paràmetres.

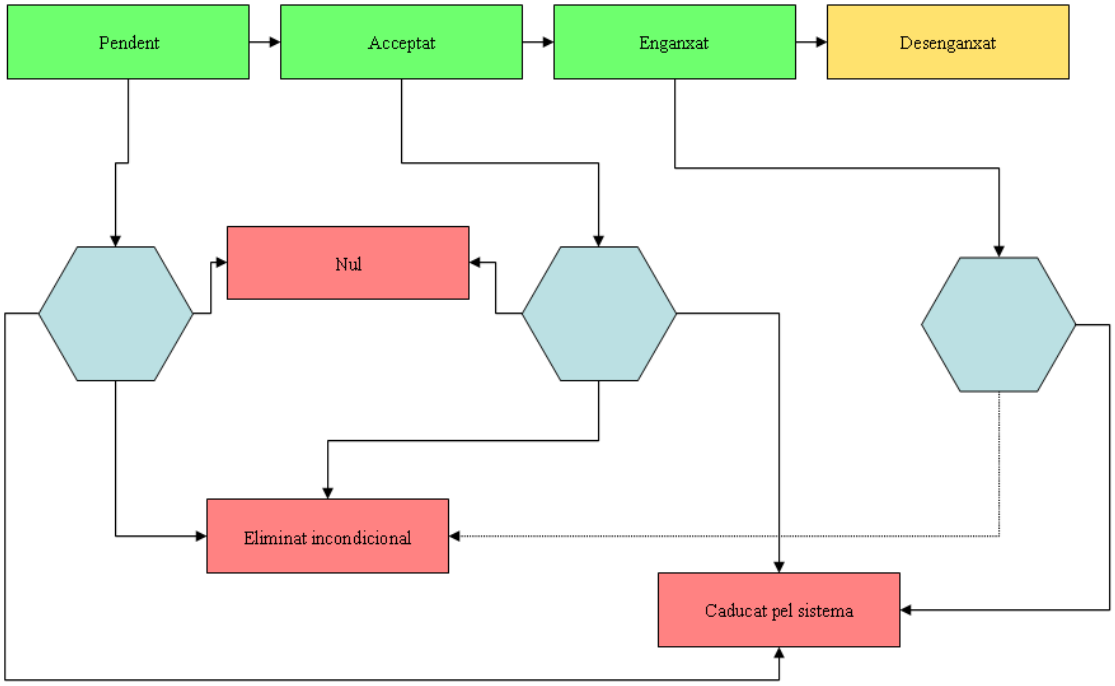
- Execució d'operacions predefinides: Apareix a MNNS, i es tracta d'un seguit d'operacions que es defineixen en base de dades i que representen realment un canvi d'estat sobre un servei d'un vehicle. Aquestes operacions van associades a un tipus de servei i a uns estats determinats.
- Validació de l'usuari de mobilitat: Prèviament a Servicleop, només calia que es validés al sistema l'usuari del centre de control. Servicleop té com a requeriment la introducció d'un usuari i paraula de pas a l' iniciar l'aplicació de mobilitat, cosa que implica noves taules i trames en el projecte que prèviament no existien.
- Còpia d'un servei finalitzat: Funcionalitat nova a Servicleop, implica la petició des de l'element de mobilitat de la generació d'una nova sol·licitud a centre de control, i l'assignació d'aquesta sol·licitud cap a la PDA que l'ha generat.
- Dualitat PDA-K8: També apareix a Servicleop, i més que d'un requeriment es tracta d'una necessitat d'implementació, ja que caldrà tractar els dos elements mòbils com si d'una sola entitat es tractés.

3.4 – Fluxogrames i parametrització específica

En aquest apartat descriurem els diferents diagrames de flux que s'han definit per als serveis en el client Servicleop, així com el diagrama per als estats dels vehicles. Conjuntament, veurem també els paràmetres associats a cada tipus d'element i els seus significats, els tipus de serveis i les operacions que es poden realitzar.

3.4.1 – Flux de serveis

En el següent diagrama podem apreciar els estats pels quals pot passar un servei de Servicleop, i en quina seqüència ha de fer-ho. En vermell s'han marcat estats finals, és a dir, que una vegada assolits aquests estats ja no hi poden haver més canvis. Aquests estats en vermell a més, són estats que en general indicaran que hi ha hagut alguna anomalia. El groc s'ha marcat l'estat final que s'espera a què arribin els serveis.



- Pendent: És l'estat amb el qual es creen els serveis. Aquest servei pot haver arribat a la PDA sense haver-se acceptat encara, o pot estar pendent d'enviar.
- Acceptat: Un servei que arriba a la PDA i l'operari accepta passa a aquest estat.
- Enganxat: Quan l'operari localitza el vehicle i l'enganxa a la grua, es marca aquest estat.
- Desenganxat: És l'estat final d'un servei normal, i representa el moment en què el cotxe és dipositat en el dipòsit o carrer de destí.
- Nul: Aquest estat es pot assolir tant per acció del centre de control com per acció de PDA. Representa el cas en què per qualsevol motiu un servei ja no és vàlid (per exemple arribar al lloc i que el vehicle ja no hi sigui).
- Eliminat incondicional: Aquest estat es va crear davant la necessitat d'anul·lar serveis sense necessitat de notificar a la PDA. És per tant un estat que assoleix un servei per acció del centre de control, i és útil en casos en què una PDA ja té un servei assignat i en volem assignar un altre de més prioritari (ja que s'ha requerit una limitació d'un sol servei en PDA).
- Caducat pel sistema: Si un servei porta massa temps (configurable) en un estat no final, el sistema el caduca automàticament. Aquesta gestió es fa per evitar sobrecarregar la memòria i les pantalles de les aplicacions amb serveis que ja no existeixen o que per algun motiu, mai s'han enviat.

3.4.2 – Tipus de serveis i operacions

A Servicleop s'ha definit un únic tipus de servei, que s'anomena *TSERVICLEOP*. Per al tipus *TSERVICLEOP*, s'ha definit com a estat inicial d'un servei l'estat *Pendent*, i com a estat final predeterminat *Desenganxat*.

Com que la interfície del client s'ha fet específicament per a aquest client, no podrem apreciar enlloc aquest tipus, però la seva definició és necessària pel funcionament del sistema.

S'han definit dues operacions per a aquest tipus:

- Anul·lació: passa el servei seleccionat a l'estat *Nul*, sempre que el fluxograma ho permeti. La PDA rep una notificació de l'anul·lació d'aquest servei, que no canviarà d'estat fins que la PDA confirmi el canvi.
- Eliminació incondicional: passa el servei a l'estat *Eliminat Incondicional*, sempre que el fluxograma ho permeti. La PDA rebrà una notificació, però el canvi d'estat serà immediat.

3.4.3 – Paràmetres de serveis

Aquí exposarem els paràmetres que conformen un servei, així com el seu significat.

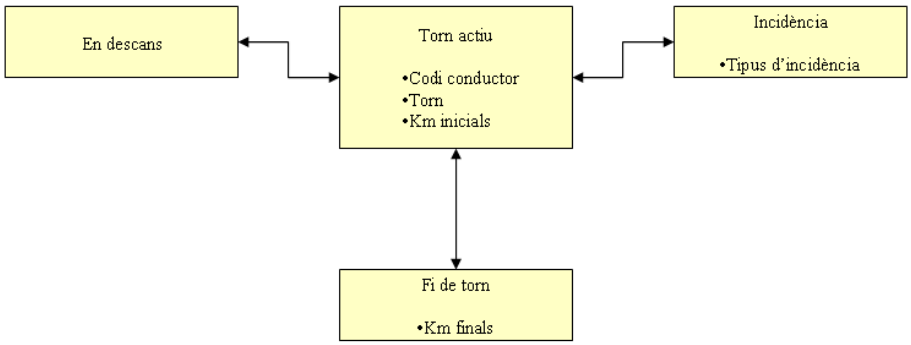
- Dipòsit destí: indica si el vehicle s'ha de deixar en un dipòsit municipal, i permet seleccionar-lo d' entre un llistat. Aquest paràmetre és opcional i s'entra des del client de centre de control, encara que la PDA podrà sobreescriure'l si així es desitja.
- Plaça dipòsit destí: complementa l' anterior, indicant a quina plaça deixar el vehicle retirat.
- Destinació: si el vehicle no ha d'anar a un dipòsit, aquí s'especifica el seu destí. Es comporta igual que els anteriors.

- Número destinació: igual que abans, complementa la informació sobre el destí del vehicle en cas que no es dirigeixi a un dipòsit.
- Dipòsit origen: totalment igual, però referint-se al lloc on es troba localitzat el vehicle i no el lloc on s'ha de dipositar (en aquest cas està en un dipòsit i cal traslladar-lo).
- Plaça dipòsit origen: complementa l' anterior indicant de quina plaça s'ha de treure el vehicle.
- Ubicació: és el carrer on es troba el vehicle a retirar. La PDA el podrà editar si la informació no és correcta. Es pot ubicar en cartografia.
- Número d'ubicació: el número del carrer on es troba el vehicle. Editable per PDA també.
- Matrícula: indica la matrícula del vehicle a retirar. La PDA podrà modificar el valor si fos erroni, validant sempre el correcte format de la matrícula.
- Observacions: s'envien des del centre de control a la PDA per completar la informació si és necessari. Aquesta les podrà modificar si no són suficients o es demostren errònies.
- Tipus de vehicle: la PDA envia el tipus de vehicle de què es tracta quan el diposita.
- Marca: la PDA envia la marca del vehicle que retira quan el diposita. Permet triar-la d'una llista o entrar-la lliurament.
- Model: la PDA envia també el model dintre de la marca. Permet triar-lo d'una llista o entrar-lo lliurament.
- Color: la PDA envia també el color del vehicle. Pot triar-se d'una llista o entrar-lo lliurament.
- Indicatiu: representa un codi que representa l'agent que ha creat el servei. Es presenta en PDA a nivell informatiu. El client permet triar-lo d'una llista.
- Hora assignació: l'hora en què el centre de control ha assignat el servei a la grua. S'envia a la PDA a nivell informatiu.
- Codi de servei: permet a la PDA triar de quin tipus de servei es tracta escollint-lo d'un desplegable.
- Codi d'agent: si el tipus de servei és *Cobrat en ruta*, des de PDA s'haurà d'introduir un codi d'agent amb format vàlid.

- Taxa: si el tipus de servei és *Cobrat en ruta*, des de PDA s'haurà d'introduir una taxa amb format vàlid.

3.4.4 – Estats i paràmetres de vehicles

En el següent diagrama apreciarem els diferents estats en què poden estar els vehicles (en aquest cas PDAs) de Servicleop, així com els paràmetres que porta associats cada estat i el seu significat.



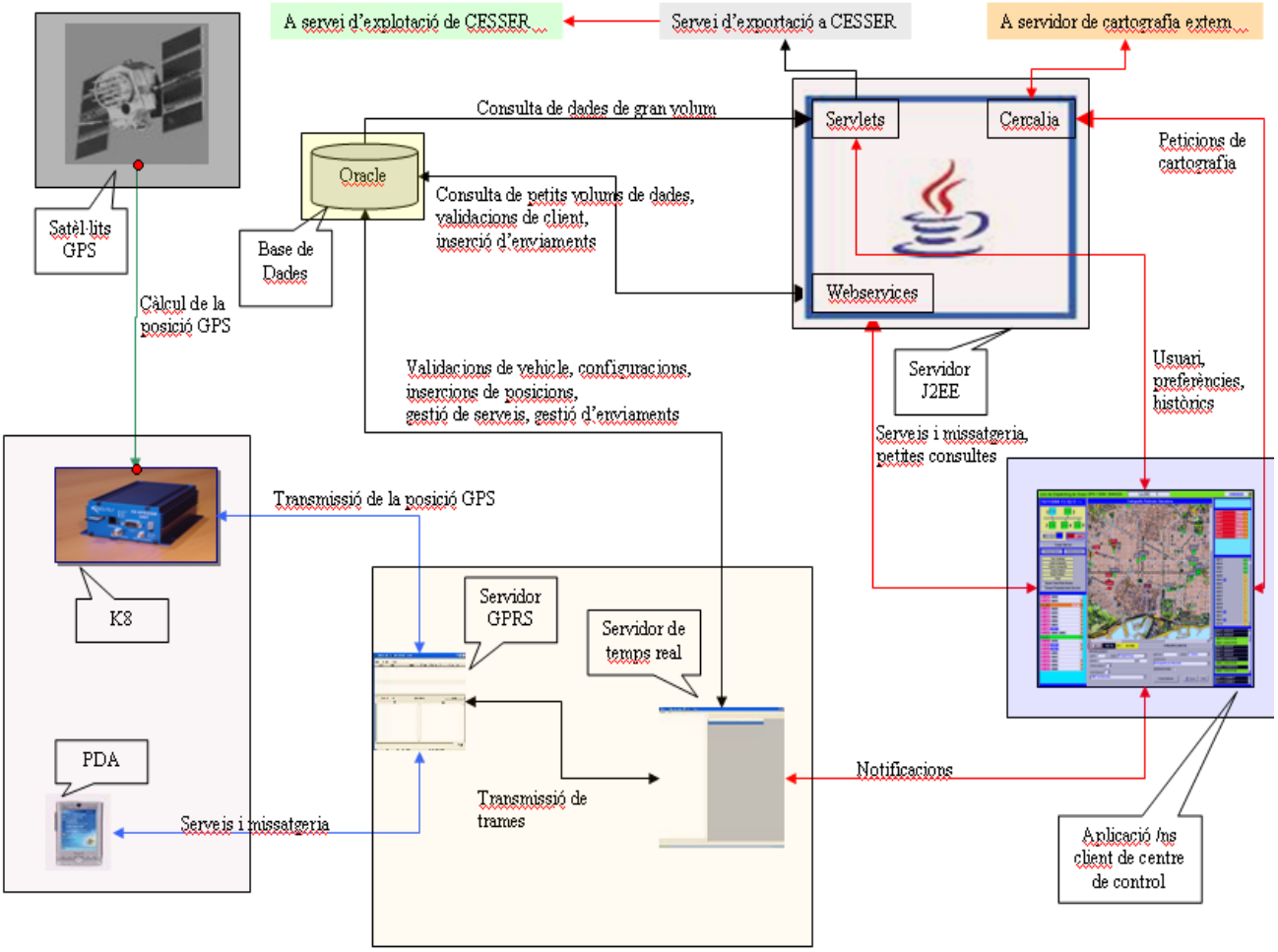
- Fi de torn: un vehicle que no té sessió activa es troba en l'estat *Fi de torn*. En el pas cap a aquest estat, s'ha d'introduir des de l'aplicació de PDA el paràmetre *Km finals*, que són els quilòmetres que marca la grua en el seu comptador.
- Torn actiu: aquest estat representa una grua que està circulant, ja sigui en un servei com sense. Es pot passar a aquest estat des de qualsevol dels altres, però quan es faci des de l'estat *Fi de torn*, es demanarà en l'aplicació PDA la introducció del conductor, el torn de treball i els quilòmetres inicials de la grua.
- En descans: representa una parada breu durant un torn de treball, motivada per exemple per l'hora d'anar a esmorzar / dinar, etc. Només es pot arribar a aquest estat des de *Torn actiu*, que és a l'únic estat al que es pot tornar des d' ell.
- Incidència: aquest estat indica que per algun motiu (que introduirem en seleccionar-lo des de la PDA) no es pot seguir treballant. Des de PDA s'ofereixen certes tipologies d'incidències. Aquest estat només és assolible des de *Torn actiu*, i només es pot tornar cap a aquest (quan la incidència està resolta).

4 – Anàlisi del sistema

Fins ara hem vist quina era l'estructura general del sistema i s'ha vist com es podia dividir en diversos elements: client de centre de control, client de PDA, i servidor. També hem vist com el servidor es dividia en diverses aplicacions. Ara veurem el detall d'aquesta estructura.

4.1 – Anàlisi general del sistema

A continuació es mostra un esquema més detallat de la distribució de la plataforma de Servicleop.



Primer de tot, aclarirem els diferents tipus de comunicació que trobem:

- Fletxa verda: la comunicació és via satèl·lit
- Fletxa blava: la comunicació es fa a través de GPRS
- Fletxa vermella: la comunicació és via Internet
- Fletxa negra: la comunicació és via xarxa local (LAN), tot i que també podria ser via Internet.

4.2 – Elements de mobilitat

Tal com ja hem comentat, els elements de mobilitat del projecte de Servicleop són les grues, però amb la particularitat que per primera vegada en un projecte basat en MNNS el terminal de dades és una PDA que fa servir les seves pròpies comunicacions per a la gestió dels serveis. Per a la localització es faran servir K8s.

Aquests dos elements s'hauran de comunicar amb el servidor central a través de GPRS, i és imprescindible que encara que cada element disposi de les seves pròpies comunicacions, es gestionin com si d'una sola entitat es tractés.

4.2.1 – PDAs

L'aplicació per gestionar els serveis instal·lada a la PDA farà les següents funcions:

- Bloquejar totalment l'accés a la PDA i el telèfon. L'aplicació a l' instal·lar-se es configura com a aplicació d' inici, i impedeix la possibilitat d'apagar-la i d'accedir als recursos de la PDA. D'aquesta manera es garanteix que l'aplicació estarà sempre engegada i s'impedeix fer un ús no desitjat de la PDA.
- Identificació de la PDA amb un vehicle: Prèviament a l' inici de sessió i tot, la PDA ha d'enviar una trama de configuració al servidor de temps real, indicant-li

a quina K8 (o grua) està associada. En principi cada PDA està associada a una grua, però es pot configurar a través d'un arxiu XML de la PDA.

- Validació remota de l'usuari: A l' iniciar l'aplicació, o després d'haver finalitzat una sessió, la PDA requerirà a l'usuari entre un codi i un password. Aquests seran validats remotament pel servidor de temps real, que respondrà a la PDA si l' inici de sessió és vàlid.
- Entrada de dades del torn: Un cop validat l'usuari, l'aplicació de PDA requerirà entrar el torn i el quilometratge inicial. Aquesta informació també serà enviada al servidor de temps real, permetent visualitzar-la en l'aplicació client del centre de control.
- Missatgeria: Permet a l'usuari rebre missatges de text, formats per un assumpte i un cos. Els missatges poden emmagatzemar-se per a una lectura posterior, o esborrar-se.
- Gestió de l'estat operatiu: Es defineix un esquema d'estats en els quals es pot trobar el recurs. L'aplicació permet canviar aquest estat segons un diagrama de flux, i n'informa al servidor de temps real. D'aquesta manera des del centre de control es pot saber si un determinat agent té algun problema amb la grua, està en el seu període de descans, etc.
- Gestió de serveis: Permet gestionar des de la PDA els serveis de retirada que es generen i envien des del centre de control. A Servicleop hi ha una limitació per especificació que diu que una PDA només pot tenir un servei en curs, però el sistema suporta múltiples serveis concurrents. La gestió d'aquest apartat es detallarà tot seguit.
- Fi de torn: Es demana a l'usuari que introdueixi novament el quilometratge, i s'envia aquesta informació al centre de control. L'aplicació de PDA retorna a la pantalla de login.

La gestió de serveis consisteix en:

- Des del centre de control es crea una sol·licitud de retirada de vehicle i s'assigna a una grua. Aquest fet dóna lloc a la creació d'un servei, que és enviat pel servidor de temps real cap a la PDA a través del servidor GPRS.
- La PDA rep el servei i notifica a l'usuari de la recepció d'aquest servei.
- L'usuari de la PDA comença a gestionar el servei. En consulta les dades i procedeix a localitzar el vehicle, resultant en l'acceptació del servei, o bé anul·la el servei si ho considera necessari.
- En arribar al lloc, comprova que el servei segueixi essent vàlid (el vehicle hi és i està mal aparcat). En cas afirmatiu, procedeix a enganxar el cotxe i així ho indica des de la PDA. En cas contrari, pot anul·lar el servei.
- Fins a aquest punt el servei podia ser anul·lat també des del centre de control. Un cop enganxat el vehicle ja no hi ha marxa enrere, per tant la PDA ignorarà les possibles peticions d'anul·lació que enviï el centre de control.
- Quan la grua arriba al dipòsit (o al lloc sol·licitat per deixar el vehicle), es procedeix al desenganxament. Aquesta operació requereix l'entrada de certes dades des de la PDA, algunes de les quals no fan més que confirmar (o sobreescriure) les rebudes des del centre de control en crear-se la sol·licitud (per exemple la matrícula o la marca del cotxe).
- Un cop confirmada l'operació de desenganxament, la PDA ofereix a l'usuari la possibilitat de demanar una còpia del servei que acaba de realitzar (per exemple si ha detectat que hi havia més vehicles mal aparcats a la zona). En cas afirmatiu, enviarà la petició al servidor, que automàticament generarà la nova sol·licitud, l'assignarà al vehicle i li retornarà per a que pugui treballar.

La PDA ha de tenir certa autonomia respecte la cobertura GPRS. És a dir, encara que ens trobem sense cobertura, ha de permetre gestionar localment el servei en curs i l'estat operatiu, que seran enviats cap al centre de control quan es recuperi. Les úniques operacions que no són possibles sense cobertura són l' inici de sessió (ja que requereix validació remota de l'usuari), i naturalment la recepció de serveis o missatges.

4.2.2 – K8s

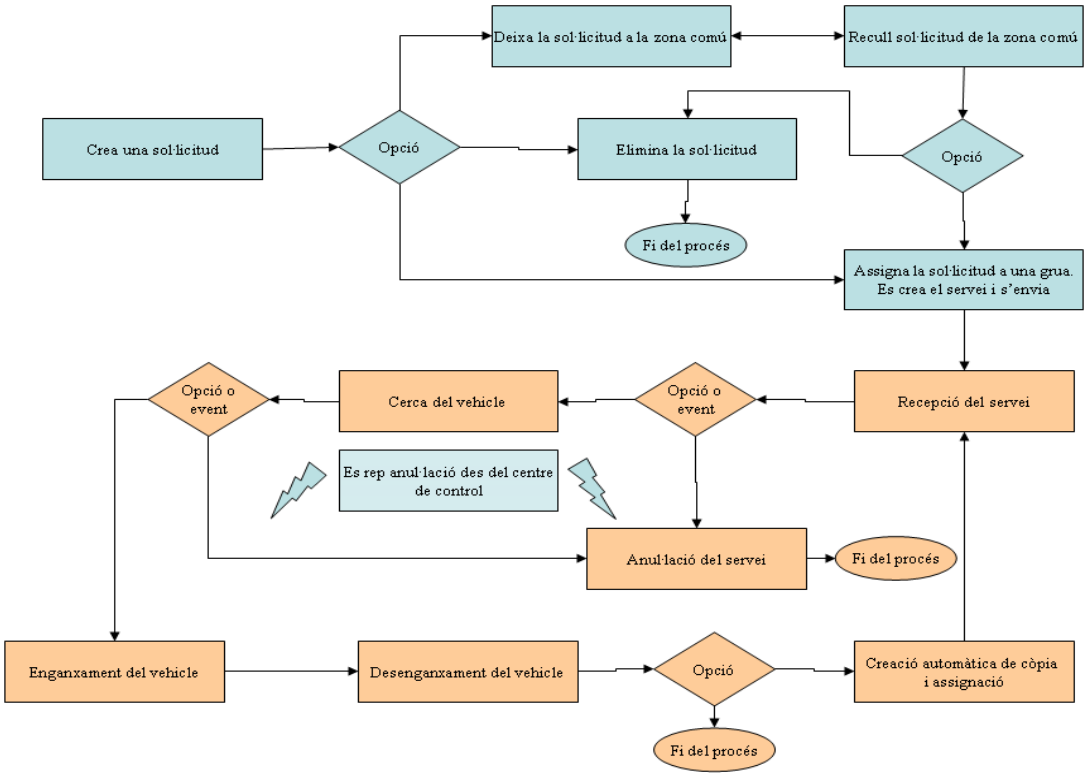
Els receptors de GPS seran necessaris a cada grua per a tal d'obtenir la posició i transmetre-la, permetent als usuaris del centre de control visualitzar els vehicles en cartografia. Aquesta és la única funcionalitat de les K8s en aquest projecte, ja que les PDAs gestionen les seves pròpies comunicacions.

4.3 – Clients del centre de control

L'aplicació client de centre de control és la que permet conèixer l'estat actual del sistema i ordenar accions.

4.3.1 – Gestió de serveis

Per tal de comprendre bé les funcionalitats del client, és necessari presentar un esquema del procés de vida d'un servei.



Blau: Accions del centre de control

Taronja: Accions de la PDA

Veiem que el punt d'entrada a la gestió de serveis és la creació d'una sol·licitud. Això es fa en resposta a una necessitat, en el cas de Servicleop, retirar un vehicle mal aparcat. Amb la sol·licitud creada, l'usuari pot decidir entre gestionar ell mateix la sol·licitud i assignar-la a una grua (originant així el que es anomenem servei), deixar-la a la zona comú per tal que un altre usuari la pugui recollir i gestionar, o bé eliminar-la si ho creu convenient.

En el cas que es generi un servei, el procés continua amb la recepció d'aquest servei a la PDA. Allà l'usuari pot decidir entre acceptar el servei, procedint a la cerca del vehicle, o anul·lar-lo si ho creu procedent.

En cas que es cerqui el vehicle, ens podem trobar en el cas que el vehicle ja no hi sigui En funció d'això, s'anul·larà el servei, o es procedirà a l'enganxament. Cal dir que fins a aquest punt el servei encara podia ser anul·lat pel centre de control. A partir de l'enganxament això ja no serà possible.

Després de l'enganxament, només queda desenganxar al vehicle on ens hagin demanat, i després decidir si volem un altre servei, o finalitzem aquí la gestió.

4.3.2 – Funcionalitat del client centre de control

Un cop coneguda la gestió d'un servei, passem a comentar les funcionalitats de l'aplicació client de centre de control:

- Control d'usuaris i perfils. Cada usuari del centre de control s'haurà d'identificar al sistema amb un usuari i una paraula de pas. Un cop fet, la informació s'enviarà al servidor, que comprovarà la validesa i calcularà el perfil de l'usuari així com les seves preferències i les retornarà cap al client. Basant-se en aquesta resposta, el client configurarà la seva vista i activarà o desactivarà certes opcions (que també seran validades en el servidor en el cas que requereixin una interacció per a més seguretat).
- Preferències. Permet al client configurar diversos aspectes, com la configuració d'IPs i ports dels servidors.
- Visió en cartografia. L'aplicació permet obrir diverses finestres de cartografia per visualitzar un o diversos vehicles sobre una finestra de mapa. La selecció de vehicles es pot fer individualment o en grup, escollint una flota sencera (el concepte de flota s'explicarà més endavant, però es pot resumir en una sub-agrupació dels vehicles que controla un client).

- Gestió de serveis. Es disposa d'una finestra on podem veure l'estat dels serveis de l'usuari actiu, així com de les seves sol·licituds encara no assignades o sol·licituds de la zona comú. També hi apareixen els serveis finalitzats recentment. Aquesta finestra està dividida en 3 parts per a distingir fàcilment cada conjunt. A través de menús contextuals, ens permetrà crear noves sol·licituds, assignar-les a vehicles, consultar-ne les dades, anul·lar serveis, etc.
- Missatgeria. Existeix una finestra a partir de la qual es pot gestionar l'enviament de missatges de text cap a un o diversos vehicles. Des d'aquesta finestra es pot veure l'estat dels missatges que hem enviat, així com enviar-ne de nous. A MNNS a més permet consultar la bústia d'entrada, per llegir missatges dels vehicles cap al centre de control, però a Servicleop aquesta funcionalitat està desactivada.
- Descàrrega d'històrics. A través d'un filtre per vehicles i dates, i opcionalment número de posicions, permet al client descarregar un històric del recorregut i l'activitat dels vehicles seleccionats. Aquest històric es pot veure en modalitat graella de dades, així com reproduir-se com si es tractés d'un vídeo sobre una finestra de cartografia. També es pot guardar a disc o imprimir.

4.4 – Servidors

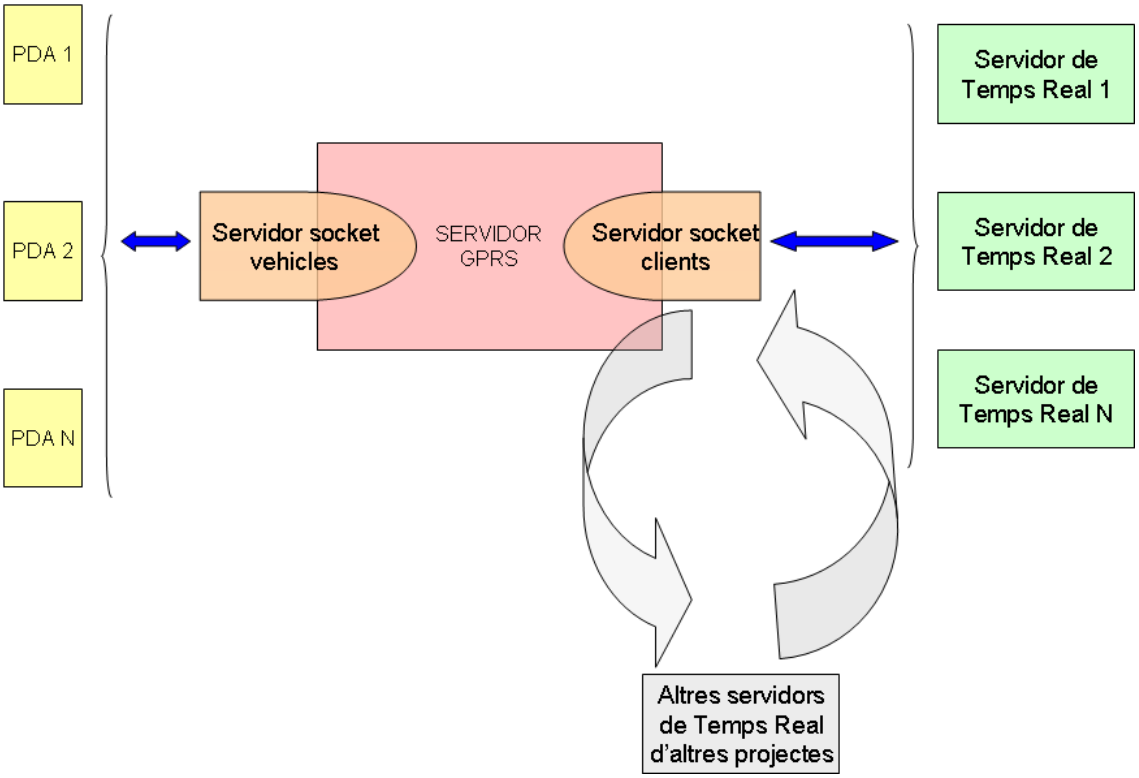
Com hem pogut comprovar en l'esquema anterior, la plataforma servidora consta de 3 aplicacions que anomenem Servidor GPRS, Servidor de Temps Real, i Servidor J2EE. En els punts següents les detallarem.

4.4.1 – Servidor GPRS

El Servidor GPRS és una aplicació desenvolupada en Delphi, amb una interfície gràfica senzilla però funcional, i és el punt on es connecten els diferents elements de mobilitat. D'aquesta manera doncs, independitza al Servidor de Temps Real de la gestió de les comunicacions, permetent que es centri en la codificació / descodificació de dades.

Aquest fet fa que el Servidor GPRS sigui un element comú que apareix a pràcticament tots els projectes de Knosos.

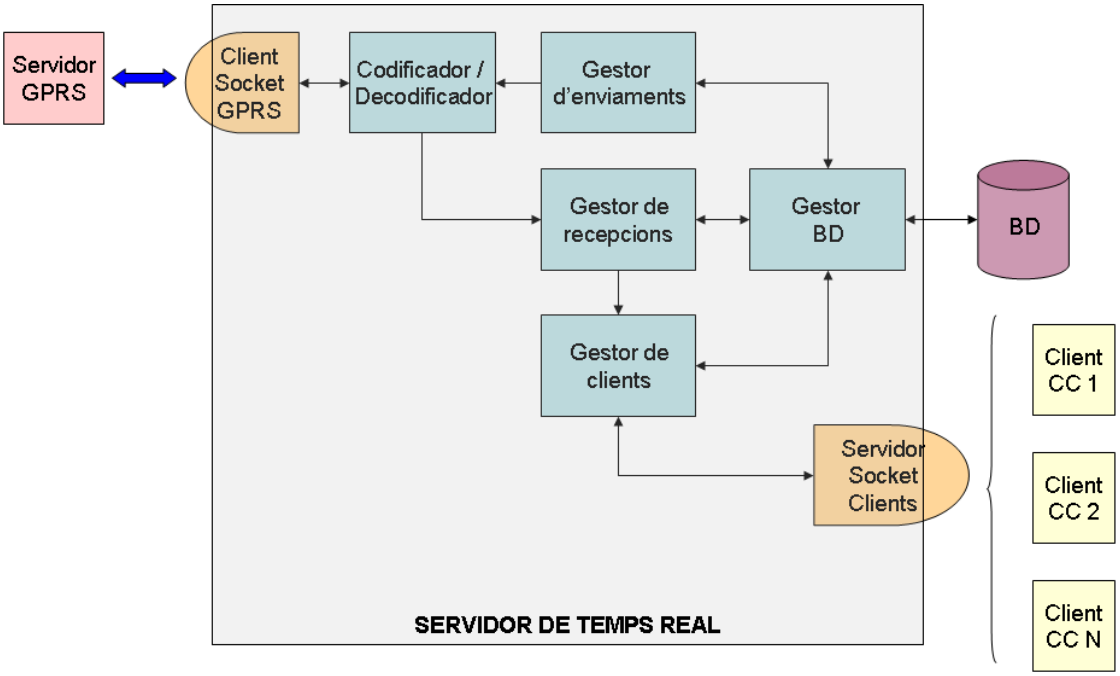
La seva tasca elemental és gestionar les comunicacions amb els vehicles. El Servidor GPRS rep una trama d'informació provinent d'un vehicle, amb un protocol comú establert. Després d'això desencapsula la informació d'interès, i l'encapsula en un nou protocol que pot desxifrar el Servidor de Temps Real. Aquest nou encapsulament és enviat pel Servidor GPRS cap a tots els clients que tingui connectats (direm que fa un Broadcast), i aquests seran els encarregats de discernir si estan interessant en aquesta trama o no i actuar en consonància. Això permet poder balancejar la càrrega d'un determinat sistema amb més d'un Servidor de Temps Real si fos necessari, i també permet que un únic Servidor GPRS ofereixi un punt d'entrada comú a vehicles de diversos projectes. En la següent figura es pot veure:



4.4.2 – Servidor de Temps Real

El Servidor de Temps Real és una aplicació desenvolupada en Delphi. Ofereix una interfície molt elemental on es poden veure els clients de centre de control connectats al sistema i tocar algun paràmetre de configuració.

La seva tasca principal és la de gestionar els enviaments i recepcions d'informació. Es connecta com a client del Servidor GPRS, per a que aquest li faci arribar les trames dels vehicles sobre els qual està interessat. Un cop disposa d'una trama, el Servidor de Temps Real la desencapsula del protocol del Servidor GPRS, la descodifica i actua en correspondència a la informació obtinguda. Això normalment implica la inserció d'una posició GPS a la base de Dades, i sovint també la notificació als clients de centre de control que algun event s'ha produït. També s'encarrega dels enviaments, i per a això es disposa d'una taula en base de dades on es guarda la informació pendent d'enviar (o de ser reconeguda) cap als vehicles. El Servidor de Temps Real llegeix un registre d'enviament, el codifica segons un protocol establert, i l'encapsula amb el protocol del Servidor GPRS. A través del socket client que té obert amb aquest li fa arribar la trama, i serà el Servidor GPRS el que veurà si el vehicle destinatari està connectat o no i transmetrà la informació o la desestimarà. Segons una política de reintents, el Servidor de Temps Real reintentarà l'enviament d'aquesta trama les vegades que calgui fins que es rebi un reconeixement (ACK) o un reconeixement negatiu (NAK) per part del vehicle, o fins que es desestimi l'enviament per antiguitat.



En aquest esquema podem apreciar els diferents elements que conformen l'aplicació Servidor de Temps Real.

- Client Socket GPRS. És qui s'encarrega d'enviar les trames codificades cap al Servidor GPRS, i en rebre les trames dels vehicles.
- Codificador / Descodificador. Decideix si una trama de recepció ens interessa o no era per a nosaltres, i en desencapsula el contingut que és traspassat al gestor de recepcions. En enviament, codifica una trama cap a vehicle i l'encapsula en el protocol del Servidor GPRS per a ser transmesa cap a aquest.
- Gestor d'enviaments. Llegeix els enviaments pendents de la base de dades, i els passa cap al codificador per a ser transmesos.
- Gestor de recepcions. Quan el descodificador li passa informació, decideix què ha de fer amb aquesta (insertar-la en BD, validar-la, descartar-la, etc.). Si creu que la recepció ha de ser notificada a algun client, li passa la informació necessària al Gestor de clients.
- Gestor de clients. S'encarrega d'enviar les notificacions necessàries a qui calgui. Porta un control de quins usuaris estan connectats, i a través de quina IP i port, per tal de fer-los arribar aquestes notificacions. També disposa d'estructures de dades per a poder discernir si una notificació ha de ser transmesa o no a un determinat client. S'encarrega també d'averiguar quins clients estan inactius (no

han enviat petició de desconnexió, però realment ja no estan al sistema), a través de la gestió de paquets de vida.

- Gestor BD. Centralitza les operacions amb base de dades i porta el control transaccional.
- Servidor socket clients: És el punt d'accés a través del qual s'envien les notificacions, i des del qual es reben els paquets de vida, etc. que mantenen viva la connexió amb els clients de centre de control.

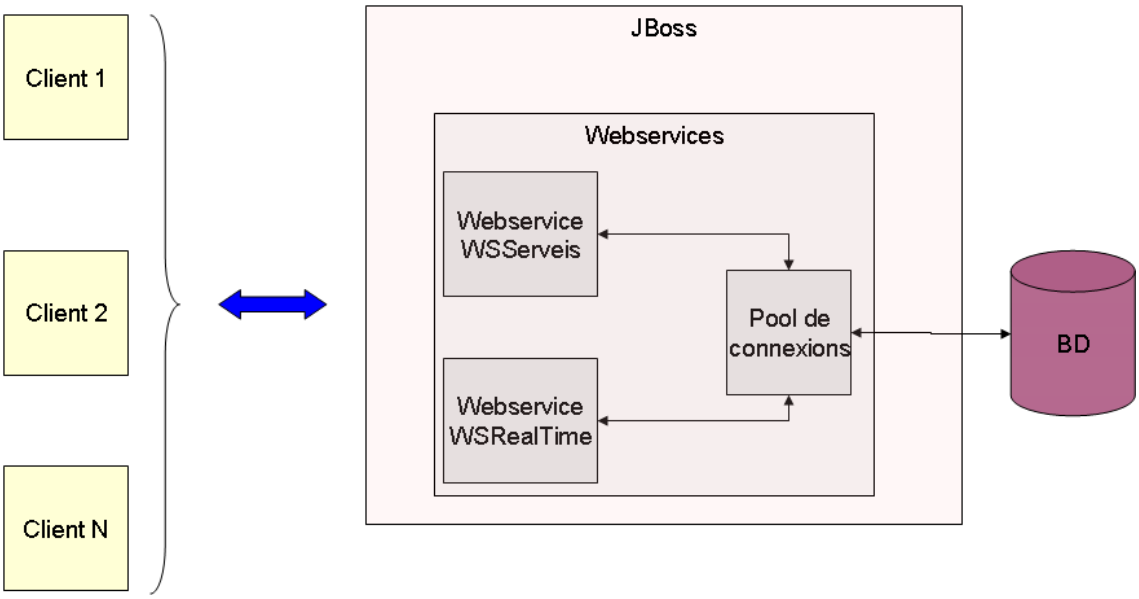
4.4.3 – Servidor J2EE

Per a l' utilització de serveis web i servlets, es requereix d'un servidor d'aplicacions J2EE. En aquest cas hem utilitzat el JBoss 4.0.2, que té la fiabilitat necessària a més de ser molt senzill d'administrar. Al tractar-se de Java, requerim d'una màquina virtual, i hem utilitzat la que ve amb la versió 1.4.0.7 del JDK.

En aquest servidor, s'executen 3 aplicacions:

4.4.3.1 – Webservices

Una de les parts més vitals del sistema per a l'aplicació client són els webservices, o serveis web. Basats en el protocol SOAP, com hem comentat en un capítol anterior, seran els que permetran als clients realitzar peticions i obtenir petits volums de dades.



A l'esquema podem observar que els clients realment no fan una petició a un servei web, sinó que és el servidor J2EE, el JBoss, que decideix en funció de la URL i de les metadades de la crida quina de les aplicacions que conté ha d'invocar.

Disposarem de 2 serveis web, que compartiran un pool de connexions:

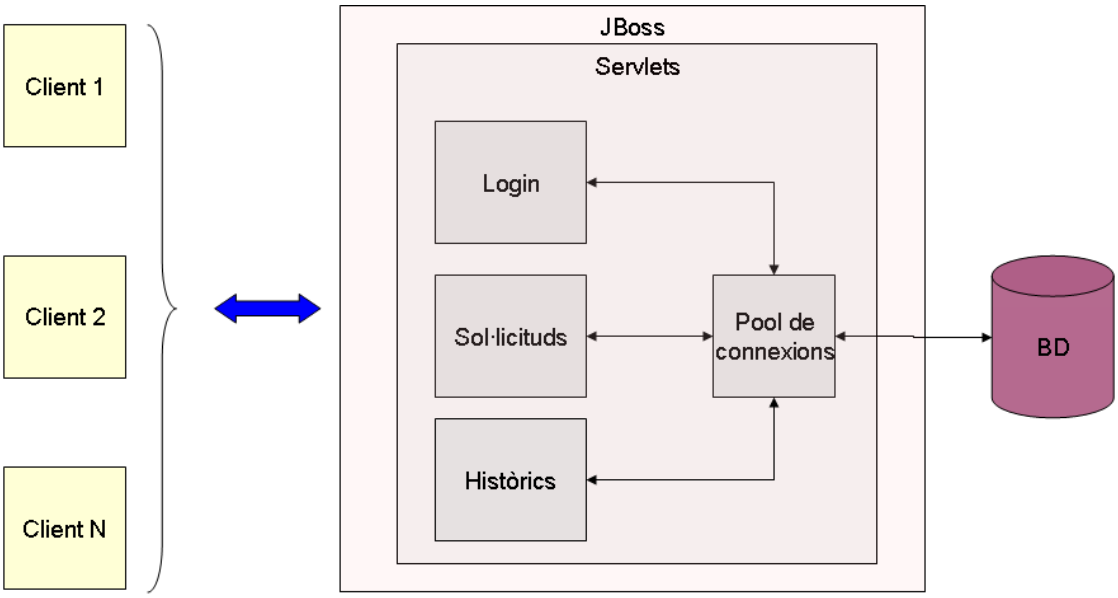
- WSServices: A través d'aquest webservice el client podrà realitzar totes les operacions relacionades amb la gestió de serveis, com creació de sol·licituds, assignació a vehicles, consultes de dades de servei, anul·lacions, etc.
- WSRealTime: Aquest webservice és l'encarregat de les funcions de missatgeria, com enviar un missatge, o retornar les dades de missatges nous en projectes amb missatgeria des de vehicle.
- Pool de connexions: Cada petició a webservice requereix de l'establiment d'una connexió a base de dades. En comptes d'utilitzar una única connexió compartida per a tots els clients, cosa que podria penalitzar greument el sistema en cas que molts usuaris estiguessin actius, o obrir i tancar una connexió a cada petició, cosa que seria ineficient ja que el temps d'establir una connexió és alt, s'ha decidit utilitzar un pool de connexions. Consisteix en tenir cert número de connexions sempre obertes, i també a configurar en quant ha de créixer aquest número si en necessitem més i a limitar el màxim de connexions que es poden tenir. D'aquesta manera, si hi ha una connexió lliure en el moment d'una nova petició, aquesta connexió s'extreu del pool, s'utilitza i és retornada al pool per a

la seva reutilització. Si no hi hagués connexions lliures, el pool creixerà en la mida especificada per poder atendre la nova petició i no penalitzar a les properes peticions que poguessin arribar. Si en un moment donat s'arriba al màxim de connexions, la petició haurà d'esperar a que una connexió quedi lliure.

4.4.3.2 – Servlets

Els servlets també són importants pel funcionament de l'aplicació client, ja que gràcies a ells podrem descarregar la informació de configuració segons l'usuari que es validi, així com les seves dades de gestió més recent.

Com es pot veure a l'esquema, la filosofia dels servlets és pràcticament idèntica a la dels serveis web, amb la diferència en que en els servlets no utilitzarem SOAP, sinó peticions HTTP.



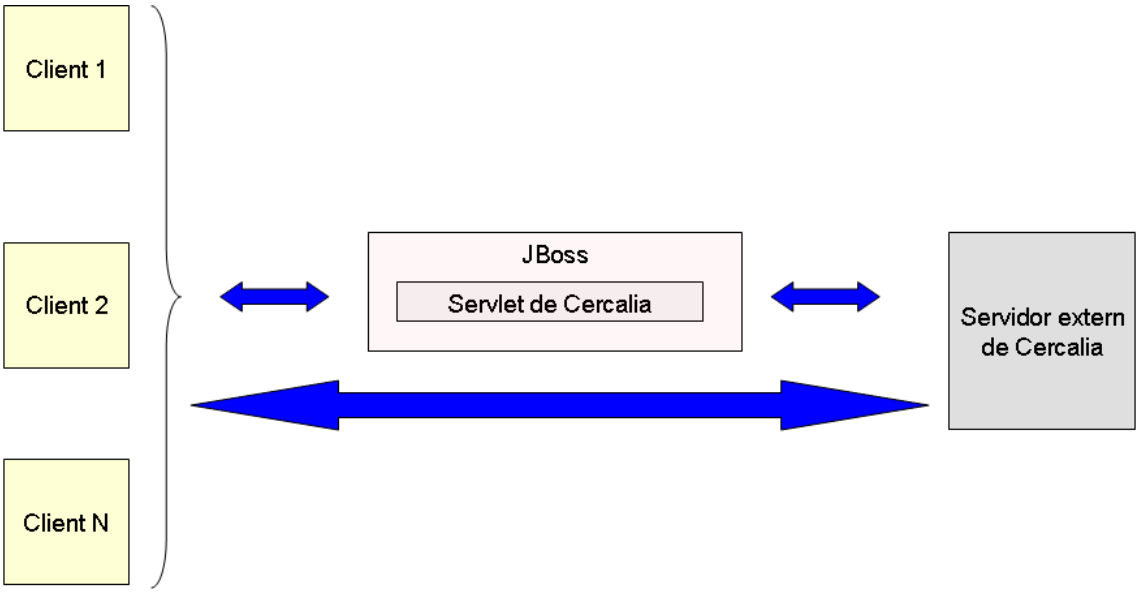
Veiem que el sistema consta de 3 servlets:

- Login: és el servlet encarregat de validar l'usuari i retornar-li la seva configuració per a l'aplicació quan aquesta s'inicia. També li retornarà la seva bústia de missatges.

- Sol·licituds: s'encarregarà de retornar a l'usuari les dades de serveis més recents que li pertanyin. S'executarà després del servlet de Login, a l' iniciar sessió.
- Històrics: quan l'usuari demana dades històriques de localització, és aquest servlet qui s'encarrega de gestionar la petició.

4.4.3.3 – Servlet de cartografia

La darrera aplicació que s'executa en el servidor d'aplicacions J2EE és un servlet que permet a l'aplicació client obtenir informació sobre la cartografia. Bàsicament el client fa una petició a aquest servlet, i aquest li respon amb les dades que necessita o bé amb una URL des de la qual descarregar-se un mapa. Això pot apreciar-se en la següent figura:



4.4.4 – Servidor Oracle

El servidor Oracle és una part vital del sistema, i on resideix gran part de la versatilitat de l'aplicació.

Pel projecte, a part de les funcions òbvies d'emmagatzament i garantir la integritat referencial (a través de constraints, triggers, etc.) s'ha utilitzat l' Oracle com a font per a la configuració dels diferents clients que poden residir en el sistema (recordem que hem comentat que el projecte Servicleop es basava en una particularització, ampliació i parametrització d'un projecte multi-client de Knosos). Per a tal fi, s'han definit una sèrie de taules que permeten configurar els següents apartats:

- Idiomes: En el cas de Servicleop es tracta d'una aplicació d'un sol idioma ja que així ho va voler el client, però MNNS suporta diferents idiomes per a la representació dels atributs en pantalla del client de centre de control.
- Paràmetres: Es poden definir tots els paràmetres que pot tenir una sol·licitud o un servei, així com les seves longituds o si són obligatoris. Aquests paràmetres seran mostrats dinàmicament en les aplicacions client (tot i que a Servicleop s'ha fet una pantalla adaptada que sacrifica l'adaptabilitat a canvi d'una interfície més amigable).
- Paràmetres de vehicle: De la mateixa manera es poden configurar paràmetres per als vehicles, de manera que per exemple podem saber qui està conduint una grua determinada.
- Estats dels serveis: Es pot definir un graf d'estats pels qual pot passar un servei, així com indicar si es tracta d'un estat final. També és possible lligar paràmetres a un canvi d'estat.
- Estats dels vehicles: També existeix un graf d'estats pels vehicles, i també porten lligats paràmetres que s'han de passar o no al fer una transició entre ells.

A part d'aquesta configuració dinàmica, a l' Oracle es guarda tot el que sigui possible, de manera que ni clients ni servidors necessitin de la presència de *Hardcode*. Així doncs, la informació dels vehicles, dels usuaris, dels clients (així com les seves preferències), etc, residiran a l' Oracle.

Una altra funció del sistema que està implementada directament a l' Oracle és la gestió de dades històriques. Quan es produeixen canvis en una sol·licitud, servei o vehicle, es guarden automàticament en taules d'històrics de canvis a través de l'execució de triggers, permetent la posterior consulta d'aquesta informació (possible a MNNS, però no inclòs a Servicleop).

5 – Disseny del sistema

Fins ara hem comentat d'una forma analítica com està dissenyat el sistema, centrant-nos en la part de servidors que és la que he dut a terme. En aquest apartat entrarem al detall de com està dissenyat el sistema.

5.1 – Disseny de base de dades

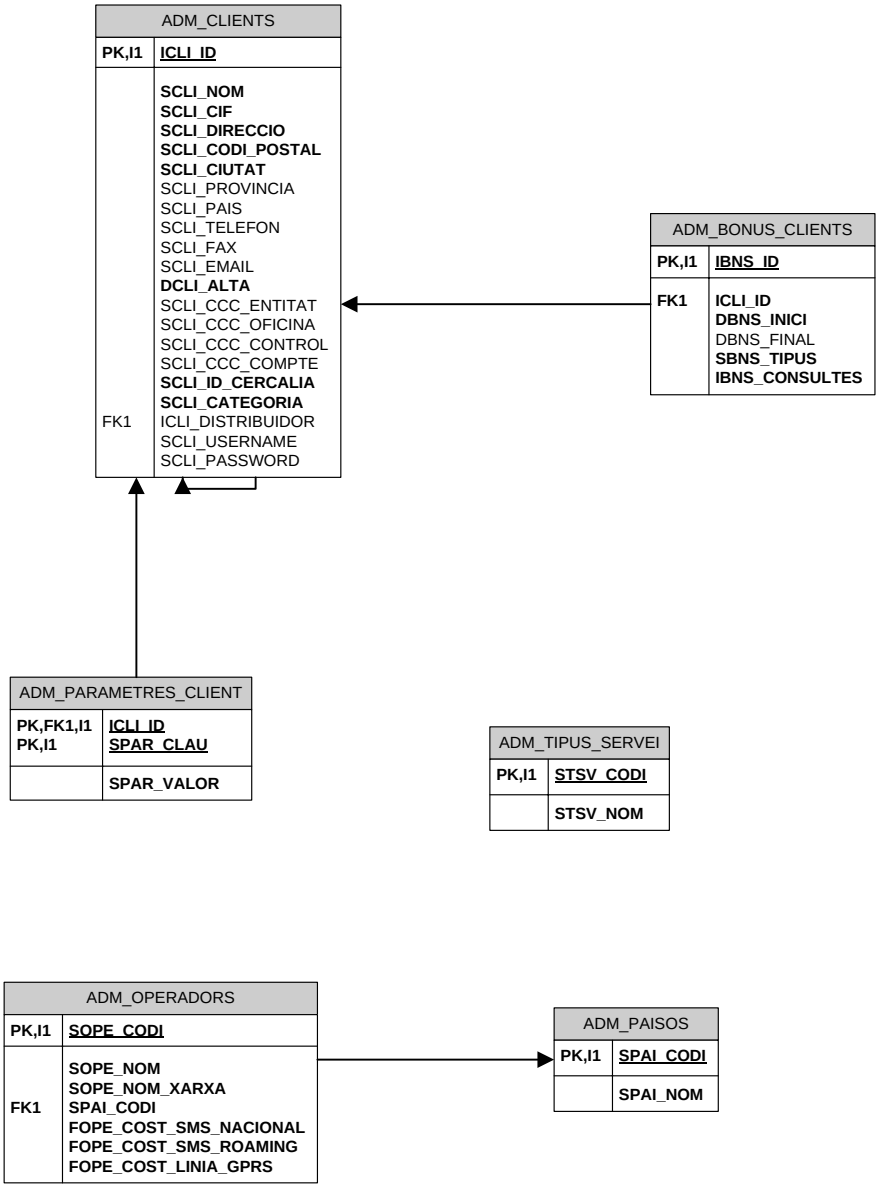
Ja hem mencionat que la base de dades és de molta importància en aquest projecte, com en el 99% dels casos, però en aquest cas és d'una criticitat enorme.

Com hem comentat, les dades de parametrització de client estan a la base de dades. Un error en la introducció d'algun valor pot fer que el sistema no es comuniqui adequadament, o que es mostri informació errònia. També pot fer que les dades siguin descartades abans del que el client desitjaria, o que es produeixi algun altre efecte no desitjat.

També es manegen volums enormes de dades (no tant a Servicleop, però sí a MNNS). Això fa que la correcta definició de les taules i sobretot dels índexs requerits per a les consultes sigui elemental, ja que el temps d'execució d'una consulta pot veure's reduït fins a 50 vegades o més (depenent de la consulta i del volum de dades de les taules involucrades). Això és d'especial importància en el Servidor de Temps Real, que ha d'oferir una resposta immediata a les peticions i que a més rep moltíssima informació (tant dels vehicles, que envien posicions periòdicament o canvis d'estats o de paràmetres), com dels clients (que realitzen enviaments i han de ser notificats de cada acció dels vehicles que controlen).

5.1.1 – Administració

En aquest apartat mostrarem les taules relacionades amb la gestió de clients i l'administració de l'aplicació, i comentarem la seva utilitat. És important deixar clar que al tractar-se d'una aplicació dissenyada per a atendre a múltiples clients, amb configuracions dispars totes les taules d'elements del sistema estaran relacionades amb la taula principal de l'aplicació, que és la de clients i s'anomena "ADM_CLIENTS".



ADM_CLIENTS: Es tracta de la taula principal de l'aplicació, i és on es donen d'alta els diferents clients que es poden tenir. El camp ICLI_ID és la clau principal de la taula, i serà clau forana a totes les taules on aparegui aquest nom de camp. En aquesta taula, a més dels codis de client, s'emmagatzemen dades com el nom o CIF de l'empresa, la direcció o la facturació. És possible que un client sigui distribuïdor d'un altre que estigui donat d'alta i és per aquest motiu que la taula es relaciona amb sí mateixa.

ADM_BONUS_CLIENTS: És una taula utilitzada per saber la configuració de consultes de posicionament dinàmiques de que disposa un client segons la seva quota.

ADM_PARAMETRES_CLIENT: Conté un conjunt de parells "CLAU-VALOR" que permet definir per a cada client certs paràmetres a controlar des de l'aplicació, com per exemple el número màxim de serveis que es poden assignar a un vehicle o el criteri per a donar una sol·licitud per finalitzada.

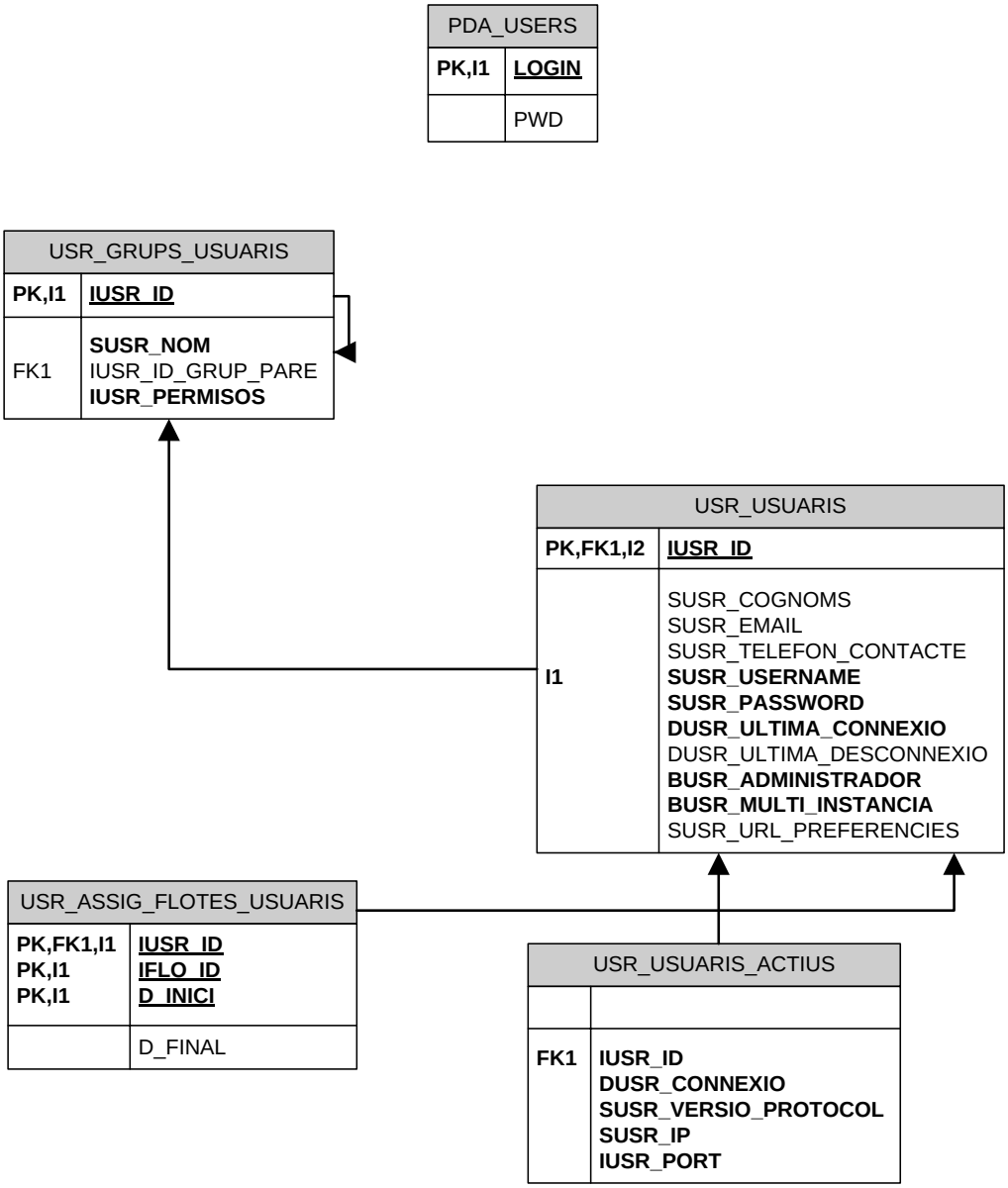
ADM_TIPUS_SERVEI: S'hi defineixen els diferents tipus de servei que es pot oferir als clients.

ADM_OPERADORS: S'hi defineixen els operadors de telefonia.

ADM_PAISOS: S'hi defineixen els països en els quals es pot oferir servei.

5.1.2 – Usuaris

Aquí expliquem les taules de gestió d'usuaris de l'aplicació. Dèiem anteriorment que l'aplicació està molt orientada a client, però dintre d'aquesta orientació, també es troba molt orientada a usuari, amb la qual cosa es veurà en força taules l'existència de camps que facin referència a la clau primària de la taula USR_USUARIS, en el camp IUSR_ID.



USR_GRUPS_USUARIS: En aquesta taula es configuren tots els usuaris, així com el seu nom i permisos (que s'explicaran més endavant). Dintre d'un client trobem grups d'usuaris, i és per aquest fet que la taula es relaciona amb sí mateixa.

USR_USUARIS: Aquesta taula és una extensió de l' anterior. En aquesta hi trobarem entre altres el nom d'usuari i contrasenya amb la qual es connecta a l'aplicació, les dates de les últimes connexions o desconnexions.

USR_USUARIS_ACTIUS: Permet saber quins usuaris es troben connectats a l'aplicació i des de quina IP i Port, així com l'hora de la connexió. Aquesta taula pateix

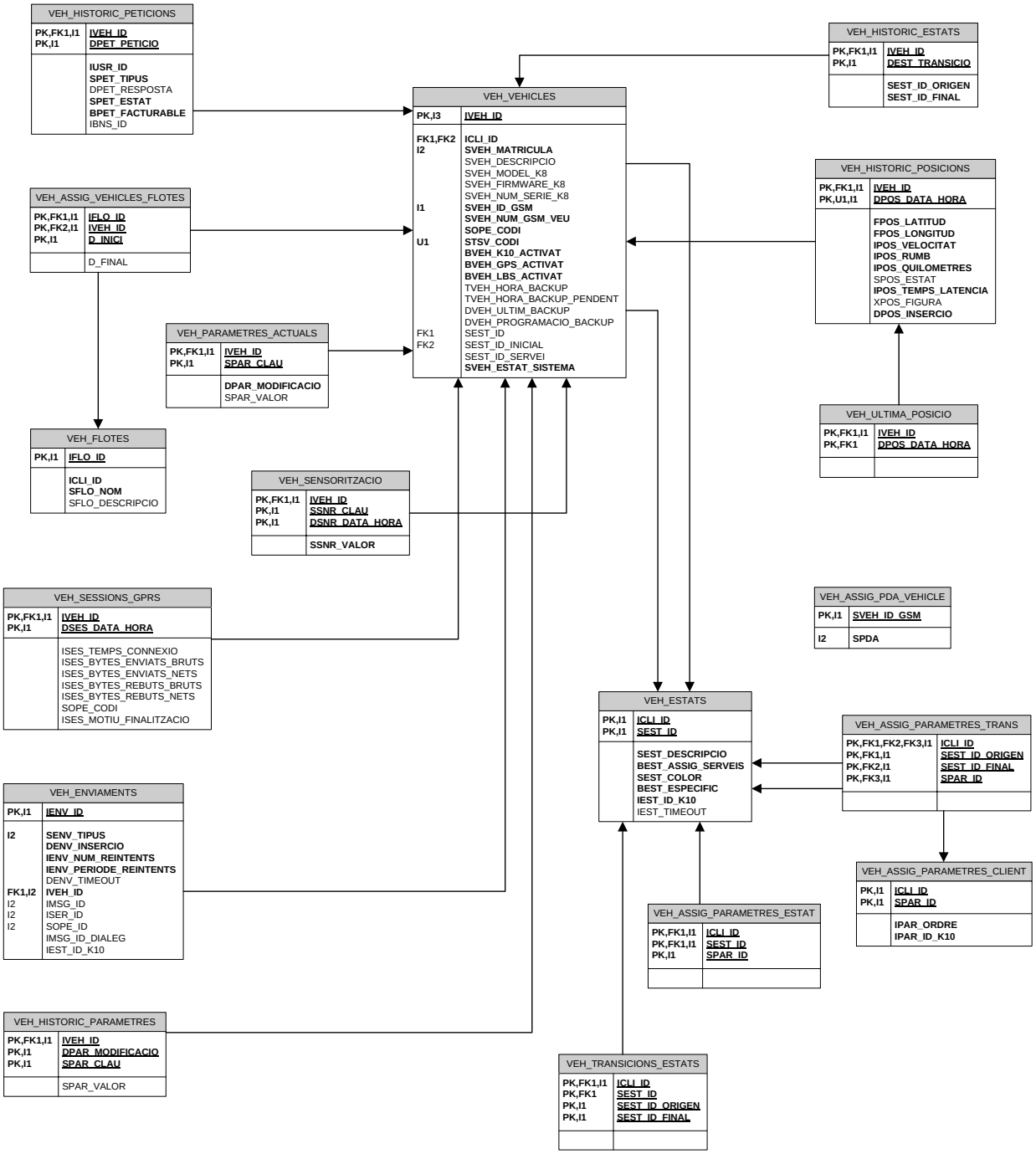
molt moviment ja que a cada connexió o desconnexió s'hi afegeixen o eliminen registres.

USR_ASSIG_FLOTES_USUARIS: Relaciona un usuari amb una o diverses flotes de vehicles. Es disposa d'una data d'inici (l'usuari no pot saber res de la flota anteriorment a aquesta data) i una de finalització (no pot saber res després d'aquesta data).

PDA_USERS: Introduïda a Servicleop, aquesta taula permet la validació de l'usuari que es connecta remotament a la PDA, guardant un nom d'usuari i una paraula de pas.

5.1.3 – Vehicles

En aquest punt expliquem la part del model de la base de dades que fa referència als vehicles.



VEH_VEHICLES: La taula principal del model actual. Aquí es defineixen els vehicles de cada client, amb dades com la matrícula, els identificadors de comunicacions, la versió de K8 que porta instal·lada i la seva referència.

VEH_ESTATS: S'hi defineixen els estats en els quals poden estar els vehicles d'un client. També hi ha camps per poder parametritzar la representació gràfica d'un vehicle en aquest estat.

VEH_HISTORIC_ESTATS: És la taula on es guarda l' historial dels estats en els quals ha estat un determinat vehicle.

VEH_TRANSICIONS_ESTATS: En aquesta taula hi ha la informació sobre els canvis d'estat permesos per als vehicles d'un client.

VEH_ASSIG_PARAMETRES_CLIENT: Aquí s'hi identifiquen els paràmetres que tindran els vehicles d'un client. Aquesta taula es relaciona amb la taula GSE_PARAMETRES d'un altre subesquema.

VEH_ASSIG_PARAMETRES_ESTAT: Defineix quins paràmetres tenen sentit en un determinat estat per als vehicles d'un client.

VEH_ASSIG_PARAMETRES_TRANS: Especifica quins paràmetres s'han d'informar durant un canvi d'estat del vehicle per a un client.

VEH_HISTORIC_PARAMETRES: És la taula on es guarda l' historial del valor dels paràmetres del vehicle.

VEH_PARAMETRES_ACTUALS: Conté el valor dels paràmetres actuals d'un determinat vehicle.

VEH_HISTORIC_POSICIONS: S'hi guarda la informació relativa al posicionament GPS dels vehicles. Degut al gran volum de dades que ha de suportar aquesta taula (cada vehicle pot enviar més d'una posició per minut), es troba particionada per mesos, cosa que permet millorar-ne el manteniment.

VEH_ULTIMA_POSICIO: La taula VEH_HISTORIC_POSICIONS executa un trigger després de la inserció d'una posició que busca en aquesta taula el vehicle que ha generat la posició i n'actualitza les dades. Això permet saber la última posició d'un vehicle determinat sense haver de buscar-la a la taula VEH_HISTORIC_POSICIONS, que com ja hem dit és gegant i lenta de consultar.

VEH_HISTORIC_PETICIONS: S'hi guarda un historial de les peticions de posicionament LBS que hagi sol·licitat un usuari sobre un vehicle.

VEH_FLOTES: Permet crear diferents flotes de vehicles per a un mateix client.

VEH_ASSIG_VEHICLES_FLOTES: Associa els vehicles d'un client a una flota. Aquestes flotes es relacionen amb la taula USR_ASSIG_FLOTES_USUARIS. Hi tenim igual que en aquesta, una data d'inici i una de finalització, que limiten en el temps la pertanyença del vehicle a la flota.

VEH_SENSORITZACIO: S'hi recull la informació dels sensors que pugui portar instal·lats el vehicle, com per exemple indicador de temperatura, de portes obertes, etc.

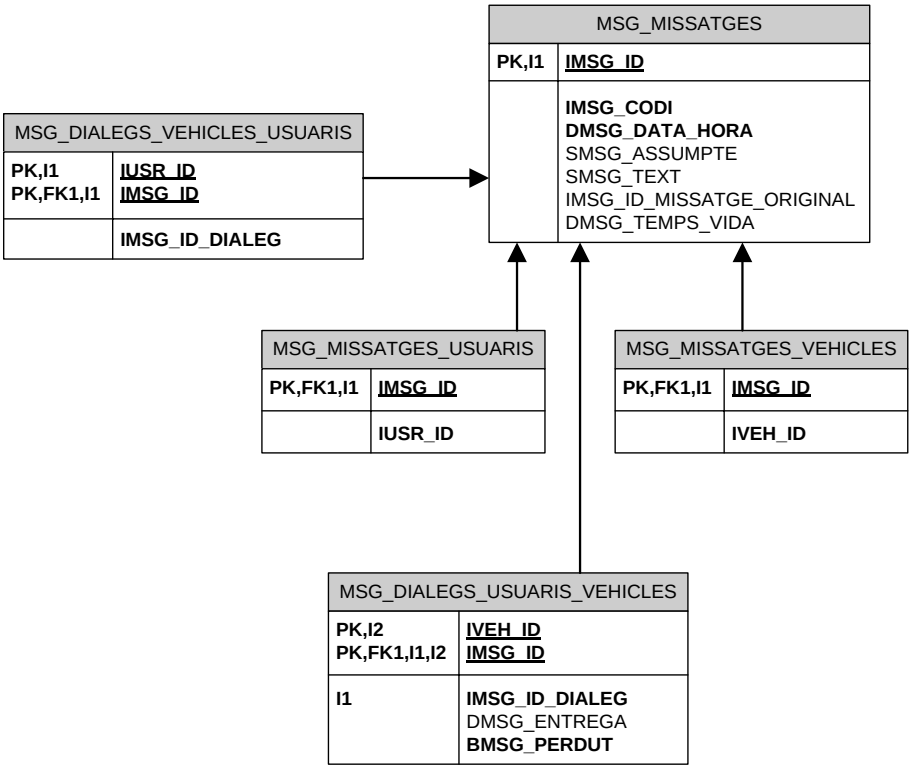
VEH_SESSIONS_GPRS: Guarda un historial de les vegades que ha entrat en sessió GPRS un determinat vehicle.

VEH_ENVIAMENTS: S'hi guarden els enviaments pendents des del centre de control cap a un determinat vehicle. Els enviaments poden ser tant de dades de serveis, com missatgeria o altres operacions. En aquesta taula es controlen el número de reintents, així com la validesa d'un enviament (si no es pot enviar en un període determinat de temps, s'ha de descartar).

VEH_ASSIG_PDA_VEHICLE: Aquesta taula es va introduir a l'esquema per al projecte de Servicleop. Degut a la dualitat en les comunicacions que existeix en aquest cas, on una K8 i una PDA representen una sola entitat, la relació entre aquestes dues es registra en aquesta taula. La relació entre elles no és persistent, i serà la PDA en iniciar una sessió de treball la que informarà al servidor amb quina K8 està treballant.

5.1.4 – Missatgeria

Aquí detallarem les taules involucrades en la funcionalitat de missatgeria.



MSG_MISSATGES: És la taula principal d'aquest mòdul, i és on es guarda l' identificador de missatge, data i hora d'enviament, assumpte, text i temps de validesa del missatge. També podrem veure si el missatge és una resposta a un altre o és un missatge independent.

MSG_MISSATGES_USUARIS: Lliga un missatge amb l'usuari que l'ha enviat.

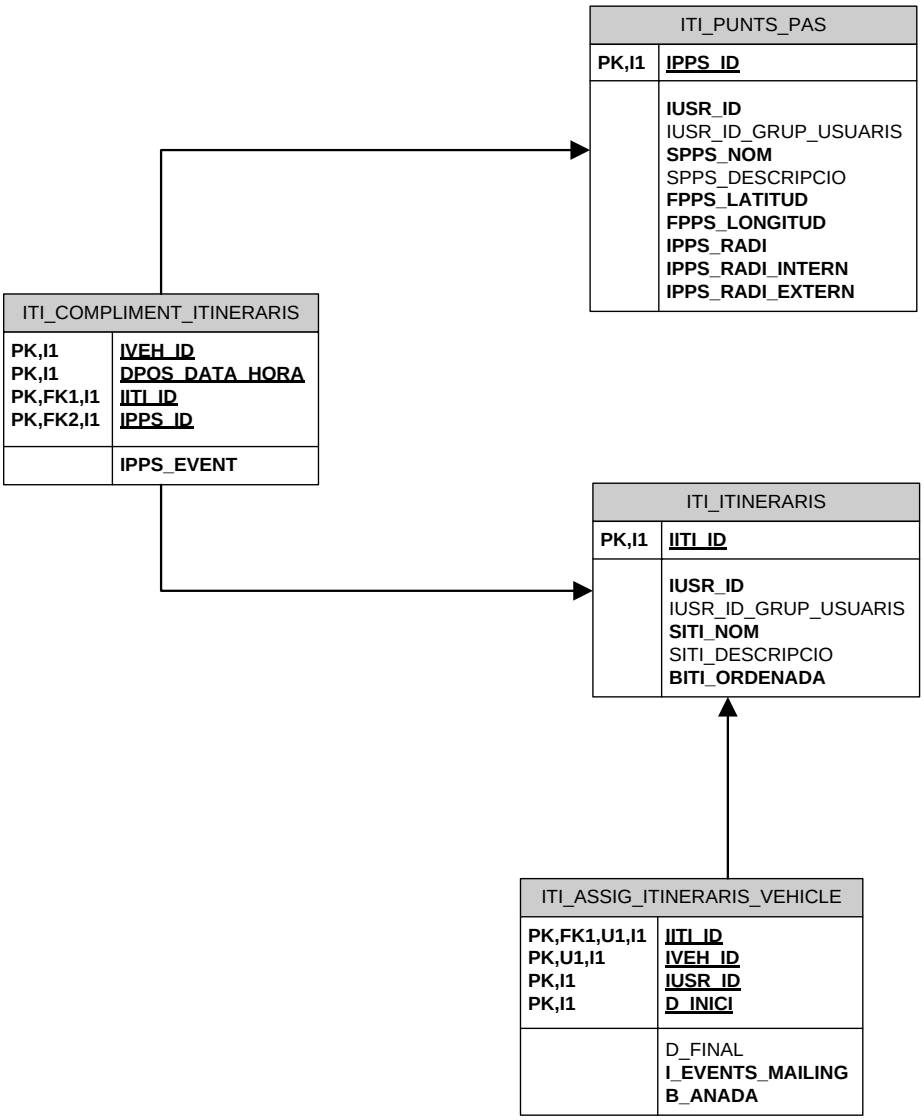
MSG_MISSATGES_VEHICLES: Lliga un missatge amb el vehicle que l'ha enviat.

MSG_DIALEGS_USUARIS_VEHICLES: Conté una entrada per cada missatge enviat a un vehicle, on s'indica l' identificador de diàleg, la data d'entrega del missatge si és que s'ha entregat, o si s'ha perdut (el temps de validesa ha caducat sense que s'hagi pogut enviar).

MSG_DIALEGS_VEHICLES_USUARIS: Conté una entrada per cada missatge enviat a un usuari, on s'indica l' identificador de diàleg.

5.1.5 – Itineraris

El subsistema de gestió d'itineraris està composat per les següents taules.



ITI_PUNTS_PAS: Defineix un punt de pas, on se li assigna un ID, l'usuari que el crea, el nom del punt de pas i una descripció, i la seva localització. Permet establir també uns radis, que atorguen certa flexibilitat al punt de pas.

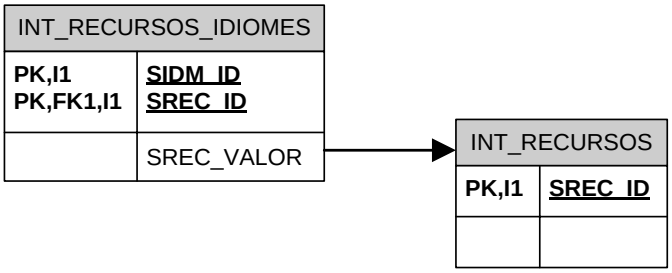
ITI_ITINERARIS: Guarda la informació d'un itinerari, com l' identificador de l'usuari que l'ha creat, i el nom i descripció de l' itinerari. Permet saber també si l' itinerari s'ha de complir ordenadament o no.

ITI_ASSIG_ITINERARIS_VEHICLE: Permet assignar a un usuari un itinerari existent a un vehicle. Es pot establir el rang de dates de validesa d'aquest itinerari, així com si cal enviar un correu electrònic en produir-se algun event sobre l' itinerari.

ITI_COMPLIMENT_ITINERARIS: Permet assignar els punts de pas a un determinat itinerari per a un vehicle. S'hi guarda també a quina hora caldria que el vehicle passés per un cert punt de pas.

5.1.6 – Internacionalització

Al tractar-se d'una aplicació multiidioma pensada per oferir servei tant a clients nacionals com estrangers, i al ser una aplicació parametritzable a través de base de dades per oferir la major flexibilitat i adaptabilitat possibles, cal que els paràmetres variables entre clients estiguin definits en diversos idiomes.



INT_RECURSOS: Estableix una clau per a un recurs de text a ser representat.

INT_RECURSOS_IDIOMES: Permet per a cada registre de la taula anterior, definir-ne la representació gràfica per als idiomes que desitgem.

5.1.7 – Gestió de serveis

Aquesta és possiblement la part més important de l'esquema, a la vegada que la més complexa i parametritzable. Aquí s'hi defineixen el que entenem per sol·licituds, serveis i tasques, així com els seus grafs d'estats i paràmetres. També hi guardem els històrics, i les relacions entre tots ells.

El primer que cal fer abans de posar el sistema en funcionament per a un client és parametritzar correctament les taules. Aquestes són les taules a parametritzar:

GSE_PARAMETRES: Defineix un paràmetre per a un client. Permet guardar-ne una descripció. S'hi registra el tipus de dades del paràmetre, així com el seu rang de valors permesos. També s'hi guarda l' identificador amb el qual es codificarà en la transmissió de dades.

GSE_SOL_TIPUS: Defineix els diferents tipus de sol·licitud que pot tenir un client. Permet guardar la descripció, el color amb el qual es pintarà per pantalla, l'estat inicial en què es trobaran els serveis per a aquest tipus de sol·licitud, i quina operació permet anul·lar-los.

GSE_SER_ESTATS: Defineix els estats en què pot estar un servei d'un client. Permet guardar la descripció, el color amb el qual es pintarà per pantalla, i l' identificador amb el que s'enviarà o rebrà en les comunicacions.

GSE_SER_TRANSICIONS_ESTATS: Defineix un graf d'estats que han de seguir els serveis d'un client, per a un tipus de sol·licitud determinat.

GSE_SER_OPERACIONS: Assigna una operació a un client. Ens permet entrar el nom de l'operació i la descripció d'aquesta.

GSE_SER_OPERACIONS_CLIENT: Per cada registre de l' anterior, ens permet associar l'operació a un tipus de sol·licitud diferent, i defineix un canvi d'estat que serà el que forçarà l'execució d'aquesta operació sobre un determinat servei.

GSE_TSC_TIPUS: Defineix els diferents tipus de tasques que poden tenir els serveis d'un client. Hi podem guardar el color amb el qual es representa per pantalla, la descripció, i l'estat inicial de les tasques d'aquest tipus.

GSE_TSC_ESTATS: És anàloga a la taula GSE_SER_ESTATS, però per a les tasques.

GSE_TSC_TRANSICIONS_ESTATS: És anàloga a la taula GSE_SER_TRANSICIONS_ESTATS, però per a les tasques.

GSE_LOCALITZACIO: Les sol·licituds i tasques poden portar associada una geolocalització, fet que farà que es mostrin en la cartografia de l'aplicació client o no. En aquesta taula es guarden les dades de les diferents localitzacions que es van assignant.

GSE_SOL_SOLICITUDS: Amb la creació d'una sol·licitud és com comença el flux de vida d'un servei. En una sol·licitud guardarem el tipus, un text descriptiu, la seva prioritat, el seu estat actual, la localització, i també l'usuari que l'ha creada i l'usuari que la gestiona. També s'hi guarden una sèrie de dates que ens indiquen quan s'ha creat, o quan s'han modificat alguns dels seus valors.

GSE_SOL_PARAMETRES: Una sol·licitud pot portar associats diferents paràmetres. Aquests es guarden en aquesta taula, que conté parells tipus CLAU-VALOR, a més de la data de modificació del valor del paràmetre. Els paràmetres no són modificables des del vehicle.

GSE_SOL_HISTORIC_ESTATS: Les sol·licituds també passen per un graf d'estats. La diferència és que aquest graf és predefinit, i per això no el guardem en una taula com es fa amb serveis i tasques. En aquesta taula queda registrat el moment en què certa tasca ha canviat d'un estat a un altre.

GSE_SER_SERVEIS: Cada sol·licitud pot generar un o diversos serveis. En aquesta taula es guarden les dades d'aquests serveis, relacionant el registre amb la sol·licitud, i guardant també a quin vehicle s'ha associat i la data de la seva creació.

GSE_SER_PARAMETRES: Taula anàloga a GSE_SOL_PARAMETRES però per a serveis. Aquests paràmetres els modifiquen els vehicles, i des de l'aplicació client només es poden consultar.

GSE_SER_HISTORIC_ESTATS: Taula anàloga a GSE_SOL_HISTORIC_ESTATS però per a serveis.

GSE_TSC_TASQUES: Cada sol·licitud pot portar associades un conjunt de tasques. En aquesta taula es guarden aquestes tasques, juntament amb l'associació a la seva localització, un text descriptiu i la data de creació.

GSE_ASSIG_TASQUES_SOLICITUD: Aquí es fa l'associació entre tasques i sol·licituds.

GSE_TSC_PARAMETRES_DEFECTE: És anàloga a GSE_SOL_PARAMETRES, però per a les tasques. Aquests paràmetres no són modificables des de vehicle, i el seu valor el tindran en comú tots els serveis que es generin que continguin aquesta tasca.

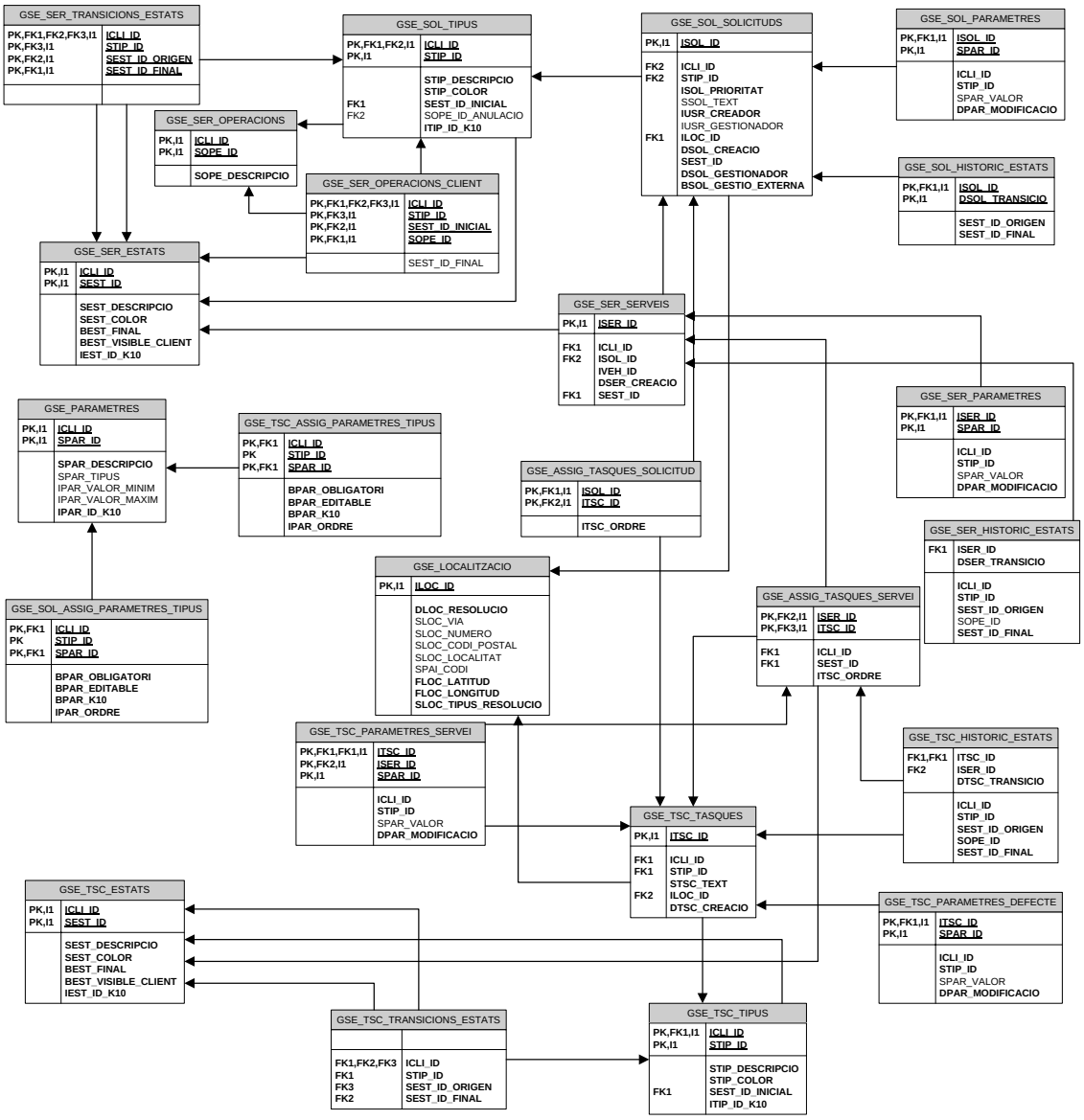
GSE_ASSIG_TASQUES_SERVEI: En el moment en què una sol·licitud que té tasques associades, s'assigna a un vehicle, es genera un servei. Al moment de fer-ho, els registres que existien a GSE_ASSIG_TASQUES_SOLICITUD són copiats a aquesta taula, creant així una associació entre la tasca i el servei.

GSE_TSC_PARAMETRES_SERVEI: Aquesta taula és una extensió de GSE_TSC_PARAMETRES_DEFECTE, però en aquest cas guarda els paràmetres associats a la tasca que està associada a un servei determinat, i no els paràmetres comuns. Aquests paràmetres són modificables només pels vehicles.

GSE_TSC_HISTORIC_ESTATS: Anàloga a GSE_SER_HISTORIC_ESTATS, però per a una tasca associada a un determinat servei.

GSE_SOL_ASSIG_PARAMETRES_TIPUS: Permet assignar els paràmetres definits als diferents tipus de sol·licituds. S'hi defineix si són obligatoris, editables, i amb quin identificador s'envien.

GSE_TSC_ASSIG_PARAMETRES_TIPUS: Anàloga a l' anterior, però per als tipus de tasques.



5.2 – Disseny de comunicacions

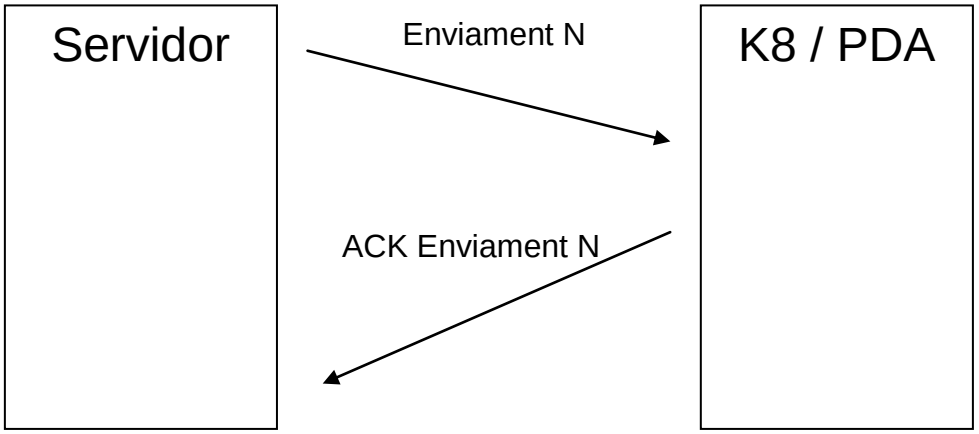
És vital en un projecte d'aquest tipus tenir un bon disseny de les comunicacions entre els diferents extrems. No fer-ho podria implicar que es rebés informació errònia, o que es perdessin dades que són vitals per als usuaris.

5.2.1 – Flux de comunicacions

En aquest apartat descriurem les diferents possibilitats que existeixen durant les comunicacions entre vehicles i servidors.

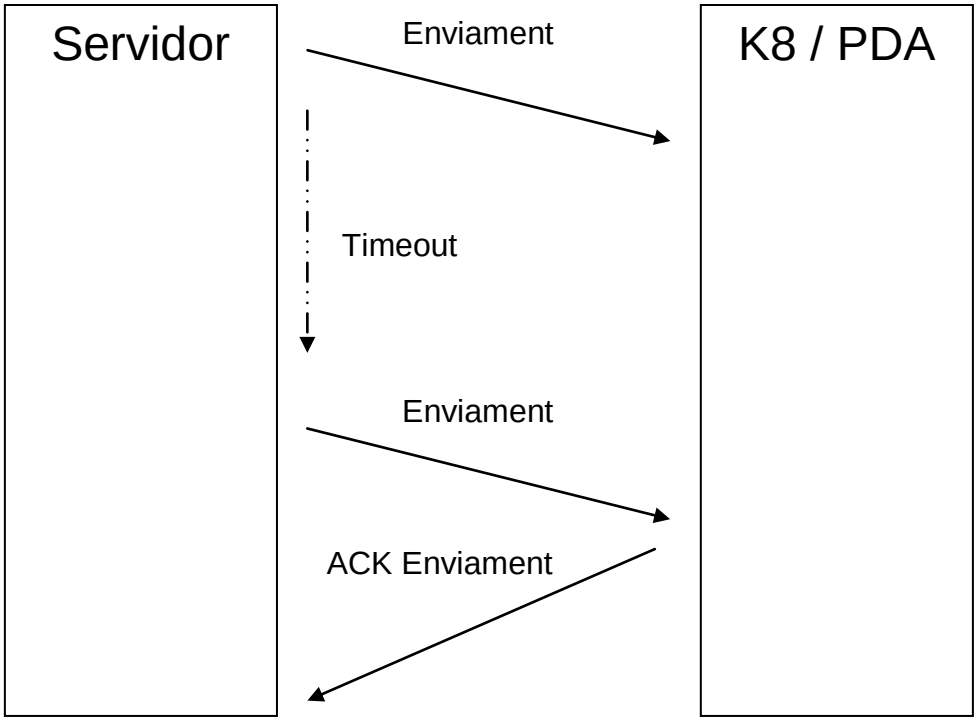
5.2.1.1 – Enviament correcte servidor-vehicle

Aquest és el cas més habitual de comunicacions en el sentit servidor-vehicle. El servidor té un enviament (identificat unívocament amb un codi) pendent d'enviar. L'envia a través de la xarxa GPRS, i el vehicle respon un ACK amb l' identificador per tal que el servidor sàpiga que ja s'ha entregat.



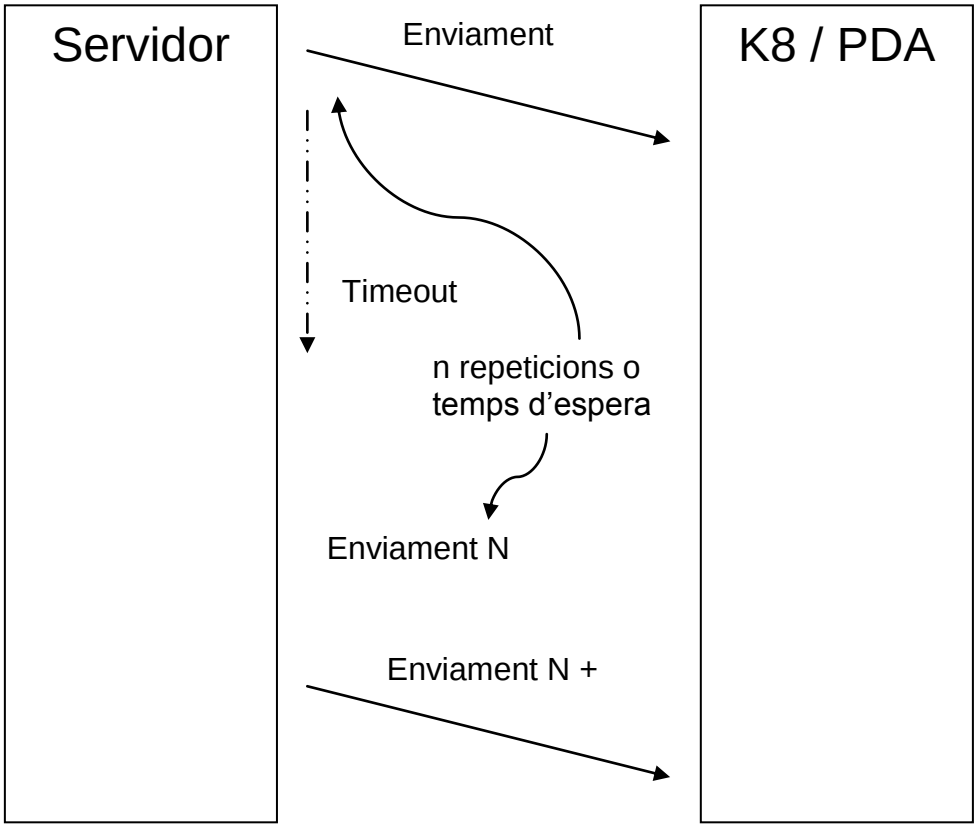
5.2.1.2 – Enviament servidor-vehicle amb reintent

Quan el servidor no rep resposta d'un enviament, ha de reintentar l'entrega del paquet. Segons el tipus de paquet i les dades i parametritzacions dels usuaris, el paquet pot ser reintentat un cert número de vegades, o descartar-se després d'un període de temps. Hi ha diversos motius pels quals un paquet pot no ser reconegut, essent els més freqüents la falta de cobertura GPRS o que el vehicle / terminal no es trobi connectat.



5.2.1.3 – Enviament servidor-vehicle amb descart

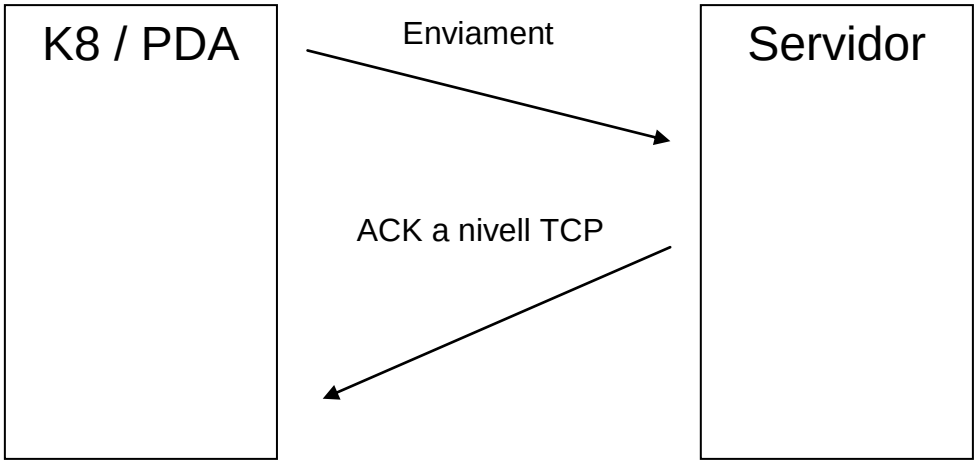
Quan un paquet ha exhaurit el seu número de reintents o el seu temps de vida sense ser reconegut, és descartat i es procedeix al següent enviament.



5.2.1.4 – Enviament correcte vehicle-servidor

Aquest és el cas més freqüent de comunicacions, ja que els vehicles contínuament estan transmetent dades de posicionament al servidor. Cal destacar que el servidor no envia un ACK de resposta de recepció al vehicle, sinó que s'aprofita el propi ACK a nivell TCP com a validació de l'entrega del paquet. Això és així ja que al partir d'un projecte on principalment es transmetien posicions GPS, no es va considerar necessari definir reconeixements negatius donada la volatibilitat i variabilitat de les dades (si arriba una posició errònia, no té sentit demanar que en torni a enviar una altra ja que igualment ho farà en pocs segons). S'està estudiant la possibilitat d'incorporar els ACKs a nivell

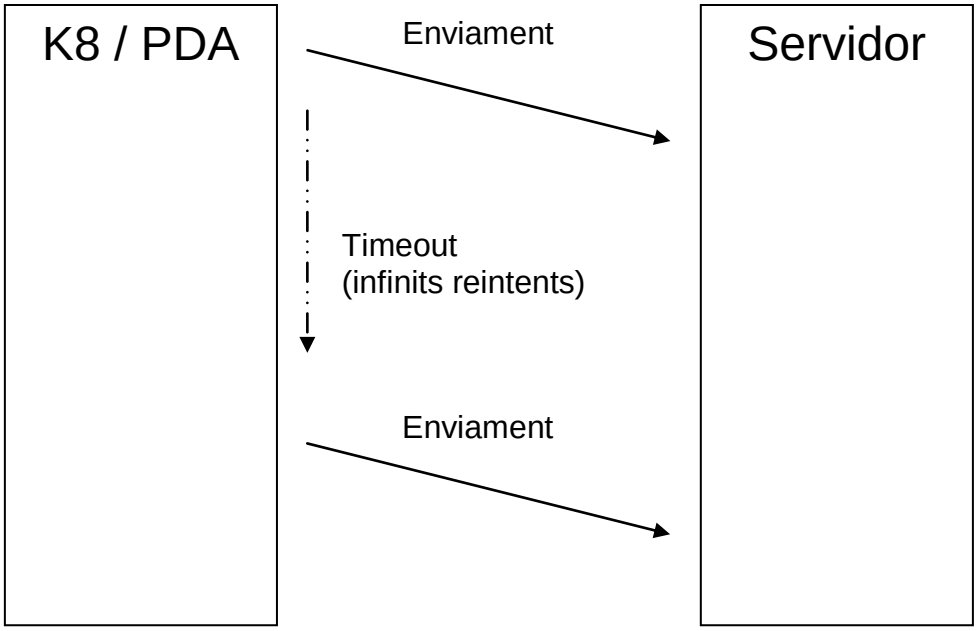
d'aplicació en els projectes que han derivat de Micronav.Net per garantir la integritat de les dades.



5.2.1.5 – Enviament vehicle-servidor amb reintent

Aquest cas és molt poc probable, i cada vegada ho és menys a mesura que la plataforma va madurant. Això és així perquè en el cas que no hi hagi reconeixement a nivell TCP i el vehicle hagi de reintentar, implica que el servidor ha deixat de respondre (o que hi ha algun error en l' enrutament de xarxa).

Els terminals mòbils estan programats per reintentar un enviament infinites vegades fins que reben l'ACK a nivell TCP.



5.3 – Trames existents

En aquest punt descriurem els diferents tipus de trames existents en l'aplicació, tant en el sentit servidor-vehicle / PDA com a la inversa. També mostrarem la seva codificació.

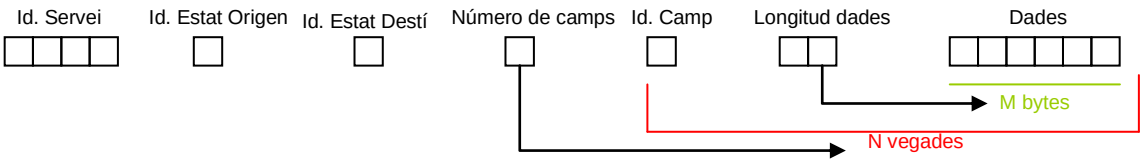
5.3.1 – Vehicle / PDA -> Servidor

Aquí presentem les diverses trames que es poden enviar des de l'equip K8 o des de la PDA cap al servidor.

5.3.1.1 – Canvi d'estat de servei

Aquesta trama és enviada per la PDA quan evoluciona un dels seus serveis, per exemple, quan es passa un servei de "pendent" a "rebut", o quan finalitza. Els canvis d'estat porten un conjunt de paràmetres associats, i per tant es codifiquen dins la trama.

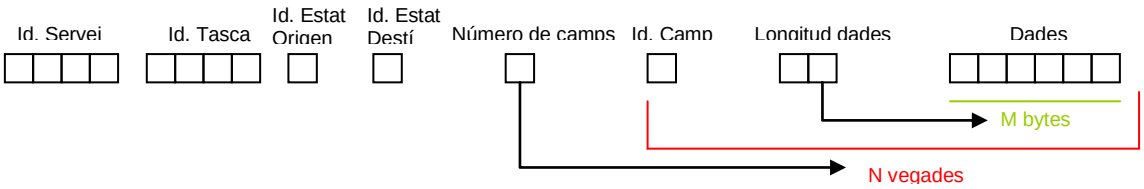
Codificació:



5.3.1.2 – Canvi d'estat de tasca

És una trama pràcticament idèntica a l' anterior, amb la diferència que fa referència a una tasca en comptes d'un servei. A Servicleop no és utilitzada ja que no hi ha tasques, però es manté el codi per si en futures ampliacions es vol utilitzar. Aquests canvis d'estat també porten paràmetres associats.

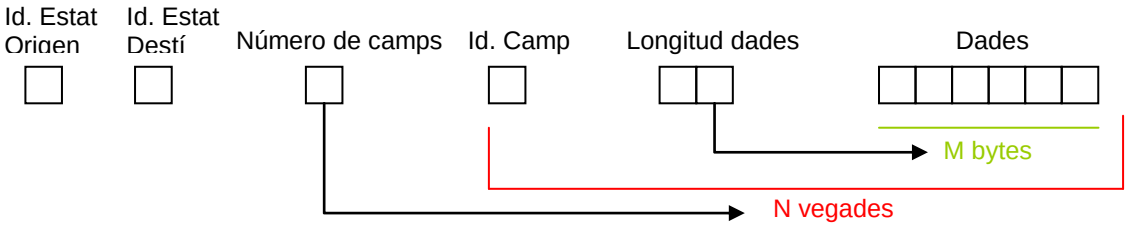
Codificació:



5.3.1.3 – Canvi d'estat de recurs

Aquesta trama l'envia un vehicle o PDA en el moment en què canvia d'estat. Això pot ser per exemple el pas de "Operatiu" a "En Descanso". Aquests canvis poden portar també paràmetres associats, essent de nou una trama molt similar a l' anterior.

Codificació:



5.3.1.4 – Resposta a canvi d'estat de servei

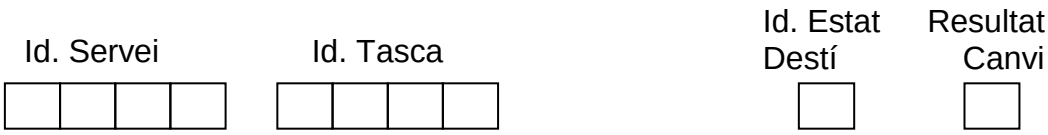
Quan des del centre de control es desitja anul·lar un servei, el vehicle / PDA ens contestarà si ha pogut realitzar o no el canvi d'estat amb aquesta trama.

Codificació:

5.3.1.5 – Resposta a canvi d'estat de tasca

Idèntica a l' anterior, però per a una tasca.

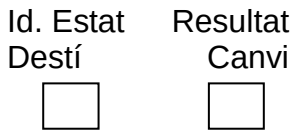
Codificació:



5.3.1.6 – Resposta a canvi d'estat de recurs

Novament idèntica a l' anterior, però per a l'estat del vehicle o PDA.

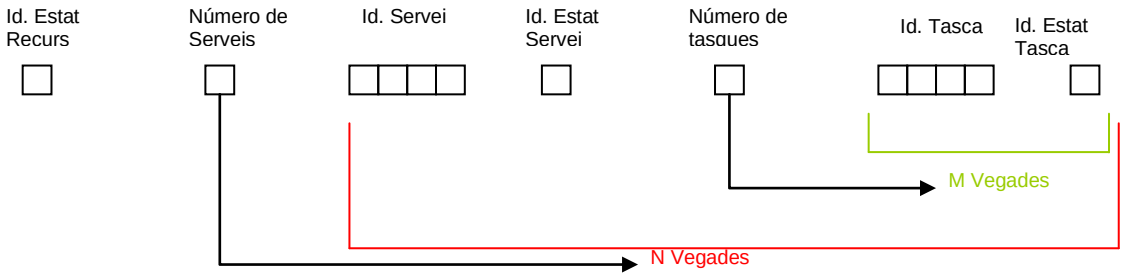
Codificació:



5.3.1.7 – Startup

Quan el vehicle o PDA arranca, envia una trama al servidor amb les dades de serveis que té actives en memòria i el seu estat. En aquest moment el servidor valida la validesa d'aquestes dades, i en cas d'incongruències actua intentant solucionar el conflicte.

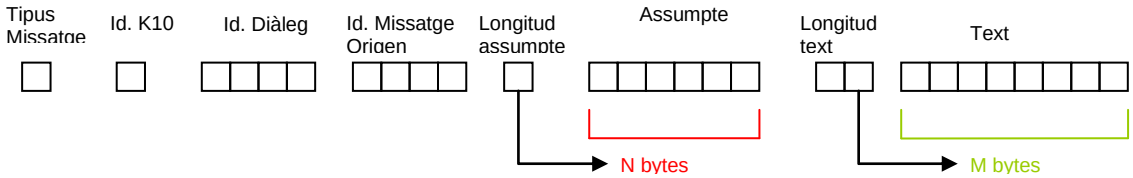
Codificació:



5.3.1.8 – Missatge de text

El vehicle o PDA pot enviar missatges de text al centre de control, a l'estil SMS. A Servicleop no hi ha implementada aquesta funcionalitat (que sí que existeix en sentit invers), però es manté per possibles ampliacions.

Codificació:



5.3.1.9 – Sync

Aquesta trama l'envia la PDA a l' engegar-se. És nova en el projecte de Servicleop, i serveix per sincronitzar l'hora de la PDA amb la del servidor.

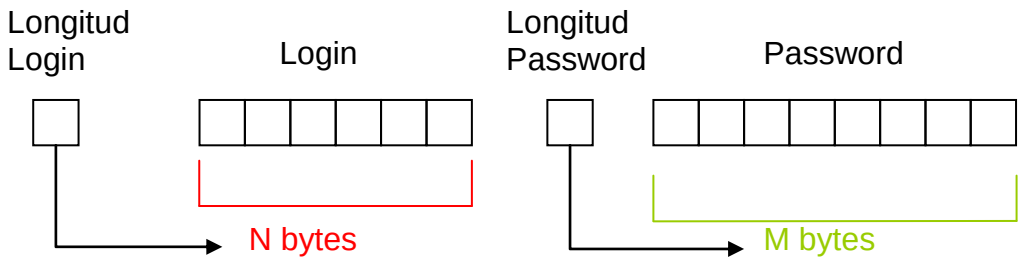
Codificació:

No hi ha cap codificació específica, només cal l' identificador de la trama perquè el servidor sàpiga com ha de contestar.

5.3.1.10 – Login

Aquesta trama també és nova de Servicleop, i es fa servir per validar remotament l'usuari de la PDA.

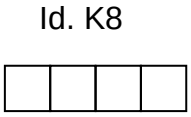
Codificació:



5.3.1.11 – Associació PDA – K8

Una altra trama nova per al projecte de Servicleop, l'envia la PDA a l' engegar-se informant al servidor de quina K8 té associada. D'aquesta manera, el servidor pot tractar d'igual manera les dades que rep a través de la PDA (serveis, missatgeria, etc.) i les que rep de la K8 (posicionament).

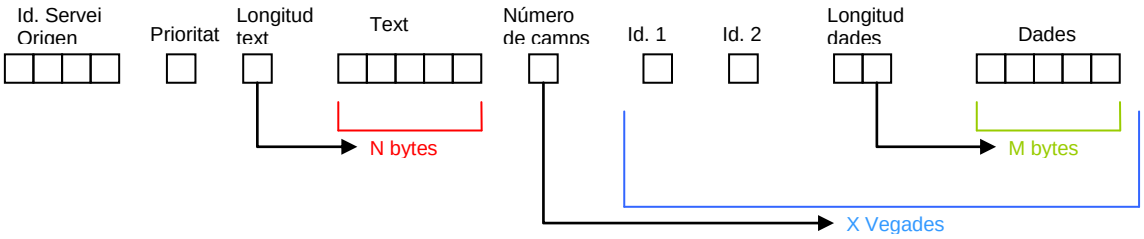
Codificació:



5.3.1.12 – Nou servei PDA

La última trama particular per a Servicleop, permet a la PDA enviar, en finalitzar un servei, una petició al servidor per a què li enviï un servei idèntic al que acaba de finalitzar. Això és útil per exemple en el cas en què la grua retiri un vehicle mal aparcat, i s'adoni que n'hi ha més de mal aparcats en el mateix carrer o aparcament.

Codificació:



5.3.1.13 – Posició GPS

Aquestes trames són enviades periòdicament per les K8 per informar de la localització dels vehicles. Totes les trames a més, porten una posició GPS associada, encara que en el cas de Servicleop les posicions que envia la PDA són invàlides (perquè el servidor no les tracti, ja que la PDA no porta GPS), i per això existeix la dualitat K8-PDA.

5.3.1.14 – ACK

És la trama que envien els vehicles / PDA per informar al servidor que han rebut un paquet i que ja el pot treure de la seva cua d'enviaments.

Codificació:

Id. Enviament			

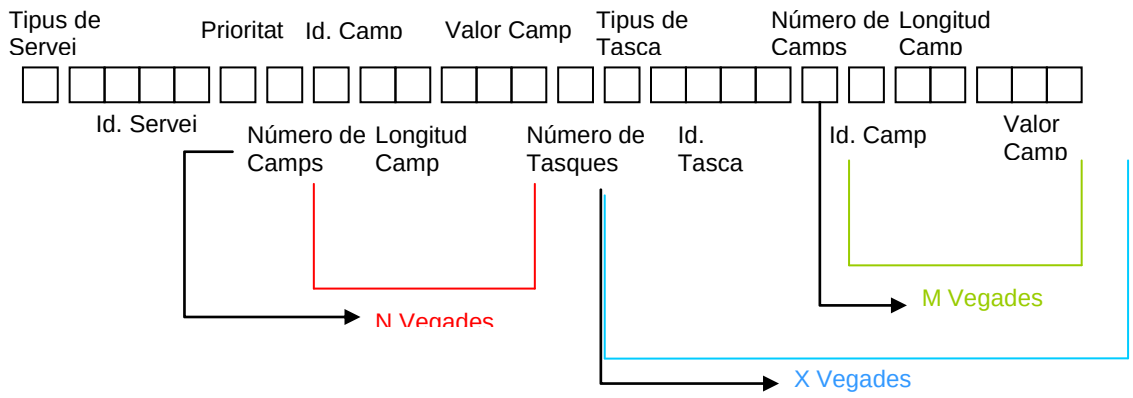
5.3.2 – Servidor -> Vehicle / PDA

En aquest punt veurem les trames en sentit invers, de servidor cap a PDA.

5.3.2.1 – Nou servei

És la trama que representa un nou servei per al vehicle (una nova retirada de grua a Servicleop). Conté l' identificador, i els paràmetres del servei, així com les tasques i els seus paràmetres si procedeix.

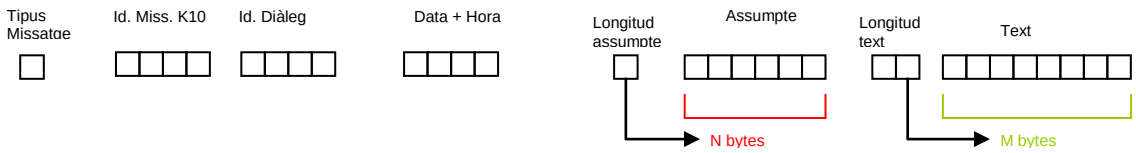
Codificació:



5.3.2.2 – Missatge de text

Representa l'enviament d'un missatge de text des de centre de control cap a un vehicle / PDA.

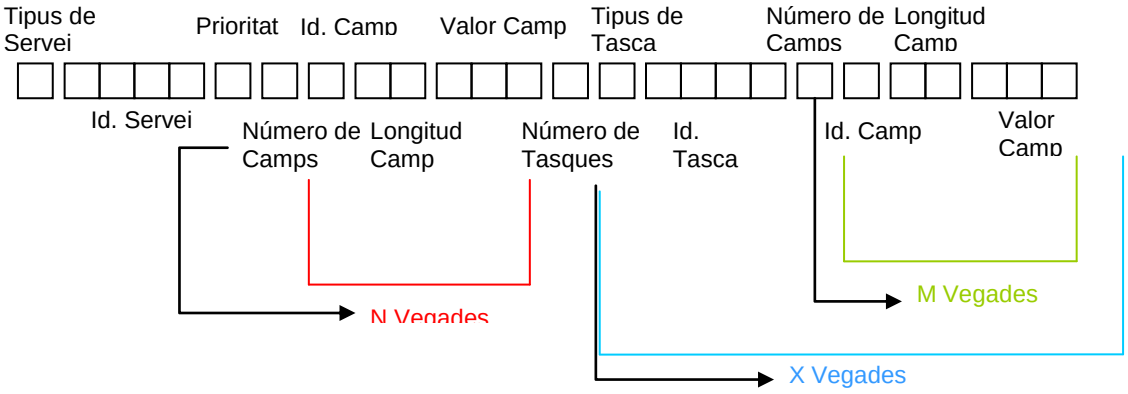
Codificació:



5.3.2.3 – Canvi de paràmetres

Si des del centre de control es modifica un servei que ja està en curs, s'enviarà una trama de canvi de paràmetres informant de les dades modificades. Aquesta trama és idèntica a la de creació de servei, excepte pel byte on s'indica la tipologia.

Codificació:



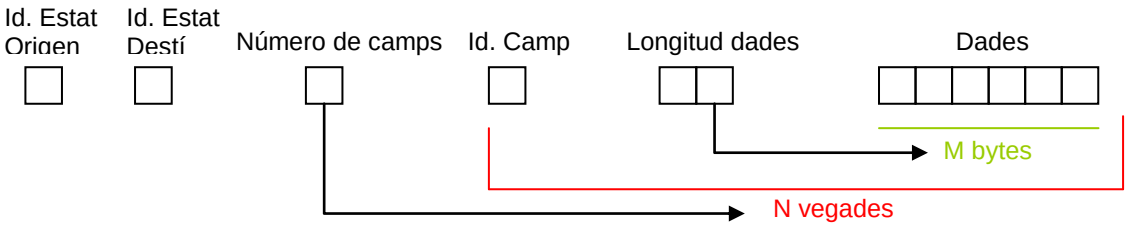
5.3.2.4 – Anul·lació

Des del centre de control és possible anul·lar un servei en curs. S'enviarà aquesta trama al vehicle, que haurà de decidir si pot o no pot portar-la a terme.

5.3.2.5 – Canvi d'estat de vehicle

Des de centre de control és possible modificar l'estat operatiu d'un vehicle. Per a fer-ho, s'utilitza aquesta trama. És possible adjuntar paràmetres de vehicle en aquesta trama.

Codificació:



5.3.2.6 – Resposta Sync

Aquesta trama és nova pel projecte de Servicleop, i és la resposta a la petició de sincronització que envia la PDA. D'alguna manera es pot considerar un ACK a nivell d'aplicació de la trama de sincronització, cosa que canvia respecte la resta de trames. S'envien la data i hora actuals.

5.3.2.7 – Resposta Login

Aquesta trama també és nova per a Servicleop, i igual que l' anterior, trenca una mica amb la resta al ser un ACK a nivell d'aplicació. La trama indica si l'usuari i password introduïts són correctes i es permet l' inici de sessió en PDA.

6 – Implementació

En aquest capítol es vol donar una visió dels equips, entorns de treball i tecnologies que s'han utilitzat per al desenvolupament de les aplicacions per al projecte Servicleop, així com les parts clau de la programació del projecte.

6.1 – Entorns, eines i tecnologies de desenvolupament

- Borland Delphi 7

S'ha utilitzat l'entorn de Borland Delphi 7 per al desenvolupament del Servidor de Temps Real, així com pel desenvolupament del Servidor GPRS i també pel servei d'exportació de dades de Cesser. La seva potència en connectivitat a base de dades, així com la velocitat tant de les aplicacions escrites en aquest llenguatge com de desenvolupament, que és orientat a objectes, han estat claus a l'hora de l'elecció.

- Borland JBuilder X

Per al desenvolupament dels Servlets i dels Webservices s'ha escollit l'eina de Borland JBuilder X. El desenvolupament en Java facilita la creació d'aplicacions amb un bon disseny orientat a objectes, i ofereix una gran seguretat. L'existència de servidors d'aplicacions gratuïts per a Java com JBoss o Apache han ajudat també a l'elecció d'aquest entorn de treball. Cal remarcar que la velocitat en aquestes aplicacions no resultava tan crítica com en el servidor de temps real.

La facilitat de desenvolupament d'aplicacions web (tant servlets com webservices) han resultat determinants per l'elecció de l'entorn de Borland.

- Eclipse

L'aplicació client del centre de control s'ha desenvolupat també amb Java, aquesta vegada però amb l'eina Open Source Eclipse. El llenguatge Java ha permès crear un

framework amb el qual es deixa a molt alt nivell la gestió de finestres i de fluxos d'informació, facilitant a costa d'un dur procés inicial la implementació de noves funcionalitats o millores.

- Microsoft Visual Studio .Net 2003

Per al desenvolupament de l'aplicació de PDA s'ha escollit una eina de Microsoft, Visual Studio .Net 2003. L'elecció ve motivada per la versatilitat de poder utilitzar el propi llenguatge C# i pràcticament els mateixos controls visuals que es farien servir en una aplicació de finestres convencional en una PDA, així com tot el framework que ofereix ja de per sí la pròpia plataforma .Net, proporcionant un disseny orientat a objectes, connexió a base de dades local (amb Oracle Lite), Sockets, FTP, etc.

- JAXB (Java Architecture for XML Binding)

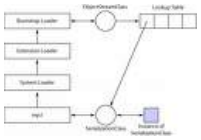
En el cas dels servlets, s'ha utilitzat una eina Open Source de Java coneguda per JAXB. Aquesta eina permet entre altres coses, convertir un model XML en una arquitectura de classes Java. L'estructura de dades que genera és Serialitzable, és a dir, pot ser transmesa i interpretada des d'un altre punt del sistema, permetent obtenir un conjunt de dades estructurat en el servidor i transmetre'l cap al client. Això s'utilitza com s'ha comentat en els servlets de l'aplicació. Per exemple, en el servlet de login s'omple una estructura de dades que conté informació sobre els vehicles, els missatges, etc. i que el client pot interpretar a alt nivell.

- WSDL (WebServices Description Language)

WSDL és un format XML que descriu serveis de xarxa com un conjunt de missatges que es transmeten entre diferents punts. Aquesta definició abstracta està diferenciada de les seves implementacions o de l'enllaç amb les dades.

- SOAP (Simple Object Access Protocol)

És un protocol estàndard que defineix com dos objectes en diferents processos poden comunicar-se a través de l' intercanvi de dades en XML. És un dels protocols utilitzats en serveis web.

- Serialització / Marshalling 

És el procés encarregat de guardar un objecte (programàticament) en un medi d'emmagatzament, com pot ser un disc dur o un buffer de memòria. També pot ser transmès per la xarxa, tant com un conjunt de bytes o en format llegible com pot ser XML. És molt utilitzat per transportar objectes a diferents punts d'una xarxa, i és molt utilitzat en els serveis web.

- Apache AXIS

AXIS és una implementació del protocol SOAP dut a terme per Apache, segons els estàndards del W3C. S'utilitza per desenvolupar serveis web en llenguatge Java.

- Jakarta Digester

És un paquet de Java que permet configurar un mòdul de conversió de XML a Java. Està basat en un conjunt de regles que, quan detecten cert patró en un document XML, executen certes accions.
S'ha utilitzat Digester en el codi del client de centre de control pe a descodificar la informació obtinguda dels servlets.

- Reflection API

És un API de Java que permet obtenir informació sobre objectes que estan actualment en la màquina virtual de Java. Permet determinar la classe d'un objecte, informació sobre els seus mètodes o atributs (així com la seva invocació), crear instàncies d'objectes dels quals no es sap el tipus en temps de compilació, entre altres.

Es fa servir contínuament en la pròpia implementació del llenguatge, i en nostre cas s'ha utilitzat per definir un framework en els serveis web que permetés treballar sempre de la mateixa manera amb molt poca generació de codi.

- Log4Java

  Log For Java és una API de Java que permet una gran potència i flexibilitat per realitzar logs en una aplicació, i penalitza molt poc la legibilitat i velocitat del codi. Es configura a través d'un fitxer pla o XML, on es parametritza com es volen guardar els logs. El destí d'aquests logs pot ser un arxiu pla, un arxiu XML, un servidor de Log4Java remot, etc. S'han utilitzat tant als serveis web com als servlets en la part de servidor.

- Oracle

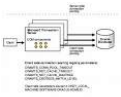
Oracle és un sistema de base de dades, fabricat per Oracle Corporation. És considerat un dels sistemes més potents, destacant el seu suport transaccional, estabilitat i escalabilitat. És per aquests motius que el projecte es desenvolupa amb aquesta base de dades, ja que el tràfic de dades en el cas de molts usuaris i vehicles és molt elevat.

- PL / SQL

PL / SQL és el llenguatge de programació que ofereix Oracle. En el projecte és àmpliament utilitzat, sobretot en Triggers que actualitzen dades en funció de les accions que realitzem o assignen identificadors automàtics amb l'ajuda de seqüències, o inserten valors per defecte. També s'utilitza en algun Stored Procedure, especialment per tasques de manteniment de la base de dades.

- Quest Software TOAD

Per a manipular les bases de dades d' Oracle hem utilitzat l'eina TOAD, que permet un ampli control sobre tots els aspectes de la base de dades oferint una interfície d'usuari molt més senzilla que la de les eines provistes per Oracle.

- Pool de connexions 

La tècnica de Connection Pooling consisteix en tenir sempre un mínim de connexions obertes amb la base de dades, de manera que quan cal atendre una petició no cal esperar el lent procés d'establir una connexió nova. Quan la petició acaba, la connexió, que s'havia retirat del pool, torna a ell. Si en un moment donat totes les connexions estan ocupades, es pot establir una nova connexió per oferir servei. El màxim de connexions és configurable, i cal tenir en compte l'escalabilitat de la plataforma per buscar un valor òptim tant del màxim com del mínim de connexions. Si s'assoleix el límit superior de connexions, la petició espera fins que queda una connexió lliure.

S'ha utilitzat en els servlets i els webservices de la plataforma, ja que han de donar servei a un alt número d'usuaris concurrents, però que fan peticions esporàdiques i que triguen poc temps en ser satisfetes.

- Altova XML Spy

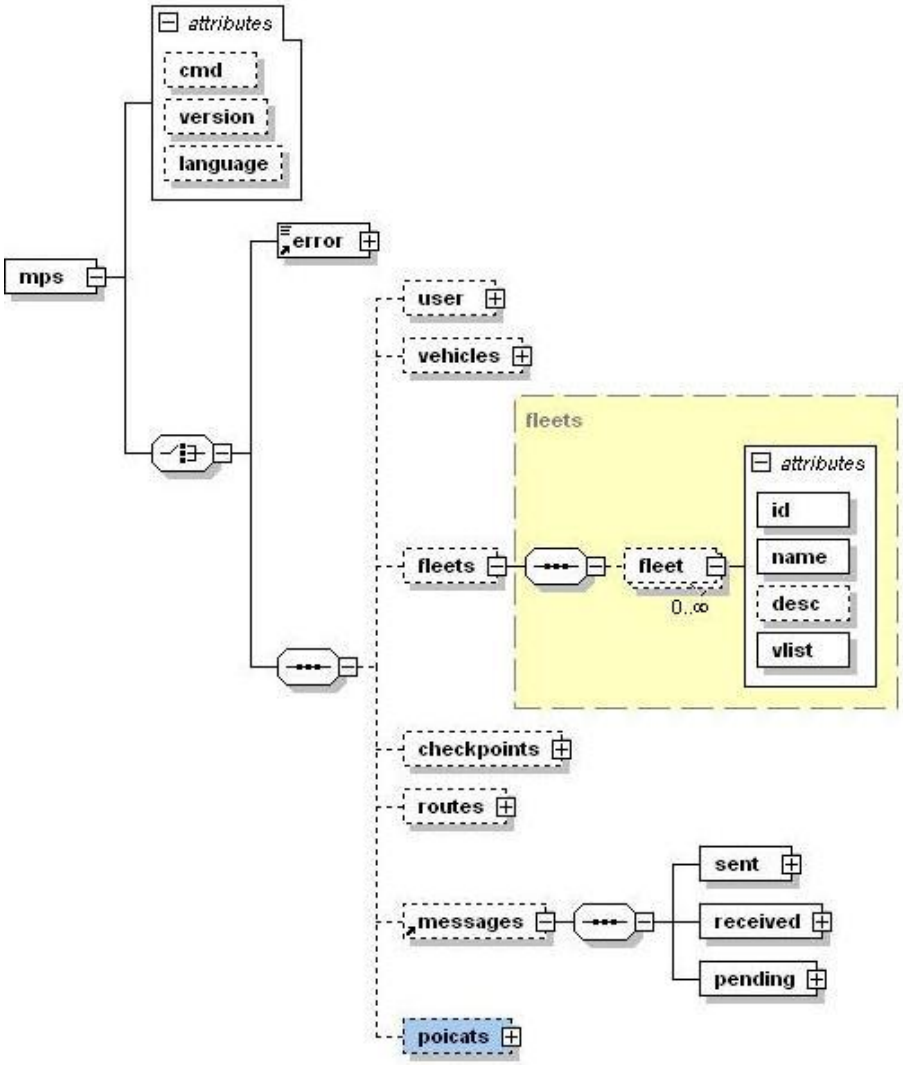
XML Spy ha estat l'eina que s'ha utilitzat per al disseny dels arxius XML a partir dels quals, utilitzant la tecnologia JAXB, s'han obtingut les classes Java per a les descàrregues d'informació a través de Servlets.

6.2 – Disseny de XMLs per als Servlets

En aquest punt presentem els esquemes XML a partir dels quals, amb l'eina JAXB, s'han generat les classes Java que permeten l' intercanvi d'informació a través de Servlets.

6.2.1 – Servlet de Login

Aquí presentem l'esquema xml que s'omple quan el client de centre de control fa una petició d'inici de sessió.



A l'arrel tenim atributs que ens permeten identificar quina versió de servidor estem utilitzant, i també l' idioma. Després, en funció de si la petició de login era correcte o no, ens retornarà un error, o bé un conjunt d'elements que ara detallem.

user: Ens retorna les dades de l'usuari actiu, com el seu identificador o el seu nom.

vehicles: Ens retorna una llista de tots els vehicles sobre els quals té visibilitat aquest usuari, és a dir, els vehicles als quals podrà enviar missatges, serveis o visualitzar en cartografia.

fleets: Són les flotes de vehicles. Permeten organitzar els vehicles en subconjunts (per exemple, un client que treballi en diverses províncies pot organitzar els seus vehicles segons aquest criteri).

checkpoints: Són punts de pas que l'usuari té definits.

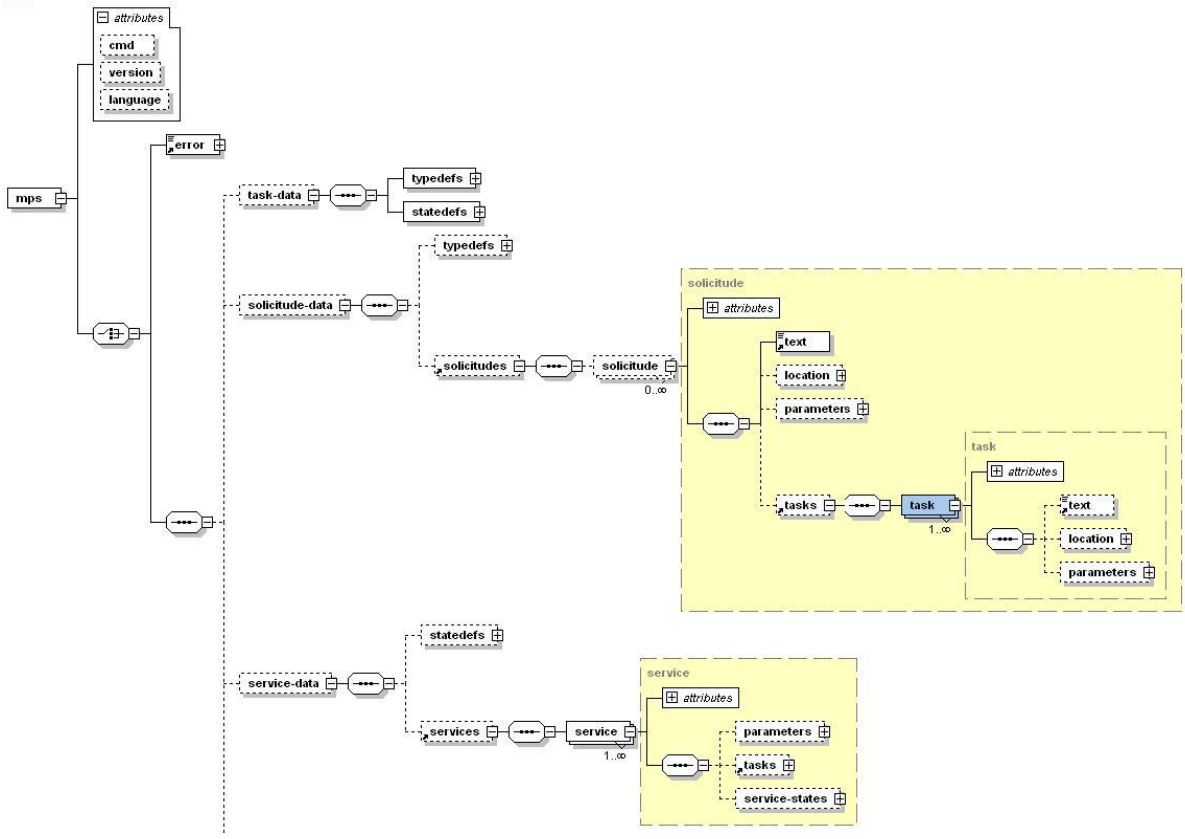
routes: Són itineraris que estan formats pels punts de pas anteriors.

messages: És la bústia de missatges d'aquest usuari. Hi ha els missatges enviats, els rebuts i els pendents d'entregar.

poicats: Defineix un conjunt de punts d'interès.

6.2.2 – Servlet de sol·licituds

Aquest servlet és invocat immediatament després del de login. Retorna totes les dades de gestió de serveis de l'usuari connectat.



Trobem com en el cas anterior, una arrel amb certs atributs, i la possibilitat de retornar un error o les dades en funció de la correcta validació de la crida al servlet.
El servlet ens retornarà la següent informació:

task-data: És la llista de tipus i estats que poden tenir les tasques.

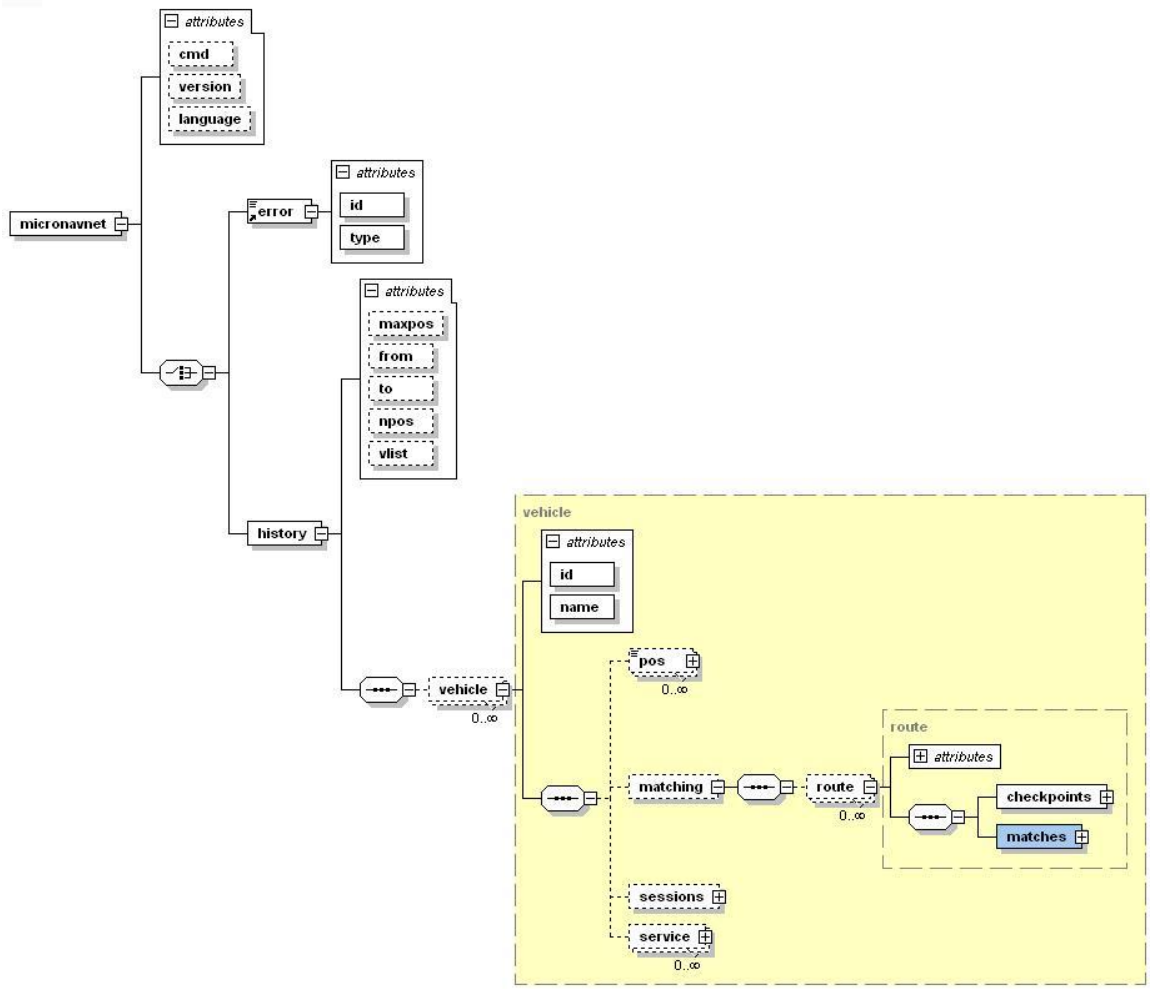
solicitude-data: Aquí hi trobem la llista de tipus de sol·licituds, i també les dades de les sol·licituds en curs de l'usuari actual. Dins les dades de la sol·licitud, també ens venen les dades de les tasques associades a aquesta.

No apareixen els estats de la sol·licitud ja que aquests són predefinits per a tots els usuaris.

service-data: Aquí apareix la llista d'estats que poden tenir els serveis (no tenim tipus ja que el tipus d'un servei és el mateix que el tipus de la sol·licitud a partir del qual s'ha creat). Hi ha la llista de serveis de l'usuari actiu, i també la llista de tasques associades a aquests.

6.2.3 – Servlet històric de posicions

Aquest servlet és invocat amb un conjunt de paràmetres variables que permet configurar l'aplicació client. En general, permet descarregar la llista de posicions d'un o diversos vehicles en un interval de temps.



Com sempre, trobem l'arrel i la possibilitat de retornar un error en cas d'arguments no correctes en la crida. Si tot és correcte, el servlet retornarà la següent informació:

vehicle: L'element de nivell superior a partir del qual s'agrupen les dades. Ens permet saber de quin vehicle són les posicions.

pos: El llistat de posicions GPS que ha generat l'equip.

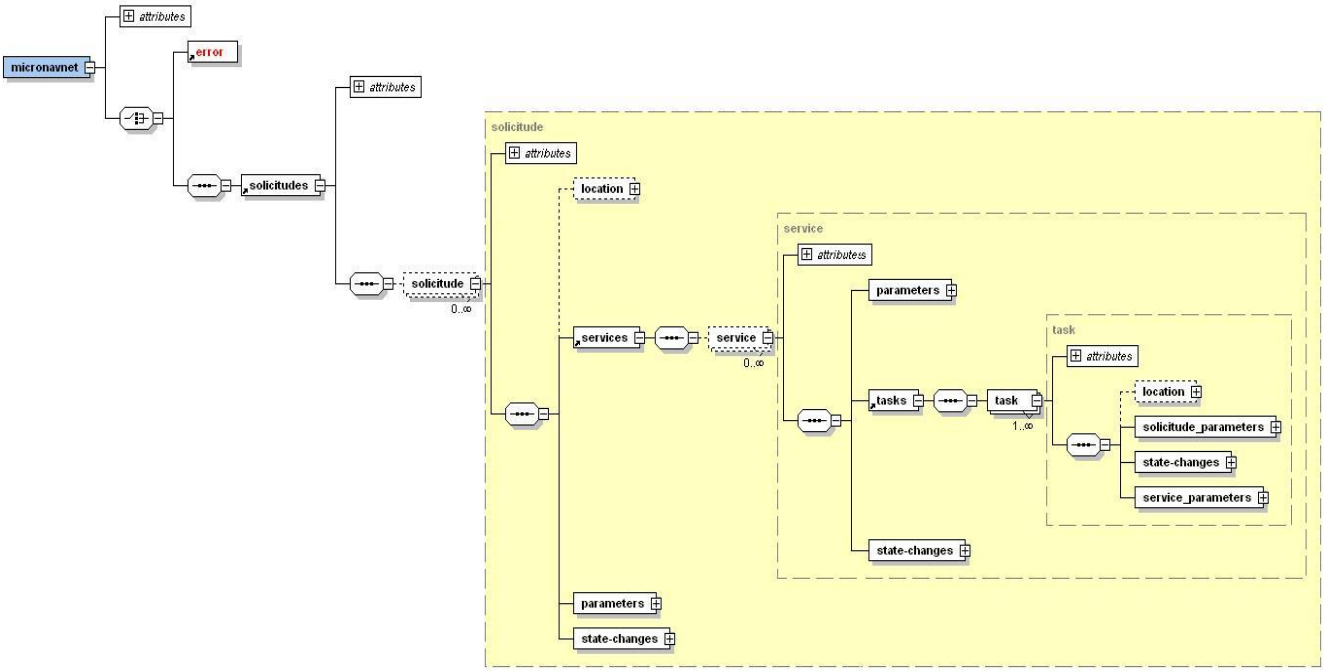
matching: Conjunt de coincidències de posicions i itineraris preestablerts.

sessions: Són les sessions GPRS que ha establert el vehicle durant el període sol·licitat.

service: Permet saber els serveis que tenia actius durant el període.

6.2.4 – Servlet històric de serveis (només Micronav.Net Serveis)

Aquest Servlet no és present a la plataforma de Servicleop, però sí que ho és a Micronav.Net Serveis i per aquest motiu l'esmentem. Aquest servlet permet recollir les dades dels serveis actius d'un o diversos vehicles entre un rang de dates. Hi ha altres opcions de filtrat addicionals, com per exemple l'estat en què es trobava el servei o el tipus de sol·licitud.



Com sempre, trobem l'element arrel seguit de l'opció de retornar un error en cas d'una crida no vàlida. En cas contrari, el servlet retorna els següents elements:

solicitudes: És l'element de nivell superior i conté les dades de la sol·licitud així com els seus canvis d'estat.

services: Dins l'estructura de la sol·licitud trobarem tots els serveis que s'han generat a partir d' ella. Dels serveis trobem els paràmetres i els seus canvis d'estat.

tasks: Dins dels serveis, trobarem totes les tasques que se'ls han assignat. Les dades de les tasques estan formades pels seus paràmetres (tant els particulars del servei, com els comuns de la sol·licitud), i els seus canvis d'estat.

6.3 – Disseny WSDL per als Webservices

En aquest punt presentem els WSDL utilitzats en els serveis web. En els WSDL, podem veure quines crides es poden efectuar, i quins paràmetres reben i quins valors retornen.

6.3.1 – WSRealTime

A continuació apareixen esquematitzades les crides que permet efectuar el servlet de temps real. Algunes crides són versions múltiples de la mateixa crida, per motius d'eficiència i velocitat, i no seran descrites en aquest document.

En els següent esquemes, veurem dues columnes. La primera d'elles, indica el nom de la funció, i portarà la paraula *Request* al final si es tracta de la petició, i *Response* si es tracta de la resposta. Dins la segona columna, veurem dues columnes més. La primera ens indica el nom del paràmetre que es passa o rep de la crida, i la segona ens indicarà el tipus de dades d'aquest paràmetre.

Totes aquestes crides seran explicades a continuació de les següents figures.

	= name	{} wsdl:part
1	doLoginNoLanguageRequest	wsdl:part (2) = name: 1 username; 2 password = type: xsd:string; xsd:string
2	doLoginNoLanguageResponse	wsdl:part (1) = name: 1 doLoginNoLanguageReturn = type: xsd:boolean
3	doLoginRequest	wsdl:part (3) = name: 1 username; 2 password; 3 language = type: xsd:string; xsd:string; xsd:string
4	doLoginResponse	wsdl:part (1) = name: 1 doLoginReturn = type: xsd:boolean
5	doLogoutRequest	
6	doLogoutResponse	
7	doTouchRequest	
8	doTouchResponse	
9	getAllEventsRequest	wsdl:part (1) = name: 1 since = type: xsd:dateTime
10	getAllEventsResponse	wsdl:part (1) = name: 1 getAllEventsReturn = type: tns1:ArrayOfWSBVehicleEvent
11	getAllowedVehicleNamesRequest	
12	getAllowedVehicleNamesResponse	wsdl:part (1) = name: 1 getAllowedVehicleNamesReturn = type: tns2:ArrayOfString

13	getAllReceivedMessagesRequest	***wsdl:*part** (1)			
				= name	= type
			1	since	xsd:dateTime
14	getAllReceivedMessagesResponse	***wsdl:*part** (1)			
				= name	= type
			1	getAllReceivedMessagesReturn	tns1:ArrayOfWSBMessage
15	getAllVehicleSensorEventsRequest	***wsdl:*part** (1)			
				= name	= type
			1	since	xsd:dateTime
16	getAllVehicleSensorEventsResponse	***wsdl:*part** (1)			
				= name	= type
			1	getAllVehicleSensorEventsReturn	tns1:ArrayOfWSBVehicleSensorEvent
17	getCurrentServerDateRequest				
18	getCurrentServerDateResponse	***wsdl:*part** (1)			
				= name	= type
			1	getCurrentServerDateReturn	xsd:dateTime
19	getReceivedMessagesRequest	***wsdl:*part** (2)			
				= name	= type
			1	vehicleName	xsd:string
			2	since	xsd:dateTime
20	getReceivedMessagesResponse	***wsdl:*part** (1)			
				= name	= type
			1	getReceivedMessagesReturn	tns1:ArrayOfWSBMessage
21	getSentMessagesStatesRequest	***wsdl:*part** (2)			
				= name	= type
			1	messageId	tns2:ArrayOfint
			2	vehicleName	tns2:ArrayOfString

22	getSentMessagesStatesResponse	wsdl:part (1)			
				= name	= type
			1	getSentMessagesStatesReturn	tns2:ArrayOfString
23	getSentMessageStateRequest	wsdl:part (2)			
				= name	= type
			1	messageId	xsd:int
			2	vehicleName	xsd:string
24	getSentMessageStateResponse	wsdl:part (1)			
				= name	= type
			1	getSentMessageStateReturn	xsd:string
25	getVehicleEventsRequest	wsdl:part (2)			
				= name	= type
			1	vehicleName	xsd:string
			2	since	xsd:dateTime
26	getVehicleEventsResponse	wsdl:part (1)			
				= name	= type
			1	getVehicleEventsReturn	tns1:ArrayOfWSBVehicleEvent
27	getVehicleLastPositionRequest	wsdl:part (2)			
				= name	= type
			1	vehicleName	xsd:string
			2	getLocation	xsd:boolean
28	getVehicleLastPositionResponse	wsdl:part (1)			
				= name	= type
			1	getVehicleLastPositionReturn	tns1:WSBPosition
29	getVehicleSensorEventsRequest	wsdl:part (2)			
				= name	= type
			1	vehicleName	xsd:string
			2	since	xsd:dateTime
30	getVehicleSensorEventsResponse	wsdl:part (1)			
				= name	= type
			1	getVehicleSensorEventsReturn	tns1:ArrayOfWSBVehicleSensorEvent

31	removeMessagesRequest	wsdl:part (1)		
			= name	= type
		1	messageIds	tns2:ArrayOfint
32	removeMessagesResponse	wsdl:part (1)		
			= name	= type
		1	removeMessagesReturn	tns2:ArrayOfint
33	replyToMessageRequest	wsdl:part (4)		
			= name	= type
		1	replyMessageId	xsd:int
		2	subject	xsd:string
		3	text	xsd:string
		4	timeToLive	xsd:int
34	replyToMessageResponse	wsdl:part (1)		
			= name	= type
		1	replyToMessageReturn	xsd:int
35	sendMessageRequest	wsdl:part (4)		
			= name	= type
		1	vehicleNames	tns2:ArrayOfString
		2	subject	xsd:string
		3	text	xsd:string
		4	timeToLive	xsd:int
36	sendMessageResponse	wsdl:part (1)		
			= name	= type
		1	sendMessageReturn	xsd:int
37	sendMessagesRequest	wsdl:part (4)		
			= name	= type
		1	vehicleNames	tns2:ArrayOfString2D
		2	subject	tns2:ArrayOfString
		3	text	tns2:ArrayOfString
		4	timeToLive	tns2:ArrayOfint
38	sendMessagesResponse	wsdl:part (1)		
			= name	= type
		1	sendMessagesReturn	tns2:ArrayOfint

doLoginNoLanguage: És una crida que donat un nom d'usuari i una paraula de pas, ens retornarà si l'usuari pot iniciar sessió al sistema o no. Aquesta crida és imprescindible efectuar-la abans que qualsevol altre, ja que en fer-ho obtenim una sessió. Si no la fem, qualsevol petició que fem al servei web ens donarà un error.

doLogin: És la mateixa crida, però ens permet passar un paràmetre que indicarà l' idioma amb el qual volem treballar en el sistema.

doLogout: És una crida que ens permet sortir del sistema formalment, així alliberem recursos més aviat que si el sistema ha de detectar que la sessió ha caducat.

doTouch: Aquesta crida permet mantenir viva la sessió. No fa res, però el simple fet d'efectuar-la demostra al servei web que l'usuari encara està connectat.

getAllEvents: Permet, donat un vehicle i una data / hora, obtenir tots els events que han succeït al vehicle en una taula.

getAllowedVehicleNames: Retorna una llista de tots els vehicles als quals l'usuari pot fer peticions.

getAllReceivedMessages: Retorna tots els missatges rebuts posteriors a la data / hora que li passem.

getAllVehicleSensorEvents: Retorna, a partir d'una data, tots els events de sensorització dels vehicles que controlem.

getCurrentServerDate: Retorna l'hora en el servidor, per a poder sincronitzar-nos.

getReceivedMessages: Retorna els missatges que ens ha enviat el vehicle que passem a partir de la data / hora que demanem.

getSentMessageState: Ens retorna, donat un identificador de missatge i un vehicle, quin és el seu estat.

getVehicleEvents: Retorna els events d'un determinat vehicle a partir d'una data / hora.

getVehicleLastPosition: Retorna la última posició del vehicle demanat. Ofereix la possibilitat d'obtenir l'adreça postal de la posició.

getVehicleSensorEvents: Obté els events de sensorització d'un vehicle a partir d'una data / hora.

removeMessage: Permet eliminar un missatge que encara no ha estat entregat a partir del seu identificador.

replyToMessage: Contesta a un missatge de vehicle. És el mateix que enviar un missatge al vehicle, però permet tenir un seguiment a nivell de conversació.

sendMessage: Permet enviar un missatge amb assumpte, text i temps de vida determinats a un cert vehicle. Si no es pot entregar dins del temps de vida, es descarta el missatge.

6.3.2 – WSServices

A continuació apareixen esquematitzades les crides que permet efectuar el servlet de serveis. Algunes crides són versions múltiples de la mateixa crida, per motius d'eficiència i velocitat, i no seran descrites en aquest document. D'altres, com les de sessions, són anàlogues a les del webservice anterior i també seran obviades.

El significat de les columnes és el mateix que en el cas anterior, i les funcions seran descrites després de les següents figures.

	name	wsdl:part
1	addTaskToSolicitudeRequest	wsdl:part (2)
2	addTaskToSolicitudeResponse	wsdl:part (1)
3	cancelSolicitudeRequest	wsdl:part (1)
4	cancelSolicitudeResponse	wsdl:part (1)
5	cancelSolicitudesRequest	wsdl:part (1)
6	cancelSolicitudesResponse	wsdl:part (1)
7	createServiceRequest	wsdl:part (2)
8	createServiceResponse	wsdl:part (1)
9	createServicesRequest	wsdl:part (2)
10	createServicesResponse	wsdl:part (1)

	name	type
1	task	tns1:WSBTask
2	solicitudeId	xsd:int

	name	type
1	addTaskToSolicitudeReturn	tns1:WSBTask

	name	type
1	solicitudeId	xsd:int

	name	type
1	cancelSolicitudeReturn	xsd:boolean

	name	type
1	solicitudeId	tns2:ArrayOfint

	name	type
1	cancelSolicitudesReturn	tns2:ArrayOfboolean

	name	type
1	solicitudeId	xsd:int
2	vehicleName	xsd:string

	name	type
1	createServiceReturn	tns1:WSBService

	name	type
1	solicitudeId	tns2:ArrayOfint
2	vehicleName	tns2:ArrayOfString

	name	type
1	createServicesReturn	tns1:ArrayOfWSBService

11	createSolicitudeRequest	***wsdl:*part** (2)			
				= name	= type
			1	solicitude	tns1:WSBSolicitude
			2	georeferenceLocation	xsd:boolean
12	createSolicitudeResponse	***wsdl:*part** (1)			
				= name	= type
			1	createSolicitudeReturn	tns1:WSBSolicitude
13	createSolicitudesRequest	***wsdl:*part** (2)			
				= name	= type
			1	solicitude	tns1:ArrayOfWSBSolicitude
			2	georeferenceLocation	tns2:ArrayOfboolean
14	createSolicitudesResponse	***wsdl:*part** (1)			
				= name	= type
			1	createSolicitudesReturn	tns1:ArrayOfWSBSolicitude
15	doLoginNoLanguageRequest	***wsdl:*part** (2)			
				= name	= type
			1	username	xsd:string
			2	password	xsd:string
16	doLoginNoLanguageResponse	***wsdl:*part** (1)			
				= name	= type
			1	doLoginNoLanguageReturn	xsd:boolean
17	doLoginRequest	***wsdl:*part** (3)			
				= name	= type
			1	username	xsd:string
			2	password	xsd:string
			3	language	xsd:string
18	doLoginResponse	***wsdl:*part** (1)			
				= name	= type
			1	doLoginReturn	xsd:boolean
19	doLogoutRequest				
20	doLogoutResponse				
21	doTouchRequest				
22	doTouchResponse				

23	endSolicitudeRequest	***wsdl:*part** (1)			
				= name	= type
			1	solicitudeId	xsd:int
24	endSolicitudeResponse				
25	executeClientOperation Request	***wsdl:*part** (2)			
				= name	= type
			1	serviceId	xsd:int
			2	operationId	xsd:string
26	executeClientOperation Response				
27	executeClientOperation sRequest	***wsdl:*part** (2)			
				= name	= type
			1	serviceId	tns2:ArrayOfint
			2	operationId	tns2:ArrayOfString
28	executeClientOperation sResponse				
29	getActiveServicesRequest				
30	getActiveServicesResponse	***wsdl:*part** (1)			
				= name	= type
			1	getActiveServicesReturn	tns2:ArrayOfint
31	getAllowedClientOperationsRequest	***wsdl:*part** (1)			
				= name	= type
			1	serviceId	xsd:int
32	getAllowedClientOperationsResponse	***wsdl:*part** (1)			
				= name	= type
			1	getAllowedClientOperationsReturn	tns1:ArrayOfWSBOperation
33	getAllowedCountryCodesRequest				
34	getAllowedCountryCodesResponse	***wsdl:*part** (1)			
				= name	= type
			1	getAllowedCountryCodesReturn	tns2:ArrayOfString
35	getCrossedKmsRequest	***wsdl:*part** (2)			
				= name	= type
			1	s_vehicleId	xsd:string
			2	when	xsd:dateTime

36	getCrossedKmsResponse	wsdl:part (1)			
				= name	= type
			1	getCrossedKmsReturn	xsd:int
37	getCurrentServerDateRequest				
38	getCurrentServerDateResponse	wsdl:part (1)			
				= name	= type
			1	getCurrentServerDateReturn	xsd:dateTime
39	getNewServicesSinceRequest	wsdl:part (2)			
				= name	= type
			1	since	xsd:dateTime
			2	stillPending	xsd:boolean
40	getNewServicesSinceResponse	wsdl:part (1)			
				= name	= type
			1	getNewServicesSinceReturn	tns2:ArrayOfint
41	getNextPossibleServiceStatesRequest	wsdl:part (1)			
				= name	= type
			1	serviceId	xsd:int
42	getNextPossibleServiceStatesResponse	wsdl:part (1)			
				= name	= type
			1	getNextPossibleServiceStatesReturn	tns1:ArrayOfWSBState
43	getNumberOfActiveServicesOfVehicleRequest	wsdl:part (1)			
				= name	= type
			1	vehicleName	xsd:string
44	getNumberOfActiveServicesOfVehicleResponse	wsdl:part (1)			
				= name	= type
			1	getNumberOfActiveServicesOfVehicleReturn	xsd:int
45	getNumberOfActiveServicesOfVehiclesRequest	wsdl:part (1)			
				= name	= type
			1	vehicleName	tns2:ArrayOfString

46	getNumberOfActiveServicesOfVehiclesResponse	wsdl:part (1)			
				= name	= type
			1	getNumberOfActiveServicesOfVehiclesReturn	tns2:ArrayOfint
47	getServiceParameterChangesRequest	wsdl:part (2)			
				= name	= type
			1	serviceId	xsd:int
			2	from	xsd:dateTime
48	getServiceParameterChangesResponse	wsdl:part (1)			
				= name	= type
			1	getServiceParameterChangesReturn	tns1:ArrayOfWSBParameterChange
49	getServiceParametersRequest	wsdl:part (1)			
				= name	= type
			1	serviceId	xsd:int
50	getServiceParametersResponse	wsdl:part (1)			
				= name	= type
			1	getServiceParametersReturn	tns1:ArrayOfWSBParameter
51	getServiceRequest	wsdl:part (1)			
				= name	= type
			1	serviceId	xsd:int
52	getServiceResponse	wsdl:part (1)			
				= name	= type
			1	getServiceReturn	tns1:WSBService
53	getServicesParameterChangesRequest	wsdl:part (2)			
				= name	= type
			1	serviceId	tns2:ArrayOfint
			2	since	tns2:ArrayOfCalendar
54	getServicesParameterChangesResponse	wsdl:part (1)			
				= name	= type
			1	getServicesParameterChangesReturn	tns1:ArrayOfWSBParameterChange2D
55	getServicesRequest	wsdl:part (1)			
				= name	= type
			1	serviceId	tns2:ArrayOfint

56	getServicesResponse	wsdl:part (1)			
				= name	= type
			1	getServicesReturn	tns1:ArrayOfWSBService
57	getServicesStateChangesRequest	wsdl:part (2)			
				= name	= type
			1	serviceId	tns2:ArrayOfint
			2	since	tns2:ArrayOfCalendar
58	getServicesStateChangesResponse	wsdl:part (1)			
				= name	= type
			1	getServicesStateChangesReturn	tns1:ArrayOfWSBStateChange2D
59	getServiceStateChangesRequest	wsdl:part (2)			
				= name	= type
			1	serviceId	xsd:int
			2	from	xsd:dateTime
60	getServiceStateChangesResponse	wsdl:part (1)			
				= name	= type
			1	getServiceStateChangesReturn	tns1:ArrayOfWSBStateChange
61	getServiceStateRequest	wsdl:part (1)			
				= name	= type
			1	serviceId	xsd:int
62	getServiceStateResponse	wsdl:part (1)			
				= name	= type
			1	getServiceStateReturn	tns1:WSBState
63	getServiceTasksRequest	wsdl:part (1)			
				= name	= type
			1	serviceId	xsd:int
64	getServiceTasksResponse	wsdl:part (1)			
				= name	= type
			1	getServiceTasksReturn	tns1:ArrayOfWSBTask
65	getSolicitudeFromPoolRequest	wsdl:part (1)			
				= name	= type
			1	solicitudeId	xsd:int

66	getSolicitudeFromPoolResponse	wsdl:part (1)			
				= name	= type
			1	getSolicitudeFromPoolReturn	tns2:ArrayOfint
67	getSolicitudeIdsRequest				
68	getSolicitudeIdsResponse	wsdl:part (1)			
				= name	= type
			1	getSolicitudeIdsReturn	tns2:ArrayOfint
69	getSolicitudeLocationRequest	wsdl:part (1)			
				= name	= type
			1	solicitudeId	xsd:int
70	getSolicitudeLocationResponse	wsdl:part (1)			
				= name	= type
			1	getSolicitudeLocationReturn	tns1:WSBLocation
71	getSolicitudeParametersRequest	wsdl:part (1)			
				= name	= type
			1	solicitudeId	xsd:int
72	getSolicitudeParametersResponse	wsdl:part (1)			
				= name	= type
			1	getSolicitudeParametersReturn	tns1:ArrayOfWSBParameter
73	getSolicitudeRequest	wsdl:part (1)			
				= name	= type
			1	solicitudeId	xsd:int
74	getSolicitudeResponse	wsdl:part (1)			
				= name	= type
			1	getSolicitudeReturn	tns1:WSBSolicitude
75	getSolicitudeServicesRequest	wsdl:part (1)			
				= name	= type
			1	solicitudeId	xsd:int
76	getSolicitudeServicesResponse	wsdl:part (1)			
				= name	= type
			1	getSolicitudeServicesReturn	tns2:ArrayOfint

77	getSolicitudeTasksRequest	wsdl:part (1)			
				= name	= type
			1	solicitudeId	xsd:int
78	getSolicitudeTasksResponse	wsdl:part (1)			
				= name	= type
			1	getSolicitudeTasksReturn	tns1:ArrayOfWSBTask
79	getTaskParameterChangesRequest	wsdl:part (3)			
				= name	= type
			1	serviceId	xsd:int
			2	taskId	xsd:int
			3	from	xsd:dateTime
80	getTaskParameterChangesResponse	wsdl:part (1)			
				= name	= type
			1	getTaskParameterChangesReturn	tns1:ArrayOfWSBParameterChange
81	getTaskParametersRequest	wsdl:part (2)			
				= name	= type
			1	serviceId	xsd:int
			2	taskId	xsd:int
82	getTaskParametersResponse	wsdl:part (1)			
				= name	= type
			1	getTaskParametersReturn	tns1:ArrayOfWSBParameter
83	getTaskStateChangesRequest	wsdl:part (3)			
				= name	= type
			1	serviceId	xsd:int
			2	taskId	xsd:int
			3	from	xsd:dateTime
84	getTaskStateChangesResponse	wsdl:part (1)			
				= name	= type
			1	getTaskStateChangesReturn	tns1:ArrayOfWSBStateChange
85	getTaskStateRequest	wsdl:part (2)			
				= name	= type
			1	serviceId	xsd:int
			2	taskId	xsd:int
86	getTaskStateResponse	wsdl:part (1)			
				= name	= type
			1	getTaskStateReturn	tns1:WSBState

87	getTransferableUsersBySolicitudeRequest	wsdl:part (1)			
				= name	= type
			1	solicitudeId	xsd:int
88	getTransferableUsersBySolicitudeResponse	wsdl:part (1)			
				= name	= type
			1	getTransferableUsersBySolicitudeReturn	tns1:ArrayOfWSBUser
89	getTransferableUsersRequest				
90	getTransferableUsersResponse	wsdl:part (1)			
				= name	= type
			1	getTransferableUsersReturn	tns1:ArrayOfWSBUser
91	getVehicleParametersChangesRequest	wsdl:part (2)			
				= name	= type
			1	vehicleName	xsd:string
			2	from	xsd:dateTime
92	getVehicleParametersChangesResponse	wsdl:part (1)			
				= name	= type
			1	getVehicleParametersChangesReturn	tns1:ArrayOfWSBParameterChange
93	getVehicleParametersRequest	wsdl:part (1)			
				= name	= type
			1	vehicleName	xsd:string
94	getVehicleParametersResponse	wsdl:part (1)			
				= name	= type
			1	getVehicleParametersReturn	tns1:ArrayOfWSBParameter
95	getVehiclesParametersChangesRequest	wsdl:part (2)			
				= name	= type
			1	vehicleName	tns2:ArrayOfString
			2	since	tns2:ArrayOfCalendar
96	getVehiclesParametersChangesResponse	wsdl:part (1)			
				= name	= type
			1	getVehiclesParametersChangesReturn	tns1:ArrayOfWSBParameterChange2D
			2	taskId	xsd:int

97	getVehiclesStateChangesRequest	wsdl:part (2)			
			= name	= type	
			1	vehicleName	tns2:ArrayOfString
			2	since	tns2:ArrayOfCalendar
98	getVehiclesStateChangesResponse	wsdl:part (1)			
			= name	= type	
			1	getVehiclesStateChangesReturn	tns1:ArrayOfWSBStateChange2D
99	getVehicleStateChangesRequest	wsdl:part (2)			
			= name	= type	
			1	vehicleName	xsd:string
			2	from	xsd:dateTime
100	getVehicleStateChangesResponse	wsdl:part (1)			
			= name	= type	
			1	getVehicleStateChangesReturn	tns1:ArrayOfWSBStateChange
101	getVehicleStateRequest	wsdl:part (1)			
			= name	= type	
			1	vehicleName	xsd:string
102	getVehicleStateResponse	wsdl:part (1)			
			= name	= type	
			1	getVehicleStateReturn	tns1:WSBState
103	setServiceStateRequest	wsdl:part (2)			
			= name	= type	
			1	serviceId	xsd:int
			2	stateKey	xsd:string
104	setServiceStateResponse				
105	setVehiclesStateAndParametersRequest	wsdl:part (3)			
			= name	= type	
			1	vehicleName	tns2:ArrayOfString
			2	stateKey	tns2:ArrayOfString
			3	parameters	tns1:ArrayOfWSBParameter

106	setVehiclesStateAndParametersResponse	wsdl:part (1)			
				= name	= type
			1	setVehiclesStateAndParametersReturn	tns2:ArrayOfString
107	setVehicleStateAndParametersRequest	wsdl:part (3)			
				= name	= type
			1	vehicleName	xsd:string
			2	stateKey	xsd:string
			3	parameters	tns1:ArrayOfWSBParameter
108	setVehicleStateAndParametersResponse				
109	transferAllSolicitudesToPoolRequest				
110	transferAllSolicitudesToPoolResponse				
111	transferAllSolicitudesToUserRequest	wsdl:part (1)			
				= name	= type
			1	newManagerUser	xsd:int
112	transferAllSolicitudesToUserResponse				
113	transferSolicitudesToPoolRequest	wsdl:part (1)			
				= name	= type
			1	solicitudeId	tns2:ArrayOfint
114	transferSolicitudesToPoolResponse	wsdl:part (1)			
				= name	= type
			1	transferSolicitudesToPolReturn	tns2:ArrayOfint2D
115	transferSolicitudeToPoolRequest	wsdl:part (1)			
				= name	= type
			1	solicitudeId	xsd:int
116	transferSolicitudeToPoolResponse	wsdl:part (1)			
				= name	= type
			1	transferSolicitudeToPoolReturn	tns2:ArrayOfint

117	transferSolicitudeToUserRequest	wsdl:part (2)			
				= name	= type
			1	solicitudeId	xsd:int
			2	newManagerUser	xsd:int
118	transferSolicitudeToUserResponse	wsdl:part (1)			
				= name	= type
			1	transferSolicitudeToUserReturn	tns2:ArrayOfint
119	updateSolicitudeLocationRequest	wsdl:part (3)			
				= name	= type
			1	solicitudeId	xsd:int
			2	location	tns1:WSBLocation
			3	georeferenceIt	xsd:boolean
120	updateSolicitudeLocationResponse	wsdl:part (1)			
				= name	= type
			1	updateSolicitudeLocationReturn	tns1:WSBLocation
121	updateSolicitudeParametersRequest	wsdl:part (2)			
				= name	= type
			1	solicitudeId	xsd:int
			2	parameters	tns1:ArrayOfWSBParameter
122	updateSolicitudeParametersResponse				
123	updateSolicitudePositionRequest	wsdl:part (3)			
				= name	= type
			1	solicitudeId	xsd:int
			2	newLatitude	xsd:double
			3	newLongitude	xsd:double
124	updateSolicitudePositionResponse				
125	updateSolicitudePriorityRequest	wsdl:part (2)			
				= name	= type
			1	solicitudeId	xsd:int
			2	priority	xsd:int
126	updateSolicitudePriorityResponse				

127	updateSolicitudeRequest	wsdl:part (2)			
				= name	= type
			1	solicitude	tns1:WSBSolicitude
			2	georeferencelt	xsd:boolean
128	updateSolicitudeResponse				
129	updateSolicitudesParametersRequest	wsdl:part (2)			
				= name	= type
			1	solicitudeId	tns2:ArrayOfint
			2	parameters	tns1:ArrayOfWSBParameter2D
130	updateSolicitudesParametersResponse				
131	updateSolicitudeTextRequest	wsdl:part (2)			
				= name	= type
			1	solicitudeId	xsd:int
			2	text	xsd:string
132	updateSolicitudeTextResponse				

addTaskToSolicitude: Afegeix la tasca demanada a la sol·licitud que li passem.

cancelSolicitude: Intenta cancel·lar la sol·licitud que passem, així com tots els serveis que depenen d' ella.

createService: Assignar la sol·licitud que li passem al vehicle demanat. Retorna les dades del servei creat.

createSolicitude: Crea la sol·licitud que li passem.

endSolicitude: Finalitza la sol·licitud que li passem.

executeClientOperation: Executa l'operació que haguem definit amb l' identificador que li passem sobre el vehicle que demanem.

getActiveServices: Retorna tots els identificadors de serveis que estan actius per a aquest usuari.

getAllowedClientOperations: Retorna totes les operacions que pot executar l'usuari connectat.

getAllowedCountryCodes: Retorna els codis de país vàlids per al sistema.

getCrossedKms: Retorna els quilòmetres recorreguts pel vehicle demanat des de la data / hora que li passem.

getNewServicesSince: Retorna els identificadors de serveis creats des d'una data / hora demanada.

getNextPossibleServiceStates: Retorna els estats als quals pot passar el servei sol·licitat en funció del seu estat actual.

getNumberOfActiveServicesOfVehicle: Retorna el número de serveis actius pel vehicle que li passem.

getServiceParameterChanges: Retorna els canvis de paràmetres d'un vehicle des d'una data / hora determinada.

getServiceParameters: Retorna els paràmetres del servei sol·licitat.

getService: Retorna la informació del servei amb l' identificador sol·licitat.

getServiceStateChanges: Retorna els canvis d'estat d'un servei des d'una data / hora determinada.

getServiceState: Retorna l'estat en què es troba el servei sol·licitat.

getServiceTasks: Retorna les tasques del servei sol·licitat.

getSolicitudeFromPool: Obté les dades d'una sol·licitud que està al pool.

getSolicitudeIds: Retorna els identificadors dels quals depèn aquesta sol·licitud.

getSolicitudeLocation: Retorna la geolocalització de la sol·licitud demanada.

getSolicitudeParameters: Retorna els paràmetres de la sol·licitud demanada.

getSolicitude: Retorna les dades de la sol·licitud amb l' identificador sol·licitat.

getSolicitudeServices: Retorna els serveis creats a partir de la sol·licitud demanada.

getSolicitudeTasks: Retorna les tasques associades a la sol·licitud demanada.

getTaskParameterChanges: Retorna els canvis de paràmetres de la tasca demanada a partir de la data / hora sol·licitada.

getTaskStateChanges: Retorna els canvis d'estat de la tasca demanada a partir de la data / hora sol·licitada.

getTaskState: Retorna l'estat de la tasca sol·licitada.

getTransferableUsersBySolicitude: Retorna els usuaris als quals es pot transferir la sol·licitud amb l' identificador demanat.

getTransferableUsers: Retorna els usuaris del mateix client connectats en aquell moment.

getVehicleParameterChanges: Retorna els canvis d'estat del vehicle sol·licitat a partir d'una data / hora demanada.
getVehicleParameters: Retorna els paràmetres actuals del vehicle sol·licitat.

getVehicleStateChanges: Retorna els canvis d'estat d'un vehicle determinat a partir d'una data / hora demanada.

getVehicleState: Retorna l'estat actual del vehicle sol·licitat.

setServiceState: Estableix l'estat d'un servei. S'utilitza per a simulacions.

setVehicleStateAndParameters: Estableix l'estat i paràmetres d'un vehicle.

transferAllSolicitudesToPool: Passa totes les sol·licituds de l'usuari actual al pool de sol·licituds.

transferAllSolicitudesToUser: Passa totes les sol·licituds de l'usuari actual a l'usuari demanat.

transferSolicitudeToPool: Traspassa la sol·licitud demanada al pool.

transferSolicitudeToUser: Traspassa la sol·licitud demanada a l'usuari especificat.

updateSolicitudeLocation: Actualitza la localització de la sol·licitud especificada amb la nova localització que li passem.

updateSolicitudeParameters: Modifica els paràmetres de la sol·licitud demanada amb els nous paràmetres.

updateSolicitudePosition: Similar a *updateSolicitudeLocation*, però no crea una nova posició sinó que l'actualitza.

updateSolicitdePriority: Modifica la prioritat de la sol·licitud especificada amb la nova prioritat.

updateSolicitude: Modifica una sol·licitud existent amb les dades de la sol·licitud que li passem.

updateSolicitudeText: Modifica el text de la sol·licitud especificada amb el nou text que li passem.

7 – Implantació

Entenem com a implantació aquella fase posterior a la implementació del sistema i prèvia a la fase de manteniment. És en aquest punt on el client ha de veure reflectit en el producte final el que s'ha acordat amb el proveïdor, i validar les característiques i funcionalitats.

En aquest capítol veurem les diferents fases de la implantació, com la instal·lació, la fase de proves, les certificacions del client fins arribar al punt del pagament del cost acordat.

Després de la posta en marxa s'inicia la fase de manteniment del sistema, i en ocasions, pot reobrir-se per a millorar funcionalitats o definir-ne de noves que el client consideri oportunes.

7.1 – Instal·lació

És la fase on s'instal·len les aplicacions, tant de servidors (en aquest cas els servidors resideixen a les oficines del client), com de software de PDA i client. Prèviament cal estar segurs que tots els requeriments de hardware han estat assolits (validar els ordinadors per a instal·lar els servidors i els clients, les PDAs i el número d'unitats disponibles, etc.).

Les tasques realitzades durant la instal·lació són les següents:

- Servidor
    - Instal·lació de màquina virtual de Java
    - Instal·lació de servidor d'aplicacions JBoss i configuració d'aquest
    - Instal·lació i configuració d'aplicacions servidores (Java i Delphi)
    - Instal·lació de servidor Oracle 9i
    - Creació de l'esquema de base de dades, usuaris, etc. i parametrització

- Clients
    - Instal·lació de màquina virtual de Java
    - Instal·lació i configuració del client
- PDA
    - Instal·lació de l'aplicació PDA automatitzada en targes SD (Secure Digital)

En general aquesta fase dura entre un i dos dies, i acaba amb la demostració al client de la correcta interacció de les aplicacions.

7.2 – Fase de proves

Un cop estan instal·lades les aplicacions, s'inicia la fase de proves. Aquesta fase és molt important tant per al client com per al proveïdor (Knosos), ja que permet detectar si realment la solució ofereix tot el que s'havia pactat, i veure si hi ha hagut alguna interpretació errònia de les especificacions.

Durant aquesta fase es demana al client la possibilitat d'instal·lar algun tipus de software de control remot (Terminal Server, PCAnywhere, etc.) per tal de poder monitoritzar les aplicacions. Això també servirà més endavant, en la fase de manteniment, per a solucionar incidències.

7.3 – Certificació del sistema

Al cap d'un temps prudencial de funcionament correcte del sistema, el client certifica el projecte. És típic entregar durant aquesta fase documents i manuals del sistema.

En aquest cas, la certificació del sistema ja ha estat obtinguda i s'està definint una nova fase.

7.4 – Cursos de formació

Si el client ho requereix, s'ofereixen cursos de formació de les diverses aplicacions, i si el client desitja portar el manteniment de la plataforma, també se li expliquen les aplicacions que formen part del servidor.

En aquest cas, s'ha realitzat formació i entregat documentació sobre les aplicacions de PDA i client de centre de control, però el manteniment de les aplicacions de servidor és tasca de Knosos.

7.5 – Manteniment

La fase de manteniment és la darrera del cicle, i és una de les més importants, sobretot de cara als clients, ja que oferint un bon servei s'aconsegueix una gran satisfacció del client que acaba resultant beneficiosa per a l'empresa perquè aquest tornarà a confiar en l'empresa en un futur i fins i tot podria recomanar-la. Aquesta fase, ja iniciada a Servicleop, ha permès detectar alguns errors que han estat ràpidament resolts. Tots els membres de l'equip de desenvolupament participem activament en aquesta fase.

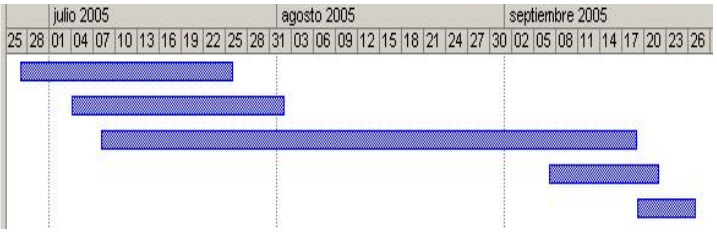
8 – Participació en el PFC

En aquest punt descriuré quina ha estat la meva participació en cadascuna de les fases d'aquest projecte, i podrem veure la durada de cadascuna de les tasques en un diagrama de Gantt. Al final del capítol podrem veure aproximadament quin ha estat el cost econòmic d'aquest PFC.

8.1 – Abast temporal

En el següent diagrama es poden veure resumides les diferents tasques del PFC, i la seva planificació. Cal tenir en compte que la dedicació no ha estat del 100%, sinó que s'estima en un 70%. Malgrat això, gran part del 30% restant va ser dedicació al projecte Micronav.Net Serveis, que evolucionava conjuntament amb el de Servicleop, introduint-se millores des d'un cap a l'altre i viceversa a mesura que es solucionaven errors o es millorava l'eficiència de les funcionalitats existents.

Nombre de tarea	Duración	Comienzo	Fin
Definició de requeriments	21 días	lun 27/06/05	lun 25/07/05
Anàlisi i disseny	21 días	lun 04/07/05	lun 01/08/05
Implementació	51 días	vie 08/07/05	dom 18/09/05
Proves	11 días	mié 07/09/05	mié 21/09/05
Implantació	6 días	lun 19/09/05	lun 26/09/05



Anem a detallar cadascuna de les fases:

- Definició de requeriments: en aquesta fase, hi ha converses amb el client sobre què espera obtenir de l'aplicació, i es van veient com seran les parametritzacions, etc.
- Anàlisi i disseny: en aquesta es veu què podem aprofitar de la plataforma existent, i què haurà de ser redissenyat o ampliat.

- Implementació: la fase més gran d'aquest projecte, sobretot perquè bona part de la feina anterior s'havia fet durant el projecte de Micronav.Net Serveis. Malgrat poder bona part del codi existent, certes exigències per part del client fan necessaris desenvolupaments importants. A més durant el procés es detecten i corregeixen errors existents en la plataforma original, i contínuament es realitzen proves d'integració tant amb PDA com amb el client del centre de control.
- Proves: una vegada acabada la implementació, s'inicia una fase de proves o control de qualitat. Durant aquesta fase es detecta algun error, tant d'implementació com d'interpretació de requeriments, i per tant es segueix desenvolupant.
- Implantació: una vegada es té la certesa que la plataforma ofereix el que s'ha pactat, es procedeix a la instal·lació del programari (client gràfic, PDAs i servidors) a casa del client. També s'ofereixen cursos de formació en les aplicacions client, tant de centre de control com de PDA.

Una vegada acabades les fases esmentades, s'inicia el manteniment de l'aplicació, que consisteix en atendre al client quan se li presenten dubtes funcionals, o resoldre incidències (que sovint impliquen modificacions en la implementació).

8.2 – Abast econòmic

En aquest punt veurem aproximadament quin ha estat el cost econòmic d'aquest PFC (tenint en compte la dedicació estimada del 70% i un cost per hora de 33,66 €, i que en tasques solapades distribuirem la dedicació a parts iguals), desglossat en cadascuna de les fases:

Tasca	**Hores**	**Cost**
Definició de requeriments	((5 dies x 8 hores) + (4 dies x 4 hores) + (12 dies x 2.66 hores)) x 0.7 % dedicació = 61.5 hores	2070 €
Anàlisi i disseny	((4 dies x 4 hores) + (12 dies x 2.66 hores) + (5 dies x 4 hores)) x 0.7 % dedicació = 47.5 hores	1599 €
Implementació	((12 dies x 2.66 hores) + (31 dies x 8 hores) + (8 dies x 4 hores)) x 0.7 % dedicació = 218.3 hores	7348 €
Proves	((8 dies x 4 hores) + (3 dies x 4 hores)) x 0.7 % dedicació = 30.8 hores	1037 €
Implantació	((3 dies x 4 hores) + (3 dies x 8 hores)) x 0.7 % dedicació = 25.2 hores	848 €
Total	383.3 hores	12902 €
Total + IVA (16%)		**14966 €**

9 – Conclusions

En aquest capítol es descriuran les conclusions obtingudes durant la realització d'aquest projecte, l'estat actual d'aquests i les perspectives de futur, així com el que m'ha aportat personalment dins de l'empresa Knosos.

9.1 – Resultats obtinguts

Com ja s'ha comentat en altres apartats, aquest projecte ha estat realitzat i derivat a partir d'altres projectes de Knosos. D'aquests projectes, Micronav.Net té un gran nombre de clients, alguns d'ells molt importants, funcionant durant ja fa temps i amb un gran nivell de satisfacció. Micronav.Net Serveis de moment té un nombre de clients molt més reduït, especialment per la menor antiguitat del projecte, però que a poc a poc van augmentant. A més, alguns clients de Micronav.Net van migrant cap a Micronav.Net Serveis en resultar-los útils les noves característiques que ofereix.

A Servicleop, els resultats que s'han obtingut han complert amb les expectatives, tant a nivell funcional com a nivell de planificació. S'han instal·lat equips de vehicle a diverses grues, i des de centre de control es poden obtenir totes les dades interessants amb la facilitat i versatibilitat esperades. Els canvis sobre la plataforma genèrica han resultat encertats i a nivell d'aplicació han estat aprovats pel client.

9.2 – Perspectives de futur

Actualment Micronav.Net es troba en explotació i amb una gran estabilitat. Es fan millores puntuals en l'aplicació client, i s'està preparant una migració del servidor de temps real cap a un altre producte estàndard de Knosos, conegut com MPS i desenvolupat en C#, que ofereix un disseny més modern que permetrà ampliar més fàcilment les característiques en el futur, l'estabilitat i sobretot, unificar diversos projectes de l'empresa. També s'està intentant migrar el client de centre de control cap a un altre que ofereix un aspecte més professional.

Micronav.Net Serveis es troba també en fase d'explotació i amb una gran estabilitat. Ara per ara, el desenvolupament d'aquesta plataforma està aturat (llevat de casos en què es presenta una incidència deguda a algun error de codificació), i es preveu que un cop finalitzada la migració de Micronav.Net, segueixi els mateixos passos. Quan això arribi, és força probable que jo tingui un paper molt important en la migració, degut a la meva implicació en aquest projecte, tant a nivell de desenvolupament com de manteniment, millores i relació amb els clients als que ofereix servei.

Finalment, Servicleop podria acabar essent migrat també cap a les noves aplicacions, però això ja seria en un futur més llunyà. Actualment s'està desenvolupant i provant un mòdul que permet a la PDA enviar fotografies a un servidor FTP, de manera que es complementa la informació del servei de retirada amb proves visuals. S'està definint també la manera en què el client visualitzarà aquestes fotografies, existint la possibilitat de veure-les des de l'aplicació client de centre de control, des d'un portal web o des d'una aplicació nova. També s'estan definint alguns paràmetres nous que poden resultar d'interès pel client.

En un futur pròxim també està previst d'augmentar el nombre de grues al que dona servei l'aplicació.

9.3 – Resultats a nivell personal

L'experiència de treballar a Knosos en aquest projecte (i en altres) ha estat i és molt enriquidora, a nivell personal i a nivell professional.

A nivell professional, m'he familiaritzat amb el funcionament d'una empresa d'enginyeria, he adquirit una metodologia de treball i també he agafat pràctica en la redacció de documentació. També he après a tractar directament amb el client final.

A nivell tècnic, m'ha permès ampliar els meus coneixements informàtics en moltes àrees, com la programació concurrent (Threads) o les comunicacions (Sockets) en Delphi. He pogut aprendre el funcionament de Webservices i Servlets en Java, així com reforçar coneixements ja adquirits, com l'orientació a objectes o les APIs de Reflection. He pogut millorar molt els meus coneixements en bases de dades, concretament Oracle, així com moltes consideracions tècniques a l'hora de programar. També m'ha estat possible treballar amb noves tecnologies, com les comunicacions GPRS i les aplicacions de PDA, encara que sense aprofundir en aquests temes.

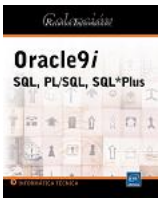
A més de tot això, he pogut conèixer molt software nou, com els servidors d'aplicacions (JBoss), sniffers (EtherDetect), i altres aplicacions que en un moment o altre he necessitat utilitzar, bé per requeriment del projecte o bé per depuració d'aquest.

A nivell personal, ha estat interessant formar part d'un equip de 4 persones, format pel cap de projecte, programador del client, programador de PDA i jo mateix en la programació de servidors i bases de dades, i participar en totes les àrees del projecte, des de prendre decisions a nivell funcional o de desenvolupament, com a fer les proves i la implantació. En general ha regnat el bon ambient en el grup, i tret d'algun moment de crisis, l'experiència ha estat molt satisfactòria.

10 – Bibliografia i planes web

- Programació
  - Delphi
    - La Cara Oculta de Delphi 6, Ian Marteens, Intuitive Sight, 2002
    - http://www.clubdelphi.com

  - Java
    - Java Web Services in a nutshell, Kim Topley, O'Reilly & Associates, 2003
    - JBuilder 8 Developer's Guide, Borland Software Corporation, 2002
    - http://java.sun.com

  - Programació en general
    - http://www.clubdevelopers.com
    - http://www.experts-exchange.com/

- Bases de dades
  - Oracle 9i SQL, PL/SQL, SQL*Plus, Jérôme Gabillaud, Ediciones ENI
  - http://www.lazydba.com
  - http://www.orafaq.com
  - http://www.dba-oracle.com