Universitat Oberta de Catalunya
www.uoc.edu

Estudis d' Informàtica i Multimèdia

Projecte Final de Carrera

Programació d' aplicacions mòbils utilitzant HTML5

(PAC Final 2013)

Alumne: Xavier Martín Bravo

Consultor: Roman Roset

Índex:

Descripció del PFC 7
Situació Actual 7
Solució Proposta 7
Objectius generals 8
Objectius específics 8
Planificació 8
Possibles punts de millora 9
Àmbit del PFC 10
Com sorgeix l' idea 10
Situació actual 11
Solució proposta 12
Esquema general de funcionament 13
Grups d' informació 14
Objectius del projecte 16
Elecció de l' idea 17
Títol del projecte 18
Motius personals 18
Que s' espera d' aquest projecte 18
Tasques lligades a l' idea 19
Get Data Task 19
List Task 20
Locator Task 20
Settings Task 20
Erase Task 21
Metodologia de Treball 21
Scrum vs Kanban 21
Diferències 24
Tasques del taulell 24
Definició de tasques 25
Situació taulell Kanban 25
Entorn de Desenvolupament 26
PhoneGap 26
Android SDK 27
Eclipse Juno 29
Accions a fer per crear un projecte PhoneGap 30

Sencha Touch 31
Estats del procés de creació d' una aplicació mòbil 32
Frameworks i Llibreries 32
HTML + CSS3 + JS 33
Sencha Touch 34
PhoneGap 34
Com funciona PhoneGap 34
Desenvolupament de l' aplicació 36
Objectiu 36
Funcionalitat 36
Descripció mòduls 37
DataTask 37
ListTask 40
MapsTask 41
EraseTask 42
SettingsTask 44
Explicació del codi 45
Mòduls que composen l' aplicació 45
Maps.js 49
CntrlMaps.js 50
MainEraseList.js 52
VehiclesEraseList.js 52
CntrlEraseTask.js 52
FormGetDataTask.js 53
CntrlGetDataTask.js 53
MainList.js 56
VehiclesList.js 56
ShowMapData.js 56
CntrlListTask.js 56
MainSettingsList.js 58
CfgTemplateList.js 58
ShowSettings.js 58
CntrlSettingsTask.js 58
ProGenerics.js 59
Cost del Projecte 60
Conclusions 61
Línies de futur 61

Annex 62
Plug-In BarCode 62
Bibliografia 64

Índex d' il·lustracions:

Figura 1: Imatge que mostra la situació actual 12
Figura 2: Imatge que mostra la solució de manera molt genèrica 13
Figura 3: imatge que mostra els grups d' informació 15
Figura 4: Imatge que mostra el sistema al complert 16
Figura 5: Imatge que mostra l' idea final implementada 17
Figura 6: Imatge que mostra la funcionalitat 21
Figura 7: Il·lustració que mostra les accions en Scrum 22
Figura 8: Il·lustració que mostra un exemple de pissarra Kanban 23
Figura 9: Ilustració que mostra la pàgina oficial de PhoneGap 26
Figura 10: Framework PhoneGap un cop descomprimit..... 27
Figura 11: Il·lustració que mostra la pantalla d' Android SDK 28
Figura 12: Il·lustració que mostra la pantalla del AVD d' Android 28
Figura 13: Il·lustració que mostra la pantalla d' edició de l' AVD d' Android..... 29
Figura 14: Il·lustració que mostra la pantalla d' instalació del Plug-In 30
Figura 15: Il·lustració que mostra els estats pel que passa el desenvolupament 32
Figura 16: Il·lustració que mostra el funcionament de PhoneGap 35
Figura 17: Il·lustració que mostra els estats de l' aplicació 37
Figura 18: Il·lustració que mostra la pantalla inicial de DataTask 38
Figura 19: Il·lustració que mostra la pantalla inicial de ListTask 40
Figura 20: Il·lustració que mostra la pantalla inicial de MapsTask 42
Figura 21: Il·lustració que mostra la pantalla inicial de EraseTask 43
Figura 22: Il·lustració que mostra la pantalla inicial de SettingsTask 44
Figura 23: Il·lustració que mostra el fitxer DataReasons.json 48
Figura 24: Il·lustració que mostra el fitxer DataProducts.json 49
Figura 25: Il·lustració que mostra la relació entre els diferents mòduls 49
Figura 26: Il·lustració d' una funció clousure..... 51
Figura 27: Funció callback utilitzada en l' eliminació d' un fitxer 52
Figura 28: Funció callback per l' accés al sistema de fitxers 54
Figura 29: Funció callback utilitzada per fer una fotografia 54

Figura 30: Funció callback utilitzada per la lectura de BarCodes 55
Figura 31: Funció callback utilitzada pel GPS 55
Figura 32: Recorregut d' elements 55
Figura 33: Push d' una vista 57
Figura 34: Esquema del projecte 63

“A la meva dona i fills per haver-me cedit el seu temps, sense el qual no hagués pogut dur a terme aquest projecte.”

Descripció del PFC

Situació Actual

Hi ha una empresa fabricant de vehicles que té un determinat problema quant dona els seus vehicles per finalitzats, o sigui ja han sortit de la línia de muntatge i per tant ja no estan ubicats seqüencialment un darrera de l' altre. Amb un volum petit de vehicles la gestió d' aquests es pot dur a terme sense masses complicacions, però quan la quantitat augmenta considerablement arribant a tenir de l' ordre de centenars de vehicles esbrinar on es troba ubicat un determinat vehicle es tota una aventura per les persones que els han de buscar. Aquesta situació provoca la pèrdua de temps per part d' alguns operaris realitzant aquesta tasca i per tant un sobrecost que l' empresa s' ha proposat eliminar.
Indicar que aquesta situació es dona molt sovint i no només dins de la mateixa factoria si no inclús a les zones on s' ubiquen els vehicles pendents de ser embarcats o modificats.

Solució proposta

Aprofitar les característiques dels telèfons mòbils (sensors de posició) per localitzar on es troben els vehicles un cop el conductor els ha deixat en una zona determinada pendent de reparar o de realitzar qualsevol altre tasca. Per aconseguir aquesta fita s' hauria de desenvolupar una aplicació que permetés enregistrar cada vehicle en quina ubicació es troba per posteriorment poder-lo localitzar si més no de manera immediata si com a mínim en quina zona es troba.
Per parlar de l' enregistrament de les dades abans hem d' indicar que tots els vehicles duen al parabrises un full enganxat per la part interior on es pot apreciar el número del vehicle. Quines possibilitats ens dona aquest fet:

- Captura d' aquest número mitjançant una fotografia.
- Introduint aquest número via teclat.

Un cop obtingut el número, el següent que hauríem de fer es capturar les coordenades obtingudes pel telèfon mòbil, ja sigui utilitzant el sistema de localització per GPS o bé per xarxa de telefonia. Posteriorment el dispositiu enviarà l' informació a un servidor extern que emmagatzemarà aquestes dades per així poder-les monitoritzar en una pantalla d' un PC.
Els telèfons mòbils també hauran de disposar de la possibilitat de mostrar per pantalla on es troba determinat vehicle respecte de la seva ubicació, d' aquesta manera l' operari podrà saber on a d' anar a buscar els vehicles i si hagués de buscar-ne més d' un poder traçar una ruta abans de començar.

Objectius Generals

L' objectiu principal d' aquest projecte és dissenyar una aplicació que pugui funcionar en dispositius mòbils, especialment 'smartphones' i que permeti localitzar un vehicle en qualsevol punt dins de la fàbrica.

Objectius Específics

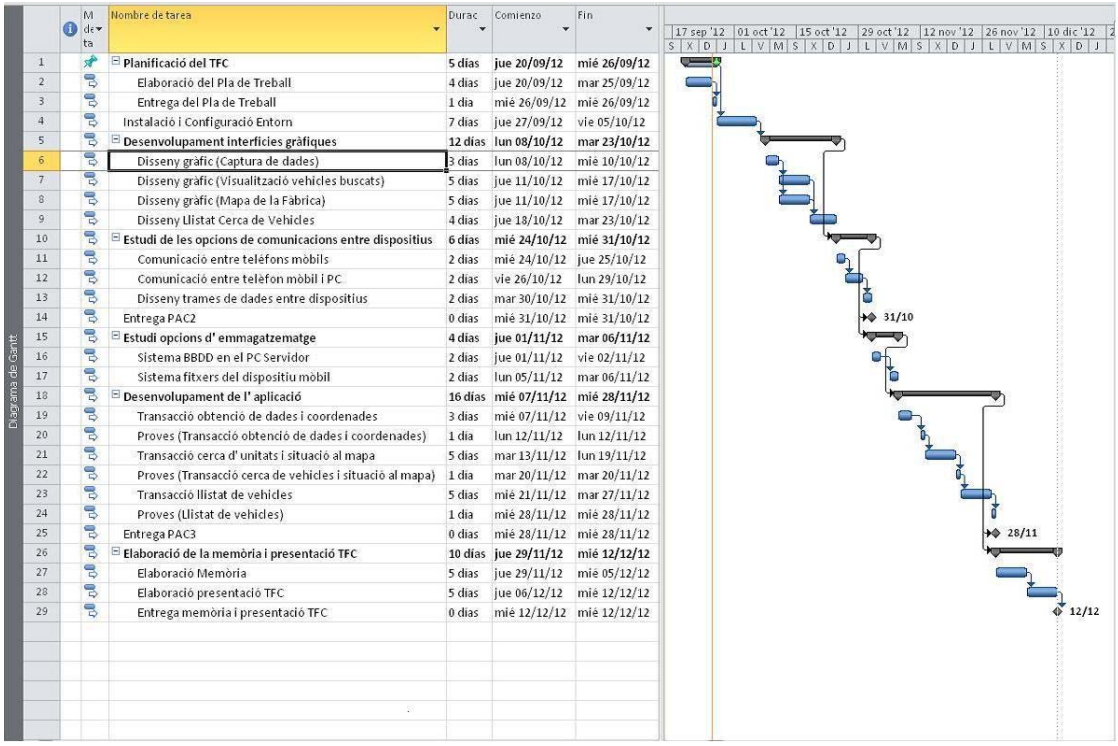
D' aquest tipus en podem trobar:

- Econòmic: menys persones i durant menys temps es dedicaran a la cerca de vehicles i per tant això implicarà un estalvi econòmic molt important.
- Qualitat: qualsevol persona de la companyia podrà saber en tot moment on es troben el vehicles així com el temps que porten parats.
- Màquines/Processos: automatització dels processos i per tant estandardització dels mateixos.
- Persones: augment de l' eficiència de les persones.
- Personal: obtenir un aprenentatge en aquests tipus de tecnologies i poder aplicar-ho a un problema real.

Planificació

El TFC s' ha dividit en 6 grans grups, els quals ara passo a comentar:

- Planificació del TFC: és aquest document en qüestió i és aquí on definirem de manera general el que es pretén fer amb la solució proposada al TFC.
- Desenvolupament interfícies gràfiques: pressa de contacte amb aquest tipus de programació, enfocant aquesta a la definició gràfica dels formularis.
- Estudi de les opcions de comunicacions entre dispositius: crec que és una part important dins del TFC. Com a introducció al problema comentaré que cadascuna de les captures dels diferents dispositius mòbils s' ha de fer saber o bé a altres dispositius o bé a un servidor extern o en ambdues.
- Estudi de les opcions d' emmagatzematge: el problema indicat a l' opció anterior ens porta en aquest, la informació la poden emmagatzemar en el propi dispositiu mòbil, en un servidor extern o en ambdues.
- Desenvolupament de l' aplicació: en aquest grup es portarà a terme tot el desenvolupament de la lògica d' aquest sistema.
- Entrega TFC: fase final on es redactarà tota la feina realitzada i es guardarà en un fitxer que posteriorment s' entregarà.



	M de ta	Nombre de tarea	Durac	Comienzo	Fin
1		**Planificació del TFC**	**5 días**	**jue 20/09/12**	**mié 26/09/12**
2		Elaboració del Pla de Treball	4 días	jue 20/09/12	mar 25/09/12
3		Entrega del Pla de Treball	1 día	mié 26/09/12	mié 26/09/12
4		Instalació i Configuració Entorn	7 días	jue 27/09/12	vie 05/10/12
5		**Desenvolupament interficies gràfiques**	**12 días**	**lun 08/10/12**	**mar 23/10/12**
6		Disseny gràfic (Captura de dades)	3 días	lun 08/10/12	mié 10/10/12
7		Disseny gràfic (Visualització vehicles buscats)	5 días	jue 11/10/12	mié 17/10/12
8		Disseny gràfic (Mapa de la Fàbrica)	5 días	jue 11/10/12	mié 17/10/12
9		Disseny Llistat Cerca de Vehicles	4 días	jue 18/10/12	mar 23/10/12
10		**Estudi de les opcions de comunicacions entre dispositius**	**6 días**	**mié 24/10/12**	**mié 31/10/12**
11		Comunicació entre telèfons mòbils	2 días	mié 24/10/12	jue 25/10/12
12		Comunicació entre telèfon mòbil i PC	2 días	vie 26/10/12	lun 29/10/12
13		Disseny trames de dades entre dispositius	2 días	mar 30/10/12	mié 31/10/12
14		Entrega PAC2	0 días	mié 31/10/12	mié 31/10/12
15		**Estudi opcions d' emmagatzematge**	**4 días**	**jue 01/11/12**	**mar 06/11/12**
16		Sistema BBDD en el PC Servidor	2 días	jue 01/11/12	vie 02/11/12
17		Sistema fitxers del dispositiu mòbil	2 días	lun 05/11/12	mar 06/11/12
18		**Desenvolupament de l' aplicació**	**16 días**	**mié 07/11/12**	**mié 28/11/12**
19		Transacció obtenció de dades i coordenades	3 días	mié 07/11/12	vie 09/11/12
20		Proves (Transacció obtenció de dades i coordenades)	1 día	lun 12/11/12	lun 12/11/12
21		Transacció cerca d' unitats i situació al mapa	5 días	mar 13/11/12	lun 19/11/12
22		Proves (Transacció cerca de vehicles i situació al mapa)	1 día	mar 20/11/12	mar 20/11/12
23		Transacció llistat de vehicles	5 días	mié 21/11/12	mar 27/11/12
24		Proves (Llistat de vehicles)	1 día	mié 28/11/12	mié 28/11/12
25		Entrega PAC3	0 días	mié 28/11/12	mié 28/11/12
26		**Elaboració de la memòria i presentació TFC**	**10 días**	**jue 29/11/12**	**mié 12/12/12**
27		Elaboració Memòria	5 días	jue 29/11/12	mié 05/12/12
28		Elaboració presentació TFC	5 días	jue 06/12/12	mié 12/12/12
29		Entrega memòria i presentació TFC	0 días	mié 12/12/12	mié 12/12/12

Possibles punts de millora

El punt més dèbil d' aquest sistema es assegurar que l' operari cada cop que mou el vehicle d' un lloc a un altre torni a fer la captura de dades per poder tornar a obtenir les coordenades de la nova ubicació.

Trobar una solució a aquest problema mitjançant un procés automàtic implicaria una inversió econòmica molt important en quant a dispositius físics a part d' haver de fer desenvolupaments de software per assegurar l' enregistrament de forma automàtica entre el número del vehicle i el seu posicionament.

Un cop el personal hagi estat informat del nou procés de treball la quantitat de no enregistraments serà mínima en comparació a la quantitat de vehicles que es mouen diàriament.

Un altre punt crític serà la comunicació WI-FI entre els dispositius mòbils i altres dispositius o bé el servidor extern (aquest fet és un punt obert dins del TFC). Què passa si es produeixen molts talls de comunicació i per tant no es pot establir la connexió un cop el dispositiu mòbil ja ha obtingut la informació del posicionament?

En apartats anteriors he parlat de la possibilitat de poder monitoritzar en la pantalla d' un Pc l' ubicació de tots els vehicles distribuïts al llarg de la fàbrica. Aquest seria un punt de millora que també està fora de l' abast d' aquest TFC però que si el resultat del sistema es satisfactori si que en un futur passaré a desenvolupar.

Ambit del PFC

Com sorgeix l' idea?

Com totes les idees, en general de preguntar-se quin problema tinc e intentar donar una solució factible en termes econòmics i tècnics. Anem a donar una breu explicació de quin és el problema i de quina és la solució que intentarem aconseguir. Indicar en aquest punt que la idea proposada permet afegir funcionalitats que en aquest projecte es comentaran però no es duran a terme.

Si pensem en buscar un element identificat numèricament i de manera consecutiva i ubicat sobre una línia física de la qual no en pot sortir, sembla molt fàcil a l' hora de trobar-ne un que hem de fer. Aniríem a qualsevol punt d' aquesta línia i sabent l' element que estic buscant i mirant l' element que està passant per davant meu sabria si he d' anar endavant o endarrere per trobar-lo. Però que passa quant els elements que vull trobar es troben en un 'núvol' d' elements sense cap tipus d' ordre, com ho faig per poder anar directament a buscar l' element que desitjo?.

A partir d' aquesta pregunta va començar a prendre forma la idea d' aquest projecte amb l' objectiu de donar una 1ª solució en aquest problema.

Moltes són les persones que cada dia s' ho han de manegar per poder treballar de la millor manera possible amb aquesta situació, alguns d' aquests implicats podrien ser encarregats de zona, operaris de taller, caps de secció, etc. En definitiva tot aquell personal que per un motiu o un altre té la necessitat de trobar un element concret. Si bé la solució proposada en aquesta 1ª fase estaria enfocada cap els operaris de taller, si que en futures ampliacions el sistema donarà servei a altres usuaris en general.

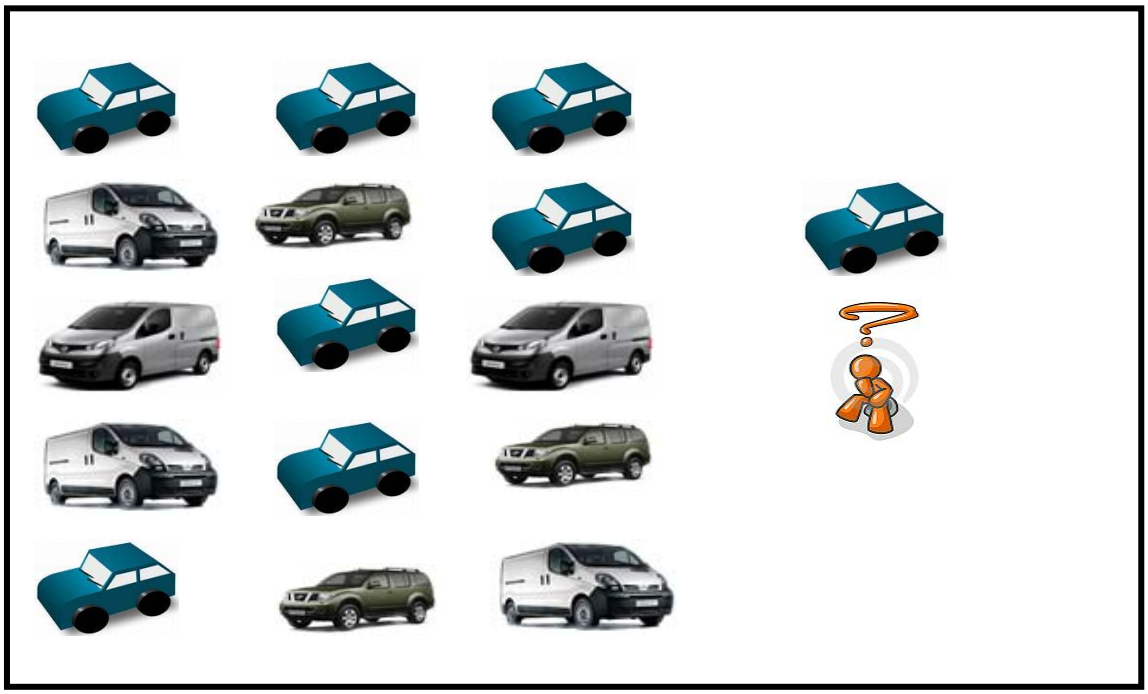
Avui en dia es estrany trobar alguna persona que no disposi d' un dispositiu mòbil i més concretament d' un que pertanyi als del grups dels 'smartphones', però aquest d' entrada no serà un problema degut a que al ser una eina per treballar serà la mateixa empresa qui haurà de subministrar aquesta eina per a poder–ne fer ús. Si ens posem a pensar que ens fa falta tenim que:

- Infraestructura de xarxa: al estar parlant de dispositius mòbils ja donem per descomptat que haurem d' utilitzar la xarxa Wi-fi de l' empresa.
- Dispositius mòbils: l' empresa ja subministra aquest tipus de dispositius a cert personal de l' empresa.
- Aplicació: en aquests moments no existeix cap aplicació que permeti automatitzar el procés abans comentat i per tant es el que es pretén crear amb aquest projecte.

Situació Actual

Un cop hem fet una breu introducció de manera genèrica a la idea que es vol dur a terme passarem a concretar amb més detall quina és la situació actual amb la que es troba aquesta empresa i quina és la situació que volem aconseguir en aquesta 1ª fase.
Fins aquest moment hem estat parlant d' elements, ara ja podem concretar més i dir que aquest són en realitat vehicles que es fabriquen en una factoria d' automòbils.
Quin és el procés a nivell d' informació que es segueix a l' hora de fabricar un vehicle?
Un cop la carrosseria ha estat pintada aquesta és enviada cap a la planta de muntatge on se li muntaran tots els components necessaris per aconseguir passar de l' estat carrosseria a l' estat cotxe llest per ser venut i per tant ser utilitzat per una persona. A cadascuna d' aquestes carrosseries se li assigna un número seqüencial que relacionat amb el número que va gravat a la planxa fa del conjunt un valor irrepetible. Per tant ja podem veure com abans s' ha explicat de manera genèrica que durant aquest procés de fabricació existeix un ordre i per tant la localització de qualsevol vehicle és ràpida i senzilla.
Però que passa una vegada el vehicle no es troba en aquesta seqüència d' elements? La resposta es fàcil de deduir, apareix el desordre. Aquest si es tractés amb quantitats petites de vehicles podria arribar inclús a ser manejable, però quant parlem de volums de l' ordre de centenars de vehicles arribant en alguns casos molt extrems al miler la situació es descontrola i trobar un cert vehicle pot arribar a ser una tasca molt costosa si parlem en termes de temps. Ja ens podem fer una idea de la magnitud del problema que volem resoldre i com utilitzant les noves tecnologies podem arribar a donar en una 1ª fase una solució que pot millorar molt la feina d' aquestes persones.
Criteri seguit per deixar aparcats el vehicles.
Quan un vehicle ha sortit de la línia principal de muntatge aquest passa tot un seguit de proves (potenciòmetre, túnel d' aigua, control lumínic, etc) que poden anar bé o malament. A part també es fa un checkeig del vehicle en general, comprovant l' estat de la pintura, la planxa i els components principals en general. Si qualsevol d' aquestes dues fases dóna un resultat negatiu el vehicle ha de ser desviat i ja no pot seguir el seu curs natural que és ser enviat cap el port.
Aquests vehicles es van aparcant amb uns criteris establerts pels encarregats dels mòduls, però a mida que el volum va augmentant el criteri es va incomplint amb més freqüència fins arribar al desordre i atenir vehicles aparcats en qualsevol lloc.
De l' explicat fins el moment ja podem concloure que a mida que aquest desordre augmenta el temps necessari per localitzar un cert vehicle també augmentarà.
En aquests moments hi han operaris que van fen recorreguts per la fàbrica anotant el número del vehicle y el problema pel qual es troba el vehicle aturat. Aquesta informació es va anotant en una llibreta y aproximadament 1 o 2 hores abans de finalitzar el seu torn van passant tota la informació anotada a un sistema informàtic que permet consultar-la via web.

Figura 1: Il·lustració que mostra la situació actual (“per on començo a buscar”)



Solució proposta

Aprofitar les característiques dels telèfons mòbils (sensors de posició) per localitzar on es troben els vehicles un cop el conductor els ha deixat en una zona determinada pendent de reparar o de realitzar qualsevol altre tasca. Per aconseguir aquesta fita s’ hauria de desenvolupar una aplicació que permetés enregistrar cada vehicle en quina ubicació es troba per posteriorment poder-lo localitzar si més no de manera immediata si com a mínim en quina zona es troba.

Per parlar de l’ enregistrament de les dades abans hem d’ indicar que tots els vehicles duen al parabrises un full enganxat per la part interior on es pot apreciar el número del vehicle. Quines possibilitats ens dona aquest fet:

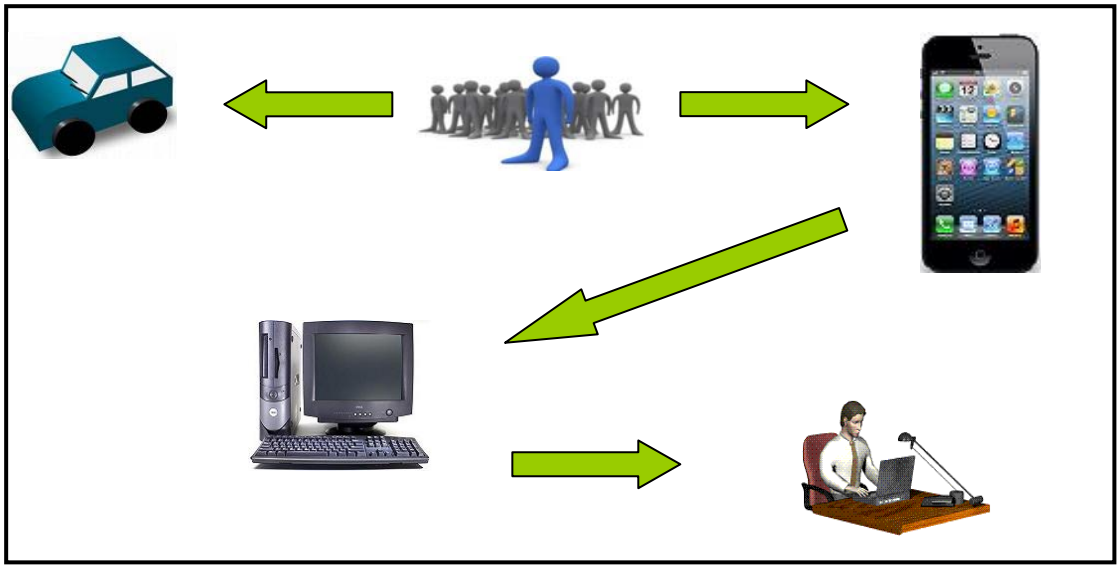
- Captura d’ aquest número mitjançant una fotografia.
- Introduint aquest número via teclat.

En aquests moments m’ estic plantejant diferents alternatives en aquest punt i estic fem I+D per a veure que es el que puc aconseguir, per una banda estic realitzant proves amb l’ opció de realitzar una fotografia del parabrises del vehicle, una altra prova seria llegir el codi de barres e interpretar-lo. En funció del que aconsegueixi hauré de fer introduir per teclat el número de vehicle o no a l’ usuari.

Un cop obtingut el número, el següent que hauríem de fer es capturar les coordenades obtingudes pel telèfon mòbil, ja sigui utilitzant el sistema de localització per GPS o bé per xarxa de telefonia. Posteriorment el dispositiu enviarà l’ informació a un servidor extern que emmagatzemarà aquestes dades per així poder-les monitoritzar en una pantalla d’ un PC.

Els telèfons mòbils també hauran de disposar de la possibilitat de mostrar per pantalla on es troba determinat vehicle respecte de la seva ubicació, d' aquesta manera l' operari podrà saber on a d' anar a buscar els vehicles i si hagués de buscar-ne més d' un poder traçar una ruta abans de començar.
El esquema de funcionament pensat per aquest solució podria ser aquest:

Figura 2: Il·lustració que mostra la solució de manera molt genèrica.



Esquema general de funcionament

El que es pretén amb aquest projecte és aconseguir desenvolupar un sistema de localització de vehicles que permeti tant als treballadors directament implicats en la feina de reparar els vehicles com als usuaris que treballen a oficines poden saber en qualsevol moment on es troba un determinat vehicle.
La tasca central i per tant més important d' aquest sistema és per tant el rigor amb que han de treballar les persones encarregades de dur els vehicles d' un lloc a un altre. El fet de que el sistema no es pot nodrir de manera automàtica de dades fa que aquesta hagi de ser manual i que per tant les persones que ho han de fer siguin molt rigoroses amb la seva feina.
Quan un vehicle hagi de ser aparcat en una zona l' operari enregistrarà una sèrie de dades necessàries pel sistema, com poden ser:

- El número d' identificació del vehicle.
- El motiu pel qual ha estat apartat del seu flux normal de producció.
- L' urgència que pot tenir el vehicle en qüestió.
- Data i hora de captura de les dades.
- Número identificador de l' operari que ha fet l' activitat.
- Etc.....

Un cop l' aplicació hagi validat les dades correctament es procedirà a obtenir les coordenades de geolocalització del vehicle i posteriorment tota aquesta informació es guardarà en el dispositiu mòbil.
En apartats anteriors he parlat de fases d' aquest projecte, concretament de la fase 1 i de la fase 2, això és degut a que l' àrea d' aquest projecte està focalitzada en aplicacions mòbils i no es centra en la part servidora. Aquesta 1ª fase que serà la que es desenvoluparà en aquest TFC no té com ha objectiu desenvolupar la part servidora, encara que per ha fer simulacions s' utilitzaran fitxers amb dades com si haguessin estats enviats pel servidor. No obstant he de recalcar que si un cop acabat el projecte aquest té una bona acceptació començaré a nivell professional a desenvolupar la 2ª fase.
Per què indico això, doncs per què un cop en aquest punt el dispositiu mòbil hauria d' enviar les dades al servidor per poder ser consultades per qualsevol usuari i en un principi això no es desenvoluparà.
Amb aquesta informació emmagatzemada en el dispositiu després l' usuari podrà realitzar cerques de vehicles per qualsevol dels diferents elements introduïts:

- Els vehicles més antics.
- Els vehicles que pertanyin a un model particular més antics.
- Els vehicles més urgents (Flotes de vehicles, campanyes per salons, detecció d' errors, etc.)
- Els vehicles amb un determinat motiu.
- Etc...

Cal destacar que l' element important que marcarà els grups de cerca serà l' antiguitat, o sigui si és marca un motiu diferent de l' antiguitat el grup de vehicles mostrat complirà amb els 2 condicionants, tant l' antiguitat com la del motiu seleccionat.
A partir d' aquest moment l' usuari podrà consultar l' ubicació del vehicle mitjançant les llibreries de Google Maps on es presentarà el plànol d' on es troba ubicada la factoria i un marcador que indicarà la situació aproximada on es pot localitzar el vehicle.

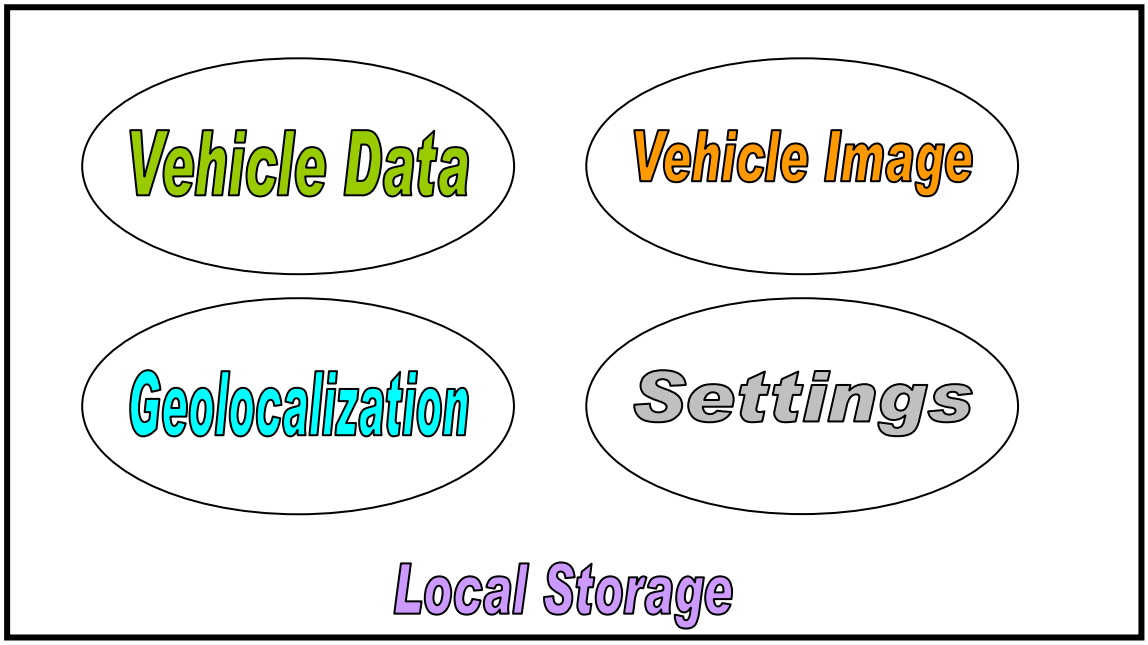
Grups d' informació

Aquest sistema permetrà treballar amb diferents tipus d' informació, en aquests moments es citaran les que s' indiquen a continuació, però és molt possible que a mida que vagi avançant en la feina apareguin de noves. Queden naturalment les portes obertes per possibles ampliacions o millores de les actuals. Aquests grups d' informació tindran la seva representació gràfica mitjançant botons i per tant cada cop que l' usuari premi un d' ells estarà accedint a un contingut diferent, encara que segurament des de dintre d' un grup podrem accedir a d' altres.

Els grups d' informació són:

- Vehicle Data: en aquest grup hi trobem tota aquella informació en format text relacionada amb el vehicle. Ja podem preveure que aquesta informació haurà de tenir una persistència en el temps i que per tant haurà d' estar emmagatzemada en local dins del dispositiu mòbil.
- Vehicle Image: informació en format fotografia on es pot apreciar tant la fulla de ruta del vehicle com si aquest té algun tipus de problema detectable visualment. De la mateixa manera que abans s' ha comentat aquest grup també haurà de ser emmagatzemat i haurà d' estar relacionat amb el grup d' abans.
- Geolocalization: informació relacionada amb la ubicació física del vehicle i per tant ha de ser obtinguda el més proper possible al vehicle en qüestió. Quan més pròxims estiguem menys error afegirem al càlcul de la posició geogràfica del vehicle. També ha de ser emmagatzemat, a més de estar relacionat amb els 2 grups anteriors. L' informació obtinguda en aquest grup podríem dir que és una tasca molt important i de la qual depèn l' èxit o no del sistema. El fet de tenir dades errònies pot arribar a posicionar vehicles en zones que no són les correctes, fent que els operaris de taller perdin confiança en el sistema ja que aniran a buscar un vehicle físicament que no es troba allà on indica el dispositiu mòbil.
- Settings: en aquest grup es troba la part de configuració dels condicionants de cerca que té com a finalitat que l' usuari modificant alguns criteris sigui capaç d' obtenir diferents cerques sense d' haver de demanar un modificació del codi. També serà necessari que l' informació sigui emmagatzemada en local per poder ser consultada i modificada per l' usuari tant com vulgui.

Figura 3: Il·lustració que mostra els diferents grups d' informació



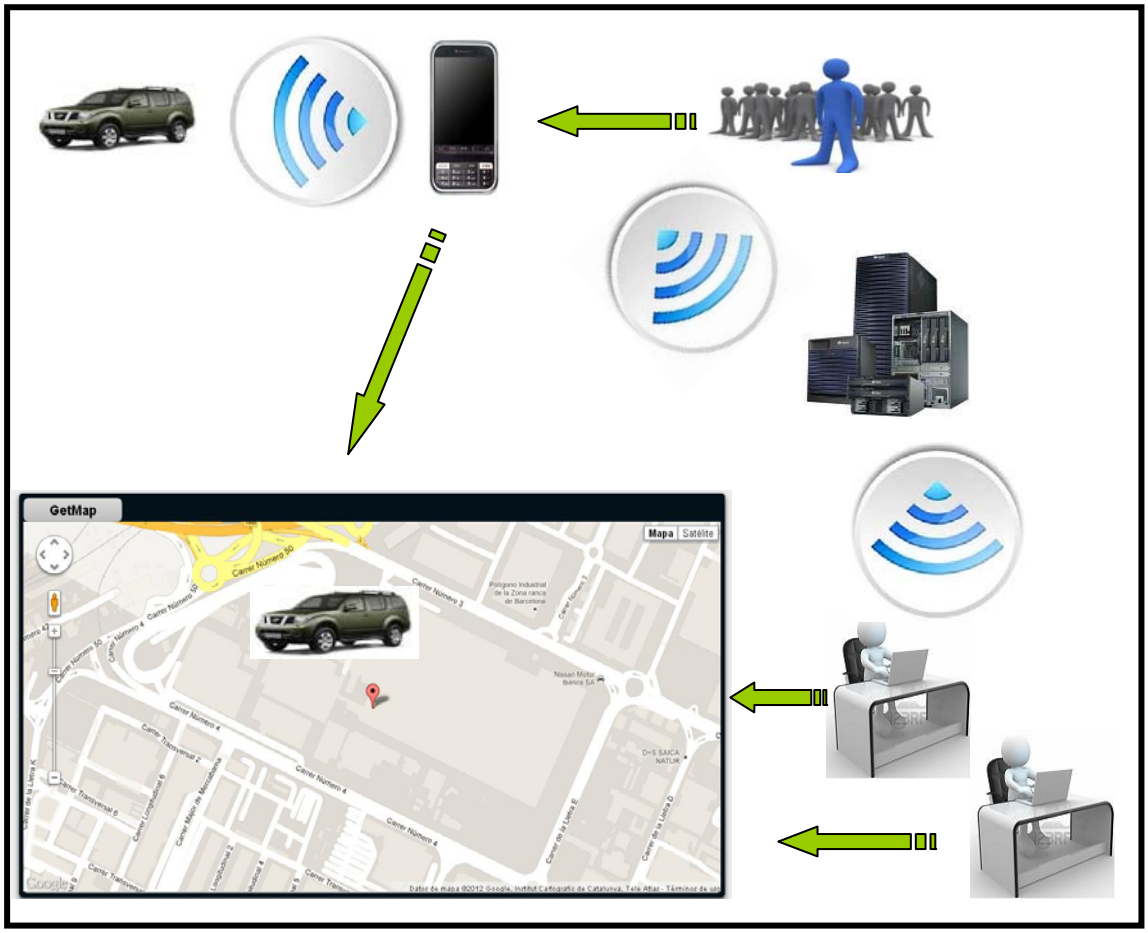
Objectius del projecte

Aquest projecte té com a objectiu principal desenvolupar una aplicació per dispositius mòbils que permeti millorar el treball de cerca de vehicles en una determinada zona. Permetent de manera fàcil i ràpida localitzar-los dins del plànol de la factoria.

Cal destacar no obstant que si aquesta idea agrada a la companyia, no seria descabellat pensar en una ampliació de la mateixa e inclús utilitzar-se en altres plantes del mateix grup.

El que es pretén es aconseguir crear una base el suficient més forta i estable d' on puguin sorgir noves idees i ampliacions del sistema inicial, amb això no estic suggerint de fer una aplicació el suficientment configurable per poder ser utilitzada en qualsevol lloc sense haver de modificar codi.

Figura 4: Il·lustració que mostra el sistema al complert.

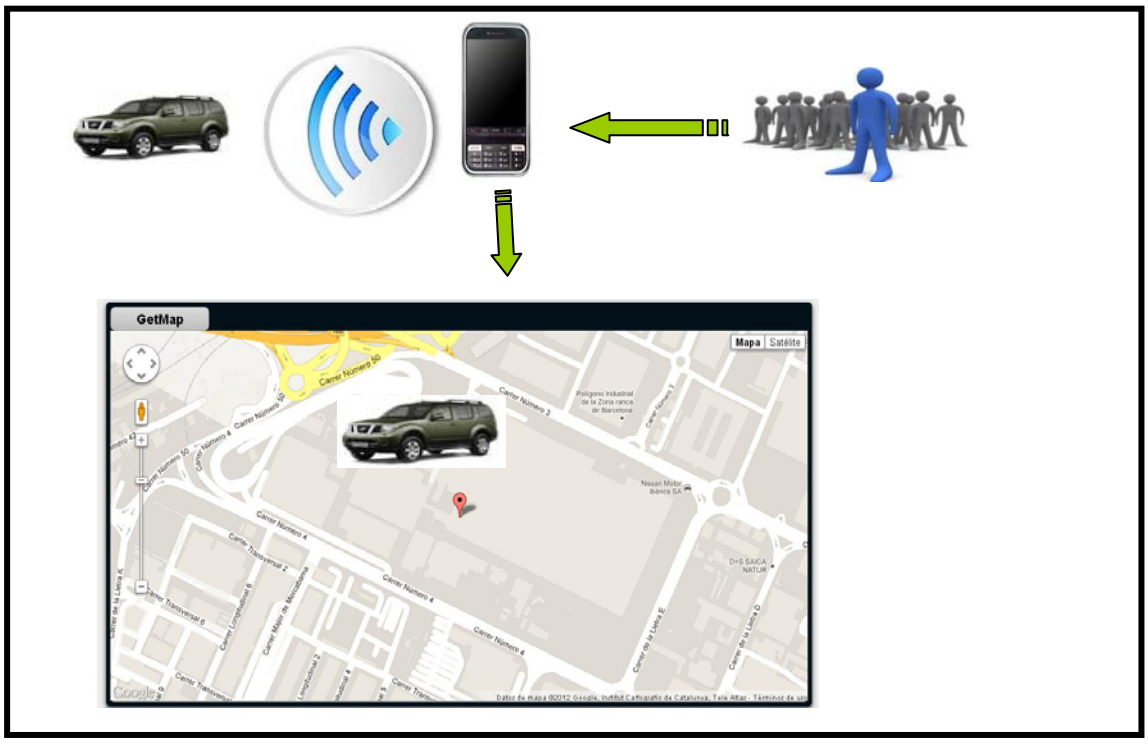


Elecció de l' idea

En l' apartat anterior s' han presentat 2 fases d' aquest sistema, la 1ª que serà la que formarà part d' aquest projecte en qüestió i que per tant es centrarà única i exclusivament en el desenvolupament d' una aplicació per ser utilitzada en un dispositiu mòbil i la 2ª que no formarà part d' aquest projecte que donarà cabuda a tota la part de desenvolupament en la banda del servidor.

Per tant tindrem una aplicació web executant-se dins d' un dispositiu mòbil i que serà utilitzada únicament pels operaris del taller i per tant en un 1ª instància seran els únics que es beneficiaran d' aquest sistema.

Figura 5: Il·lustració que mostra l' idea final a implementar.



I la 2ª opció que seria una aplicació web que s' executi en els dispositius mòbils i que a més enviï informació a un servidor per a ser utilitzada posteriorment pel personal que ho desitgi. L' esquema d' aquesta última ja ha estat presentat anteriorment.

Per tal de centrar-me en l' àmbit establert per la temàtica escollida desenvoluparé la 1ª idea deixant per una 2ª fase la finalització del sistema al complert, per tant aquesta fase no serà presentada en el projecte i per tant tampoc es desenvoluparà.

Títol del projecte

Car Locator System in House (CLSH)

El punt que cal destacar del títol és la seva part final 'in house', o sigui sistema pensat per a ser utilitzat dins de la factoria o inclús en alguna de les zones properes on es deixen els vehicles pendents de rebre petits retocs abans d' embarcar.

Motius personals

Varis i diferents són els motius personals que m' han dut a pensar que l' idea que tenia es podria dur a terme fins el punt d' arribar a ser una aplicació útil per un col·lectiu de treballadors:

- La inquietud generada per trobar-me davant d' una tecnologia nova totalment per a mi.
- El saber que aquesta eina pot tenir una utilitat real.
- El aprendre tecnologies i entorn de treball nous.
- L' estalvi econòmic que aquesta solució pot aportar a l' empresa.
- El fet de saber que si aquest sistema té èxit l' ampliació del mateix està assegurada i per tant podria finalitzar el mateix amb la 2ª fase.

Que s' espera d' aquest projecte

Aquest projecte ha esta pensat i desenvolupat amb l' objectiu clar de crear una aplicació mòbil que pugui millorar i facilitar la feina a un col·lectiu de treballadors. No es una eina que servirà per arreglar millor i més ràpidament un vehicle amb problemes, però si que permetrà localitzar vehicles perduts per la factoria i que poden tenir un client esperant al carrer. Aquest fet fa que es pugui considerar una eina de 2º nivell, o sigui no es clau i determinant per fabricar amb qualitat un vehicle però si útil i necessària per optimitzar la gestió dels recursos. Si ens fitxem en les últimes paraules podem apreciar indicis econòmics darrera d' elles, o sigui sembla plausible dir que hi haurà un estalvi de temps en l' activitat i per tant com a conclusió l' empresa obtindrà un benefici econòmic que pot venir per 2 besants:

- Directe a la companyia:

  L' estalvi de temps i l' eficiència del sistema farà que certs treballadors puguin optimitzar la seva jornada laboral. Aquest fet comporta 2 possibles situacions, o bé als treballadors afectats se'ls hi pot donar més feina o bé es poden eliminar alguns treballadors i ubicar-los en altres departaments. En definitiva un estalvi econòmic per la companyia.

- Indirecte a la companyia:

  Aconseguir reduir el temps per localitzar un vehicle pot significar en moltes ocasions entregar el vehicle abans i això en definitiva ho agrairà el client que està esperant-lo. Finalment tindrem una milloria com a marca, e inclús deixar a un client content pot significar que aquest recomani el producte a un altre persona i que aquesta acabi comprant i per tant l' empresa n' obtingui un benefici.

Com a contrapartida ens podem trobar amb un rebuig per part dels treballadors que l' han d' utilitzar, ja que poden preveure canvis que els puguin afectar. Aquest fet no és nou i generalment sempre que hi ha modificacions de sistemes que s' estan usant o inclús s' afegeixen sistemes nous es produeix un rebuig si no per tot el col·lectiu si per un grup important de treballadors.

Tasques lligades a l' idea

Ara ja hem arribat a un punt on s' ha de començar a donar forma a la nostra idea, indicant quina activitat associada té el fet de posar-la en marxa. D' entrada dividirem aquestes tasques segons la seva funcionalitat i per tant hi trobarem:

- Get Data Task: tasca que s' ha de realitzar quan s' obtenen les dades d' un vehicle. Serà la nostra font de dades i el punt inicial d' on és nodrirà el sistema. El fet de ser l' inici de tot ja ens fa veure que és necessari ser molt rigorosos amb ella per poder a posteriori donar un bon servei.
- List Task: tasca que ens permetrà llistar tots els vehicles que tinguem emmagatzemats en el nostre dispositiu mòbil, mostrant una petita fotografia i un breu comentari que l' identifiqui.
- Locator Task: tasca molt important ja que serà la que donarà confiança a l' usuari final de que el sistema funciona o no. Evidentment està estretament lligada amb la tasca <Get Data Task>, o sigui si falla la 1ª també fallarà la 2ª.
- Settings Task: tasca enfocada a donar cert dinamisme a l 'usuari a l' hora de realitzar la cerca de vehicles, permetent fer simples modificacions en els filtres de cerca per aconseguir diferents resultats en funció de les seves necessitats.
- Erase Task: tasca encarregada d' eliminar tots aquells vehicles que ja han estat entregats i que per tant en una situació típica i normal no haurien de tornar a tenir defectes o errades.

Get Data Task

Aquesta serà l' entrada principal de dades del nostre sistema i haurà de permetre realitzar aquesta acció de manera ràpida i sense que provoqui dubtes a l' usuari. En aquest punt indicarem els seus requeriments per a poder dur a terme amb èxit la tasca:

- L' aplicació ha de poder capturar amb una imatge el parabrises del vehicle.
- Indicar el número de vehicle afectat. Si és possible aquesta introducció s' hauria de fer mitjançant codis de barres. [?????]
- Si el motiu pel qual està el vehicle apartat és pot mostrar amb una imatge que sigui possible realitzar aquesta acció.
- Indicar el motiu d' una llista definida de motius ja preestablerts.
- Enregistrar l' hora en la que s' obtenen les dades.

- Obtenir les coordenades on es troba el vehicle per poder després situar-lo dins del mapa.
- Indicar el producte al qual pertany el vehicle.
- Indicar el model específic del vehicle.
- Indicar el color del vehicle.

He d' indicar que ens aquests moments ja he realitzat proves amb els sensors del dispositiu mòbil, tals coms l' "acelerometer" i el "gps" obtenint resultats positius. Amb l' acció d' obtenir fotografies crec que tampoc tindré problemes i per tant es podrà aconseguir, però amb la lectura del codi de barres encara no ho tinc clar i per tant ho deixo amb un interrogant. No obstant el que s' ha de llegir és un codi numèric de 6 xifres i si no pogués aconseguir fer que el dispositiu mòbil ho llegís llavors seria l' operari de taller qui ho hauria d' introduir a mà mitjançant el teclat.

List Task

Aquesta tasca és la responsable de mostrar en una llista tots els vehicles que es troben emmagatzemats en el dispositiu local. Els requisits que ens demana l' usuari es centren en la funcionalitat d' aquesta llista demanant que es pugui navegar amunt i avall i que al polsar sobre un vehicle aparegui tota la informació de la que disposa el dispositiu mòbil. El que es pretén és disposar d' un llistat ordenat per un camp (segurament serà el número del vehicle la clau principal del registre) on es pugui navegar de manera ràpida i còmoda fins localitzar el vehicle desitjat.

Locator Task

La funcionalitat d' aquest mòdul es centra única i exclusivament en posicionar un o varis vehicles dins del plànol de la factoria. En cap moment es pretén localitzar i posicionar tots els vehicles que es troben en les diferents zones, ja que això faria que el plànol s' omplis de marques i enlloc de ser una ajuda seria un perjudici. L' objectiu és poder ubicar com a màxim un volum de 5 a 10 vehicles a més de la posició d' on es troba l' operari de taller per què així d' una manera ràpida i visual pugui saber on es troben. Com a requisit afegit se'ns demana la possibilitat de que aparegui una petita finestra amb informació un cop es polsi sobre la marca seleccionada.

Settings Task

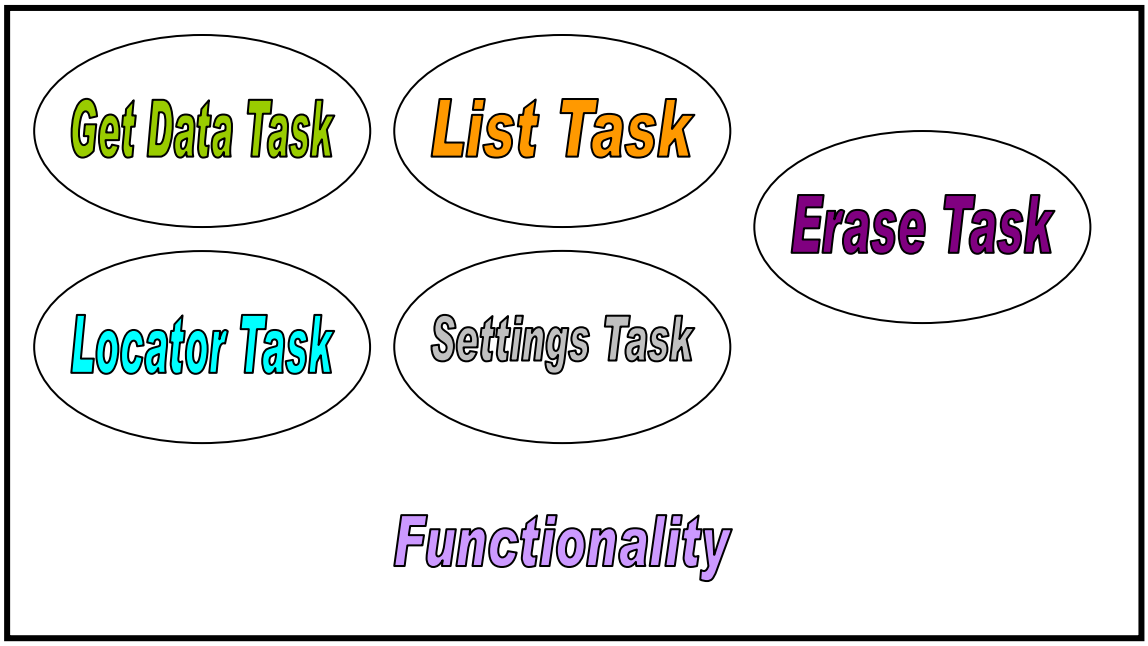
Aquesta és una tasca que té com a objectiu poder canviar els filtres de cerca en temps real de l' aplicació per l' usuari que estigui utilitzant el dispositiu mòbil. Passem-ho a explicar amb un exemple que servirà per a entendre-ho millor; suposem que un criteri de cerca sigui "els 5 vehicles més antics que pertanyin a un producte <X> i que a més siguin del model <Y>", passa el temps hi ens surt una nova necessitat: volem distingir pel "color" i no ens fa falta saber el

model. Amb aquesta tasca l' usuari ha de poder fer aquest petit canvi en el criteri de cerca i poder a partir d' aquest moment utilitzar aquest nou filtre.

Erase Task

Com no disposem d' un servei que ens informi o bé des del mateix dispositiu mòbil preguntar a un servei quins són els vehicles que ja han estat identificats com a "OK" i que ja no fa falta que estiguin emmagatzemats en el nostre "local storage", haurem de fer aquesta acció de manera manual cridant en aquesta tasca. El funcionament serà el següent; un cop l' operari ha entregat el vehicle per a ser reparat aquest marcarà el vehicle com a llest i serà en aquest moment quan s' eliminarà i deixarà d' estar present en el sistema. Si a l' hora de reparar-lo es detecta un altre problema i s' hagués de tornar a desviar s' hauria de tornar a introduir en el sistema com si fos la 1ª vegada.

Figura 6: Il·lustració que mostra les tasques funcionals.



Metodologia de Treball

Scrum vs Kanban

Abans de decidir quina serà l' eina de treball utilitzada donarem unes petites pinzellades de cascuna d' aquestes, farem una petita comparativa entre elles i finalment escollirem la que més ens interessa per realitzar aquest projecte.

Que és SCRUM? Doncs bé Scrum és un procés en el que s' apliquen de manera regular un conjunt de bones pràctiques per treballar en col·laboració en equip i obtenir per tant el millor resultat possible. Es van realitzant entregues parcials i regulars del producte final, això si prioritzades pel benefici que aporten al receptor del projecte. Es per això que Scrum està

especialment pensat e indicat per projectes en entorns complexos, on es necessari obtenir resultats el més aviat possible i on els requeriments son canviants o estan molt pocs definits. També podem dir d' Scrum que s' utilitza per resoldre situacions compromesses entre client i proveïdor de software, algunes d' elles podrien ser p.e.; el client no està rebent allò que necessita, les entregues no es fan en els terminis estipulats, els costos estan sobrepassant allò que s' havia acordat d' entrada o la qualitat del software presentat no està a l' alçada del que s' esperava.

Un projecte que utilitzi aquesta eina s' haurà d' executar en blocs temporals curts i fixes (d' un mes, 2 setmanes, etc). Aquests blocs es coneixen com interaccions i han de donar un resultat complet, o sigui aconseguir sempre poder entregar la feina portada a terme en aquest bloc al client quan aquest ho sol·liciti. Una altre particularitat que hi podem trobar en un projecte que utilitzi aquesta eina és que evitarà en la mesura del possible generar documentació, això no vol dir que s' hagi de documentar sinó que no és una exigència per començar el projecte.

Aquesta metodologia distingeix 2 aspectes a tenir en compte, d' una banda els actors i de l' altre les accions, em centraré només en aquestes últimes. Hi podem trobar de 3 tipus:

- Product Backlog: és el conjunt de tasques, funcionalitats i requeriments a realitzar.
- Sprint Backlog: és un subconjunt del Product Backlog i per tant engloba 1 o més tasques d' aquest grup. Aquest subconjunt de tasques s' ha de dur a terme com ja he indicat abans entre 2 i 4 setmanes. Indicar també que dins de l' Sprint les tasques no poden ser modificades un cop aquestes han començat.
- Daily Scrum Meeting: és una tasca repetitiva que es realitza tots els dies que duri l' Sprint amb l' equip de treball. Es una reunió operativa e informal d' un màxim de 30 minuts.

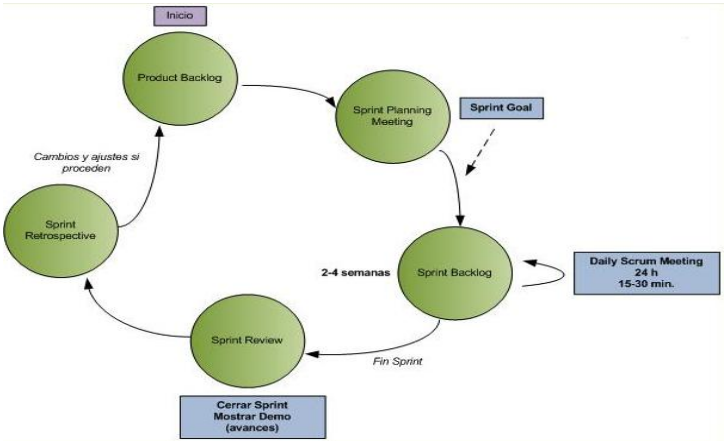


Figura 7: Il·lustració que mostra les accions en Scrum

Que és Kanban? Kanban és una paraula japonesa on hi trobem 2 parts; 'kan' significa 'visual' i 'ban' significa 'targeta' i per tant una possible traducció podria ser 'targeta visual'. Aquesta eina va ser creada per Toyota i s' utilitza per controlar l' avanç del treball dins del context d' una línia de producció. Encara que no va néixer per donar solucions a projectes informàtics, aquests poc

a poc han anat utilitzant-la degut a que el seu gran objectiu es gestionar de manera general com es van completant les tasques. Aquesta eina té 3 regles principals:

- Visualitzar les fases del flux de treball: Kanban està basat en el desenvolupament incremental, dividint el treball en parts més petites. Una aportació d' aquesta metodologia és que utilitza tècniques visuals per veure la situació de cadascuna de les tasques, normalment aquestes anotacions es fan sobre 'post-it' que s' enganxen en un taulell o pissarra; el conegut com a 'taulell Kanban'. Aquesta pissarra té tantes columnes com estats diferents por tenir una determinada tasca. El fet de plasmar la situació de l' evolució del projecte de manera visual permet clarificar el treball a realitzar, saber en que està treballant cada persona en cada moment i tenir una prioritat ben definida de les diferents tasqes.
- Determinar el límit del WIP (work in progress): poden dir que una de les principals idees d' aquesta metodologia es saber el màxim de tasques que es poden realitzar en cada fase o com hem indicat abans estats. Això vol dir que s' han de definir quantes tasques com a molt es poden realitzar a cada fase del cicle de treball, és per tant aquest número de tasques limitat el que rep el nom de 'work in progress'.
- Mesurar el temps en finalitzar una tasca: en tot moment s' ha de tenir controlat el temps que es triga en finalitzar una tasca, en aquest concepte se li diu 'lead time' i vol identificar el temps des de que es fa una petició fins que es fa l' entrega de la mateixa.

Figura 8: Il·lustració que mostra un exemple de pissarra Kanban

Diferències

Un cop hem vist una petita introducció en aquestes 2 eines, passarem a fer una breu comparativa per posteriorment acabar indicant amb quina he decidit treballar.

Kanban	Scrum
Mètrica: Lead Time	Mètrica: Velocitat
No fa cap al·lusió a com s' han de dividir les funcionalitats	Les funcionalitats s' han de poder dividir en parts que puguin ser completades en un 'Sprint'
Permet afegir tasques, sempre que es pugui	No es poden afegir tasques en mig d' una iteració
No existeixen rols de funcionalitat	Com a mínim d' entrada hi ha 3 rols diferents.
Sempre es manté el taulell complert	Cada cop que es finalitza un 'Sprint' es neteja el taulell de seguiment
No s' obliga a que les iteracions tinguin una durada fixada	Les iteracions han de tenir una durada fixa.

Encara hi podríem posar més discrepàncies entre les 2 eines, però he considerat que aquestes eren les més interessants de destacar. Un cop ja han estat situades les 2 metodologies i observats els seus pros i contres he decidit acabar utilitzant el taulell Kanban per les següents raons:

- Ofereix una major flexibilitat a l' hora d' enfrontar-se amb les tasques ha desenvolupar, es possible trobar-se amb un problema que no havia estat previst d' entrada i ens permet afegir-lo encara que estem realitzant altres tasques.
- En el meu cas tenir una visió de totes les tasques que s' han de dur a terme em permet estar situat en tot moment.
- No tenir una dependència de la durada de les iteracions, es difícil a priori poder assignar uns temps a les tasques.

Tasques del taulell

El primer que farem serà establir els diferents estats en els que pot estar una tasca. Aquesta divisió no només ens permetrà situar-nos en quan a la evolució que porta la feina si no que a més ens marcarà un ordre de treball, degut a que cada tasca haurà d' anar passant per cadascuna d' elles en l' ordre preestablert. Els diferents estats són:

- ***Stack***: en aquest estat anirem posant totes les tasques en les que s' ha dividit el projecte, només es marcaran aquelles que tenen a veure amb el desenvolupament del software.

- ***Seleccionat***: de totes les tasques que tenim en cua d' espera ja marquem quines seran les pròximes a treballar.
- ***Desenvolupar***: aquest estat el dividirem en dos sub-estats marcant la diferència entre una tasca en curs i una finalitzada.
- ***Proves***: aquí ens hi trobarem aquelles tasques que s' estan provant.
- ***Producció***: un cop la tasca ja ha passat per tots els estats han un resultat satisfactori ja podem considerar que està llesta per a ser utilitzada en un entorn real.

Definició de tasques

Aquesta seria una petita mostra de les diferents tasques de les que es composa el projecte i com es reflecteixen dins del taulell Kanban, indicar aquí que aquesta divisió està feta amb un zoom bastant ampli i que de cadascuna d' aquestes tasques hi surten més però es suficient per poder fer un petit exemple de l' us de l'eina.

- Disseny interfase gràfic (A)
- Obtenció de les coordinades (B)
- Obtenció d' una fotografia (C)
- Magatzem de dades (D)
- Posicionament al planell (E)
- Aprenentatge de l' entorn (F)
- Lectures bibliografia d' eines a utilitzar (G)
- Accés a sistema de fitxers (H)
- Creació / Modificació / Eliminació de dades (I)
- Canvis en la funcionalitat dels botons (J)
- Adaptació disseny gràfic al dispositiu (K)

Stack (n)	Seleccionat (2)	Desenvolupar (4)		Test (2)	Producció (n)
		En curs	Finalitzat		
E H J	D	F G I	C	K B	A

Un fet destacable en aquest taulell és el nombre ubicat en cadascuna de les columnes, que indica exactament aquest nombre?, doncs bé permet de manera visual saber quantes tasques a l' hora hi poden pertànyer al mateix estat. Si ens fixem en la columna 'Desenvolupar' que està dividida en 2 sub-estats veurem que aquesta està tipificada amb un 4, per tant estem indicant que com a molt entre els 2 sub-estats hi hauran 4 tasques en desenvolupament; si les tasques (F) i (G) estaran quasi durant tota la durada del projecte sols ens queden a representar 2 possibles escenaris:

- 2 tasques en curs i cap d' elles finalitzades: opció poc probable per què haurien de ser tasques que estiguessin molt relacionades per poder estar desenvolupant-les a l' hora.
- 1 tasca en curs i l' altre finalitzada: opció més probable i que s' anirà repetint al llarg del projecte.

Entorn de desenvolupament

Per poder dur a terme el desenvolupament d' aquest projecte des del punt de vista del software ens faran falta diverses eines que a continuació passaré a descriure:

- Android SDK
- PhoneGap
- Sencha Touch
- Eclipse Juno

PhoneGap

Per poder utilitzar aquest framework necessitarem poder-lo descarregar en el nostre ordinador. El lloc on podrem realitzar aquesta tasca és www.phonegap.com

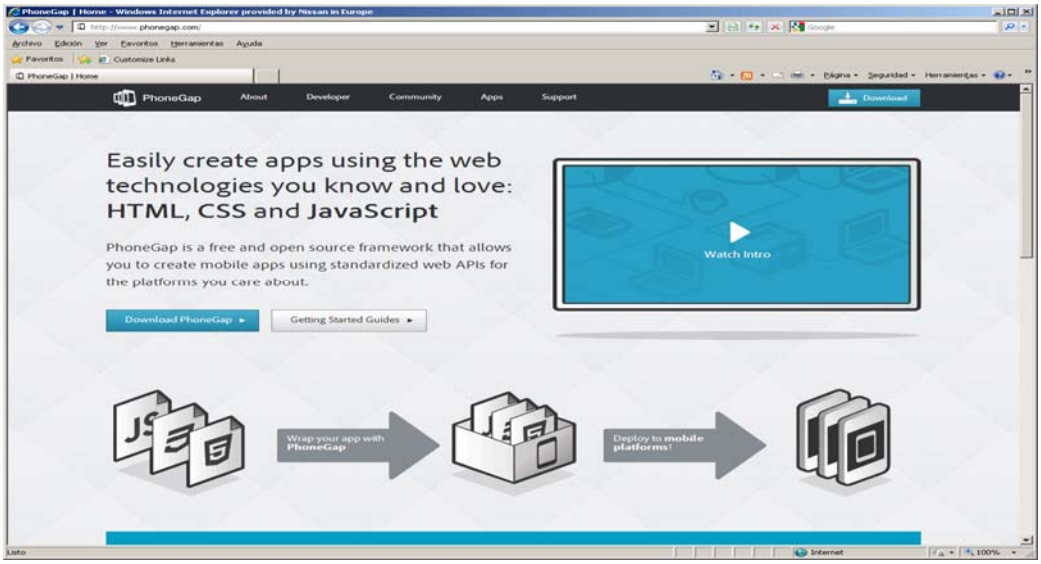


Figura 9: Il·lustració que mostra la pàgina oficial de PhoneGap

Indicar que aquest framework està distribuït en format .Zip i que la seva instal·lació serà tant simple com descomprimir-ho i desar-ho allà on vulguem. Per evitar possible problemes posteriors serà recomanable que s' instal·lin aquest fitxers en una carpeta que pengi directament del 'root' i a més si pot ser que el seu nom no tingui espais en blanc. Un cop feta aquesta feina podrem observar com tenim una carpeta diferent per a cada plataforma mòbil a la que aquest framework dona suport.

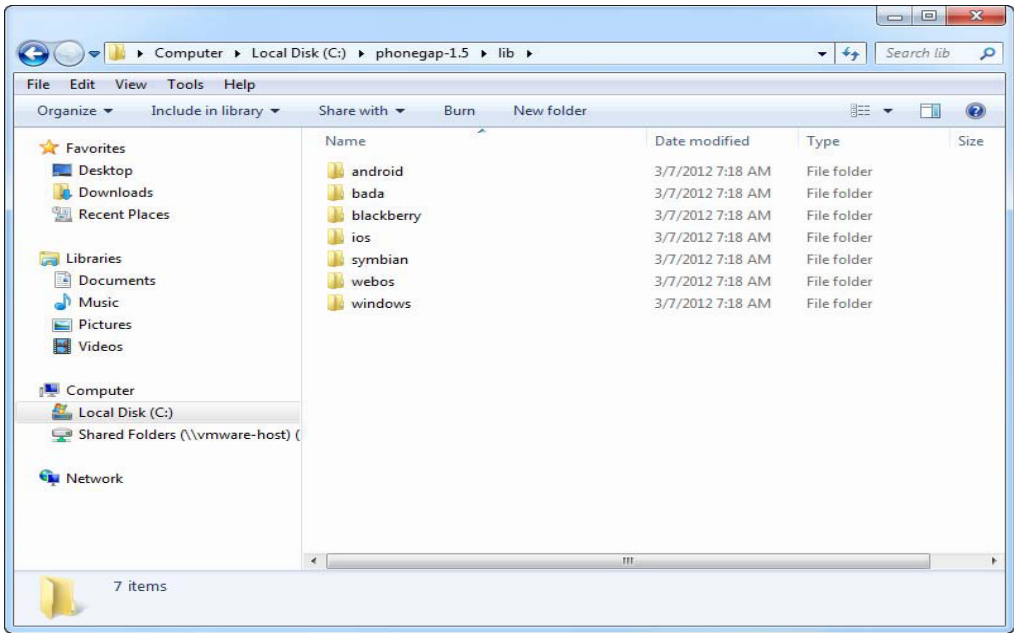


Figura 10: Il·lustració que mostra el framework PhoneGap un cop descomprimit

Per tant en el meu cas com estic desenvolupant una aplicació per dispositius que utilitzen Android com a S.O. utilitzarem la carpeta destinada en aquest.

Android SDK

Ens connectarem a la web http://developer.android.com/sdk/index.html i ens descarregarem el SDK indicat. Un cop hem fet l' instal·lació ja podrem executar el programa SDK.exe que ens mostrarà una finestra similar en aquesta:

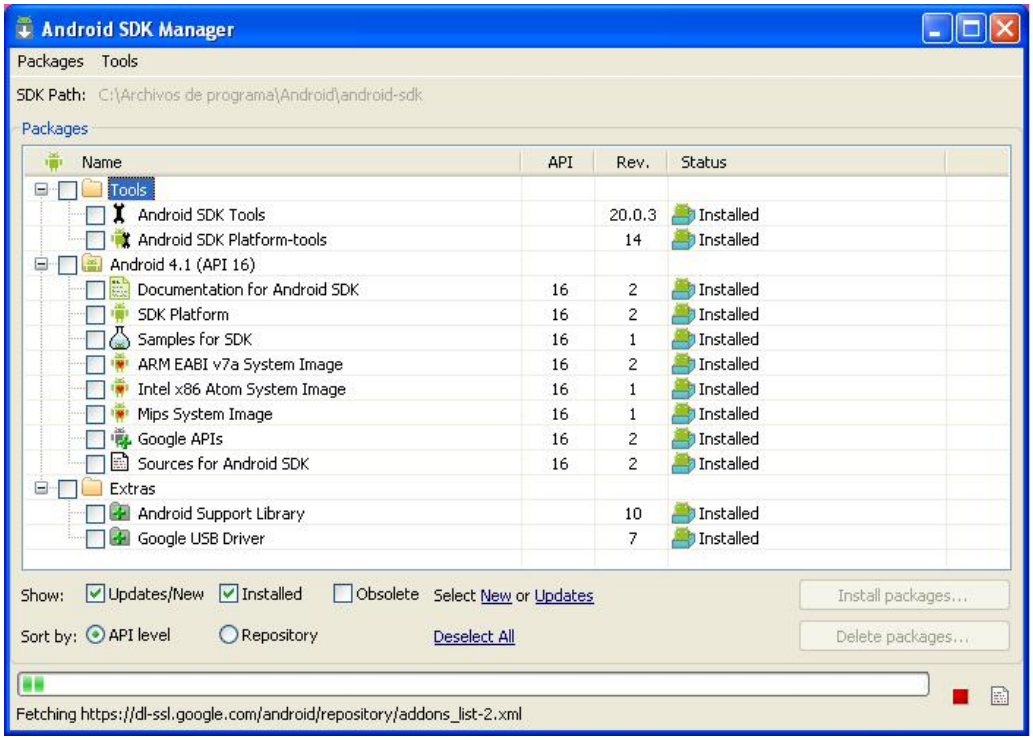


Figura 11: Il·lustració que mostra la pantalla d' Android SDK

Un cop en funcionament aquest intentarà connectar amb la web d' Android per mostrar tots aquells paquets disponibles per a ser descarregats. Hem de tenir en compte que cadascun d' aquests paquets pot fer referència a una versió d' Android diferent.

Ja tenim l' SDK instal·lat a la nostre màquina i ara el que ens caldrà serà aconseguir tenir un dispositiu virtual que ens permeti simular el funcionament de la nostra aplicació en un entorn de PC. Aquesta funcionalitat ens bé definida pel AVD (Android Virtual Device) i serà la que ens permetrà testejar les nostres aplicacions que treballen amb PhoneGap. Un cop haguem definit un dispositiu virtual ens apareixerà una pantalla similar en aquesta:

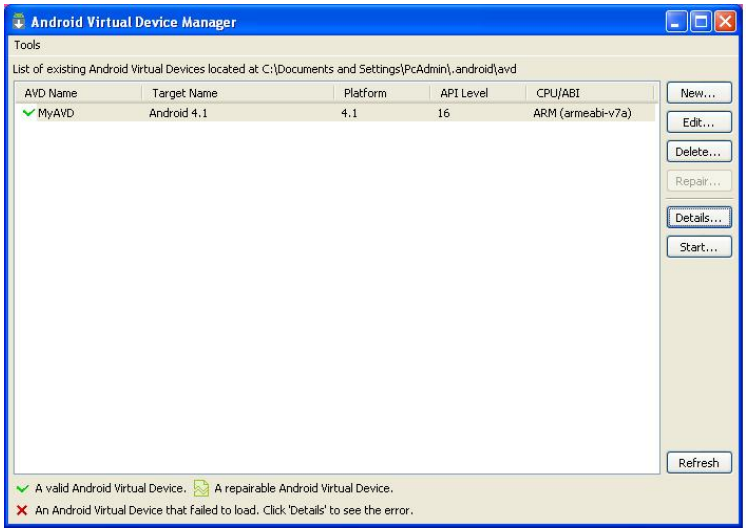


Figura 12: Il·lustració que mostra la pantalla del AVD d' Android

Aquest dispositiu permet ser configurat adaptant-se el més possible a les nostres necessitats. Algunes de les característiques que permet alterar per obtenir un comportament diferent de l' emulador les poden apreciar en la següent imatge:

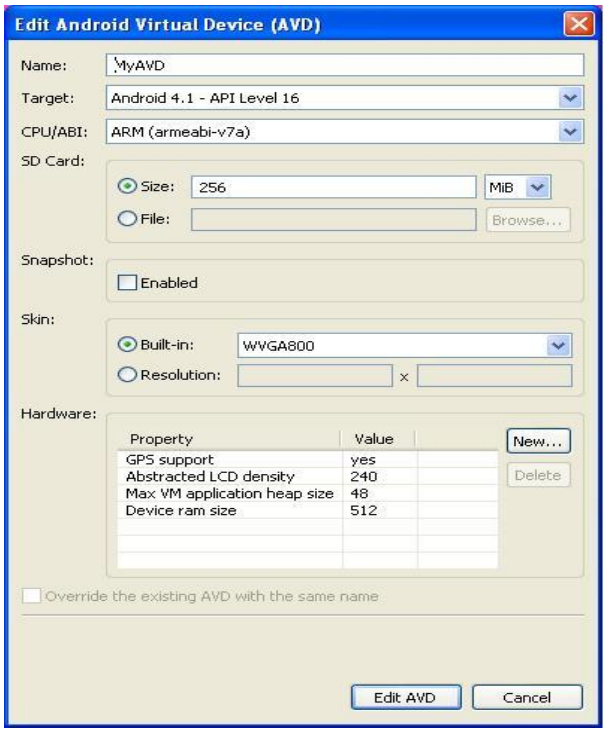


Figura 13: Il·lustració que mostra la pantalla d' edició de l' AVD d' Android

Per sort per a nosaltres Google ha desenvolupat un plug-in per a Eclipse que ens facilita molt el desenvolupament d' aplicacions amb Android. En el següent punt on parlarem d' Eclipse explicarem que hem de fer per poder treballar amb aquest plug-in.

Eclipse Juno

Eclipse es un entorn integrat de desenvolupament de codi lliure i obert utilitzat majoritàriament per programadors en Java i desenvolupadors WEB. Per poder fer ús d' ell ens haurem de connectar a la web www.eclipse.org/downloads i descarregar aquella versió que ens interessi, per aquest projecte en qüestió he escollit Eclipse Juno. Un cop aquí per poder instal·lar el plug-in que ens interessa el que farem serà accedir a l' opció que ens permet fer instal·lacions de software nou dins de l' entorn Eclipse i col·locar la següent adreça https://dl-ssl.google.com/android/eclipse/, aquest fet farà que ens connectem a la web de descarregues d' Android i es descarregui de manera automàtica informació de les diferents opcions disponibles.

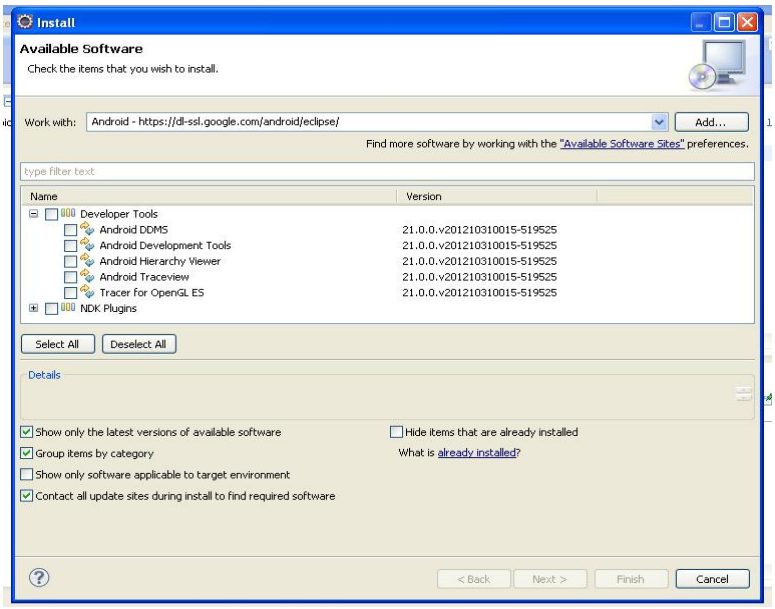
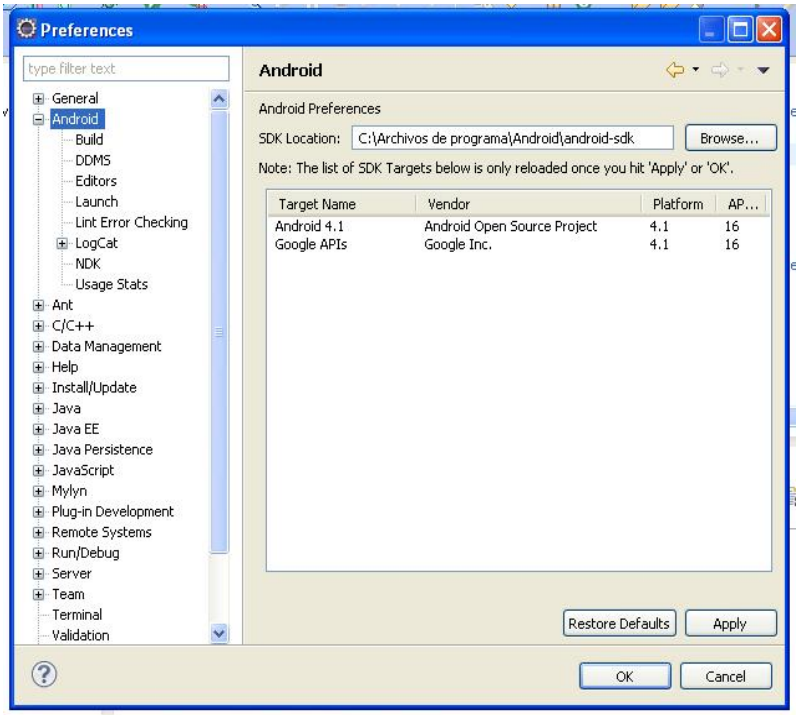


Figura 14: Il·lustració que mostra la pantalla d' instal·lació del Plug-In

La part final d' aquesta instal·lació serà indicar-li a Eclipse quina és la ruta on s' ha instal·lat l' Android SDK. Un cop ja s' ha indicat aquesta ubicació i s' han aplicat els canvis Eclipse actualitzarà la llista que pot tenir de les diferents versions de l' Android SDK.



Accions a seguir per crear i executar un projecte PhoneGap en Eclipse

Ja disposem de tot el software necessari, excepte Sencha (més endavant explicaré que cal fer amb Sencha per poder construir un projecte dins d' Eclipse), de moment ens centrarem únicament en PhoneGap.

Un cop ja hem creat un nou projecte Android i Eclipse en ha mostrat el 'template' que tenia associat amb aquest tipus de software hem de passar a realitzar uns petits retocs per què tot acabi de funcionar de la millor manera possible:

- Crear 2 carpetes noves dins del root del projecte, els seus noms seran 'libs' i 'assets/www'.
- Copiar alguns fitxers de l' instalació de PhoneGap dintre d' aquesta estructura de nova creació, 1º copiarem el fitxer <cordova-2.1.0.js'> dins de la carpeta <assets/www>, a continuació copiarem el fitxer <cordova-2.1.0.jar> dins de la carpeta <libs>, finalment copiarem tota la carpeta <xml> dins de la carpeta <res>
- Modificar alguns fitxers per poder convertir el projecte a un de tipus PhoneGap.
- Modificar el fitxer <AndroidManifest.xml> afegint alguns permisos necessaris que necessitarà la nostra aplicació per poder accedir al telèfon mòbil.
- Finalment crear el fitxer <index.html> dins de la carpeta <assets/www>.

Hem acabat el nostre programa i ara el que volem és poder-lo executar, aquesta és la tasca més fàcil, només haurem d' indicar-li a Eclipse que volem executar el nostre software com si fos una 'Android Application'. Si tots els passos previs han estat realitzats correctament, tant els interns d' Eclipse com els externs de PhoneGap, Eclipse compilarà i llençarà el dispositiu virtual d' Android per què es pugui provar l' aplicació si no s' ha trobat cap error.

Sencha Touch

Ens connectarem a la web www.sencha.com i ens descarregarem el Sencha Touch 2 SDK que ens permetrà poder programar tota la funcionalitat de l' aplicació amb codi JavaScript, així com la SDK Tools que ens permetrà generar les aplicacions mitjançant la línia de comandes.
La carpeta on s' ubicaran els fitxers d' aquesta instal·lació ha de ser accessible des de el web server.
Quan executem les SDK Tools per crear la nostre aplicació se'ns crearà una estructura de directoris on es trobaran tots els fitxers necessaris per què la nostra aplicació funcioni correctament. Indicar no obstant que la magnitud d' aquesta estructura dependrà molt de la llibreria utilitzada de Sencha.
Amb aquesta aclaració feta aprofito per recordar que en el punt anterior quan parlava de la creació i execució d' un projecte PhoneGap no he comentat com intervé Sencha dins de tota aquesta teranyina de tecnologies. Doncs bé un cop hem construït la nostre aplicació Sencha sense errors, utilitzant per aquesta fita les llibreries que ocupin menys pes, l' únic que haurem de fer serà copiar l' estructura de fitxers i directoris creada i ubicar-la dins de la carpeta <assets/www> del projecte definit en l' entorn d' Eclipse, executar el projecte com si fos una aplicació Android i esperar que el dispositiu virtual mòbil aparegui i ens mostri l' aplicació resultant.

Estats del procés de creació d' una aplicació mòbil

Ara que ja s' ha explicat com obtenir e instal·lar les eines necessàries per poder desenvolupar una aplicació mòbil amb tecnologia WEB, és important saber quins són els diferents estats en el que es trobarà el nostre projecte software des de que comença fins que acaba vist des d' un punt de vista del desenvolupador.

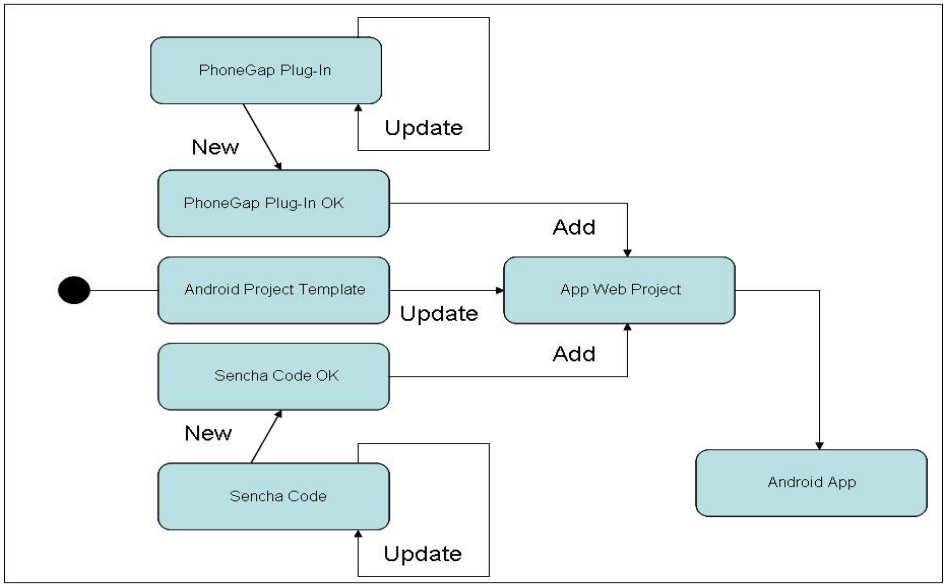


Figura 15: Il·lustració que mostra els estats pel que passa el desenvolupament

El fet de disposar d' un 'template' de projecte Eclipse ens facilita molt les coses com bé es pot apreciar a l' imatge anterior. Ara sols ens hem de preocupar de la programació d' un Plug-In, si fos el cas i fes falta i de la programació amb Sencha. Ambdues tasques es poden fer per separat, validar el seu funcionament i a posteriori afegir-les al projecte Eclipse.
Indicar que per aquest projecte he fet servir un Plug-In que permet fer lectures de codis de barres tant en format 3 de 9 com en format QRCode.

Frameworks i Llibreries

Per a dur a terme una aplicació d' aquest estil com a mínim farà falta treballar i conèixer les següents tecnologies:

- HTML + CSS + JavaScript, o sigui HTML5
- Sencha Touch
- PhoneGap (opcionalment), però en el meu cas és necessari.

HTML + CSS + JavaScript

Al conjunt d' aquestes tecnologies més algunes de servidor és el que coneixem com a HTML5. Això si encara que formin un grup cadascuna d' elles té un rol particular i específic:

Aquí indico algunes de les seves principals característiques:

- Inclou nous tags que descriuen parts d' un document. Inclús s' han inclòs tags especifiquis per elements de navegació, articles, seccions, etc...
- Apareixen nous elements pertanyen a un formulari, ens podem trobar diferents versions de l' element <input>.
- S' inclou suport nadiu al vídeo i àudio.
- Permet construir gràfics interactivament gràcies al nou tag <canvas>.
- Permet l' emmagatzematge de dades (LocalStorage)
- Permet obtenir Geolocalització

Indicar que aquests 3 últims punts són accessibles mitjançant JavaScript.

Es impossible parlar de HTML5 sense parlar de CSS3, aquesta agrupació de tecnologies ha fet que ambdues estiguin estretament relacionades. L' objectiu bàsic que pretén és separar el contingut de la presentació. Algunes de les seves principals característiques poden ser:

- Permet dibuixar fonts en la pantalla encara que aquestes no estiguin instal·lades en el S.O.
- S' han incorporat nous tipus de selector, s' ofereixen més possibilitats a l' hora de seleccionar els <tags>.
- S' han millorat molt els efectes visuals, transparències, ombres, animacions, etc.

Fins ara hem descrit quines són les parts d' un document (HTML5), com s' han de visualitzar (CSS3), ara ens falta definir com volem que actuïn (JavaScript).

Aquest llenguatge de programació ens permet de manera dinàmica modificar elements estàtics definits i creats per les altres tecnologies.
Per simplificar el desenvolupament, a part d' utilitzar al màxim el potencial que ens ofereix aquest llenguatge s' utilitzen llibreries de codi JavaScript que encapsularan diferents funcionalitats d' alt nivell. En el cas d' aquest projecte s' ha utilitzat Sencha Touch que a continuació descriurem breument.

Es un framework que utilitzant les tecnologies HTML5 i JavaScript permet desenvolupar aplicacions web per a dispositius mòbils, independentment del S.O. i per tant suportant tant Objective-C com Android. Aquest fet ens permet desenvolupar aplicacions sense haver d' aprendre diferents llenguatges de programació específics dels dispositius, fent per tant que focalitzem el nostre aprenentatge en tecnologies ja conegudes a priori.
A part també guanyem en l' àmbit d' entrega del producte ja que podrà ser utilitzat el mateix codi en diferents plataformes mòbils. Aquest framework ha estat construït específicament per aprofitar al màxim les tecnologies HTML5, CSS3 i JavaScript intentant donar el màxim de flexibilitat i d' optimització.

PhoneGap és un framework de codi obert que permet construir aplicacions mòbils per a diferents plataformes utilitzant tecnologies Web, com són HTM5, CSS3 i JavaScript, o sigui realitza la mateixa tasca que Sencha i genera aplicacions híbrides (més endavant explicarem la diferència entre aquestes aplicacions i les natives). Però llavors si diem que tots 2 frameworks fan el mateix per què els utilitzem conjuntament? doncs per qué encara que tots 2 tinguin el mateix objectiu final PhoneGap permet una interacció directa amb certes funcionalitats dels dispositius mòbils, com poden ser; l' accelerometer, la càmera, la connexió a xarxa, els contactes, la brújula, el sistema de fitxers, el GPS, etc. Indicar que aquest projecte utilitza alguna d' aquestes funcionalitats que s' explicaran més endavant i per aquest motiu s' ha hagut d' utilitzar aquest framework.

Com funciona PhoneGap

Quan executem una aplicació al dispositiu mòbil aquest el que fa es carregar l' aplicació web (normalment index.html) dins de la vista web de que disposa i passar-li el control a l' usuari per què aquest pugui interactuar amb l' aplicació. Però que és la vista web del dispositiu mòbil? És

un component nadiu del dispositiu que permet dibuixar contingut web dins d' una finestra o pantalla. Aquesta funcionalitat permetrà obrir qualsevol fitxer HTML així com executar codi escrit en JavaScript.
Ja tenim una petita pinzellada de com funciona però encara no podem utilitzar els components del dispositiu que necessitem en el nostre projecte. Aquesta darrere funcionalitat ens vindrà donada mitjançant l' API de PhoneGap que permet a una aplicació web que s' està executant tenir accés als components del dispositiu que es troben fora de la vista web. Un cop es fa una crida a una funció d' aquesta API el que fa PhoneGap és traduir aquesta crida a l' API nativa corresponent del dispositiu.

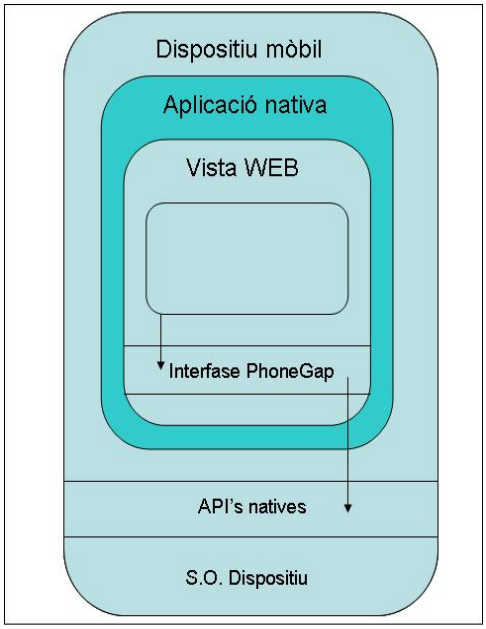


Figura 16: Il·lustració que mostra el funcionament de PhoneGap

Abans he comentat les paraules 'híbrides' i 'natives' fent referència a aplicacions mòbils i ara després d' haver fet una petita introducció a les tecnologies utilitzades, com s' instal·len, que fan i que es pot aconseguir d' elles ja estem en situació de poder explicar breument les diferències que hi ha entre aquest 2 tipus d' aplicacions.

Començarem reflectint les avantatges e inconvenients de cadascuna d' elles:

Aplicacions híbrides	
Avantatges	Inconvenients
Utilitza tecnologies estandard com poden ser (HTML5, CSS3, JavaScript)	El seu funcionament és més lent ja que s' ha d' interpretar.
El mateix codi generat pot ser utilitzat en diferents plataformes	No permet interactuar amb el hardware específic del dispositiu.
Al utilitzar tecnologies obertes és més fàcil trobar programadors que les coneguin en profunditat	

Aplicacions natives	
Avantatges	Inconvenients
Les aplicacions generades s' executen molt més ràpid.	Cada plataforma funciona amb el seu S.O. propi
Permeten explotar al màxim els recursos del dispositiu mòbil en quant al hardware.	Per donar el mateix servei en quant a capacitat de dispositius s' ha de tenir molt més coneixement.

La taula dalt mostrada ens permet veure clarament on ens movem amb cadascun del tipus d' aplicacions. Totes 2 tenen característiques que són interessants i particularitats que no les fan tant atractives, per tant quedarà en mans del desenvolupador decidir quina opció pren, prioritzant aquella característica a la que ell li dona una major importància o pes específic.
En aquest projecte en qüestió m' he decantat per les aplicacions híbrides pel següent motiu; el fet de tenir com a usuari final una empresa multinacional fa que no sàpigues a priori qui serà el proveïdor de dispositius mòbils i per tant un any poden oferir com a eina dispositius amb Android i un altre amb RIM de BlackBerry o inclús amb Objective-C (poc probable), a part de la curiositat e inquietud per conèixer el seu funcionament.

Desenvolupament de l' aplicació

Objectiu

Eina que permeti al treballador que té com a missió localitzar els vehicles distribuïts per la fàbrica realitzar aquesta de manera automàtica.

Funcionalitat

L' aplicació C.L.S.H. es pot descompondre en 5 grups diferents en quant a la seva funcionalitat:

- DataTasK: és la tasca inicial, és aquí on l' usuari obtindrà les dades necessàries pel sistema.
- MapsTaks: permet ubicar un grup de vehicles dins d' un planell. Quins seran els posicionats i la quantitat vindrà definida pel la configuració dels botons.

- ListTask: mostra la llista de vehicles que tenim emmagatzemada localment en el dispositiu mòbil. Indicar que aquesta tasca és l' inici de l' acció modificar les dades d' un vehicle.
- EraseTask: tasca que marca el final de l' existència de les dades d' un vehicle en el magatzem local del dispositiu.
- SettingsTask: permet donar-li dinamisme a la cerca de vehicles, ja que dona la possibilitat a l' usuari de modificar comportaments de cerca.

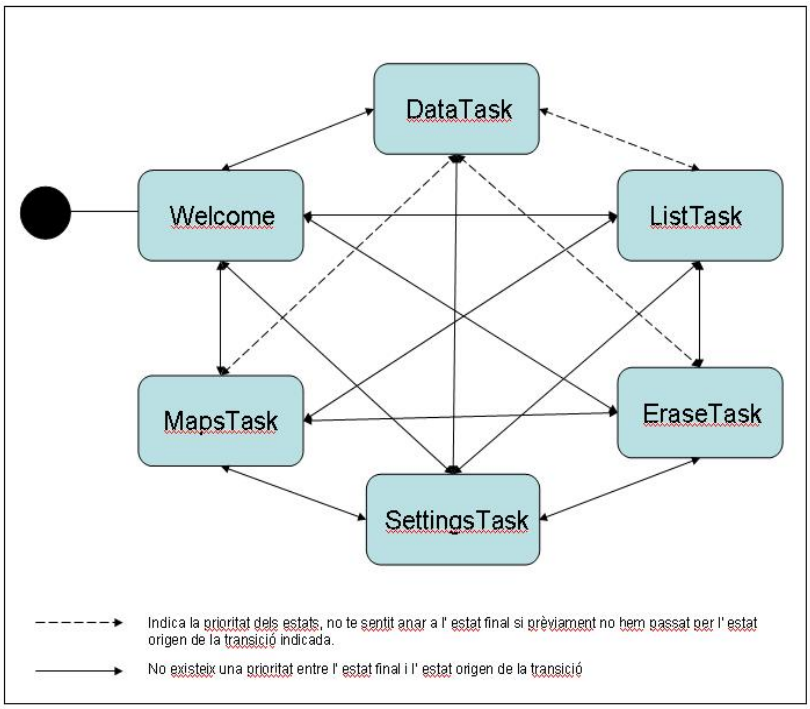


Figura 17: Il·lustració que mostra els estats de l' aplicació

Descripció Mòduls

Nom:	**DataTask**

Descripció:

El sistema no tindrà sentit si la aplicació no passa en algun moment o altre per aquest estat. Es el responsable d' obtenir les dades necessàries per poder després donar l' informació necessària a l' usuari.

Validacions:

- Comprovar el funcionament del lector de codis de barres.
- Comprovar la captura d' una fotografia i la seva eliminació posterior si és requerit per l' usuari.
- Comprovar l' obtenció correcta del nom del fitxer que conté la fotografia capturada.

- Comprovar l' obtenció de les coordinades latitud i longitud
- Comprovar la correcta visualització de la caixa de diàleg que presenta els colors.
- Comprovar que els filtres de les dades actuen correctament.
- Comprovar que les dades s' emmagatzemen al LocalStorage

Disseny:

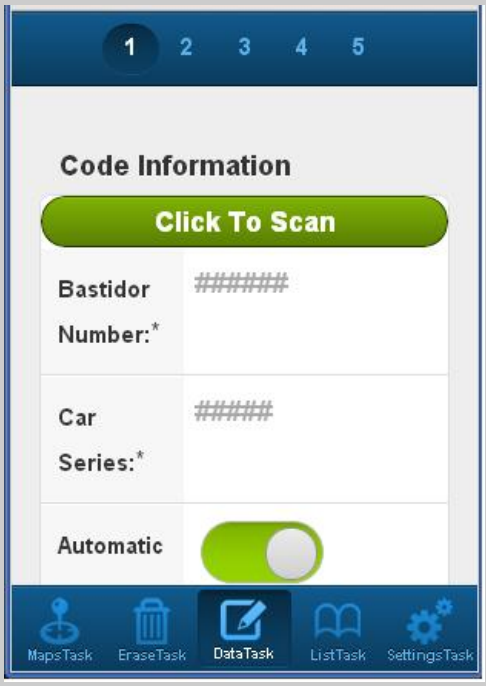


Figura 18: Il·lustració que mostra la pantalla inicial de DataTask

A l' imatge anterior es pot veure l' interfase utilitzat per fer la crida al lector de codis de barres. A més també observem una sèrie de nombres en la part superior de la mateixa que ens indiquen que per cadascun d' ells hi ha un altre formulari que utilitzarem per obtenir dades. El significat de cada nombre és:

1. Obté el número de bastidor i el car series de manera automàtica o manual.
2. Permet fer una fotografia del defecte del vehicle, així com obtenir el nom del fitxer que conté la mateixa. Un cop realitzada i mostrada al formulari l' usuari podrà decidir eliminar-la o no. També permet introduir el nom del responsable del defecte.
3. Obté les coordenades latitud i longitud.
4. Permet introduir el model del vehicle i el seu color. Dades importants en alguns casos per ser utilitzades en filtres de cerca.
5. Emmagatzema les dades, si aquestes han passat les validacions, al LocalStorage del dispositiu.

El Model que utilitzarem i que finalment serà emmagatzemat al dispositiu local és:

- NBastidor (tipus: String, longitud: 7): número que identifica a un vehicle.

- CarSeries (tipus: String, longitud: 4): cadena de text que agrupa els vehicles. Per tant poden haver 2 ‘NBastidor’ iguals en el conjunt de les dades, però no poden haver 2 ‘NBastidor’ iguals amb el mateix ‘CarSeries’.
- PhotoName (tipus: String, longitud: 25): cadena de text indicant el nom del fitxer que conté la fotografia.
- Reason (tipus: String, longitud: 15): cadena de text indicant qui és el responsable de solucionar el defecte.
- Latitude (tipus: String, longitud: 15): número que identifica la coordenada <latitud> del punt geogràfic on està ubicat el vehicle.
- Longitude (tipus: String, longitud: 15): número que identifica la coordenada <longitud> del punt geogràfic on està ubicat el vehicle.
- Model (tipus: String, longitud: 4): cadena de text identificant el model al qual pertany el vehicle, hi ha un model diferent per cada tipus de vehicle (actualment sols disposem de 3 models).
- Color (tipus: String, longitud: 15): cadena de text indicant el color exterior del vehicle.
- Date (tipus: Date): dia i hora en la que s’ ha enregistrat el vehicle.

Com funciona?

L’ usuari pot començar l’ introducció de dades per l’ opció que desitgi, la enumeració segueix un ordre lògic d’ introducció però no necessàriament ha de ser respectat. Això es compleix excepte en el <Tab5> que tot just és on es realitza la validació de les dades introduïdes anteriorment, o sigui que si alguna dada es considera errònia ja sigui per format o per absència el formulari apareixerà en blanc i el <Botó> per emmagatzemar les dades desactivat. Pel contrari si un cop arribats en aquest punt es compleixen totes les restriccions i per tant s’ accepten les dades introduïdes, apareixerà el formulari amb les dades i el <Botó> per emmagatzemar activat.
Si l’ usuari decideix modificar alguna dada ho pot fer en qualsevol moment, sempre i quant encara no hagi fet un <click> sobre el <Botó> moment en el qual les dades seran emmagatzemades en el LocalStorage del dispositiu i ja no podran ser modificades des d’ aquesta tasca.
Les validacions de les dades les realitza el Model directament i si en el moment d’ accedir al <Tab5> hi troba alguna discrepància mostrarà una finestra indicant tots aquells camps que no compleixen les condicions preestablertes.

Nom:	ListTask

Descripció:

Visualitza en una llista tots els vehicles emmagatzemats en el dispositiu mòbil. Permet fer una cerca del vehicle desitjat mitjançant un patró de cerca i un cop seleccionat ens mostrarà on es troba físicament ubicant-lo dins del mapa.

A més també ens permetrà un cop tenim el vehicle seleccionat modificar algunes de les dades del Model.

Validacions:

- Comprovar que la llista contingui tots els vehicles emmagatzemats.
- Comprovar que les dades i la fotografia associada al model del vehicle siguin coherents.
- Comprovar que a l' utilitzar el patró de cerca aquest funciona correctament.
- Comprovar que al realitzar un <click> sobre l'element seleccionat ens apareix el vehicle en qüestió ubicat dins del mapa
- Comprovar que la modificació de dades del vehicle funciona i que un cop realitzada s' emmagatzema al LocalStorage.

Disseny:

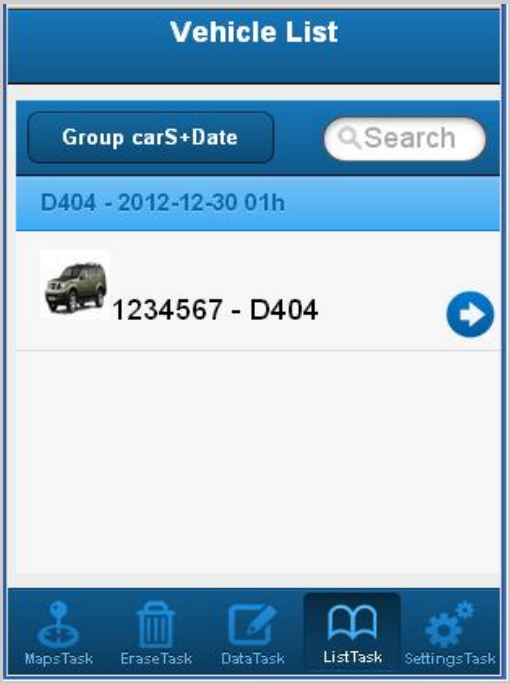


Figura 19: Il·lustració que mostra la pantalla inicial de ListTask

A l' imatge anterior es pot veure l' interfase utilitzat per visualitzar els vehicles dins d' una llista. Cada un dels elements de la mateixa té associat una imatge que indica el model, a més del número del bastidor i el 'car series'.

Com funciona?

Un cop se'ns mostra la llista de vehicles, l' usuari podrà navegar amunt i avall de la llista. Podrà seleccionar un vehicle, ja sigui per què l' ha trobat amb la navegació o bé per què ha utilitzat l' element de cerca. Fen un <click> sobre el vehicle desitjat li apareixerà un formulari on veurà posicionat sobre un mapa el vehicle en qüestió, a més de tenir la possibilitat de fer un <click> sobre el botó <Edit> que li permetrà modificar algunes dades d' aquest vehicle, com poden ser; les noves coordenades, el responsable d' arreglar el defecte, el nou defecte d' existir-hi.

Nom:	**MapsTask**
Descripció: Visualitza en el mapa un conjunt de vehicles, permetent a l' usuari localitzar la seva ubicació de manera ràpida i fàcil.	
Validacions: • Comprovar que l' ubicació mostrada en el mapa correspon a la realitat. • Comprovar que el resultat de prémer un botó (dels 3 possibles) ens mostra els vehicles correctes, tenint en compte quantitat de vehicles, agrupació i filtres de dades	

Disseny:

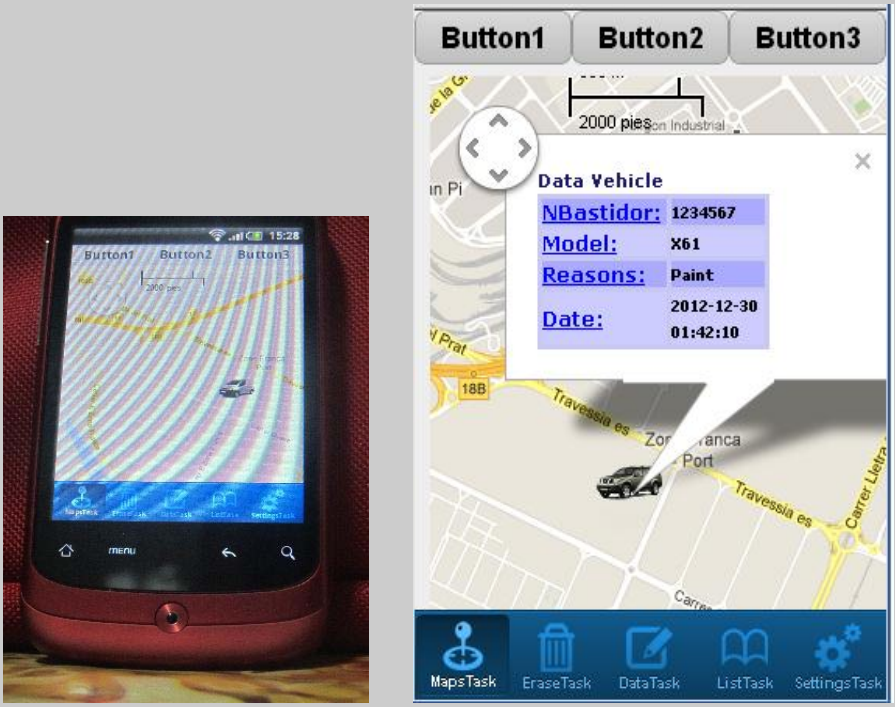


Figura 20: Il·lustració que mostra la pantalla inicial de MapsTask

A l' imatge anterior es pot veure l' interfase utilitzat per visualitzar els vehicles posicionats dins del mapa.

Com funciona?

Un cop realitzem un <click> sobre el botó apareixeran reflectits dins del mapa tots aquells vehicles que compleixen els filtres indicats, tant de quantitat de vehicles com del tipus. Un exemple de filtre podria ser: "cerca els últims 5 vehicles del model X amb motiu Y".

Nom:	**EraseTask**

Descripció:

Visualitza en una llista tots els vehicles emmagatzemats en el dispositiu mòbil. Permet fer una cerca del vehicle desitjat mitjançant un patró de cerca i un cop seleccionat ens preguntarà si desitgem eliminar-lo o no.

Validacions:

- Comprovar que la llista contingui tots els vehicles emmagatzemats.
- Comprovar que les dades i la fotografia associada al model del vehicle siguin coherents.

- Comprovar que a l' utilitzar el patró de cerca aquest funciona correctament.
- Comprovar que al realitzar un <click> sobre l'element seleccionat ens apareix una caixa de diàleg preguntant si volem o no eliminar el vehicle
- Comprovar que l' eliminació de dades del vehicle funciona i que un cop realitzada el LocalStorage s' ha actualitzat.
- Comprovar que al eliminar el vehicle de la llista si aquest té un fitxer de fotografia associat també s' eliminarà.

Disseny:

Figura 21: Il·lustració que mostra la pantalla inicial de l' opció EraseTask

Com funciona?

Un cop l' usuari ha entregat el vehicle al reparador de l' últim defecte tipificat s' haurà d' eliminar del LocalStorage, per aconseguir això haurà de seleccionar el vehicle desitjat de la llista e indicar que el vol eliminar confirmant aquesta acció. No necessàriament s' hauran d' eliminar els vehicles d' un en un, l' usuari pot marcar varis registres i després prémer el botó  per eliminar-los tots junts.

Nom:	SettingsTask

Descripció:

Visualitza en una llista els botons utilitzats en la tasca ‘MapsTask’, en concret mostrarà 3 registres. Cadascun d’ aquests registres conté informació que serà utilitzada a posteriori per a realitzar les cerques de vehicles. Les característiques que poden ser modificades són:

- Agrupar per: es dona la possibilitat de poder agrupar les dades de 2 maneres diferents; per model, per ‘carSeries’.
- Filtrat per: de la mateixa manera que abans també es donen 2 opcions a escollir; per motiu i per model.
- Quantitat a mostrar: ens permet indicar quans vehicles volem que siguin mostrats sobre el mapa quan fem un ‘click’ sobre el botó. Aquesta quantitat té un límit superior de 10 vehicles.

Validacions:

- Comprovar que la llista contingui els 3 botons.
- Comprovar que les dades de la configuració de cadascun dels botons siguin correctes.
- Comprovar que al realitzar un <click> sobre l’element seleccionat ens apareix un altre formulari on poder modificar les dades.
- Comprovar que la modificació de les dades funciona i que un cop realitzada el LocalStorage s’ ha actualitzat.

Disseny:

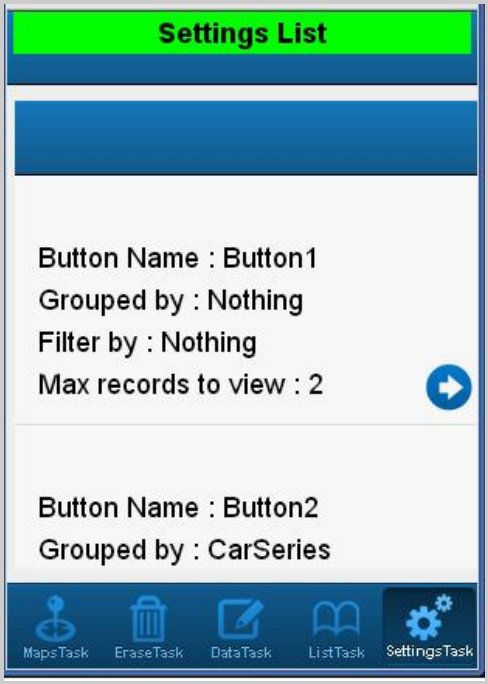


Figura 22: Il·lustració que mostra la pantalla inicial de l’ opció SettingsTask

Com funciona?

Cada cop que l' usuari faci un 'click' sobre qualsevol dels 3 registres li apareixerà un altre formulari amb la possibilitat de modificar dades. L' usuari podrà modificar tant el nom del botó, com els motius d' agrupació i filtre que s' aplicaran a les dades dels vehicles, així com la quantitat dels mateixos que es desitja poder visualitzar al mapa. Un cop fetes les modificacions l' usuari podrà prémer el botó 

per a fer aquests canvis definitivament consistents.

Explicació del codi

Mòduls que formen l' aplicació

L' aplicació ha estat construïda seguint un patró MVC. Aquest disseny s' ha aplicat individualment per a cadascuna de les 5 tasques, quedant de la següent manera la distribució de fitxers:

- Main.js (Vista): objecte de tipus <Ext.tab.Panel> que conté l' entrada a la resta de formularis. Aquest objecte té 5 objectes fills, un per cadascuna de les 5 tasques a desenvolupar.
- Maps.js (Vista): objecte de tipus <Ext.Container> que conté 3 botons i una imatge que serà on s' incrustarà el mapa de la fàbrica.
- cntrlMaps.js (Controlador): objecte de tipus <Ext.app.Controller> encarregat d' atendre totes les peticions enviades des de la vista <Maps.js>.
- MainEraseList.js (Vista): objecte de tipus <Ext.Panel> que conté una barra de navegació per poder navegar entre diferents vistes, així com un seguit de botons per a poder realitzar diferents tasques.
- VehiclesEraseList.js (Vista): objecte de tipus <Ext.dataview.List> que conté un 'template' a aplicar als registres dels vehicles que es mostren.
- cntrlEraseList.js (Controlador): objecte de tipus <Ext.app.Controller> encarregat d' atendre totes les peticions enviades des de les vistes <MainEraseList.js> i <VehiclesList.js>. El fet de disposar d' un navegador de vistes fa que aquest controlador hagi de rebre events de qualsevol de les 2 vistes.
- FormGetDataTask.js (Vista): objecte de tipus <Ext.tab.Panel> que conté 5 panels fills que seran utilitzats per introduir les dades necessàries de cadascun dels vehicles.
- cntrlGetDataTask.js (Controlador): objecte de tipus <Ext.Controller> encarregat d' atendre les peticions que li són enviades des de la vista <FormGetDataTask.js>.

- MainList.js (Vista): objecte de tipus <Ext.Panel> que conté una barra de navegació per poder navegar entre diferents vistes, així com un seguit de botons per a poder realitzar diferents tasques.
- VehiclesList.js (Vista): objecte de tipus <Ext.dataview.List> que conté un 'template' a aplicar als registres dels vehicles que es mostren.
- ShowMapData.js (Vista): objecte de tipus <Ext.Container> que conté 2 panels diferents dels quals sempre sols hi pot haver 1 de visible; el panel que presenta les dades per poder ser modificades i per tant emmagatzemades i el panel que mostra un 'template' amb algunes de les dades del vehicle a més del seu posicionament dins del mapa.
- cntrlListTask.js (Controlador): objecte de tipus <Ext.Controller> encarregat d' atendre les peticions que li són enviades des de les vistes <MainList.js>, <VehiclesList.js> i <ShowMapaData.js>. El fet de rebre events des de 3 punts diferents ens bé condicionat per l' utilització d'un navegador de vistes.
- MainSettingsList.js (Vista): objecte de tipus <Ext.Panel> que conté una barra de navegació per poder navegar entre diferents vistes, així com un seguit de botons per a poder realitzar diferents tasques.
- cfgTemplateList.js (Vista): objecte de tipus <Ext.dataview.List> que conté un 'template' a aplicar als registres dels botons que es mostren.
- ShowSettings.js (Vista): objecte de tipus <Ext.Container> que mostra el formulari que permet modificar les dades dels botons.
- cntrlSettingsTask.js (Controlador): objecte de tipus <Ext.Controller> encarregat d' atendre les peticions que li són enviades des de les vistes <MainSettingsList.js>, <cfgTemplateList.js> i <ShowSettings.js>. El fet de rebre events des de 3 punts diferents ens bé condicionat per l' utilització d'un navegador de vistes.
- proGenerics.js: classe estàtica que conté en seguit de mètodes comuns que poden ser utilitzats per la resta d' objectes.

En aquestes vistes i controladors hem d' afegir-hi els models de dades associats, així com els <stores>:

- modGeolocation.js: model de dades utilitzat per tractar les dades referents a la geolocalització. Els seus camps són:
  - Latitude: coordenada latitud.
  - Longitude: coordenada longitud.
  - Altitude: coordenada altitud.
  - locOptions: objecte utilitzat per condicionar el funcionament de l' obtenció de les coordenades mitjançant el GPS. Propietats que el composen:
    - timeout: indica el temps màxim que s' esperarà des del moment que es executada l' acció fins que la funció <success> es executada.

    - maximumAge: indica el temps en ms que les dades es mantindran a la cache. Per forçar que a cada petició es tornin a calcular les coordenades es convenient que aquesta propietat tingui un valor petit.
    - enableHighAccuracy: indica si volem o no que el càlcul d' obtenció de les coordenades sigui precís o no.
- modGetDataTask.js: model de dades utilitzat per tractar les dades referents a l' introducció de nous vehicles. Els seus camps són:
    - idBastCar (String): és la clau primària d' aquest model de dades i està composada pels camps 'nBastidor' i 'carSeries'
    - nBastidor (String): conté 7 xifres que indican el número de bastidor del vehicle.
    - carSeries (String): conjunt de 4 caràcters utilitzat per agrupar en grups diferents vehicles d' un mateix model. Per exemple 2 vehicles del mateix model 'X61' poden tenir 'carSeries' diferents ('D404' i 'D405') en funció si un és un model 'SUV' o 'pick-up'.
    - Reasons (String): indica quin és el departament responsable de que aquest vehicle estigui apartat del circuit normal de producció.
    - URLPhotoDefect (String): nom del fitxer que conté la fotografia del defecte pel qual el vehicle està apartat del circuit.
    - Latitude (String): indica la coordenada latitud d' on es troba el vehicle.
    - Longitude (String): indica la coordenada longitud d' on es troba el vehicle.
    - Model (String): indica el model del vehicle.
    - URLPhotoModel (String): nom del fitxer que conté la fotografia del model al qual pertany el vehicle.
    - Color (String): color exterior del vehicle.
    - dateCreated (Date): data en la qual el vehicle es enregistrat en el sistema CLSH.

  Indicar també que dins d' aquest model hi podem trobar els criteris utilitzats per realitzar les diferents validacions a les dades, hi podem trobar criteris de presència de dades, de longitud, de format, etc.
- modDataReasons.js: model utilitzat per tractar les dades referents als noms dels departaments responsables de la desviació dels vehicles del circuit estàndard. Els seus camps són:
    - text (String): el nom del departament responsable que apareixerà a la caixa de diàleg.
    - value (String): el valor que se li dona al text seleccionat.

- modDataProducts.js: model utilitzat per tractar les dades referents als diferents productes que es fabriquen actualment. Els seus camps són:
    - id (int): identificador del producte.
    - text (String): el nom del producte que apareixerà a la caixa de diàleg.

- value (String): el valor que se li dona al text seleccionat.

- btnCfg.js: model utilitzat per a tractar les dades referents als diferents botons utilitzats a la cerca de vehicles múltiple dins del mapa. Els seus camps són:
  - idBtn (String): identificador del botó.
  - BtnName (String): text que apareixerà dins del botó quant aquest es mostri dins del formulari.
  - Grp1 (String): nom del camp pertanyent al model < modGetDataTask.js> utilitzat pel tractament dels vehicles pel qual s' agruparan les dades.
  - Flt1 (String): nom del camp pertanyent al model modGetDataTask.js> utilitzat pel tractament dels vehicles pel qual es filtraran les dades.
  - Flt2 (String): quantitat de vehicles que es mostraran dins del planell.
- strGetDataTask.js: objecte de tipus <Ext.data.Store> que s' inicialitza amb la següent configuració:
  - Les dades estaran ordenades pel camp <dateCreated> de manera ascendent i per tant la data més llunyana estarà sempre a principi de llistat.
  - Les dades estaran agrupades pels camps <carSeries + dateCreted>.
  - Les dades s' emmagatzemaran al <LocalStorage>

  Aquest <Store> està directament relacionat amb el model <modGetDataTask.js>.
- strDataReasons.js: objecte de tipus <Ext.data.Store> directament relacionat amb el model <modDataReasons.js>.

  Aquest <Store> obté les dades d' un fitxer de tipus .Json. El path on es troba aquest fitxer és: </resources/statup/DataReasons.json> i la seva estructura interna la podem observar a continuació:

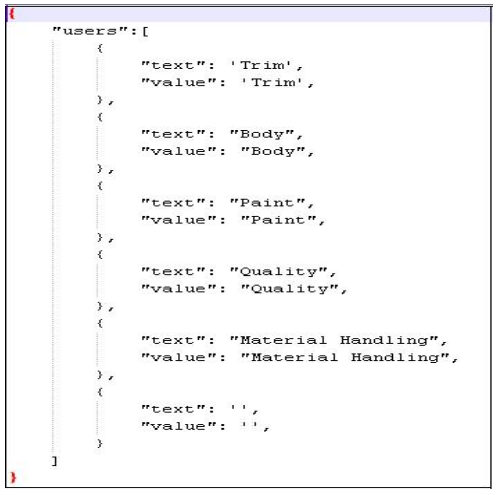


Figura 23: Il·lustració que mostra el fitxer DataReasons.json

- strDataProducts.js: objecte de tipus <Ext.data.Store> que s' inicialitza amb la següent configuració:
  - Les dades estaran ordenades pel camp <id> de manera ascendent.

Està directament relacionat amb el model <modGetDataTask.js> i obté les dades d' un fitxer de tipus .Json. El path on es troba aquest fitxer és: </resources/statup/DataProducts.json> i la seva estructura interna la podem observar a continuació:

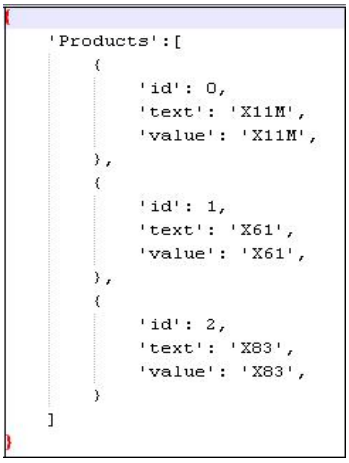


Figura 24: Il·lustració que mostra el fitxer DataProducts.json

- Settings.js: objecte de tipus <Ext.data.Store> que està directament relacionat amb el model <btnCfg.js>. Les seves dades s' emmagatzemaran dins del <LocalStorage>.

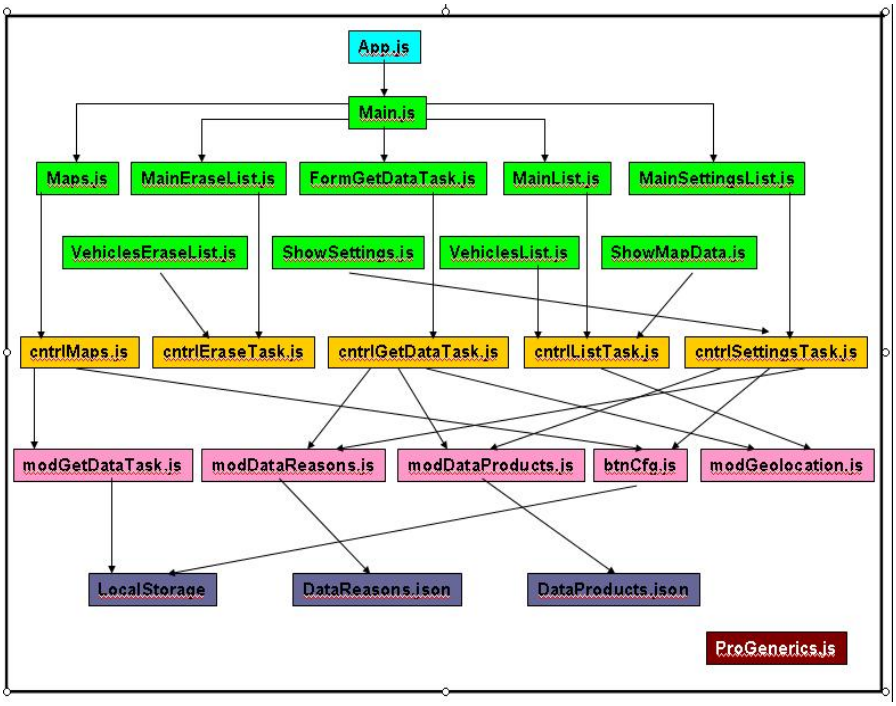


Figura 25: Il·lustració que mostra la relació entre els diferents mòduls

Vista Maps.js

Aquesta vista hereta de <Ext.Container> i té com a objectes fills 3 <Ext.Button> i un <Ext.Image>. L' objectiu d' aquesta vista es dibuixar els 3 botons en la part superior de la

mateixa i deixar la resta d' espai del contenidor per visualitzar el mapa on s' ubicaran els vehicles cercats. Els events que es generin per prémer qualsevol dels 3 botons (event *tap*) aniran dirigits al controlador <cntrlMaps.js>.

Controlador cntrlMaps.js

Aquest controlador té els següents mètodes:

doActivateMapsTask: aquest mètode és executat cada cop que es realitza un 'click' sobre el tab 

, en aquest moment es tornaran a carregar les dades de configuració de cadascun dels 3 botons ja que si no fos així es podria donar el cas de modificar quelcom dins la tasca <Settings> i no ho veuríem reflectit en aquesta vista.

doInitialize: aquest mètode és executat quant s' inicialitzen el botons i el que fa es fer la crida en un altre mètode <doInitializeBtns>.

doInitializeBtns: cada cop que s' inicialitzen els botons es fa una consulta al <LocalStorage> per obtenir el text de cadascun dels botons.

doGetRecordSettings: aquest mètode obté els 3 registres que composen del <LocalStorage> i assigna el text al botó en funció del botó que hagi estat pres. Si és el botó de l' esquerra se li assigna el valor del 1º registre, si és el botó del mig llavors els valors del 2º registre i finalment si és el de la dreta els del 3º registre.

doGoogle: aquest és el mètode més important d' aquest controlador i és el que ens permet visualitzar el mapa i els vehicles cercats. Com l' objecte <Ext.Button> es passat per paràmetre podem saber sobre quin botó s' ha fet un 'click' i per tant també quina configuració té definida dins del <LocalSotage>. Un cop obtinguda aquesta informació ja estem en situació d' agrupar i filtrar les dades dels vehicles si la configuració ho demana. Abans de mostrar els vehicles s' haurà de fer una comprovació més, es podria donar el cas que la quantitat de vehicles a mostrar per configuració fos superior al total de registres obtinguts després del filtrat i per tant aquest valor s' haurà de recalcular.

Ara ja tenim la informació necessària i passem a fer un recorregut per tots els registres del vehicles aplicant la configuració abans indicada. Per a cada vehicle:

- S' obtenen els camps <latitude> i <longitude> i es crea un objecte de tipus <google.maps.LatLng()>, aquestes seran les coordenades de posicionament dins del mapa.
- Es crea una variable <contentInfoWindow> de tipus <String> que conté codi HTML a més de dades particulars del registre vehicle tractat. L' aplicació ha estat construïda seguint un patró MVC. Aquest disseny s' ha aplicat individualment per a cadascuna de les 5 tasques, quedant
- Es crea un objecte de tipus <google.maps,LatLng()> per ubicar el centre del mapa.
- S' aplican els criteris de configuració que volem que tingui el mapa mostrat.
- Es crea un objecte de tipus <google.maps.Map()> passant-li com a paràmetres l' objecte on volem que es visualitzi, així com la configuració del mapa.

- Es marquen els límits del mapa, d' aquesta manera no farà falta indicar un valor de zoom. El mateix objecte <Map> en funció del tamany de la finestra on es mostri apareixerà amb un zoom o un altre. Aquesta acció es realitza cridant a un mètode extern que es troba dins de l' objecte <Profile.Generics> i el nom és <doLimitMap(map)>.
- Es crea un únic objecte de tipus <google.maps.InfoWindow> per a tots els marcadors, aquest objecte ens permet visualitzar informació del vehicle seleccionat. Li assignem alguns valors a les seves propietats:

  Content: no té un valor inicialment ja que se li assignarà quan es faci un 'click' sobre el marcador.

  disableAutoPan: fa que el mapa es desplaci per poder mostrar tot l' objecte dins del mapa.

  maxWidth: màxima amplada que tindrà aquest objecte

  pixelOffset: el desplaçament que se li vol donar respecte a les coordenades del marcador al qual està relacionat, en el meu cas cap.
- Afegim tots els marcadors sobre el mapa, això ho farem crenat objectes de tipus <google.maps.Marker>, on se li indicarà sobre quin mapa s' han de ubicar, si els dibuixos tindran alguna animació i les coordenades de posicionament dins del mapa.
- L' imatge associada a cada marcador la resolem en temps d' execució en funció del valor del camp 'model' del registre vehicle tractat.
- Finalment es configuren els events que desitgem controlar. Cada cop que es faci un 'click' sobre qualsevol dels marcadors succeirà el següent:
  - Si el marcador té una animació assignada la finalitzarà, si no en cas contrari la començarà.
  - Se li assignarà un <String> a la propietat <content> de l' objecte <infoWindow>.
  - S' obrirà aquest objecte sobre el mapa i marcador indicats.
  - Es tornarà a posicionar el mapa.

Aquesta última tasca ha estat realitzada utilitzant una funció <clousure> que ens permet utilitzar variables d' un altre àmbit dins de l' àmbit actual d' execució. En el nostre cas s' han utilitzat les variables <markers> i <contentInfoWindow> d' un altre àmbit dins de la definició de l' event.

```
(function(marker, contenido){
    google.maps.event.addListener(marker, 'click', function()
                                  {
                                      if (marker.getAnimation() == null || marker.getAnimation() == google.maps.Animation.DROP)
                                      {
                                          marker.setAnimation(google.maps.Animation.BOUNCE);
                                          infoWindow.setContent(contenido);
                                          infoWindow.open(map,marker);
                                      }else{
                                          marker.setAnimation(null);
                                          infoWindow.close();
                                          GS.profile.proGenerics.doLimitMap(map);
                                      }
                                  });
    google.maps.event.addListener(infoWindow,'closeclick', function()
                                                  {
                                                      //Colocamos los límites para visualizar el mapa
                                                      GS.profile.proGenerics.doLimitMap(map);
                                                      marker.setAnimation(null);
                                                  });
})(markers[x],contentInfoWindow[x]);
```

Figura 26: Il·lustració d' una funció clousure

Vista MainEraseList.js

Aquesta vista hereta de <Ext.Panel> i té com a objectes fills un <Ext.navigation.View> amb 3 objectes, 2 de tipus <Ext.Button> i 1 de tipus <Ext.field.Search> i un altre fill de tipus <VehiclesEraseList>.

Vista VehiclesEraseList.js

Aquesta vista hereta de <Ext.dataview.List> i serà el template que utilitzarem per a mostrar els diferents registres de vehicles. Algunes propietats d' aquesta llista són:

- Permet seleccionar més d' un element de la llista
- Permet agrupació d' elements
- Cada cop que es fa un 'click' sobre un element hi ha un canvi de format en l' element.

Controlador cntrlEraseTask.js

Aquest controlador té els següents mètodes:

doGroup: aquest mètode es executat cada cop que fem un 'click' sobre el botó

. En funció del text del botó agruparem les dades per data o per <carSeries + data>.

doShowList: aquest mètode és cridat cada cop que la llista anterior es pintada i el que fa es torna a carregar les dades de vehicles des del <LocalStorage>.

doEraseItemList: mètode responsable de l' eliminació dels registres tant dels mostrats a la llista com dels que es troben dins del <LocalStorage>. Obtindrem tant el total de elements seleccionats com els elements en si. Un cop aquí mostrarem una caixa de missatges indicant si l' usuari vol realment eliminar els elements o no. Si és el cas és cridarà una funció callback per realitzar aquesta tasca.

```
onConfirm = function(buttonId, value, opt)
            {
                var x = 0;  //Indice utilizado para recorrer el array de items

                if(buttonId == 'yes')
                {
                    // Se obtiene el almacen de datos
                    vehStore = Ext.getStore('strGetDataTask');

                    //Se realiza una recorrido por todos los items seleccionados
                    for(x=0; x<itemsSelect; x++)
                    {
                        //console.log(itemsList[x].data.nBastidor + ' - ' + itemsList[x].data.carSeries);
                        //this.getRefEraseItemList().remove(itemList[x],false);

                        //Eliminamos el fichero que contiene la fotografía (de ser el caso)
                        //Si el campo <URLPhotoDefect> no contiene el nombre del fichero
                        //no haremos nada en caso contrario borraremos.
                        if (itemsList[x].data.URLPhotoDefect != '')
                        {
                            this.doErase(itemsList[x], null);
                        }
                        //Eliminamos el registro del <Store>
                        vehStore.remove(itemsList[x]);
                    }

                    //Confirmamos los datos eliminados
                    vehStore.sync();
```

Figura 27: Funció callback utilitzada en l' eliminació d' un fitxer

doFindEraseList: mètode per buscar i seleccionar un element de la llista.
doErase: mètode que fa una crida a un altre mètode que és el que realment esborrarà el fitxer associat a la fotografia del defecte, de ser el cas.

Vista FormGetDataTask.js

Aquesta vista hereta de <Ext.tab.Panel> i té com objectes fills 5 <Ext.Panel> un per cada tipus de dada que es vol obtenir. Al 1º panel s' obtindrà el número de bastidor i el 'carSeries', al 2º el nom del responsable i el nom del fitxer que conté la fotografia del defecte, al 3º s' obtenen les coordenades latitud i longitud, al 4º el model al qual pertany el vehicle i el color exterior del mateix i finalment al 5º es validaran totes les dades introduïdes anteriorment.

Controlador cntrlGetDataTask.js

Aquest controlador té els següents mètodes:
doLoadModel: aquest mètode és cridat per validar el contingut de les dades introduïdes als 5 panels citats anteriorment. Un cop assignem les dades a un registre del model <modGetDataTask> aquest es validarà si no es troba cap incompatibilitat es mostraran les dades al formulari, en cas contrari apareixerà una caixa de diàleg indicant les anomalies trobades.

doSave: aquest mètode és executat un cop fem 'click' sobre el botó 

. Es busca dins del <LocalStorage> si ya existeix un altre registre amb la mateixa clau primària, de ser així apareixerà una caixa de diàleg confirmat aquesta situació, en cas contrari afegirem un nou registre al magatzem permanent.
doShowColor: aquest mètode és cridat quant fem un 'click' sobre el camp de text que demana el color exterior del vehicle. Un cop succeeix aquest event apareixerà un nou formulari modal mostrant 16 colors diferents, donant l' opció a l' usuari d' escollir-ne un, un cop fet farem un altre 'click' fora del formulari i el nom del color seleccionat apareixerà dins de la caixa de text citada anteriorment.
doDeletePhoto: el que fa aquest mètode es eliminar un fitxer del sistema de fitxers del dispositiu mòbil. Aquest mètode es cridarà sempre que prenem el botó 

indicat per fer aquesta acció. Utilitzarem varies funcions callback per realitzar aquesta tasca, la 1ª funció cridada es farà si l' accés al sistema de fitxers ha estat satisfactori, dins d' aquesta cridarem a una 2º funció si hem accedit correctament al fitxer i finalment a una 3º que ens indicarà si l' acció d' esborrar ha tingut èxit.

```
                                        };
                        alert(fs.name + " " + fs.root.name);
                        //Se solicita un descriptor para el fichero solicitado por parámetro
                        fs.root.getFile(GS.controller.cntrlGetDataTask.PATHPHOTO + Ext.getCmp('textFieldNamePhoto').getValue(),
                        alert("Obtenido el descriptor....");
                    };
var onErrorFunction = function(e)
                    {
                        alert("Can not access the file system. Error: " + e + " ,Error Code: " + e.code);
                    };
//Se solicita el acceso al sistema de ficheros del dispositivo móbil
window.requestFileSystem(LocalFileSystem.PERSISTENT, 0, onSuccessFunction, onErrorFunction);
alert("Acceso al FilesSystem...");
```

Figura 28: Funció callback per l' accés al sistema de fitxers

doPhoto: permet realitzar una fotografia utilitzant la càmera del dispositiu mòbil, aquesta acció també la realitza mitjançant funcions callback. La crida a la càmera es realitzarà mitjançant PhoneGap. El fet d' utilitzar la càmera del dispositiu fa que aquest mètode ens permeti via propietats modificar una mica el comportament de la mateixa; característiques que podem canviar poden ser el tipus de destinació, el tipus de codificació, l' amplada de la fotografia, l' alçada de la fotografia.

```
    var onCameraSuccess = function(imageURL)
                          {

                              //Se obtiene la posición del último "/" encontrado en el path
                              var index = imageURL.lastIndexOf("/");
                              alert("Photo file: " + imageURL.substring(index + 1) + "\n" +
                                    "Path Photo: " + imageURL);
                              //Se obtiene el nombre del fichero únicamente, se desestima toda
                              //la ruta entera de carpetas
                              Ext.getCmp('textFieldNamePhoto').setValue(imageURL.substring(index + 1));
                              //Se activa el botón por si se desea eliminar la fotografía realizada.
                              Ext.getCmp('buttonDeletePhoto').setDisabled(false);
                              //Se desactiva el botón que permite realizar fotografías
                              Ext.getCmp('buttonMakePhoto').setDisabled(true);
                              //Asignamos la ruta de un fichero al objeto <imagen>
                              Ext.getCmp('imgPhoto').setSrc(imageURL);
                          };
    var onCameraError =
         function(e){alert("Isn't possible to make photo. Error: " + e + " ,Error Code: " + e.code);};
    navigator.camera.getPicture(onCameraSuccess, onCameraError,{quality: 90,
                                                                sourceType : Camera.PictureSourceType.CAMERA,
                                                                destinationType: Camera.DestinationType.FILE_URI,
                                                                encodingType: Camera.EncodingType.JPEG,
                                                                targetWidth: 200,
                                                                targetHeight: 121
                                                               });
},
```

Figura 29: Funció callback utilitzada per fer una fotografia

doChange: cada cop que fem lliscar l' objecte <Slider> estem fent un canvi d' automàtic a manual i al contrari, o sigui estem indicant que la captura del codi de barres es faci de manera automàtica mitjançant un lector de codis o que bé l' introducció del número de bastidor i el <carSeries> es faci de manera manual.

doScan: mètode que crida a un plugin creat amb PhoneGap per realitzar la lectura de un codi de barres.

```
window.plugins.barcodeScanner.scan(function(result)
                                   {
                                       var nBastidor = '';
                                       var carSeries = '';

                                       alert("We got a barcode\n" +
                                           "Result: " + result.text + "\n" +
                                           "Format: " + result.format);

                                       //Se controla la longitud del barcode leído
                                       //Nbastiodr: un número de 7 cifras
                                       //CarSeries: 4 caracteres
                                       if (result.text.length == GS.controller.cntrlGetDataTask.LENGTHBARCODE)
                                       {
                                           nBastidor = result.text.substring(0,GS.controller.cntrlGetDataTask.LENGTHNBASTIDOR)
                                           Ext.getCmp('textFieldBastidor').setValue(nBastidor);
                                           carSeries = result.text.substring(GS.controller.cntrlGetDataTask.LENGTHNBASTIDOR);
                                           Ext.getCmp('textFieldCarseries').setValue(carSeries);
                                           Ext.getCmp('textFieldBastidor').setReadOnly(true);
                                           Ext.getCmp('textFieldCarseries').setReadOnly(true);
                                           alert("NBastidor: " + nBastidor + "\n" + " CarSeries: " + carSeries);

                                       }else{
                                           alert('BarCode length wrong');
                                       }
                                   }, function(error)
```

Figura 30: Funció callback utilitzada per la lectura de BarCodes

doGeolocation: aquest mètode permet obtenir mitjançant PhoneGap les coordenades actuals d' on es troba el dispositiu mòbil. Per aconseguir-ho crea un objecte de tipus <model.modGeolocation> el qual s' haurà d' inicialitzar amb els següents valors:

- Timeout: 15000 ms, és el temps màxim d' espera que té el sistema des de que es fa la crida a la funció i la funció calbback associada és executada.
- maximumAge: 5000 ms, el temps que estan a la caché les dades obtingudes prèviament.
- enableHighAccuracy: true, ens indica que volem que el dispositiu sigui elmàxim de precís.

Aquest objecte té un únic mètode, <getLocation()>, al qual li passem un objecte del mateix tipus. El que farà serà utilitzar una funció callback que ens retornarà les coordenades si ha tingut èxit en la seva obtenció.

```
getLocation:
    function(gps)
    {
        var onGeolocationSuccess = function(loc){gps.setLatitude(loc.coords.latitude);gps.setLongitude(loc.coords.longitude);gps.setAltitud
        var onGeolocationError =
            function(e){alert("Geolocation error! Clearing Watch ID");};
        navigator.geolocation.getCurrentPosition(onGeolocationSuccess, onGeolocationError, gps.getLocOptions());
    }
```

Figura 31: Funció callback utilitzada pel GPS

doValidation: aquest mètode realitza un recorregut pet tots els errors trobats durant la validació dels valors dels camps del registre vehicle i els mostra amb una caixa de diàleg.

```
//Variable que contendrá todos los mensajes de error de la validación
var errStr = '';
Ext.each(errors.items, function(error, index, errors)
                       {
                           errStr += error.getField() + ' : ' + error.getMessage() + '\n';
                       }
        );
alert(errStr + ' \n\n' + 'You must revised the data introduced into form.');
```

Figura 32: Recorregut d' elements

Vista MainList.js

Aquesta vista hereta de <Ext.Panel> i té com a objectes fills un <Ext.navigation.View> amb 6 objectes, 5 de tipus <Ext.Button> i 1 de tipus <Ext.field.Search> i un altre fill de tipus <VehiclesEraseList>. Aquests 5 botons apareixeran i desapareixeran en funció de quina vista tinguem activa. A la vista principal apareixen els botons de canviar d' agrupació les dades i el de localitzar un element de la llista, a la 2ª vista apareix el botó que permet editar l' element seleccionat i finalment a la 3ª vista apareixen els botons que permeten realitzar una fotografia, obtenir les coordenades, eliminar una fotografia i emmagatzemar de manera permanent les dades modificades.

Vista VehiclesList.js

Aquesta vista hereta de <Ext.dataview.List> i serà el template que utilitzarem per a mostrar els diferents registres de vehicles. Algunes propietats d' aquesta llista són:

- Permet agrupació d' elements
- Cada cop que es fa un 'click' sobre un element hi ha un canvi de format en l' element.

Vista ShowMapData.js

Aquesta vista hereta de <Ext.Container> i té com a objectes fills 2 contenidors més, un que contindrà el mapa que es mostrarà per ubicar el vehicle seleccionat de la llista i un altre per mostrar les dades d' aquest vehicle que es volen modificar.
Té un únic mètode:
updateRecord: aquest mètode és necessari per poder mostrar les dades de l' element seleccionat tant en la 2ª vista com en la 3ª. Cada cop que es realitza un 'click' sobre un element de la llista es crearà aquesta vista <ShowMapData> passant-li el registre de l 'element seleccionat, llavors en funció de la vista que mostrem es veuran uns valors dels camps o uns altres.

Controlador cntrlListTask.js

Aquest controlador té els següents mètodes:
doGroupList: aquest mètode es executat cada cop que fem un 'click' sobre el botó

. En funció del text del botó agruparem les dades per data o per <carSeries + data>.
doPaint: aquest mètode és molt semblant al mètode <doGoogle()> (veure controlador <cntrlMaps.js> per les explicacions).
doErase: mètode que elimina el fitxer que conté la fotografia del defecte. Per dur a terme aquesta tasca crida al mètode <profile.proGenerics.doDeletePhoto()> (veure aquesta classe per més explicacions).

doPhoto: mètode que permet utilitzar la càmera del dispositiu mòbil. Per dur a terme aquesta tasca crida al mètode <profile.proGenerics.doPhoto()> (veure aquesta classe per més explicacions).

doPop: cada cop que tornem enrere mitjançant el botó 

es fa una crida en aquest mètode. Independentment de la vista en la que ens trobem, 2ª o 3ª, sempre tornarem a la vista inicial on es veu el llistat de vehicles. El que fa aquest mètode es descarregar de la pila les vistes que s' han a nat mostrant.

doSaveData: una vegada ja estem segurs dels canvis que hem realitzat procedirem a emmagatzemar-los definitivament utilitzant aquest mètode.

doEditItemList: aquest mètode s' executarà capa cop que es faci 'click' sobre el botó 

. El text d' aquest botó canviarà entre 2 valors possibles: "Edit" i "Save" i en funció d' aquests el comportament d' aquest mètode serà diferent. Si estem en la funció d' editar llavors es presentaran els valors dels camps del registre seleccionat per què l' usuari pugui modificar les dades, si pel contrari ens trobem davant la funció de gravar llavors cridarem al mètode <doSaveData>.

doClickItem: cada cop que es faci un 'click' sobre un element de la llista es cridarà en aquest mètode. Es crearà un objecte de tipus <ShowMapData> (veure aquesta classe per més explicacions) amb la vista configurada per defecte i se li passarà el registre amb les dades de l' element seleccionat. Per a posar aquesta vista com a activa es farà un 'push' de la mateixa.

```
if ( Params['recordModel'] != null)
{
    //Guardamos los valores de los campos del registro seleccionado
    GS.controller.cntrlListTask.record = Params['recordModel'];

    refShowPanel.setRecord(Params['recordModel']);

    //Cargamos la foto en el objeto <Image>
    if(GS.controller.cntrlListTask.record.data.URLPhotoDefect != '')
    {
        this.getRefFormUpdImg().setSrc(GS.controller.cntrlListTask.PATHPHOTO + GS.co

    }else{
        alert("Error no hay nombre de foto ");
    }

    //this.getRefFormContainerData().setHidden(true);
    //this.getRefFormContainerMap().setHidden(false);
    Ext.getCmp('navigation').push(refShowPanel);

    //Se añade un manejador de eventos al objeto id: 'PanelMap'
    this.getRefPanelMap().on('initialize',this.doPaint(this.getRefPanelMap()));
```

Figura 33: Push d' una vista

doShowList: cada cop que es pinta la llista <VehiclesList> es cridarà en aquest objecte, que té com a missió tornar a carregar les dades del <LocalStorage> i refrescar les dades de la llista.
doFindList: mètode per buscar i seleccionar un element de la llista.

Vista MainSettingsList.js

Aquesta vista hereta de <Ext.Panel> i té com a objectes fills un <Ext.navigation.View> amb 1 objecte de tipus <Ext.Button>. Aquest botó només serà visible quan es faci un 'click' sobre algun dels 3 elements de la llista

Vista cfgTemplateList.js

Aquesta vista hereta de <Ext.dataview.List> i serà el template que utilitzarem per a mostrar els diferents registres dels botons de configuració utilitzats al controlador <cntrlMaps>.

Vista ShowSettings.js

Aquesta vista hereta de <Ext.dataview.List> i serà el template que utilitzarem per a mostrar els diferents registres dels botons de configuració utilitzats al controlador <cntrlMaps>. Conté 3 mètodes que tenen com a finalitat marcar els objectes <Radio> i <Select> utilitzats per indicar la configuració del botons:
updateRecord: com ja he explicat anteriorment aquest mètode es utilitzat per rebre un registre de dades des d' un altre objecte. Es comprova si el valor del camp <Flt1> existeix dins del <LocalStorage>, aquesta informació serà utilitzada a posteriori quan es realitzi un recorregut per tots els objectes <Radio>. El recorregut juntament amb el valor del camp indicat abans serviran per poder mostrar l' estat que tenia el botó seleccionat en un moment anterior.
onChange: cada cop que es produeix un canvi de selecció d' element s' executarà aquest mètode, que té como a funcionalitat emmagatzemar aquest canvi en un objecte invisible del formulari.
onCheck: cada cop que es produeix un canvi d' objecte <Radio> s' executarà aquest mètode, que té com a funcionalitat emmagatzemar aquest canvi en un objecte del formulari.

Controlador cntrlSettingsTask.js

Aquest controlador té els següents mètodes:
doInitialize: cada cop que s' inicialitzi l' objecte <Ext.navigation.View> s' executarà aquest mètode que tindrà com a missió donar d' alta 3 configuracions de botons per defecte si no existeixen en el moment de l' inicialització, o sigui si ja existeixen degut a una arrancada prèvia no es crearà cap registre en cas contrari si.

doPop: cada cop que tornem enrere mitjançant el botó 

es fa una crida en aquest mètode i es farà un 'pop' de la vista actual.

doSaveData: cada cop que fem un 'click' al botó 

es cridarà en aquest mètode. L' objectiu es fer permanent qualsevol canvi que s' hagi produït al formulari. Hem de tenir en compte que sempre hi hauran sols 3 registres i que només es donaran d' alta quan a l' iniciar l' objecte <Ext.navigation.View> no existeixen els registres, per tant amb aquesta opció sempre modificarem dades ja existents.

doClickItem: quan fem un 'click' sobre algun element de la llista cridem en aquest mètode. El que fa es crear un objecte de tipus <View.showSettings> i fer un 'push' a la pila de vistes, tot seguit enviarem en aquesta vista que s' ha posat com a activa la informació de l' element seleccionat per poder-lo tractar en la vista següent.

ProGenerics.js

Classe estàtica que conté un sèrie de mètodes utilitzats pels altres objectes:

maskFormSaveData: màscara que apareix quan s' estan emmagatzemant de manera permanent les dades. Només ens ho trobarem en 2 situacions, al finalitzar el procés d' alta d' un vehicle i després per fer possibles modificacions.

maskFormGPS: màscara utilitzada quan s' estan fent els càlculs per obtenir la posició de latitud i longitud.

doGetNet: mètode que utilitzant PhoneGap ens permet saber quin tipus de connexió està utilitzant el dispositiu mòbil en certs moments, sobre tot es farà aquest control quan s' hagin de presentar vehicles dins d' un mapa. Si a l' arrencar l' aplicació no hi havia una connexió de xarxa establerta i tampoc dades a la cache aquest mètode ens indicarà en quina situació ens trobem i no es mostraran els mapes, però si pel contrari a l' arrencar si hi havia xarxa i la desconnexió ha succeït a posteriori, el mètode ens detectarà la situació però el mapa es continuarà mostrant degut a que es trobava a la cache, això si al no tenir xarxa s' aniran produint errors interns que faran alentir el funcionament de l' aplicació.

doPhoto: mètode que utilitzant PhoneGap permet realitzar una fotografia i emmagatzemar-la en un fitxer.

doDeletePhoto: mètode que utilitzant PhoneGap permet eliminar un fitxer del sistema de fitxers del dispositiu mòbil.

doLimitMap: aquest mètode es utilitzat per establir els límits del mapa que volem que es mostri. El fet d' utilitzar límits cartesians farà que el mateix objecte s' ajusti a ells i el zoom no s' hagi de tocar per res.

Cost del Projecte

Com en quasi tots els projectes ha hagut algunes desviacions del previst inicialment amb el resultat final. He de dir que la part que m' ha portat més temps ha estat la d' I+D de les noves tecnologies, anar llegint molta documentació i fent petits desenvolupaments per provar les diferents funcionalitats que havia de tenir l' aplicació final. El cost final ha estat el següent:

Tasques	Setmanes	Hores	Total Hores
Lectura documentació llibres	12	5	**60**
Lectura documentació articles	6	2	**12**
Desenvolupament/Proves Barcode	1	16	**16**
Desenvolupament/Proves Càmara	1	3	**3**
Desenvolupament/Proves GPS	1	10	**10**
Desenvolupament/Proves Mapes	1	15	**15**
Desenvolupament/Proves Sistema Fitxers	1	5	**5**
Desenvolupament/Proves Network	1	3	**3**
Desenvolupament/Proves Disseny Interfases Gràfiques	4	5	**20**
Unificació dels Codis	1	2	**2**
Test de l' aplicació	2	4	**8**
Memòria del Projecte	4	7	**28**
Obtenció de requeriments	1	8	**8**
Unificació de mòduls i proves conjuntes	2	8	**16**
Total hores projecte			**206**

Indicar a més que aquest cost s' ha vist molt incrementat per les ganes d' aprendre la nova tecnologia amb profunditat a la que se li han dedicat moltes hores, també ha influït en aquest augment d' hores el fet de tractar amb activitats desconegudes per a mi. Cada nova tasca implicava quasi per complert tornar a començar de 0, en definitiva si hagués estat dedicat única i exclusivament a fer aquesta tasca durant 8 hores al dia, hauria necessitat 8 setmanes de feina per a dur-la a terme.

Conclusions

Durant aquests últims mesos he estat mol capficat en dur a terme aquest projecte i en aconseguir una eina útil per l' empresa per la qual treballo. He de dir no obstant que el principal objectiu que tenia en cap quant vaig escollir aquesta branca era l' aprenentatge de les tecnologies utilitzades. Fins aquest moment no havia mai tingut contacte amb les aplicacions mòbils i he de dir que és un món que m' ha intrigat bastant i que segurament continuaré explotant. En aquesta branca de les aplicacions híbrides i en concret amb les eines utilitzades, com per exemple Sencha, m' he trobat amb forces dificultats. Problemes i comportaments que a priori no s' haurien de donar i que fins que no trobes alternatives fan perdre molt de temps, en especial comportaments dels interfases gràfics.
Seguint en aquesta línia també cal comentar els comportaments del <LocalStorage>, nova característica de HTML5 que a vegades et dona sorpreses no desitjades com la desaparició de dades sense un motiu clar.
Fets com els comentats han provocat mols mal de caps, però finalment s' ha aconseguit finalitzar la feina i entregar una aplicació amb la funcionalitat indicada des del començament. Com ja he indicat anteriorment el meu principal objectiu era l' aprenentatge d' aquestes noves tecnologies i es per això que aplicant aquest principi he desenvolupat codi amb una mateixa funcionalitat de diferents maneres, per poder conèixer quan més millor del Sencha. Alguns casos podrien ser la creació d' objectes, la creació d' events, la referència a objectes, etc.

Línies de futur

Al llarg del desenvolupament d' aquest projecte he anat observant punts que en un futur es podrien arribar a implementar i a posar en marxa, alguns d' aquests podrien ser:

- Evitar la connexió inicial a Internet per a obtenir una referència als objectes que ofereix Google en les seves llibreries, així com l' obtenció de les coordenades per GPS. Com sempre estem tractant amb les mateixes coordenades del mapa, aquest el podríem arribar a dibuixar, posant sobre d' ell una quadrícula per permetre a l' usuari que seleccionés en quina casella de les dibuixades es troba el vehicle defectuós. Aquest punt donaria major temps de resposta a l' aplicació.
- La solució enfocada en aquest projecte al problema presentat sols dona cabuda a l' utilització de l' eina per usuaris de manera individual, o sigui cada usuari portarà el dispositiu amb l' aplicació i anirà obtenint el posicionant dels vehicles. Aquest fet fa que els vehicles capturats per un usuari no puguin ser vistos per la resta. La millora en aquest punt passaria per desenvolupar una aplicació servidora que anés rebent l' informació de cada vehicle capturat, de tal manera que quant un usuari desitgés obtenir

dades el que hauria de fer el dispositiu mòbil es fer una petició al servidor amb les dades del vehicle que vol consultar. D' aquesta manera tots els usuaris podrien treballar amb les dades obtingudes per altres usuaris. El fet de pensar en un repositori multi-usuari ens podria plantejar dubtes com per exemple, que passa amb les col·lisions, o sigui 2 ó més usuaris accedint a les mateixes dades? Doncs bé en el nostre cas res, ja que un mateix vehicle sols pot estar en un lloc i per tant mai hi hauran 2 usuaris modificant les seves dades.

- Amb l' objectiu d' aprendre a fer un mateix desenvolupament de varies maneres, com ja he comentat abans, hi han algunes parts de codi que es podrien millorar, com per exemple el cas del formulari dels colors. En aquest cas volia crear el formulari en temps real d' execució sense fitxers externs de dades, cosa que provoca que tinguem una limitació inicial de colors als que s' han definit dins del codi.

- La visualització dels vehicles des de qualsevol PC de l' Intranet, el fet de disposar d' una aplicació servidora que pot rebre les actualitzacions de les dades dels vehicles ens permet la possibilitat de mostrar l' ubicació dels vehicles a qualsevol usuari d' oficines que els vulgui consultar.

Plug-In BarCode

Indicar que aquest plugin no ha estat realitzat per mi i que ha estat aconseguit a la següent adreça: //github.com/phonegap/phonegap-plugins/tree/master/Android. Dins d' aquesta carpeta hi podem trobar diferents plugins que ens poden facilitar la feina, en el meu cas he utilitzat el que permet fer lectures de codis de barres. A continuació faré una breu explicació dels passos a seguir per poder utilitzar-lo en les nostres aplicacions:

- Descarregar els fitxers de l' adreça indicada anteriorment.
- Crear un nou projecte a Eclipse i generar com a resultat una llibreria.
- Afegir aquesta llibreria al projecte que desitgem.
- Copiar el fitxer <barcodescanner.js> dins la carpeta <assets/www>.
- Crear un nou package dins de la carpeta <src> amb el nom <com.phonegap.plugins.barcodescanner> i copiar-hi el fitxer <BarcodeScanner.java>.
- Obrir el fitxer <res/xml/config.xml>.
- Posicionar-se dins del tag <plugins></plugins>
- Afegir el següent tag:

```
<plugin name="BarcodeScanner"
value="com.phonegap.plugins.barcodescanner.BarcodeScanner"/>
```

- Obrir el fitxer <AndroidManifest.xml>.
- Posicionar-se dins del tag <application></application>.

- Afegir els següents tags:

```
<activity android:name="com.google.zxing.client.android.CaptureActivity"
    android:screenOrientation="landscape"
    android:configChanges="orientation|keyboardHidden"
    android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
    android:windowSoftInputMode="stateAlwaysHidden">
  <intent-filter>
    <action android:name="com.phonegap.plugins.barcodescanner.SCAN"/>
    <category android:name="android.intent.category.DEFAULT"/>
  </intent-filter>
</activity>
<activity android:name="com.google.zxing.client.android.encode.EncodeActivity"
    android:label="@string/share_name">
  <intent-filter>
    <action android:name="com.phonegap.plugins.barcodescanner.ENCODE"/>
    <category android:name="android.intent.category.DEFAULT"/>
  </intent-filter>
</activity>
```

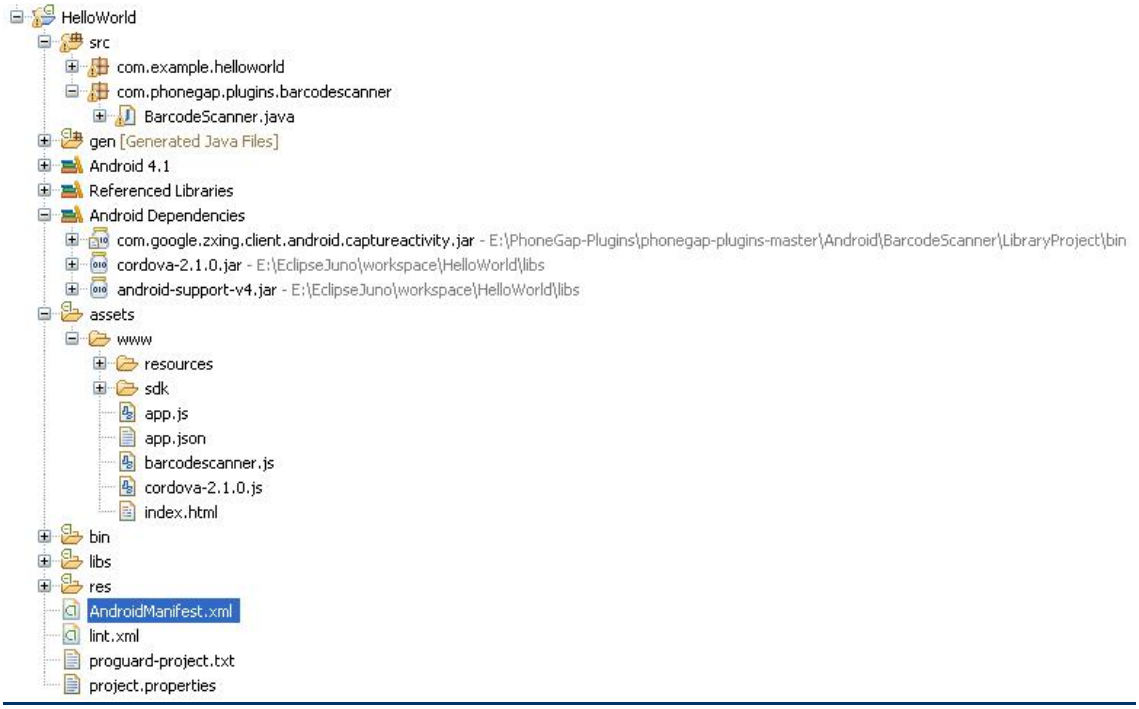


Figura 34: Esquema del projecte utilitzat

- Finalment haurem d' afegir dins de la carpeta on es troba la nostra aplicació <Sencha> el fitxer <barcodescanner.js> i a més afegir una referència a ell dins del fitxer <app.json>.
- Un cop realitzats tots els passos indicats anteriorment ja estarem en disposició d' utilitzar el codi següent:

```
window.plugins.barcodeScanner.scan( function(result) {
        alert("We got a barcode\n" +
                  "Result: " + result.text + "\n" +
                  "Format: " + result.format + "\n" +
                  "Cancelled: " + result.cancelled);
    }, function(error) {
        alert("Scanning failed: " + error);
    }
);
```

Bibliografia

[1] Ext JS 4 First Look (Ed. Packt Publishing).

[2] Ext JS 4 Web Application Development Cookbook (Ed. Packt Publishing).

[3] Learning Ext JS 3.2 (Ed. Packt Publishing).

[4] Sencha Touch Cookbook (Ed. Packt Publishing).

[5] Sencha Touch Mobile JavaScript Framework (Ed. Packt Publishing).

[6] Begining Android Web Apps Development (Ed. Apress).

[7] HTML5 Mobile Development Cookbook (Ed. Packt Publishing).

[8] HTML5 and JavaScript Projects (Ed. Apress).

[9] Pro HTML5 and CSS3 Design Patterns (Ed. Apress).

[10] HTML5 Step by Step (Ed. Microsoft Press).

[11] HTML5 Architecture (Ed. O'Reilly).

[12] HTML5 Geolocation (Ed. O'Reilly)

[13] Teach YourSelf HTML5 Mobile Application Development in 24 Hours (Ed. Sams).

[14] PhoneGap Essentials. Building Cross-Platform Mobile Apps. (Ed. Addison Wesley).

[15] PhoneGap Mobile Web Framework for JavaScript and HTML5 (Ed. Apress).

[16] 20 Recipes for Programming PhoneGap (Ed. O'Reilly).

[17] http://www.phonegap.com/developer

[18] http://www.mobiledevelopersolutions.com/home/start/twominutetutorials/tmt3

[19] http://miamicoder.com/2012/how-to-create-a-sencha-touch-2-app-part-1/

[20] http://senchaexamples.com/

[21] http://www.quizzpot.com/blog/

[22] http://code.google.com/p/google-maps-utility-library-v3/wiki/Libraries

[23] https://developers.google.com/maps/documentation/javascript/tutorial

[24] https://developers.google.com/maps/

[25] http://developer.android.com/develop/index.html

[26] https://github.com/phonegap/phonegap-plugins/tree/master/Android/BarcodeScanner

[27] Help de Sencha