Disseny i implementació d'una base de dades relacional per a un concessionari de vehicles
- Memòria -

Ignasi Fosch Alonso
Enginyeria en Informàtica

Juan Martínez Bolaños

13 de desembre de 2012

Resum

Aquest projecte s'ha desenvolupat aplicant metodologies àgils amb eines compatibles, pensades i triades per al desenvolupament amb el sistema proposat per l'àrea, és a dir, la bases de dades d'Oracle.

Per al desenvolupament s'han aplicat les següents tècniques:

Desenvolupament dirigit per proves

Per cada funcionalitat identificada, es desenvolupa un codi que permetrà confirmar que la funcionalitat que s'implementarà és correcta. Aquests tests es mantenen dins del projecte per validar que tot segueix funcionant correctament a mida que s'afegeixen noves funcionalitats.

Desenvolupament iteratiu

El desenvolupament es realitza en iteracions, de durada determinada i relativament curta, que permeten a l'usuari veure el progrés del desenvolupament, validar la satisfacció de la feina realitzada i introduir canvis en les especificacions i requeriments prèviament definits.

Control de versions

Per mantenir l'ordre en el codi, s'utilitza un sistema de control de versions que permet emmagatzemar, recuperar i comparar les diferències entre dues versions del codi, identificant quins són els canvis que s'han fet entre les situacions del sistema.

Simplicitat i refactorització

Degut a la possibilitat de canvi convé mantenir els dissenys i les implementacions el més simples possible i estar disposat a modificar el que calgui, en cas de ser necessari.

Amb l'aplicació d'aquesta metodologia en el disseny i implementació de la base de dades, es vol demostrar que les eines definides accepten, toleren i, en definitiva, estan preparades per la utilització d'aquestes tècniques que faciliten la satisfacció dels usuaris i clients i que organitzen la feina dels enginyers, incrementant-ne la qualitat dels productes finals.

Taula de contingut

Introducció

Aquest capítol conté la informació general sobre el projecte, organitzada en les següent seccions:

Justificació

Primerament, es presenten la justificació i el context del projecte i el seu desenvolupament, indicant-ne el punt de partida i quina és l'aportació del projecte.

Objectius

En la segona secció, s'expliquen quins són els objectius del projecte.

Enfocament i metodologia

Donada la importància de la metodologia per aquest projecte, aquesta secció la descriu en profunditat.

Planificació

La quarta secció detalla la planificació realitzada, assenyalant els canvis que ha calgut realitzar per ajustar-se a cada una de les situacions que s'han presentat.

Productes obtinguts

Aquesta secció descriu quins són els elements que s'han produït durant el desenvolupament del projecte.

Descripció de la resta de capítols

Finalment, es presenta una descripció de la resta de capítols de la memòria.

Justificació

Des de l'àrea de Bases de Dades, per a la realització del projecte de final de carrera, s'ha proposat el disseny i desenvolupament d'una base de dades relacional per a un concessionari de vehicles.

Aquest projecte contempla la tasca encomanada des de la presa de requeriments fins a l'entrega del codi necessari, tenint present un disseny escalable i plantejant una metodologia àgil que permeti el desenvolupament posterior necessari segons les necessitats dels usuaris.

L'aportació principal del projecte és la definició d'una metodologia adient per al disseny i implementació de bases de dades, que utilitzi tècniques reals i viables per al plantejament d'altres projectes similars.

Objectius

L'objectiu principal del projecte és el disseny i implementació d'una base de dades relacional que satisfaci les necessitats indicades a l'enunciat. Per aquest motiu, inclou la identificació de requeriments, el seu anàlisi i implementació.

Tot i que en un segon pla, un altre objectiu molt important és l'aplicació d'una metodologia que permeti un desenvolupament del projecte adaptat a les necessitats dels usuaris. Aquesta metodologia inclou els plantejaments adients per a garantir la funcionalitat de cada versió.

Finalment, el tercer objectiu més important del projecte és demostrar la viabilitat de les tècniques aplicades.

Enfocament i metodologia

Ja fa uns quants anys que van començar a aparèixer, en el món del desenvolupament de programari, el que s'anomenen metodologies àgils, ja basant-se en els mètodes incrementals, que van començar a aparèixer el 1957 [Larman-Basili, 2003]; continuen evolucionant durant els anys 70, [Edmonds, 1974]; després es viu l'aparició d'algunes tècniques més avançades, com *Scrum* [Takeuchi-Nonaka, 1986] o *Extreme Programming* [Beck, 1999], durant els anys 80 i 90, per acabar amb la declaració de l'*Agile Manifesto* [Beck-et.al, 2001], el febrer de 2001.

A partir d'aquí, apareixen tot un seguit de principis, tècniques i metodologies, que es solen combinar i adaptar a cada cas i situació en la que s'apliquen. És per això que en aquest projecte, s'ha triat per aplicar una combinació de tècniques i mètodes que es detallen en aquesta secció.

Metodologia general

El projecte es desenvoluparà utilitzant desenvolupament guiat per tests (TDD, *Test driven development*, en anglès) i iteracions cícliques, més conegudes com *sprints*, en anglès. Així, en primer lloc s'utilitzarà un llistat de requeriments inicial, que s'anomena *backlog*. Cada un d'aquests requeriments es dividirà en les tasques necessàries per assolir-los, es valorarà una estimació de temps necessari per implementar cada tasca, assignant-les-hi un ordre de prioritat. A partir d'aquest *backlog* ordenat i estimat, s'establiran una sèrie d'iteracions en les que s'aniran implementant, completant el producte resultant. La definició de les tasques que es faran en cada iteració es realitzarà en començar-la, un cop acabada l'anterior, en funció dels costos estimats de les tasques, la prioritat definida pels usuaris i el temps previst per la nova iteració. Aquest temps s'ha establert, inicialment, en 20 hores per iteració, de forma que tenint present que es compten al voltant de 10 hores per setmana, s'hauria d'alliberar una nova versió cada dues setmanes.

Cal indicar que aquestes valoracions i estimacions de temps i costos són aproximades i que poden fer variar el temps dedicat a cada iteració a mida que es progressi en el desenvolupament. Aquest sistema de mesura de recursos s'abstrau de mesures sovint errònies o variables, com desenvolupadors disponibles, temps de treball, línies de codi escrites per hora i d'altres. És evident que és un sistema que pot variar molt segons la situació, però també reflexa clarament la realitat.

Arrancada del projecte

El projecte s'inicia, doncs, amb la definició dels requeriments que s'identifiquen a l'enunciat entregat. Cada un dels requeriments es pren com una història d'usuari (US, *user story*) que explica el que es vol aconseguir, amb el màxim detall possible, però sense entrar en llenguatge formal ni definicions tècniques, és a dir, utilitzant el llenguatge propi de l'usuari.

Per cada US, s'identificaran totes les tasques necessàries per a satisfer la necessitat descrita, afegint-hi una estimació aproximada del temps necessari per a realitzar-la. Finalment, tenint en compte les necessitats i interessos dels usuaris, s'estableix un ordre de realització de les tasques. El resultat final és una llista de tasques que s'anomena *backlog*.

Processos comuns de cada iteració

Al començament de cada iteració, hi ha una primera fase que s'anomena llançament o *kick off*, en la què es revisen les tasques que estan al *backlog* i els possibles canvis que convingui realitzar, segons el que hagi succeït a la iteració anterior i els possibles canvis en les necessitats actuals dels usuaris.

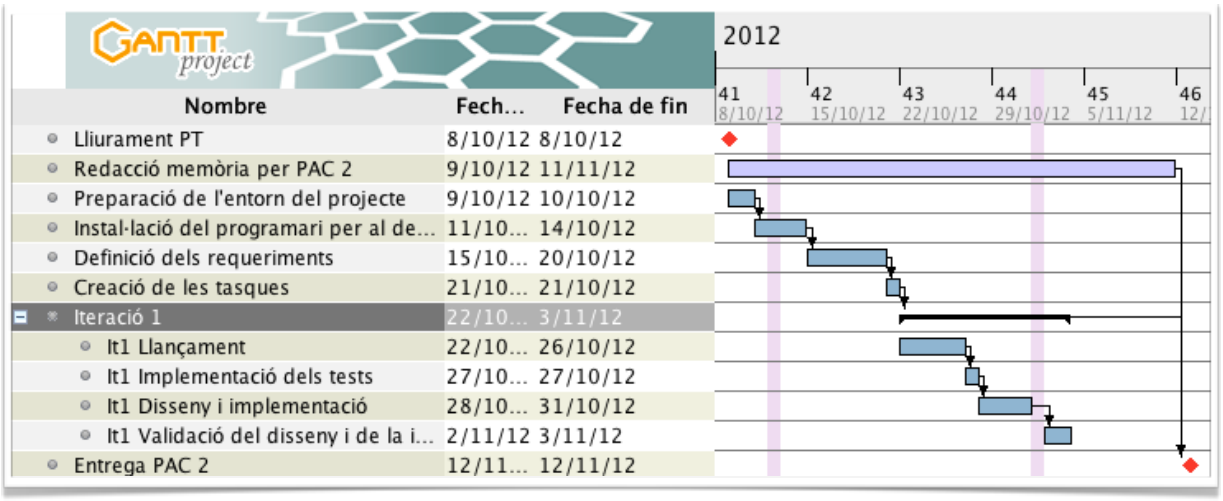
Per cada funcionalitat de la llista associada a l'actual iteració, primer cal implementar tests que serveixin per validar-ne la implementació, i el disseny necessari, i, un cop aquests tests estiguin implementats, es començarà el disseny i la implementació del que sigui necessari per tal de satisfer els tests prèviament implementats.

Per finalitzar la implementació, es comprovarà l'execució completa dels tests implementats, en totes les iteracions, de forma que es validi que totes les funcionalitats implementades segueixen funcionant. Del resultat d'aquests tests i de la feina feta, es realitza un resum retrospectiu que engloba possibles accions de millora, de cara a optimitzar el procés de desenvolupament.

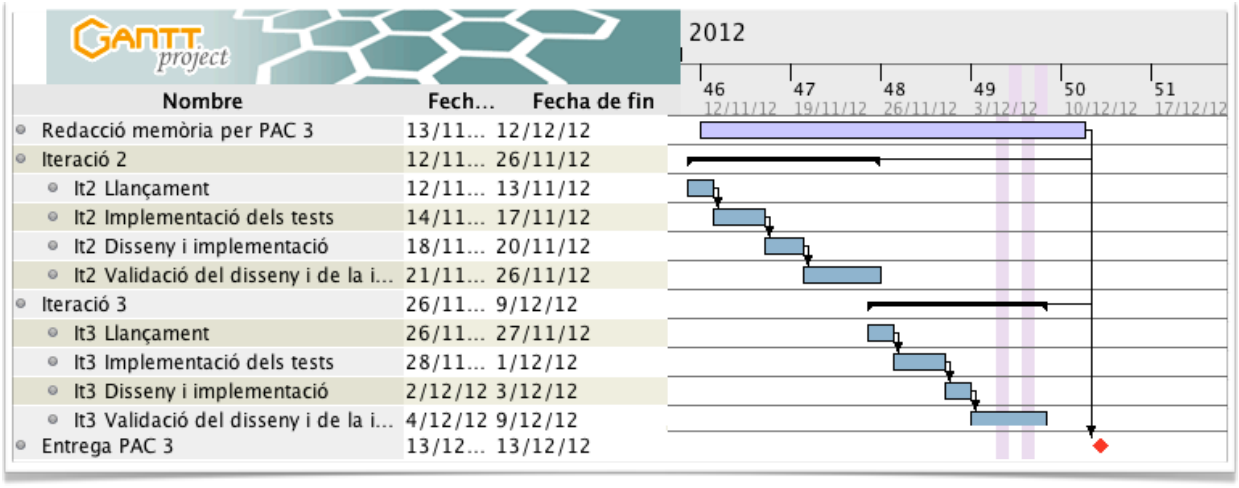
Planificació

Per la realització del projecte, s'han planificat sis fases:

1. Preparatius i arrancada del projecte: En aquesta fase es realitzen les tasques necessàries per trobar les eines i components necessaris per als entorns de desenvolupament i test. L'ultim pas d'aquesta fase és recopilar les US a partir de l'enunciat i dividir-les en les tasques necessàries.

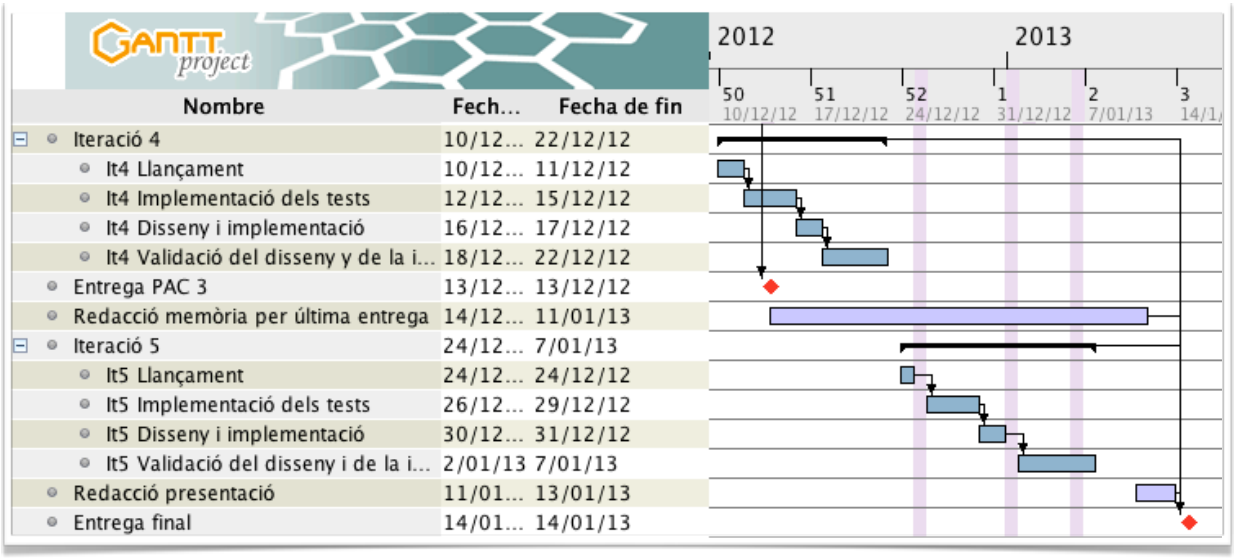


2. Primera iteració: En aquesta primera iteració es realitzaran las tasques que s'assignin a la primera iteració. Una bona part d'aquestes tasques inclouran el disseny i creació de les estructures de dades i codi genèric que s'anirà ampliant en les posteriors iteracions. La finalització de la iteració està prevista per permetre'n incloure-la en l'entrega de la PAC 2.
3. Segona iteració: La segona iteració completarà més tasques corresponents a US definides afegint estructures de dades necessàries,



modificant-ne algunes de les creades en l'anterior iteració i completant codi ja començat en l'anterior iteració. Donat que es modificaran estructures i que es completarà codi preexistent és molt important que es verifiquin les tasques de l'anterior iteració.

4. Tercera iteració: Com l'anterior, aquesta inclourà el disseny i implementació de més tasques, amb els seus tests que s'afegiran a la base de tests per les posteriors iteracions. L'entrega d'aquesta iteració, juntament amb la segona, es preveu per l'entrega de la PAC 3.
5. Quarta iteració: Completant més tasques del *backlog*, es completaran més US i, per tant, més requeriments. També s'inclouran més tests a la base existent.



6. Cinquena iteració: Finalitzant les últimes tasques i preparant l'entrega final. L'entrega final inclourà les dues últimes iteracions.

Productes obtinguts

Els productes obtinguts d'aquest projecte són, principalment, els codis de creació, inicialització i testeig de la base de dades resultant.

Descripció de la resta de capítols

En el capítol *Preparatius del projecte*, es detallen les eines i tasques que, tot i no considerar-se com a part del desenvolupament del projecte, s'han dut a terme.

El capítol *Definició de requeriments* detalla la relació inicial de requeriments que s'han identificat a l'inici del projecte, sobre la qual es construirà la llista de tasques a realitzar, que, al seu temps, és la base del treball de cada fase o iteració.

Per cada iteració realitzada durant el projecte, existeix un capítol que n'explica els seus continguts, el seu disseny i implementació i un breu resum de la retrospectiva.

A continuació, s'inclou una Valoració econòmica, un Glossari, la relació de Bibliografia que s'ha referenciat al llarg de tota la memòria, en forma de llistat.

Finalment, els annexes contenen informació sobre l'estat de desenvolupament de cada iteració, generada a partir de l'eina de gestió que s'utilitza en el projecte i referenciada en la secció corresponent a cada iteració.

Preparatius del projecte

Per a poder treballar còmodament amb les tasques a desenvolupar, es va plantejar la utilització d'una eina de gestió de tasques, anomenada Redmine. Aquesta eina, i la seva utilització, es detalla en l'apartat *Gestió del projecte amb Redmine*.

Així, mateix, per tal de tenir un control acurat de cada canvi que es realitza, es va posar a disposició del projecte un servidor *Mercurial*, en el que es va crear un repositori de codi. Aquests elements es descriuen en més detall a l'apartat *Utilització del repositori del projecte*.

D'altra banda, s'han realitzat les següents tasques preparatives:

- **Anàlisi de les eines de desenvolupament**: Donat que hi ha moltes eines, tant pròpies d'Oracle com d'altres fabricants, per realitzar el desenvolupament, s'ha volgut analitzar-les per conèixer la conveniència d'algunes d'elles. L'apartat *Eines de desenvolupament* tracta sobre aquestes eines.
- **Anàlisi de les eines de comprovació**: Ja que el projecte s'anava a desenvolupar, principalment, amb Test Driven Development, abreujat TDD, es va dedicar un cert temps a considerar diferents opcions i mètodes, que queden detallades a l'apartat *Eines de test*.

Cal assenyalar que aquestes dues tasques, s'han considerat alienes al projecte, ja que formarien part de la part de desplegament de l'equip de desenvolupament i es podrien aprofitar en posteriors projectes o, de fet, podrien estar-hi ja presents d'anteriors projectes. Malgrat tot, també s'ha considerat interessant destacar-ho com a punt d'informació per poder observar l'envergadura de la feina realitzada.

Gestió del projecte amb Redmine

Redmine [Redmine] es una eina de gestió de tasques. Permet crear projectes separats en els què treballar amb tasques i feines organitzant-ne els fluxes, mantenint un conjunt d'usuaris amb rols.

El Redmine del projecte

Per a la realització del projecte, s'ha habilitat, en el servidor del projectista [Redmine Y10K], un projecte de Redmine, anomenat PFC, completament privat, és a dir, del qual no es pot accedir sense un compte expressament permès. Així mateix, s'ha habilitat un compte d'usuari per tal de poder observar les tasques realitzades, amb les següents credencials:

- Usuari: pfc
- Contrasenya: uoc2013pfc

Gestió del projecte

Els projectes de Redmine s'organitzen en tiquets o *issues* que tenen camps i classificacions. Una d'aquestes classificacions, que es sol utilitzar, en Redmine, per identificar els camps de què es disposen, és el que s'anomena *tracker* i, per defecte, Redmine incorpora *Feature*, *Bug* i *Support*.

Per a la correcta aplicació de la metodologia explicada a la secció *Metodologia*, s'han creat els *trackers User story* i *Task* i s'han associat al projecte PFC per a la seva realització.

Així, es mantindran per una banda, els requeriments dels usuaris en forma de *User stories* que es resoldran en *Tasks* associades a la seva corresponent *User story*. Les *Tasks* tindran un parell de camps diferents dels altres *issues*: *Estimation* i *Queue Number*.

El camp *Estimation*, s'utilitzarà per enregistrar la quantitat d'hores estimades que caldrà dedicar a aquesta tasca. El camp *Queue Number* servirà per posar un valor numèric en funció de la prioritat que la tasca tingui i així poder-los ordenar en llistat.

Els *trackers Feature* i *Support* es desactiven del projecte, mentre que es manté el *tracker Bug*, que s'utilitzarà per a registrar els errors que s'hagin detectat que pertoqui corregir. Els *Bugs* seran oberts per corregir detalls que s'han detectat com errors en versions suposadament correctes.

El procés d'anàlisi de requeriments

Les *User stories* són declaracions concretes en llenguatge natural i amb explicacions normals que el client realitza. No necessiten més que ser detallades, però no han de ser representades en llenguatge formal. El que sí que interessa molt és que representin processos concrets o entitats tancades. No es pot representar més d'una funcionalitat en una mateixa història.

Les *User stories*, un cop creades, son analitzades i desglossades en *Tasks* a les que, utilitzant el camp *Estimation*, se les assigna un número de punts en funció del temps que cal invertir. Aquests punts, en la gestió d'aquest projecte, tindran una correspondència directa amb les hores que s'hi esperen dedicar, però es seguirà l'escala de Fibonacci [Tamrakar, 2012].

Per tal de preparar els llistats de tasques a realitzar en cada iteració, un cop estimades, s'assignen valors a *Queue Number*, per definir l'ordre en el què aquelles es resoldran, dins de la iteració.

Kick off de cada iteració

Sobre aquest valor, s'ordena la llista de tasques que hi ha i es separen les tasques, seguint aquest llistat, fins que s'acumulen tants punts del camp *Estimation* com punts es poden fer en una iteració.

Per fer això, existeixen dos llistats predefinits, un que es diu *backlog* i un altre que és el de cada iteració. A més, dins del projecte, per cada iteració es crea una versió. Veient el contingut de la versió, es pot veure quants punts hi ha assignats.

Tancament de cada iteració

Habitualment, en cada tancament, es fa un repàs reflexiu dels problemes que s'han presentat i de com resoldre'ls endavant, anomenat retrospectiva. És freqüent analitzar el nombre de punts de versió en versió, per tal de validar que la capacitat de treball està correctament ajustada.

Utilització del repositori del projecte

Sistemes de control de versions: Centralitzats i distribuïts

Una de les eines tecnològiques que més s'han desenvolupat en els últims anys són els sistemes de control de versions, sobretot degut a la gran quantitat d'aplicacions que s'hi han trobat. Alguns s'han integrat amb d'altres programaris, com els processadors de text (com [Microsoft Community, 2010] i [Apple Support, 2009]), o, fins i tot, amb sistemes d'emmagatzament (com [Novatech, 2012] o [SugarSync, 2012]); però, sobretot, el camp en el què segueixen sent més coneguts, és el del desenvolupament de programari, concretament, per les tasques de gestió de la configuració.

Pel que fa a aquests últims, podem distingir clarament una categorització que depèn, principalment, de la seva arquitectura de funcionament:

- **Centralitzats**: Consisteixen en una instal·lació centralitzada, on es conté el repositori i del que tots els desenvolupadors en depenen. Quan un nou desenvolupador necessita el repositori, es descarrega, mitjançant els protocols previstos pel sistema en qüestió, una còpia de treball, o *working copy*, que es guarda en el seu equip. Amb aquesta còpia, el desenvolupador pot accedir a qualsevol de les versions que hi havia del repositori. Sobre la còpia, el desenvolupador continuarà desenvolupant i haurà de comunicar els canvis que hagi fet al servidor central per tal de què la resta de desenvolupadors puguin revisar-lo en els seus equips.
- **Distribuïts**: En aquest tipus, la instal·lació no és centralitzada, donat que cada desenvolupador té un repositori complet, no només una còpia de treball. La diferència més clara és que entre dos desenvolupadors poden compartir-se les seves versions directament, sense haver de passar per un servidor central.

Mercurial

Mercurial [Mercurial] és un sistema de control de versions distribuït força utilitzat. Es troba disponible per els sistemes operatius més comuns, com Windows, MacOS o Linux [Mercurial Download], i és força senzill d'instal·lar. Per a utilitzar-lo en Windows, hi ha prou amb descarregar-ne la versió corresponent, i fer-ne la instal·lació seguint les instruccions proporcionades [Mercurial WindowsInstall].

Malgrat que la interfície per defecte de Mercurial és en línia de comandes, existeixen eines gràfiques per a interactuar-hi, com el TortoiseHg [TortoiseHg], disponible per Windows, MacOS o Linux, i molt senzill d'utilitzar [TortoiseHg QuickStart].

El repositori del projecte

Les dades d'accés del repositori del projecte, on s'emmagatzema el codi i part de les eines necessàries per al desenvolupament, són les següents:

- URL del repositori: ssh://hg.y10k.ws/pfc
- Usuari: pfc
- Contrasenya: uoc2013pfc

Per tal d'accedir al projecte correctament, cal seguir les instruccions que es donen en el punt següent, *Operacions típiques*, sota el nom *Clonar el repositori*.

Operacions típiques

Clonar el repositori (clone)

Per tal de clonar el repositori, s'ha de triar una ubicació dins de l disc de l'ordinador, i realitzar l'operació de clonació. Per tal d'invocar la operació de clonació del repositori des de la línia de comandes, un cop al directori d'on es vol que pengi, n'hi ha prou amb invocar el següent:

```
hg clone ssh://pfc@hg.y10k.ws/pfc
```

Si es vol clonar en un directori amb un nom específic, es pot afegir com a argument a continuació.

Per tal de fer-ho amb el TortoiseHg, cal anar a la carpeta en la que es vol contenir el repositori local i sobre el fons de la carpeta, desplegant el menú contextual, accedir a l'opció Clone... del submenú TortoiseHg.

En el diàleg que s'obrirà, es poden introduir les dades d'accés al repositori i, prement el botó de Clone, s'iniciarà el clonat.

Cal tenir present que, depenent del contingut del repositori, aquesta funció pot trigar una mica, ja que es pot trobar que ha de descarregar fitxers més o menys grans i en una quantitat possiblement força elevada.

Canviar de versió al repositori (*update*)

Normalment, en el repositori hi haurà diferents versions. Algunes de les versions, a més, estaran etiquetades, per correspondre a alguna fita especial en el desenvolupament del projecte. Així, per exemple, és possible que sigui necessari canviar a la versió etiquetada com It1, des de la versió etiquetada com It2, és a dir, anar enrere. Això es pot fer utilitzant la següent comanda, executant-la des de dins del directori del repositori:

```
hg update It1
```

O bé, amb el TortoiseHg, utilitzant l'opció `Update To...` del submenú desplegable, apareix una finestra que mostrarà una llista de les diferents versions.

És molt important tenir en compte que els fitxers canviaran físicament, per tant es podrien donar corrupcions en aplicacions, com l'Oracle, que estiguessin amb aquests fitxers oberts mentre es fa un *update*. Aquesta operació es realitza de forma local, exclusivament, pel que no hi ha cap comunicació i no hauria de comportar massa temps.

Actualitzar una versió nova del servidor (*pull*)

Un cop ja es disposa d'un repositori local, per tal d'actualitzar-lo amb noves versions, es pot utilitzar la següent comanda, dins del directori del repositori:

```
hg pull -u
```

Això actualitzarà les versions del repositori local amb les del que hi ha al servidor i posarà la còpia local a la última versió disponible. Això implica que si hi ha fitxers grans, l'operació podria trigar més temps de l'esperat. Com que l'opció -u provoca un canvi de versió implícitament, cal tenir present les mateixes implicacions respecte a l'estat dels fitxers, per evitar-ne cap corrupció.

Pel que fa al TortoiseHg, el procediment és molt similar, doncs s'ha d'utilitzar l'opció `Pull...`, el diàleg de la qual permetrà triar si fer l'actualització a la última versió o no.

Eines de test

Per a la validació dels tests s'ha dissenyat una instal·lació d'Oracle virtualitzada que es pot desplegar de forma automàtica, permetent que l'execució dels tests estigui el més automatitzada possible. Aquesta instal·lació es troba inclosa dins del codi escrit pel projecte, però requereix la instal·lació del programari indicat en aquest apartat. A banda d'això, per poder fer el desplegament, requereix connectivitat a Internet.

Oracle VirtualBox

VirtualBox [VirtualBox] és un programari que permet virtualitzar sistemes en un altre equip, de forma que un sistema operatiu es pot executar en el mateix maquinari que un altre, al temps que permet la comunicació entre ells.

Tot i així, no en tots els equips funciona amb la mateixa flexibilitat, depèn de la configuració de maquinari. Aquest programari es pot descarregar gratuïtament i la seva instal·lació no és complexa [VirtualBox Install].

Vagrant

Vagrant [Vagrant] és un conjunt d'eines, desenvolupades en Ruby, que permeten definir i gestionar infraestructura virtualitzada amb VirtualBox. La seva instal·lació és força senzilla [Vagrant Install].

PL/SQL Unit Testing for Oracle (PLUTO)

PLUTO és un conjunt de llibreries de tests unitaris (*unit testing*, en anglès) per PL/SQL i Oracle. Aquestes llibreries aporten mètodes per a crear les situacions a testejar, sense afectar a la base de dades en producció i poder-ne validar els resultats, donant per bo el codi creat. Cal assenyalar que, en un principi, s'havia decidit utilitzar les llibreries utPLSQL, però en la primera iteració es va observar que la versió utilitzada d'Oracle no era compatible i el resultat esperat no era correcte.

Utilització

Per arrencar l'entorn de test, cal entrar, per la línia de comandes, al directori test del projecte i executar `hg serve` i `vagrant up`. En condicions normals, aquesta comanda crearà i arrencarà una màquina virtual amb Ubuntu Lucid on instal·larà l'Oracle. Un cop aquesta màquina hagi arrencat correctament, utilitzant Chef [Chef], s'hi instal·larà i configurarà una Oracle Database XE, instal·lant-hi les llibreries de *unit testing* i el codi desenvolupat al projecte. Per tal de poder completar aquestes tasques, cal tenir accés a Internet.

Per fer els tests, cal entrar a l'Oracle amb l'SQL*Plus utilitzant l'usuari `concessionari`, amb la contrasenya `concessionari`, i executar les següents comandes:

```
SET SERVEROUTPUT ON
SET FEEDBACK OFF
EXEC RUNTESTS
```

Eines de desenvolupament

Per a la realització del projecte, des del punt de vista del desenvolupament, s'utilitzaran les eines descrites en aquest apartat. Per cada eina, es dóna una descripció del motiu per utilitzar-la, una breu explicació de com instal·lar-la i les possibles observacions.

Val la pena que per al desenvolupament, s'utilitza la mateixa instal·lació que la de test, definida en el apartat anterior, Eines de test, de la que n'és una extensió, afegint-hi les eines de desenvolupament.

Oracle Instant Client

L'Oracle Instant Client [OIC Download] l'eina bàsica per interactuar amb l'Oracle, és convenient instal·lar-la a l'estació de desenvolupament. Especialment, perquè inclou l'SQL*Plus que permet treballar en línia de comandes amb l'Oracle.

Aquesta eina és especialment complexa d'instal·lar en MacOS, doncs cal instal·lar la versió de 32 bits, sigui quin sigui el format de l'arquitectura.

Oracle SQL Developer Data Modeler

L'Oracle SQL Developer Data Modeler [Oracle Data Modeler] és una eina de disseny de bases de dades que facilita la implementació dels models, a partir de crear-los. La instal·lació és molt senzilla, només cal seguir els passos que presenta l'instal·lador. Tant mateix, aquesta eina no s'ha arribat a utilitzar fins ara, doncs requereix de la existència d'un diccionari de dades o de certes fonts de les què no es disposa. A més de tractar-se d'una aplicació amb molt consum de processador i memòria.

Oracle JDeveloper

JDeveloper [Oracle JDeveloper] és una suite de desenvolupament per Java i Oracle, que permet desenvolupar en PL/SQL amb facilitat. En aquest cas, la instal·lació és molt senzilla, ja que es tracta d'un fitxer JAR que conté el programa. Requereix que hi hagi un entorn d'execució Java a l'equip on s'executarà. Malgrat que, com l'anterior, aquesta eina consumeix massa recursos, s'ha fet servir, però només en algun equip en què no es disposava de l'SQL*Plus i no hi havia forma d'instal·lar-lo.

Definició dels requeriments

A partir de l'enunciat del projecte, s'han considerat quatre històries d'usuari prou concretes que es detallen en els apartats d'aquesta secció. Totes aquestes històries d'usuari s'han introduït, tal i com estan aquí descrites al projecte en Redmine per a la seva gestió dins de les iteracions posteriors.

Com es veurà al llarg del projecte, la idea és que cada història d'usuari es dissenyi i desenvolupi en iteracions separades, però només es desenvoluparan les funcions i estructures necessàries per satisfer les necessitats de la història d'usuari corresponent.

Per demostrar l'escalabilitat del disseny i com la metodologia respon als canvis en les necessitats inicials, eventualment durant les iteracions, es presentaran alguns canvis en les necessitats que provocaran canvis en el disseny i en la implementació de la base de dades resultant.

Pel que fa al desenvolupament del sistema de registre de les accions preses mitjançant la base de dades i a les estadístiques necessàries, també s'implementaran dins de cada iteració.

Compra d'un vehicle

Les botigues del concessionari poden realitzar compres de vehicles, podent-se tractar de vehicles de primera mà o de segona mà, tenint present que els vehicles de primera mà, poden no tenir assignada cap matrícula.

De la botiga es registrarà el nom, l'adreça, el telèfon, el fax i el correu electrònic.

La compra es realitza a, en el cas dels vehicles de primera mà, la fàbrica de la marca del concessionari, o, en el cas dels vehicles de segona mà, a venedors independents, dels quals se'n guardarà el nom, l'adreça postal, el telèfon, el fax i el correu electrònic. En el cas de la fàbrica, es considerarà un venedor més, amb les mateixes dades.

El vehicle comprat tindrà associat un número de chassis o bastidor, que seguirà el format internacional definit [VIN], la marca, el model, la motorització, el color, l'equipament extra de què disposi, i la matrícula, en cas que en tingui.

A més, de cada compra, cal enregistrar l'empleat de la botiga que ha efectuat la compra, la data de compra, l'import i si l'adquisició és nova o de segona mà.

Pel que fa a les estadístiques, es volen saber quants vehicles s'han comprat per botiga, marca del vehicle, quants de primera i quants de segona mà; també es volen conèixer les compres per empleat.

Venda d'un vehicle

Qualsevol botiga del concessionari ha de poder vendre vehicles, ja siguin nous o de segona mà, als clients que ho demanin. Evidentment, la botiga ha d'haver comprat prèviament el vehicle. En cas que sigui nou, al vehicle se li ha d'assignar una matrícula.

L'operació de venda la realitzarà un comercial de la botiga, el qual ha de quedar registrat. Del client caldrà guardar el nom, l'adreça postal, el telèfon, el fax i el correu electrònic. A més, es guardarà la data de venda, i l'import. En cas que el vehicle sigui de segona mà, caldrà guardar el quilometratge que tenia en vendre'l.

Respecte a les estadístiques, es vol saber quants vehicles s'han venut, en total, per botiga, per comercial, i quants n'eren de primera mà i quants, de segona.

Revisió d'un vehicle

Totes les botigues disposen d'un taller al qual els clients poden dur els seus vehicles comprats en qualsevol botiga, per a fer-los revisions mantenint, així, la garantia. Aquestes revisions les executarà un mecànic al taller de la botiga, les dades del qual quedaran enregistrades, així com les dates i hores d'entrada i sortida del vehicle i l'import que se li ha cobrat al client.

Les estadístiques més significatives en aquest cas són el nombre de revisions realitzades, per taller i mecànic, i el temps que han trigat en completar-se.

Reparació d'un vehicle

Així mateix, els clients poden dur els seus vehicles a reparar al taller de qualsevol botiga. S'emmagatzemarà la mateixa informació que per una revisió, afegint-hi el motiu de la reparació.

Les estadístiques que interessen són les mateixes que en el cas de les revisions.

Matriculació

Tot just després del desenvolupament de la segona iteració, concretament, després d'haver completat les tasques per la història d'usuari sobre les vendes, es va observar que l'aplicació permetia vendre vehicles sense matricular, cosa que no s'havia especificat en un principi.

La qüestió és que s'ha de controlar que tot cotxe que es vengui estigui matriculat i que mai es matriculin dos cotxes amb la mateixa matrícula.

Gestió de personal

Després de les tres primeres iteracions, es va observar que s'havia desenvolupat tot més ràpid del que s'esperava i es va decidir afegir algunes històries d'usuari més, com aquesta.

La història d'usuari defineix que el concessionari pot contractar nous empleats o, també, acomiadar-los, per tant cal que l'aplicació contempli aquesta possibilitat. S'ha de tenir present que un empleat acomiadat no pot realitzar cap de les tasques indicades, cas en el que l'aplicació ho ha d'indicar amb un missatge d'error.

Gestió de tendes

Una altra de les històries d'usuari afegides durant el desenvolupament del projecte, molt similar a la de gestió d'empleats, però consisteix en poder indicar quan s'han inaugurat noves tendes o se n'han clausurat d'existents. El principal detall a tenir present és que tots els empleats d'una tenda clausurada s'assignaran a una altra tenda.

Avisos de revisió

L'última de les històries d'usuari que s'han afegit durant el projecte, consisteix en poder obtenir un llistat dels clients, i les seves dades de contacte, que tenen cotxes que en algun moment han passat per la botiga, ja sigui per venda, reparació o revisió, i que fa més d'un any que no han passat la revisió anual. Cal tenir present que aquesta revisió s'ha de fer a tots els cotxes que s'hagin venut fa més de quatre anys, i s'ha de fer cada any.

Iteració 1

En aquest capítol es detalla com ha anat la primera iteració, una de les més complicades en tractar-se la creació de totes les estructures inicials, provar el sistema de tests, i tota la infraestructura que s'utilitzarà.

Kick off

La primera iteració té els següents objectius:

- ➡ Completar les tasques assignades.
- ➡ Iniciar la línia de base del projecte que s'anirà completant en les properes iteracions. Aquesta línia de base inclou el disseny d'estructures que es reutilitzaran, com les que donen suport al registre o al magatzem de dades, i les eines per al desenvolupament, testeig i desplegament, com els guions d'instal·lació o les pròpies receptes d'instal·lació, que es podran aprofitar per desplegar la plataforma de producció.
- ➡ Validar i aprendre a utilitzar la llibreria de *unit testing*.
- ➡ Confirmar l'aproximació de la velocitat d'implementació per iteració.

Sobre les tasques del *backlog*, s'han triat totes les tasques que corresponen a la US de la compra de vehicles, però en l'ordre que permet la seva implementació ordenada. A continuació, s'inclou una llista de les tasques seleccionades:

#	Titol	Creada	Queue Number	Estimation
48	Identificar les entitats implicades a la compra d'un vehicle	10/25/12 8:23	10	1
49	Identificar i prototipar els processos implicats en la compra d'un vehicle	10/25/12 8:23	20	1
52	Ampliar el disseny per incloure'n un registre de les accions preses per l'aplicació	10/25/12 8:23	21	1
50	Implementar els tests per a la compra d'un vehicle	10/25/12 8:23	30	3
53	Implementar tests per validar el registre de les accions de l'aplicació	10/25/12 8:23	31	3
55	Dissenyar els elements necessaris per les estadístiques de les compres de vehicles	10/25/12 8:24	32	1
51	Implementar la compra de vehicles	10/25/12 8:23	40	3

54	Implementar el registre d'accions de l'aplicació	10/25/12 8:23	41	3
56	Implementar les validacions de les estadístiques de la compra de vehicles	10/25/12 8:24	42	3
57	Implementar les estadístiques de la compra de vehicles	10/25/12 8:24	50	3

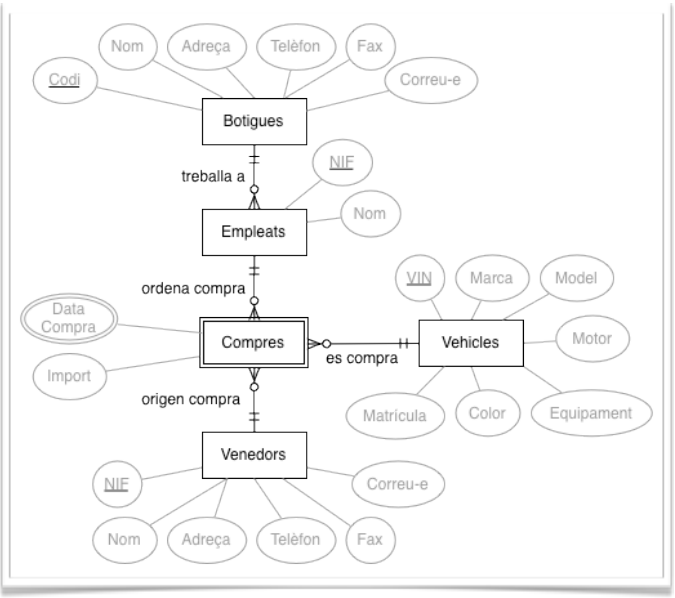
Casos d'ús

De cara a la lògica de negoci, la història d'usuari que s'està satisfent reflexa els següents casos d'ús:

- **Compra d'un vehicle**: Qualsevol empleat, de qualsevol botiga, pot fer la compra d'un vehicle, que quedarà registrat.
- **Addició d'un nou venedor**: Per a poder comprar vehicles de segona mà, es contempla la possibilitat de que es puguin afegir nous venedors.
- **Registre d'accions**: Tots els processos que es realitzin, han de quedar reflexats en el registre d'accions.
- **Execució de l'ETL**: Ja sigui de forma periòdica o puntual, cal que es pugui executar el procés de càrrega del magatzem de dades de forma automatitzada.

Disseny

S'han identificat les següents entitats:



Botigues

És l'entitat que representa cada una de les botigues del concessionari. Com indica la US, es consideren les dades següents, a banda d'un codi enter identificatiu únic, que serà el camp clau:

- Nom: Consistirà en una cadena de caràcters de 100 caràcters com a màxim.

- Adreça: Consistirà en una cadena de caràcters amb longitud màxima de 200.
- Telèfon: Consistirà en una cadena de caràcters de 9 caràcters com a màxim. Només s'han considerat numeracions nacionals.
- Fax: Consistirà en una cadena de caràcters amb les mateixes consideracions que el telèfon.
- Correu-e: Consistirà en una cadena de caràcters de 50 caràcters com a màxim.

Empleats

Es considera necessari poder identificar a l'empleat que realitza la compra com una entitat pròpia. A més, la utilització d'aquesta entitat permet simplificar la definició de la relació que representen les compres, que es defineix a l'últim punt d'aquesta enumeració. Es consideren els següents camps:

- NIF: Consistirà en una cadena de caràcters de màxim 9 caràcters, considerant la numeració del NIF. Serà el camp clau.
- Nom: Consistirà en una cadena de caràcters de 200 caràcters com a màxim.
- Codi Botiga: Necessari per implementar la relació 1:N amb l'entitat Botigues, per tant, consistirà en un enter.

Vehicles

Representarà cada un dels vehicles. Per a la identificació del vehicle s'ha triat el *VIN*, o número de bastidor, ja que no tots els vehicles han de tenir matrícula. En cas que el vehicle no tingui vehicle, la compra es considerarà de primera mà; si, pel contrari, en té, aleshores, la compra s'haurà de considerar de segona mà. L'enumeració de dades recollides és la següent:

- VIN: Consistirà en una cadena de caràcters de 17 caràcters, basant-se en la definició que es dóna a la Viquipèdia
- Matrícula: Consistirà en una cadena de caràcters de longitud màxima de 7 caràcters, ja que només s'han considerat matrícules nacionals. No és obligatòria la seva presència.
- Marca: Consistirà en una cadena de caràcters de, com a màxim, 30 caràcters.
- Model: Consistirà en una cadena de caràcters amb una longitud màxima de 50 caràcters.

- Motor: Consistirà en una cadena de caràcters de 50 caràcters com a màxim.
- Equipament: Consistirà en una cadena de caràcters de 200 caràcters de mida màxima.
- Color: Consistirà en una cadena de caràcters amb un màxim de 20 caràcters.

Cal destacar que no s'ha considerat, ja que no s'ha indicat a la història d'usuari, la possibilitat d'estandarditzar la descripció del vehicle, a nivell de marca, model, motor, color, equipament.

Venedors

Aquesta entitat representa l'individu, empresa o institució a la que se li compra un nou vehicle i se'n consideren els següents camps:

- NIF: Consistirà en una cadena de caràcters de màxim 9 caràcters, considerant la numeració del NIF. Serà el camp clau.
- Nom: Consistirà en una cadena de caràcters de 200 caràcters com a màxim.
- Adreça: Consistirà en una cadena de caràcters amb longitud màxima de 200.
- Telèfon: Consistirà en una cadena de caràcters de 9 caràcters com a màxim. Només s'han considerat numeracions nacionals.
- Fax: Consistirà en una cadena de caràcters amb les mateixes consideracions que el telèfon.
- Correu-e: Consistirà en una cadena de caràcters de 50 caràcters com a màxim.

Com que la compra quedaria representada per una relació a tres bandes entre Empleats, Vehicles i Venedors, cal implementar una entitat explícita per representar-la:

Compres

La clau de la qual estarà composta pels NIFs de l'Empleat i del Venedor, el VIN del Vehicle i la Data de la compra. Considera els següents camps:

- NIF Empleat: Consistirà en una cadena de caràcters de màxim 9 caràcters, considerant la numeració del NIF.
- NIF Venedor: Com l'anterior.

- VIN Vehicle: Consistirà en una cadena de caràcters de 17 caràcters, basant-se en la definició que es dóna a la Viquipèdia
- Data de la compra: Serà un camp de tipus data que no contindrà l'hora, doncs no cal.
- Import: Serà un camp de tipus numèric amb dos decimals, que representarà l'import de la compra en Euros.

Pel que fa als processos necessaris:

Addició d'un nou venedor

Utilitzat únicament en el cas que s'hagi d'incloure un nou venedor al que se li està comprant un cotxe. El seu prototip seria:

```
nouVenedor(NIF, Nom, Adreça, Telèfon, Fax, Correu-e)
```

Compra d'un nou vehicle

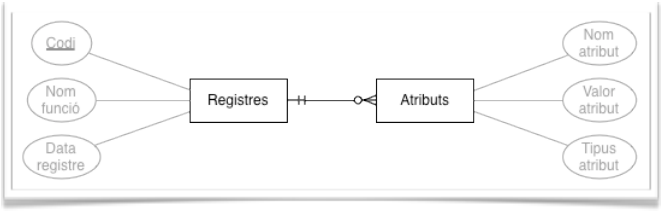
Realitzarà la compra efectiva. El seu prototip és:

```
compra(NIF Empleat, NIF Venedor, VIN, Matrícula, Marca, Model, Motor, Equipament, Color, Data de la Compra, Import, Quilòmetres)
```

El camp dels quilòmetres és opcional, si no s'inclou, es considerarà 0, és a dir, nou, per defecte.

No es considera que la compra d'un vehicle hagi de proveir la incorporació d'un nou empleat ni l'addició de noves botigues, per tant, els tests hauran de proveir-ne les dades d'aquestes entitats explícitament. Aquesta consideració no s'aplica per als venedors ja que es considera el cas en què s'estigui fent una compra a algú que ha dut un vehicle per vendre'l per primer cop, per exemple.

Per la inclusió d'un subsistema de registre d'accions de les aplicacions, s'han considerat les següents entitats:



Registres

És l'entitat que representa cada registre. Consta dels següents camps:

- Codi: És el camp clau, consistent en un nombre enter incrementat automàticament.

- Nom funció: És el nom de la funció que ha provocat el registre i es representa com una cadena de caràcters amb longitud màxima de 100 caràcters.
- Data registre: És el moment en què s'ha produït el registre, ha d'incorporar la màxima precisió possible.

Atributs

És l'entitat que representa diferents atributs per cada registre. S'ha considerat que, com hi haurà diferents atributs implicats depenent de l'acció que s'estigui registrant, convé permetre indicar una llista clau-valor d'atributs. La clau es compondrà de l'identificador del registre al què correspon i del nom de l'atribut. Conté els següents camps:

- Codi Registre: És el camp clau que identifica el registre, per tant és un enter.
- Nom: Representa, amb una cadena de caràcters de 100 octets de longitud, el nom de l'atribut que representa.
- Valor: Conté el valor en una cadena de caràcters de 200 octets de longitud.
- Tipus: Conté el nom del tipus de valor, en una cadena de caràcters de 200 octets de longitud.

Caldrà d'implementar la següent operació:

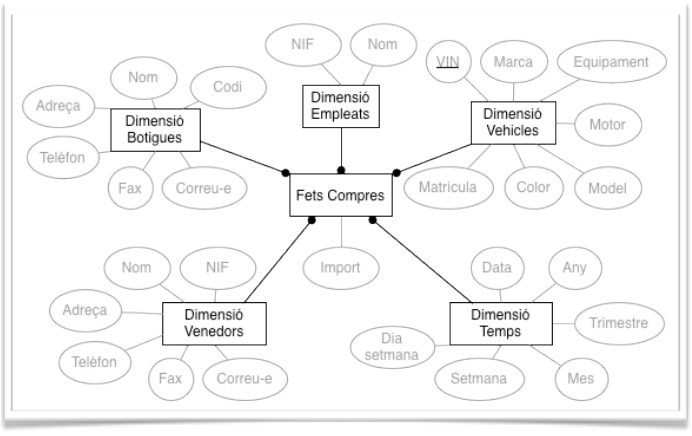
Addició d'un nou event

Enregistrarà una acció que s'ha realitzat, amb el següent prototip:

```
registra(Nom funció, Atributs)
```

Finalment, queda el magatzem de dades. Aquesta tasca implica crear els cubs OLAP utilitzant tècniques ROLAP, ja que l'Oracle és relacional i no multidimensional.

Pel que fa al disseny, s'ha triat utilitzar un esquema d'estrella per simplicitat. Se pot donar el cas que, més endavant, es veiés necessari, per exemple, posar categorització en diferents nivells.



S'ha considerat crear una taula de fets per les compres

i s'han considerat les dimensions venedors, botigues, empleats, vehicles i temps. Pel que fa a la granularitat de la dimensió de temps, es considerarà diària. A més, també amb granularitat diària, es consideren taules agregades per resoldre els dubtes més generals, si bé, fins a aquest moment, no s'ha definit cap dubte sobre les compres.

El model físic que s'ha definit és basa en els camps del model físic OLTP de l'aplicació. Val a dir que s'emmagatzemen, per totes les dimensions, els mateixos camps amb els mateixos formats.

Una de les principals diferències és que les claus primàries del model OLTP no es consideraran vàlides, només les que tinguin algun significat (com els NIF o el VIN) es mantindran en el model OLAP, però ja no com a camps clau. Per a totes les dimensions, els camps clau seran numèrics generats automàticament. El model físic resultant és com segueix:

Dimensió Empleats:
- **Id**
- NIF
- Nom

Dimensió Botigues:
- **Id**
- Nom
- Adreça
- Telèfon
- Fax
- Correu-e

Dimensió Venedors:
- **Id**
- NIF
- Nom
- Adreça
- Telèfon
- Fax
- Correu-e

Dimensió Vehicles:
- **Id**
- VIN
- Matrícula
- Marca
- Model
- Color
- Motor
- Equipament

Dimensió Temps:
- **Id**
- Data
- Any
- Trimestre
- Mes
- Setmana
- Dia de la setmana

Fets Compres:
- **Id Empleat**
- **Id Botiga**
- **Id Venedor**
- **Id Vehicle**
- **Id Temps**
- Import

Pel que fa a processos, només hi haurà un, `etl`, que no rebrà cap paràmetre, però s'encarregarà de comprovar quines accions del registre no s'han integrat al magatzem de dades i les afegirà, utilitzant altres procediments especialitzats.

Creació dels tests

Per la creació dels tests relacionats amb les compres de vehicles, ha calgut començar per crear part del model:

- S'ha creat un *tablespace* per la base de dades anomenat `concessionari`. Cal indicar que s'ha configurat aquest *tablespace* per permetre'n el creixement automàticament.
- S'ha creat l'esquema `concessionari`, amb la contrasenya `concessionari` i s'atorguen els permisos necessaris per poder-se connectar, associat amb el *tablespace* recent creat.
- Connectant-se al nou esquema, es creen les taules.
- Per implementar la clau primària amb increment automàtic de la taula `botigues`, es crea una seqüència que va generant els identificadors i un *trigger* que actua en els *inserts* sobre la taula, donant-li el següent valor de la seqüència al camp numèric `codi`.
- El camp `codiBotiga` de la taula `empleats` és una clau forània que fa referència al camp `codi` de la taula `botigues`.
- La taula compres té les següents peculiaritats:
  - El camp `nifEmpleat` és una clau forània que referencia al camp `nif` de la taula `empleats`.
  - El camp `nifVenedor` és una clau forània referent al camp `nif` de la taula `venedors`.
  - El camp `vinVehicle` és una clau forània que fa referència al camp `vin` de la taula `vehicles`.
  - La clau primària, que s'anomena `pk_compres`, està composta pels camps `nifEmpleat`, `nifVenedor`, `vinVehicle` i `data`.

Per tal de facilitar-ne la gestió del model, s'han preparat uns guions SQL que permeten crear, esborrar i reiniciar el model. Es troben al directori `code`, s'anomenen `create.sql`, `delete.sql`, `reset.sql` i es poden invocar des de l'SQL*Plus de la forma següent:

```
SQL> @/camí/al/repositori/code/create.sql
```

Tot seguit, per la implementació dels scripts de test, s'ha fet servir la llibreria PLUTO per implementar els tests, utilitzant els procediments descrits. El codi resultant, ha quedat escrit en un fitxer dins del directori de codi, que es pot carregar al SQL*Plus amb la següent instrucció:

```
SQL> @/camí/al/repositori/code/tests/tasca50.sql
```

Igualment, per la creació dels tests del registre d'accions, primer s'ha creat el model, ampliant el codi que hi ha a `create.sql` amb el codi necessari per crear les taules registres i atributs, creant una nova seqüència i disparador pel camp `codi` de `registres`. La taula `atributs` s'ha creat tenint present la clau forània `codiRegistre` i la clau primària composta per `codiRegistre` i `nom`.

Tot seguit, s'ha vist que la funció `log`, en realitat, ja existeix, així que s'ha substituït per `registra`, amb el mateix prototipus i s'ha implementat el test tenint en compte aquest detall.

Val a dir que per a la utilització del segon paràmetre, es preveu crear una taula PL/SQL, com es descriu a [Tablas PL/SQL], a la què s'afegeixen tots els atributs.

Per tal d'implementar els tests de les estadístiques en condicions, en primer lloc s'han afegit, al guió `create.sql`, les estructures detallades al disseny de les estructures per les estadístiques. Aquestes estructures reuneixen les següents característiques:

- El model pel magatzem de dades s'ha dissenyat en un esquema separat.
- Totes les dimensions i la taula de fets tenen un identificador numèric incrementat automàticament separat dels que s'utilitzen en l'aplicació.
- Com no s'han definit consultes concretes per les compres, s'ha considerat que es comptaran les compres únicament, però tampoc s'han considerat taules agregades. En tractar les posteriors històries d'usuari, es podrien detectar més canvis addicionals que comportin noves estructures.

Finalment, s'ha implementat el test bàsic plantejat per les estadístiques de les compres.

Implementació

Pel que fa a la satisfacció de la funcionalitat bàsica de les compres, s'han implementat els procediments `compra` i `nouVenedor`, que satisfan els tests anteriorment implementats.

El procediment `compra`, senzillament, crea un nou registre a la taula de compres amb les dades corresponents, les que s'han rebut per paràmetre, i registra l'acció al registre de l'aplicació.

El procediment `nouVenedor`, com en el cas anterior, insereix el venedor a la taula corresponent, amb les dades rebudes per paràmetre, i registra l'acció al registre de l'aplicació.

En primer lloc, per codificar el procediment `registra` s'han creat dos tipus nous dins de l'esquema `concessionari`: `atribut` i `tAtributs`, que són, respectivament, un objecte i un vector, `VARRAY`, d'`atribut` que permeten gestionar els paràmetres del procediment. En variar el prototip del procediment, ha calgut modificar el test implementat, però, un cop corregit, s'ha pogut validar la implementació realitzada.

Aquesta forma aleatòria de passar paràmetres ha sigut necessària en descobrir que l'Oracle no permetia la utilització d'una taula SQL, com es pretenia. En trobar aquest obstacle, es va provar fer-ho d'aquesta altra forma.

D'altra banda, també s'han modificat els procediments `compra` i `nouVenedor` per què en facin ús del registre d'accions.

Per a implementar l'ETL del magatzem de dades, s'han creat els següents procediments i funcions:

> Procediment `etl`: Aquest procediment és l'encarregat de dirigir el procés ETL principalment. Per fer-ho, primer busca al registre d'accions l'últim cop que es va executar l'`etl`. Si no troba cap entrada d'execució, considerarà que ha de llegir tot el registre des de la primera entrada. Aleshores, comença a recórrer cada entrada del registre, des de l'últim registre trobat o l'inici. Per cada registre que troba, mira el nom de la funció registrada i, si és `compra`, crida el procediment `etlCompra`, passant-li l'identificador del registre. No es comproven les crides de `nouVenedor`, perquè així només es carreguen els venedors que han fet alguna compra. Finalment, registra aquesta execució de l'`etl` al registre d'accions, cosa que permetrà identificar el punt de càrrega en la següent execució.

Procediment `etlCompra`: Aquest procediment s'encarrega de carregar les dades d'un event de `compra`, detectat pel procediment etl en llegir del registre un event que correspon. El seu funcionament, es basa en recollir cada atribut del registre que se li ha indicat per paràmetre i, utilitzant les funcions adients, assegurar-ne la presència de les dades al magatzem i anotar-lo per a, finalment, crear la fila corresponent a la taula de fets de compres. El procediment coneix 5 atributs possibles: El NIF del client, el del venedor, el VIN del vehicle, la data de compra i l'import. Pels quatre primers, s'utilitzen les funcions creades àdhuc per a gestionar les dimensions relacionades; el cinquè atribut, es mantindrà dins de la taula de fets. Si es recuperés algun atribut que no és cap d'aquests, es continuaran llegint la resta, però es mostrarà un missatge informant que es desconeix l'atribut. Un cop s'han recollit tots els atributs, les dades recollides s'insereixen en la taula de fets.

Funció `etlNovaData`: Aquesta funció s'encarrega de comprovar i assegurar que la data, en format `VARCHAR2`, que s'indica per paràmetre s'emmagatzema a la dimensió de dates (`dimTemps`) correctament, i en retorna el registre resultant. En primer lloc, crea un registre en memòria de la dimensió, transformant la data en la cadena de caràcters en cada un dels atributs de la dimensió. Després tracta de recuperar-ne un d'idèntic de la taula de dimensió i el retorna com a resultat de la funció. Si no existeix cap d'idèntic, l'insereix i el recupera.

Funció `etlNouVehicle`: Aquesta funció serveix per mantenir la dimensió de vehicles actualitzada, comprovant i assegurant que existeix un vehicle amb el VIN que es rep per paràmetre. El que fa és recuperar les dades del vehicle corresponent a aquest VIN de la base de dades OLTP. Com pot ser que a la base de dades OLTP la matrícula tinguin valor nul, utilitza la funció `etlNullVarchar2` per convertir-lo en una cadena d'un sol espai. Tot seguit, mira de recuperar un registre de la dimensió de vehicles que coincideixi amb les dades corresponents per retornar-lo com a resultat de la funció. Si no es troba cap registre coincident a la dimensió, n'insereix un de nou, i el retorna.

Funció `etlNouVenedor`: Aquesta funció serveix per assegurar-ne la presència del venedor amb el NIF indicat per paràmetre. Recupera, de la base de dades transaccional, la resta de dades corresponents al venedor, i els consulta a la dimensió, per retornar-ne el resultat. Si no

troba les dades correctes a la dimensió, els inserirà i després els retornarà igualment.

Funció `etlBotigaEmpleat`: Aquesta funció s'encarrega de mantenir les dades relatives a botigues en la dimensió, que vinguin relacionades amb l'empleat indicat per paràmetre. Actualitza un registra en memòria amb les dades OLTP de la botiga que està relacionada amb l'empleat corresponent, i busca un registre que concordi a la dimensió de botigues per retornar-lo com a resultat. Si no troba cap registre a la dimensió, l'inserirà.

Funció `etlNouEmpleat`: Aquesta funció mantindrà les dades dels empleats en el magatzem. Recupera les dades de l'empleat al sistema OLTP, per comprovar si hi són al magatzem, retornant-les en cas de trobar-les. Si no les troba, les insereix i les retorna.

Funció `etlNullVarchar2`: Aquesta funció serveix per convertir un valor nul en una cadena de caràcters amb un sol espai. Comprova si el valor rebut per paràmetre és un valor nul. Si ho és, retorna una cadena de caràcters contenint només un espai. Si no ho és, retorna el valor que contingui.

Validacions

Per l'execució de la màquina virtual, cal:

- Arrencar el Mercurial localment: En un terminal, dins del directori on es tingui el repositori, executar `hg serve`.
- Arrencar la màquina virtual: En un altre terminal, entrar al directori test del projecte i executar `vagrant up`.

Aquesta última operació inicia el procés d'arrencada, configuració i instal·lació, que triga aproximadament 15 minuts, depenent si s'ha de descarregar els paquets d'instal·lació d'Oracle o ja els té, o si s'ha arrencat prèviament i s'ha parat o s'ha esborrat.

Un cop la màquina virtual s'ha arrencat i instal·lat, es pot accedir i entrar a l'Oracle amb les següents comandes:

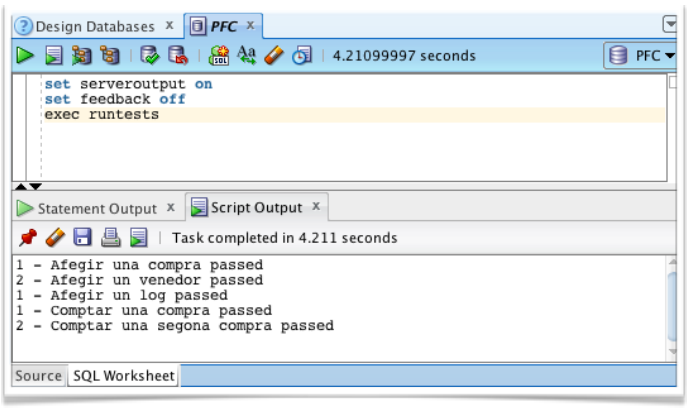
```
$ vagrant ssh
$ su - oracle
$ sqlplus concessionari/concessionari@XE
SQL> set serveroutput on
```

```
SQL> set feedback off
SQL> exec runtests
```

I el resultat és:

```
1 - Afegir una compra passed
2 - Afegir un venedor passed
1 - Afegir un log passed
1 - Comptar una compra passed
2 - Comptar una segona compra passed
```

De forma similar, es pot comprovar amb el JDeveloper:



És a dir que tots els tests implementats són correctes.

Retrospectiva

En primer lloc, cal dir que la tria inicial de llibreria de *unit testing* inicial ha estat un error, ja que no era compatible i ha calgut perdre força temps en trobar i provar un alternativa funcional, que ha resultat ser el PLUTO.

Un altre problema ha estat no tenir una forma de provar-ho tot directament. El principal obstacle és que la màquina virtual fa anar molt lent l'equip de treball. Per la realització del projecte potser no és un impediment, però per un desenvolupament continuat caldria un altre infraestructura, potser utilitzant servidors virtualitzats.

Pel que fa a la velocitat de l'*sprint*, s'han realitzat 28 punts de 25 que hi havien estimats. Donat que no és gaire diferència, es pot entendre que l'estimació ha estat força ajustada.

A posteriori de l'entrega, ja durant la implementació de la següent iteració, s'han observat dos errors força importants:

- **Dimensió Temps del magatzem de dades**: S'ha observat que malgrat que el disseny del magatzem de dades indicava que hi havia una dimensió Temps, aquesta no s'ha implementat. El principal motiu d'això ha estat que en el moment d'implementar els tests, s'ha descartat per error, i que no s'ha comprovat el detall de la tasca, ni de cap de les següents relacionades. Per evitar això, s'ha de procurar detallar millor les tasques que s'han de realitzar. El problema puntual, però, s'ha resolt afegint l'estructura corresponent i el procediment per carregar-lo.
- **No hi ha cap forma d'extreure les estadístiques**: Finalment, s'ha vist que, tot i que hi ha un magatzem de dades i que s'hi afegeixen dades, no hi ha una forma senzilla d'accedir a les dades. El problema s'ha resolt afegint-hi unes vistes que permeten obtenir les estadístiques desitjades.

Iteració 2

En aquest capítol es detalla com ha anat la segona iteració, en la que s'ha implementat la funcionalitat de venda de vehicles, fixant i assentant els processos de desenvolupament que s'han iniciat a la iteració anterior.

Kick off

Aquesta iteració té els següents objectius:

- ➡ Completar les tasques assignades.
- ➡ Completar la línia de base del projecte, afegint i modificant tot el que calgui per completar les tasques assignades.
- ➡ Per procurar evitar la lentitud de l'equip de treball, s'ha instal·lat tot l'entorn de testeig en un altre equip al què s'accedeix remotament i s'executen els tests.
- ➡ Verificar la relació entre l'estimació i el temps dedicat.

Sobre les tasques del *backlog*, s'han triat totes les tasques que corresponen a la US de la venda de vehicles, però en l'ordre que permet la seva implementació ordenada. A continuació, s'inclou una llista de les tasques seleccionades:

#	Titol	Creada	Queue Number	Estimation
58	Identificar les entitats implicades en la venda d'un vehicle	10/25/2012 08:26	51	1
59	Identificar i prototipar els processos implicats en la venda d'un vehicle	10/25/2012 08:27	60	1
65	Verificar el subsistema de registre d'accions per la història de venda d'un vehicle	10/25/2012 09:05	61	1
62	Dissenyar els elements necessaris per les estadístiques de les vendes de vehicles	10/25/2012 09:02	62	1
60	Implementar els tests per a la venda d'un vehicle	10/25/2012 08:28	70	3
63	Implementar les validacions de les estadístiques de la venda de vehicles	10/25/2012 09:04	71	3
61	Implementar la venda de vehicles	10/25/2012 08:29	80	3
64	Implementar les estadístiques de la venda de vehicles	10/25/2012 09:05	81	3

Casos d'ús

De cara a la lògica de negoci, la història d'usuari que s'està satisfent reflexa els següents casos d'ús:

- **Venda d'un vehicle**: Només els comercials, de qualsevol botiga, poden fer una venda d'un vehicle, que quedarà registrat.
- **Addició d'un nou client**: Per a poder vendre vehicles, es contempla la possibilitat de que es puguin afegir nous clients.

Disseny

Per a les vendes, s'han identificat les següents entitats:

Clients

És l'entitat que representa els clients que compren els vehicles a les botigues. Com indica la història d'usuari, es consideren les dades següents:

- NIF: Consistirà en una cadena de caràcters de màxim 9 caràcters, considerant la numeració del NIF. Serà el camp clau.
- Nom: Consistirà en una cadena de caràcters de 100 caràcters com a màxim.
- Adreça: Consistirà en una cadena de caràcters amb longitud màxima de 200.
- Telèfon: Consistirà en una cadena de caràcters de 9 caràcters com a màxim. Només s'han considerat numeracions nacionals.
- Fax: Consistirà en una cadena de caràcters amb les mateixes consideracions que el telèfon.
- Correu-e: Consistirà en una cadena de caràcters de 50 caràcters com a màxim.

Comercials

És una entitat que hereta de l'entitat Empleats. No té cap camp addicional, però permet establir la diferència entre un empleat i un comercial. Com hereta de Empleats contindrà el camp NIF, que serà clau foràinia i primària, a l'hora.

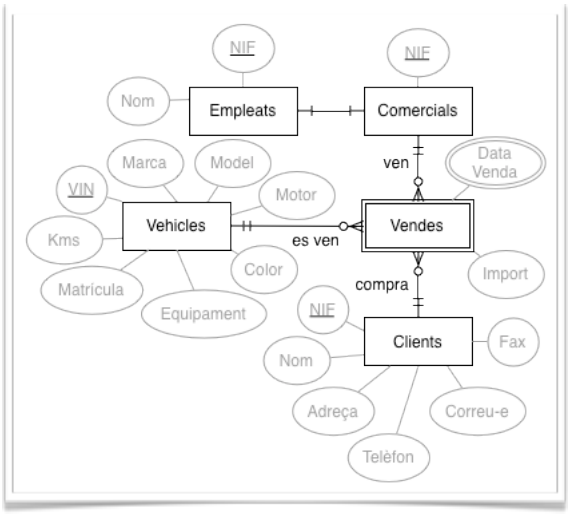
Per aquesta entitat, s'han contemplat altres opcions:

- No implementar res més que la relació entre les vendes i els empleats. El problema que presenta aquesta opció és que, en el moment de veure quants cotxes ha venut cada comercial, en no

poder-se diferenciar entre els empleats els que són comercials, es podrien donar més entrades de les que realment es volen.

- Afegir un camp diferencial a l'entitat d'empleats. Aquesta opció considera que hi ha més categories i, com fins ara no han estat necessàries, convindria establir-les totes.

Com que la venda quedaria representada per una relació a tres bandes entre Comercials, Vehicles i Clients, cal implementar una entitat explícita per representar-la:



Vendes

La clau de la qual estarà composta pels NIFs del Comercial i del Client, el VIN del Vehicle i la Data de la venda. Considera els següents camps:

- NIF Comercial: Consistirà en una cadena de caràcters de màxim 9 caràcters, considerant la numeració del NIF.
- NIF Clients: Com l'anterior.
- VIN Vehicle: Consistirà en una cadena de caràcters de 17 caràcters, basant-se en la definició que es dóna a la Viquipèdia
- Data de la venda: Serà un camp de tipus data que no contindrà l'hora, doncs no cal.
- Import: Serà un camp de tipus numèric amb dos decimals, que representarà l'import de la venda en Euros.

Per a implementar les vendes, s'han especificat els següents procediments:

Addició d'un nou client

Utilitzat únicament en el cas que s'hagi d'incloure un nou client al que se li està venent un cotxe. El seu prototip seria:

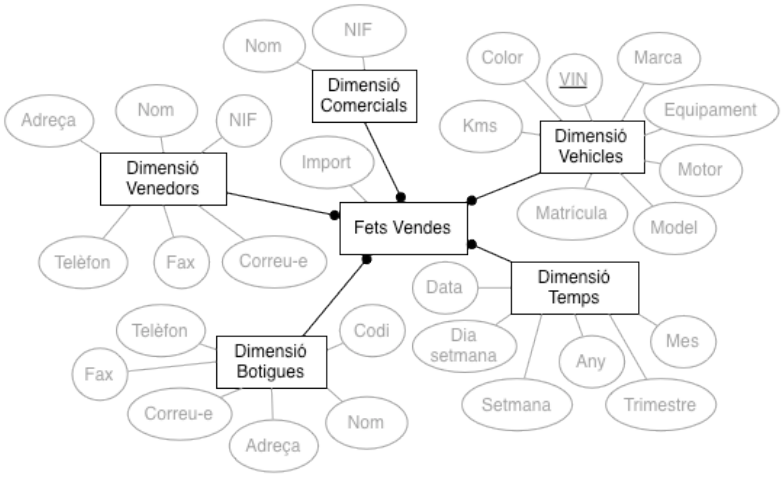
```
nouClient(NIF, Nom, Adreça, Telèfon, Fax, Correu-e)
```

Venda d'un nou vehicle

Realitzarà la venda efectiva. El seu prototip és:

```
venda(NIF Comercial, NIF Client, VIN, Matrícula,
Marca, Model, Motor, Equipament, Color, Data de la
Compra, Import)
```

Pel que fa al magatzem de dades, s'ha considerat afegir les dimensions pels comercials i pels clients, així com una taula de fets nova, explícita per les vendes:



Deixant el model físic següent:

Dimensió Comercials:

- **Id**
- NIF

Dimensió Clients:

- **Id**
- NIF
- Nom
- Adreça
- Telèfon
- Fax
- Correu-e

Fets Vendes:

- **Id Comercial**
- **Id Botiga**
- **Id Client**
- **Id Vehicle**
- **Id Temps**
- Import

Per a la correcta implementació de l'ETL, la integració de les vendes inclou modificar el procediment principal per cridar als procediments especialitzats en introduir les dades corresponents, tant pel client com per la venda mateixa.

Creació dels tests

Pel que fa als tests de les vendes, no són gaire diferents als de les compres, compten quantes vendes s'han realitzat, executen una compra i una venda i tornen a comptar les vendes, comparant tots dos resultats per validar que s'ha afegit una venda. Per testejar els clients es comporten de forma similar.

Per les validacions de les estadístiques, en canvi, s'ha preferit unificar tant el testeig de les estadístiques de les compres com les de les ventes, de forma que el nou codi substitueix les validacions creades en l'anterior iteració. Per comprovar que el nou codi de test no invalidava el codi anterior, abans de començar la implementació de les noves funcionalitats, s'ha verificat que els tests de les estadístiques de compres seguien acceptant el codi encara present.

Implementació

Pel que fa a la implementació de les vendes, s'han implementat els procediments de `venda` i `nouClient`.

El procediment de `venda`, com en el cas de les compres, insereix les dades rebudes per paràmetre a la taula de vendes i registra l'event al log d'aplicació.

El procediment de `nouClient`, de forma molt similar a l'addició d'un nou venedor, insereix les dades del nou client a la taula de clients i ho registra.

Pel que respecta a l'ETL del magatzem de dades, s'han creat o modificat els següents procediments i funcions:

> Procediment `etl`: El procediment s'ha modificat des de la versió anterior en que identifica les accions de `venda` i `nouClient`, i que si no identifica l'acció trobada al registre, deixa un missatge d'error.
>
> Procediment `etlVenda`: Aquest és el procediment que s'encarrega de registrar la venda, per la qual cosa és cridat pel procediment `etl`. El que fa és recórrer tots els atributs pel codi de registre que se li passa per paràmetre, recollint els valors, verificant-ne l'existència en les dimensions corresponents i transformant el valor de l'import, guardant-ho tot en memòria, fins que, un cop recorreguts tots els atributs, n'insereix el registre a la taula de fets de vendes. Si troba algun atribut no reconegut, ho informa escrivint-ho a la sortida del servidor, i continua.

Procediment `etlNouClient`: Aquest procediment serveix per atendre la dimensió de clients. S'ha considerat que en altres històries d'usuari hi haurà clients relacionats, per la qual cosa, convé tenir aquest procediment i que sigui invocat des del procediment d'`etl`. Aquest procediment, a diferència de l'`etlVenda` o l'`etlCompra`, actualitza atributs locals, per tant no verifica cap altre dimensió, només assigna cada atribut al camp que pertoca d'un registre en memòria, i n'intenta recuperar un registre coincident de la dimensió. Si no el recupera, n'insereix el que té en memòria.

Funció `etlClient`: Aquesta funció serveix per mantenir la dimensió de clients. El seu funcionament consisteix en recuperar les dades del client de la base de dades transaccional i tractar de recuperar-lo de la dimensió, per retornar-lo. Si no el pot recuperar, el crea i el retorna.

Funció `etlNouComercial`: Aquesta funció serveix per mantenir la dimensió de comercials. El seu funcionament consisteix en recuperar les dades del comercial de la base de dades transaccional i tractar de recuperar-lo de la dimensió, per retornar-lo. Si no el pot recuperar, el crea i el retorna.

Així mateix, s'han implementat tres vistes per facilitar l'extracció de les estadístiques. Permeten obtenir les vendes per botiga i per comercial, i la quantitat de vehicles de primera i de segona mà venuts.

Validacions

Per a la realització de les validacions, així com de la codificació sencera, s'ha utilitzat únicament el JDeveloper, aplicació que ha facilitat en gran part el treball amb la base de dades, ja que l'SQL*Plus no permet utilitzar historial i s'han d'escriure les comandes un i altre cop.

```
No Errors.
1 - Afegir una compra passed
2 - Afegir un venedor passed
1 - Afegir un client passed
2 - Afegir una venda passed
1 - Afegir un log passed
1 - Comptar una compra passed
2 - Comptar una venda passed
3 - Comptar una segona compra passed
4 - Comptar una segona venda passed

Connection created by CONNECT script command disconnected
```

Retrospectiva

La nova forma d'executar els tests i la consolidació de l'ús de la llibreria de *testing*, serien punts positius força importants. La manca de canvis en el sistema de registre, també. I l'encaixament del disseny de les noves funcionalitats en el disseny existent, el més important de tots. Finalment, aquest cop s'ha inclòs la creació de les vistes per extreure les estadístiques.

Com a punts de millora, es considera que la utilització dels tests no és gaire completa, són una mica massa simples, malgrat que han cobert les necessitats que s'han anat presentant fins ara.

Pel que fa a la velocitat d'implementació, s'ha dedicat el temps exacte (16 punts), però com s'ha dedicat més temps a la correcció dels errors detectats de la iteració anterior, no ha estat possible dedicar-hi tant de temps com al primer. És possible que aquesta sigui la velocitat normal d'una iteració, però s'hauria de poder veure millor amb la tercera iteració.

Després de veure la versió funcional, s'ha observat que el sistema permet vendre cotxes sense matricular, cosa que ha fet que s'afegeixi una nova història d'usuari per a incloure les funcions de matriculació al *backlog*. Per a la correcta implementació, es considera que els empleats poden afegir matrícules manualment, posteriorment a haver-les sol·licitat a la Direcció General de Trànsit, o DGT. El sistema emmagatzemarà totes les matrícules dels vehicles que passin per les botigues, així com les que tingui disponibles per associar a un vehicle nou. També es considera que el procediment de compra s'ha de modificar per verificar si la matrícula del nou cotxe existeix. Finalment, el procediment de venda ha de comprovar si el vehicle que s'està venent té assignada una matrícula. Si no la té, comprovarà si en queden i, en cas afirmatiu, n'assignarà una al vehicle. Si no quedessin, es generarà un error que avisi al comercial que fa la venda per tal que n'aconsegueixi.

Finalment, s'ha observat que no s'emmagatzemava la informació sobre els quilometratges dels vehicles.

Iteració 3

En aquest capítol es detalla com ha anat la tercera iteració, en la que s'ha implementat la funcionalitat de matriculació de vehicles, afegida després d'observar-ne la necessitat, i la de revisió.

Kick off

Aquesta iteració té els següents objectius:

- ➡ Completar les tasques assignades.
- ➡ Seguir completar la línia de base del projecte, afegint i modificant tot el que calgui per completar les tasques assignades.
- ➡ Verificar la relació entre l'estimació i el temps dedicat.

Sobre les tasques del *backlog*, s'han triat totes les tasques que corresponen a la US de la matriculació de vehicles i la de les revisions, però en l'ordre que permet la implementació ordenada. A continuació, s'inclou una llista de les tasques seleccionades:

#	Titol	Creada	Queue Number	Estimation
91	Analitzar la necessitat de modificar el sistema de registre per les matriculacions	11/26/2012 23:31	30	1
92	Analitzar la necessitat de modificar el magatzem de dades per la matriculació	11/26/2012 23:34	40	1
87	Dissenyar els elements necessaris per la implementació de les matriculacions	11/26/2012 23:13	50	1
88	Identificar i prototipar els processos implicats en la matriculació	11/26/2012 23:17	60	1
89	Implementar els tests per a la matriculació	11/26/2012 23:20	70	3
90	Implementar la matriculació	11/26/2012 23:23	80	3
66	Identificar les entitats implicades en la revisió d'un vehicle	10/25/2012 08:26	90	1
67	Identificar les entitats implicades en la revisió d'un vehicle	10/25/2012 08:27	100	1
70	Verificar el subsistema de registre d'accions per la història de revisió d'un vehicle	10/25/2012 09:05	101	1
71	Dissenyar els elements necessaris per les estadístiques de les revisions de vehicles	10/25/2012 09:02	102	1

68	Implementar els tests per a la revisió d'un vehicle	10/25/2012 08:28	110	3
72	Implementar les validacions de les estadístiques de la revisió de vehicles	10/25/2012 09:04	111	3
69	Implementar la revisió de vehicles	10/25/2012 08:29	120	3
73	Implementar les estadístiques de la revisió de vehicles	10/25/2012 09:05	121	3

Casos d'ús

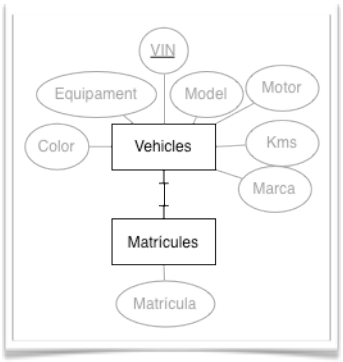
De cara a la lògica de negoci, les històries d'usuari que s'estan satisfent reflexen els següents casos d'ús:

- **Addició d'una nova matrícula**: Aquest cas d'ús permetrà als empleats introduir noves matrícules que la DGT hagi assignat al concessionari.
- **Compra d'un vehicle**: Quan s'adquireix un vehicle de segona mà, cal introduir la seva matrícula al sistema. Això implica modificar el procediment actual.
- **Venda d'un vehicle**: Com el concessionari no pot vendre vehicles sense matricular, caldrà que el procediment de venda busqui una matrícula disponible i li assigni. Si no en queden, caldrà que adverteixi a l'usuari per tal que n'aconsegueixi.
- **Entrada d'un vehicle per revisió**: Quan un client porta un vehicle per una revisió, el rep un mecànic que realitzarà l'albarà d'entrada i l'atendrà.
- **Sortida d'un vehicle després de ser revisat**: Un cop el vehicle ha estat revisat, el client vindrà a recollir-lo i caldrà que el mecànic registri la retirada.

Disseny

Per tal d'implementar les funcionalitats relacionades amb les matriculacions, s'ha identificat una sola entitat, molt senzilla, que relacionarà les matrícules amb els VIN que corresponguin.

Matrícules



És l'entitat que representa les matrícules que s'assignen. Com indica la història d'usuari, es consideren les dades següents:

- Matrícula: Consistirà en una cadena de caràcters de màxim 7 caràcters. Serà el camp clau.
- VIN: Consistirà en una cadena de caràcters de 17 caràcters com a màxim, que podrà ser nul i està vinculat a la clau de l'entitat de Vehicles.

Una matrícula disponible, no tindrà cap VIN assignat, mentre que per qualsevol vehicle matriculat que entri al taller, s'haurà de registrar la matrícula.

Per a la implementació del cas d'ús, es proposa un procediment amb el següent prototipus:

```
novaMatrícula(matricula)
```

Respecte a la US de les revisions, s'han considerat les següents entitats:

Mecànics

És l'entitat que representa els treballadors que són mecànics. De forma similar a com es va fer pels comercials, es considera únicament mantenir el NIF, com a clau primària, per vincular-lo amb l'entitat d'Empleats, per tant també serà clau forània.

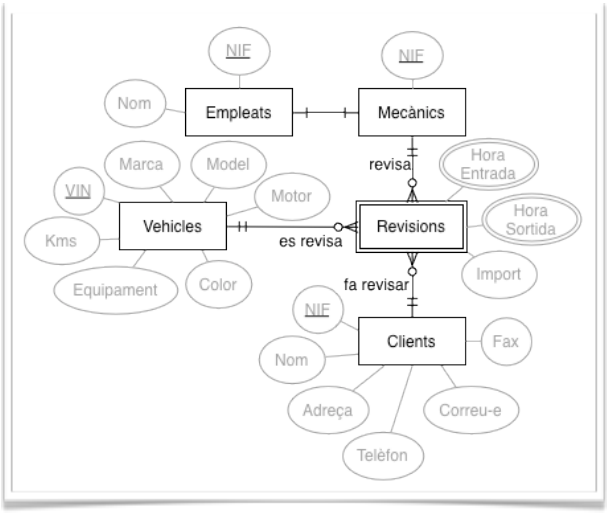
Revisions

És l'entitat que relaciona les entitats Mecànics, Clients i Vehicles, per establir-ne la relació que es produeix en fer-se una revisió. Es consideren els següents camps:

- NIF Mecànic: Consistirà en una cadena de caràcters de màxim 9 caràcters, considerant la numeració del NIF. És clau forània relacionada amb el NIF de l'entitat de Mecànics.
- NIF Client: Com l'anterior, i també és una clau forània, però relacionada amb l'entitat de Clients.
- VIN Vehicle: Consistirà en una cadena de caràcters de 17 caràcters, basant-se en la definició que es dóna a la Viquipèdia. És una clau forània amb l'entitat de Vehicles.

- Data i hora d'entrada: Serà un camp de tipus data que contindrà l'hora, necessària per poder calcular el temps utilitzat.
- Data i hora de la sortida: Serà un altre camp de tipus data que contindrà l'hora.
- Import: Serà un camp de tipus numèric amb dos decimals, que representarà l'import de la revisió en Euros.

Aquesta entitat tindrà una clau primària formada per diversos camps, que seran el NIF del Mecànic i el del Client, el VIN del Vehicle i la data i hora d'entrada.



Pel que fa als casos d'ús, es contemplen les següents funcions:

Entrada d'un vehicle per revisió

És el procediment que s'utilitzarà quan el mecànic registri l'entrada. El seu prototip seria:

`entraRevisió(NIF Mecànic, NIF Comercial, VIN Vehicle, Matrícula, Marca, Model, Motor, Equipament, Color, Quilometratge Actual)`

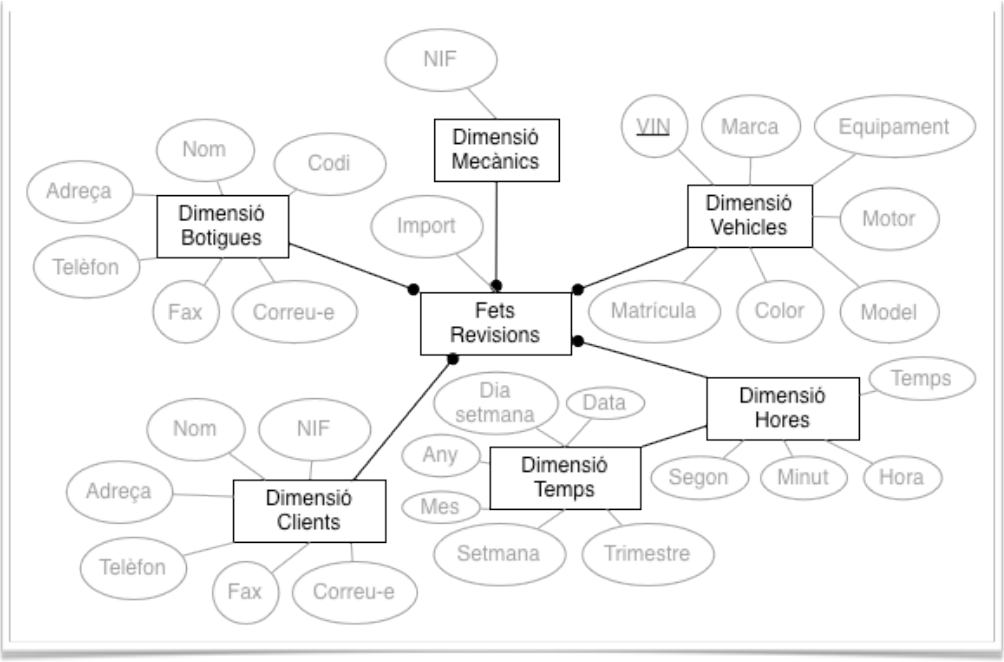
Sortida d'un vehicle per revisió

Serveix per registrar-ne la sortida del vehicle. El seu prototip és:

`surtRevisió(NIF Mecànic, NIF Client, VIN, Import)`

Respecte al magatzem de dades, la història d'usuari sobre la matriculació no inclou cap mena d'estadística pel que no cal modificar res del magatzem de dades.

Per les estadístiques de revisions, en canvi, cal afegir una taula de fets i una altra de mecànics, molt similar a la de comercials, i una nova dimensió amb les hores d'entrada i sortida:



Deixant el model físic següent:

Dimensió Mecànics:

- **Id**
- NIF

Dimensió Hores:

- **Id**
- Id Temps
- Temps
- Hora
- Minut
- Segon

Fets Revisions:

- **Id Mecànic**
- **Id Botiga**
- **Id Client**
- **Id Vehicle**
- **Id Hora Entrada**
- **Id Hora Sortida**
- Import

Creació dels tests

Els tests de la matriculació són una mica diferents als de les vendes o les compres. Per una banda, requereixen comprovar diferents procediments i, per l'altra, s'ha de comprovar que davant de certes situacions, es produeixen les excepcions adients. Així que s'han implementat els següents tests:

- Afegir una matrícula manualment.
- La detecció d'una matrícula duplicada.
- Afegir una matrícula en comprar un cotxe.
- L'assignació automàtica d'una matrícula en vendre un cotxe nou.
- La detecció de que manquen matrícules per assignar.

Com que no inclou cap estadística no ha estat necessari fer cap test pel magatzem de dades o per l'ETL.

Pel que fa als tests de les revisions, no hi ha gaire diferència amb els de les compres o les vendes. La principal diferència és que hi ha dos moments diferents, descrits pels dos casos d'ús, per tant, s'ha implementat un test per validar l'entrada i un altre per validar la sortida.

El test per validar l'entrada fa una compra, després una venda i realitza una entrada per revisió del mateix vehicle, amb un nou quilometratge, per tant, es comprova que l'entrada per revisió d'un cotxe ja existent modifica el quilometratge. Per comprovar-ne la inserció d'un nou vehicle, es fa una entrada d'un vehicle que no existeix prèviament i es comprova que hi ha un nou vehicle a la taula corresponent.

Pel que fa a la validació de la sortida, es realitza una entrada d'un nou vehicle, es compten el nombre de revisions, es fa la sortida i es compten les revisions que tenen import a 0, és a dir, que no s'han cobrat. El primer valor ha de ser d'un 1, doncs s'acaba d'afegir un vehicle per revisar, i el segon, un 0, doncs ja no hi ha cap revisió pendent de cobrar.

Finalment, s'han implementat un test que realitza una entrada per revisió, executa l'ETL i comprova que s'ha registrat a la taula de fets de revisions i que el total de temps dedicat és 0. Després, en fa la sortida i una altra entrada, torna a executar l'ETL i comprova que hi ha una revisió més a la taula de fets i que el temps total dedicat és superior a 0.

Implementació

Per la implementació de la història d'usuari sobre les matriculacions, s'ha implementat el procediment `novaMatricula` i s'han modificat els procediments `compra` i `venda`. El procediment `novaMatricula`, que accepta una cadena amb la matrícula, la insereix a la taula de matrícules, amb el VIN a nul.

Pel que fa al procediment de `compra`, s'ha modificat per comprovar si la matrícula que s'ha passat per paràmetre és nul·la o no. Si no ho és, afegeix la matrícula a la taula.

Per la refactorització del procediment de `venda`, s'ha afegit, abans de fer la inserció de la venda, es comprova si el vehicle ja té matrícula assignada. Si no en té, es comprova si queden matrícules sense assignar a la taula. Si en queden, se n'assigna una.

Pel que fa a les revisions, s'han implementat dos procediments: `entraRevisio` i `surtRevisio`. El procediment `entraRevisio`, guarda la data i hora en què s'executa, recupera les dades del vehicle, insereix la nova revisió, actualitza el quilometratge del vehicle i ho registra tot. Si no pot recuperar les dades del vehicle, les afegeix a la taula de vehicles, així com la matrícula, i la revisió, deixant-ne el registre corresponent. Pel que fa al procediment `surtRevisio`, només guarda l'hora en què s'executa i actualitza la revisió de la taula per afegir-hi l'hora de sortida i l'import, finalment registra l'acció.

Pel que respecta a l'ETL del magatzem de dades, s'han creat o modificat els següents procediments i funcions:

> Procediment `etl`: El procediment s'ha modificat des de la versió anterior en que identifica les accions de `venda` i `nouClient`, i que si no identifica l'acció trobada al registre, deixa un missatge d'error.
>
> Procediment `etlEntraRevisio`: Aquest procediment és responsable d'atendre les entrades del registre que fan referència a les entrades de vehicles per revisió, per això el procediment `etl` l'invoca quan en troba una. El seu codi fa que recuperi tots els atributs i, mitjançant les funcions adients, els inclou en les dimensions corresponents, per acabar inserint les dades resultants a la taula de fets de revisions.
>
> Procediment `etlSurtRevisio`: Aquest procediment és utilitzat pel d'`etl`, quan es troba un registre d'una sortida d'un vehicle de revisió,

per registrar-ne les dades. El seu codi recorre tots els atributs relacionats amb el registre en qüestió, i, utilitzant les funcions adients per cada un, n'assegura la presència a les dimensions corresponents, per tots els atributs menys l'import, que el transforma en nombre i el guarda al registre en memòria. Finalment, recupera la revisió per verificar-ne l'existència; i n'actualitza els camps de l'hora de sortida, l'import de la revisió, i del càlcul del temps empleat.

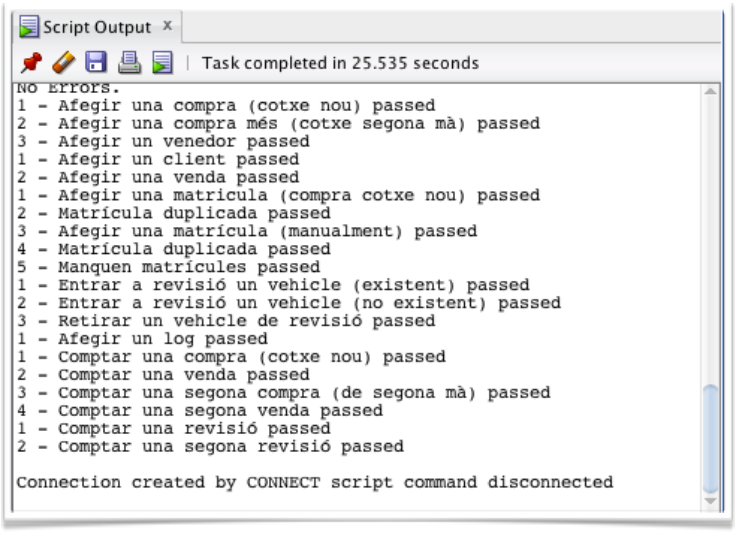
Funció `etlNouVehicle`: Aquesta funció s'ha modificat per eliminar-ne les referències a les matrícules. Es podria haver fet més canvis, però no s'han considerat necessaris.

Funció `etlNouMecanic`: Aquesta funció permet mantenir actualitzada la dimensió dels mecànics. Com la majoria d'aquestes funcions, recupera les dades del mecànic de la base de dades OLTP i comprova si coincideixen amb les que hi ha a l'OLAP, per retornar-les com a resultat. Si no hi ha les dades al magatzem, les insereix i les retorna com a resultat.

Funció `etlNovaHora`: Aquesta funció és la que s'encarrega de mantenir la dimensió d'hores. Treballa de forma molt similar a la funció `etlNovaData`, i en fa ús per recuperar el registre de la dimensió de temps que necessita per fer-hi referència. Així doncs, converteix el paràmetre d'entrada en una data, amb hora inclosa, i en fa diferents transformacions per cada camp, com, per exemple, recuperar-ne el registre de la dimensió de dates per relacionar-lo amb el d'hores.

Validacions

Per executar les validacions, s'ha utilitzat, de nou, el JDeveloper:



Retrospectiva

En aquesta iteració, s'han fet unes validacions una mica més complexes, fent més comprovacions i de forma una mica més avançada.

El temps dedicat a la iteració ha estat, en total de 26 punts, exactament el que estava estimat, i força ajustat al valor inicial. Per això, es fa difícil pensar que es pugui augmentar el ritme, sense incrementar el temps dedicat al projecte o els recursos.

Iteració 4

En aquest capítol es detalla com ha anat la quarta iteració, en la que s'ha implementat la funcionalitat de reparacions de vehicles i de gestió de personal.

Kick off

Aquesta iteració té els següents objectius:

- ➡ Completar les tasques assignades.
- ➡ Seguir completar la línia de base del projecte, afegint i modificant tot el que calgui per completar les tasques assignades.
- ➡ Verificar la relació entre l'estimació i el temps dedicat.
- ➡ Completar la gran majoria de funcionalitats més importants i iniciar-ne la de les que són més secundàries.

Sobre les tasques del *backlog*, s'han triat totes les tasques que corresponen a la US de les reparacions de vehicles i la de la gestió de personal. A continuació, s'inclou una llista de les tasques seleccionades:

#	Titol	Creada	Queue Number	Estimation
74	Identificar les entitats implicades en la reparació d'un vehicle.	10/25/2012 08:26	130	1
75	Identificar i prototipar els processos implicats en la reparació d'un vehicle	10/25/2012 08:27	140	1
78	Verificar el subsistema de registre d'accions per la història de reparació d'un vehicle	10/25/2012 09:05	141	1
79	Dissenyar els elements necessaris per les estadístiques de les reparacions de vehicles	10/25/2012 09:02	142	1
76	Implementar els tests per la reparació d'un vehicle	10/25/2012 08:28	150	3
80	Implementar les validacions per les estadístiques de les reparacions de vehicles	10/25/2012 09:04	151	3
77	Implementar la reparació de vehicles	10/25/2012 08:29	160	3
81	Implementar les estadístiques de la reparació de vehicles	10/25/2012 09:25	161	3
100	Identificar i prototipar els processos implicats en la gestió de personal	12/18/2012 19:59	210	1

101	Implementar els tests per la gestió de personal	12/18/2012 20:07	220	2
102	Implementar la gestió de personal	12/18/2012 20:58	230	2
103	Identificar canvis als procediments de l'ETL per les gestions de personal	12/18/2012 21:51	240	1
104	Verificar els tests existents de l'ETL involucrats amb la gestió de personal	12/18/2012 21:55	250	2
105	Implementar els canvis a l'ETL per la gestió de personal	12/18/2012 21:59	260	2

Casos d'ús

Pel que fa a la implementació de les reparacions de vehicles, s'han identificat els següents casos d'ús:

- **Entrada d'un vehicle per reparació**: Quan un client porta un vehicle per una reparació, el rep un mecànic que realitzarà l'albarà d'entrada i l'atendrà, registrant-ne les dades del vehicle, els quilòmetres que marca el comptaquilòmetres i el motiu de la reparació. Si el vehicle ja figura a la base de dades, s'actualitzaran les dades corresponents.
- **Sortida d'un vehicle després de ser reparat**: Un cop el vehicle ha estat reparat, el client vindrà a recollir-lo i caldrà que el mecànic en registri la retirada.

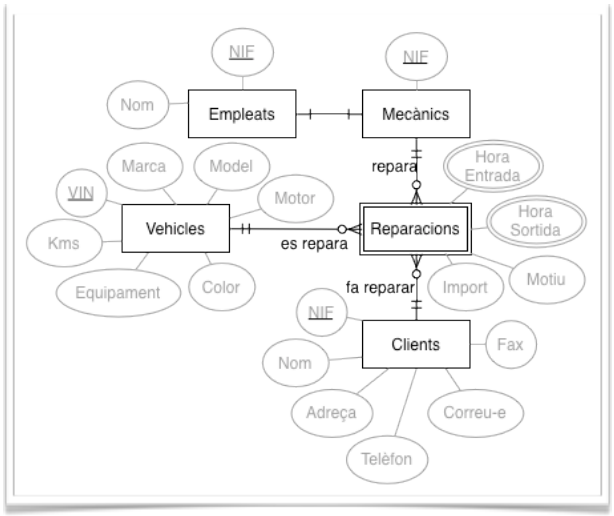
Pel que fa a la gestió de personal, s'han identificat els següents casos d'ús:

- **Contractació**: És el cas d'ús pel qual s'incorpora un nou treballador. Caldrà guardar-ne el nom, NIF, botiga al que se l'assigna, i si és comercial o mecànic.
- **Acomiadament**: L'acomiadament, sigui pel motiu que sigui, només marcarà un mecànic de forma que no se'l consideri en actiu. Els empleats que no estan en actiu, no podran realitzar les tasques habituals.

Disseny

Per la implementació de la història de les reparacions de vehicles, s'ha considerat afegir una nova entitat, molt similar a la de les revisions, amb la següent estructura:

Reparacions



És l'entitat que relaciona les entitats Mecànics, Clients i Vehicles, per establir-ne la relació que es produeix en fer-se una reparació. Es consideren els següents camps:

- NIF Mecànic: Consistirà en una cadena de caràcters de màxim 9 caràcters, considerant la numeració del NIF. És clau forània relacionada amb el NIF de l'entitat de Mecànics.
- NIF Client: Com l'anterior, i també és una clau forània, però relacionada amb l'entitat de Clients.
- VIN Vehicle: Consistirà en una cadena de caràcters de 17 caràcters, basant-se en la definició que es dóna a la Viquipèdia. És una clau forània amb l'entitat de Vehicles.
- Data i hora d'entrada: Serà un camp de tipus data que contindrà l'hora, necessària per poder calcular el temps utilitzat.
- Data i hora de la sortida: Serà un altre camp de tipus data que contindrà l'hora.
- Motiu: Serà una cadena de 200 caràcters.
- Import: Serà un camp de tipus numèric amb dos decimals, que representarà l'import de la revisió en Euros.

Pel que fa als casos d'ús, es contemplen les següents funcions:

Entrada d'un vehicle per reparació

És el procediment que s'utilitzarà quan el mecànic registri l'entrada. El seu prototip seria:

`entraReparació(NIF Mecànic, NIF Comercial, VIN Vehicle, Matrícula, Marca, Model, Motor, Equipament, Color, Quilometratge Actual, Motiu)`

Sortida d'un vehicle per reparació

Serveix per registrar-ne la sortida del vehicle. El seu prototip és:

`surtReparació(NIF Mecànic, NIF Client, VIN, Import)`

Pel que fa a la part de les estadístiques, com en el cas de les revisions, també cal afegir la taula de fets corresponent, però per les dimensions, es podran fer servir les mateixes que per les revisions:

Portant al següent model físic:

Fets Reparacions:

- **Id Mecànic**
- **Id Botiga**
- **Id Client**
- **Id Vehicle**
- **Id Hora Entrada**
- **Id Hora Sortida**
- Import
- Motiu

Pels casos d'ús de la contractació i de l'acomiadament, l'única modificació a fer en l'esquema de la base de dades és afegir, a l'entitat d'Empleats, l'atribut `enActiu` que serà un valor booleà.

En quant als prototipus de les funcions per cada cas d'ús, són els següents:

Contractació

És el procediment que s'utilitzarà per introduir un nou empleat. El seu prototip seria:

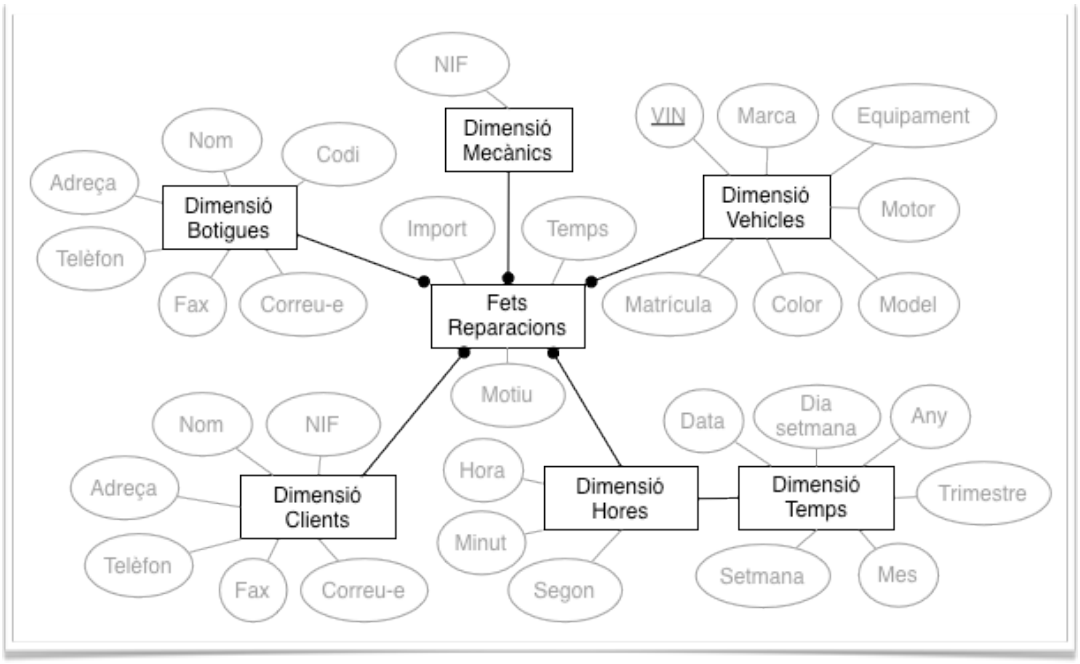
`contractació(NIF, Nom, Codi Botiga, Tipus Empleat)`

Acomiadament

Quan s'acomiadi un empleat, s'indicarà amb aquest procediment. El seu prototip és:

`acomiadament(NIF)`

Pel que fa al magatzem de dades, es mantindrà un atribut, anomenat baixa, de tipus data que indicarà el moment en el que l'empleat és acomiadat.



Creació dels tests

Pel que fa als tests de les reparacions, no hi ha gaire diferència amb els de les compres o les vendes. La principal diferència és que hi ha dos moments diferents, descrits pels dos casos d'ús, per tant, s'ha implementat un test per validar l'entrada i un altre per validar-ne la sortida.

El test per validar l'entrada fa una compra, després una venda i realitza una entrada per reparació del mateix vehicle, amb un nou quilometratge, per tant, es comprova que l'entrada per reparació d'un cotxe ja existent modifica el quilometratge. Per comprovar-ne la inserció d'un nou vehicle, es fa una entrada d'un vehicle que no existeix prèviament i es comprova que hi ha un nou vehicle a les taules de vehicles i de reparacions.

Pel que fa a la validació de la sortida, es realitza una entrada d'un nou vehicle, es compten el nombre de reparacions, es fa la sortida i es compten les reparacions que tenen import a 0, és a dir, que no s'han cobrat. El primer valor ha de ser d'1, doncs s'acaba d'afegir un vehicle per reparar, i el segon, un 0, doncs ja no hi ha cap reparació pendent de cobrar.

Finalment, s'han implementat un test que realitza una entrada per reparació, executa l'ETL i comprova que s'ha registrat a la taula de fets de reparacions i que el total de temps dedicat és 0. Després, en fa la sortida i una altra entrada, torna a executar l'ETL i comprova que hi ha una reparació més a la taula de fets i que el temps total dedicat és superior a 0.

Pel que fa a les contractacions i als acomiadaments, s'han fet vuit tests diferents:

Un test valida el procediment de contractació, verificant que tant si es contracta un mecànic com un comercial, tant el nombre d'empleats com el de les respectives taules s'incrementen.

D'altra banda, el segon test valida que en acomiadar, tant un mecànic com un comercial, el nombre de registres, de la taula empleats, que tenen el camp `enActiu` a `'Y'`, es decrementa.

La resta de tests, validen que en acomiadar-se un empleat, si es fa qualsevol de les operacions com compra, venda, entrada o sortida a revisió o a reparació, amb l'empleat acomiadat, es genera una excepció expressament creat per aquesta situació.

Pel que fa a l'ETL i les contractacions i acomiadaments, s'ha fet un sol test que comprova que quan es contracta un comercial o un mecànic, o s'acomiada un empleat de qualsevol tipus, després d'executar el procediment d'ETL, les taules de dimensió corresponents s'actualitzen convenientment.

Implementació

Per la implementació de la història d'usuari de les reparacions, s'han implementat els procediments `entraReparació` i `surtReparació`.

El primer, recull l'hora actual per associar-la com l'hora d'entrada, busca el vehicle que s'ha entrat per reparació, guarda la nova reparació i actualitza el quilometratge del vehicle. En cas que no trobi el vehicle, s'introdueix a la taula de vehicles, es guarda la seva matrícula i es registra la nova reparació.

El segon, recull l'hora actual per associar-la com l'hora de sortida i actualitza el registre de la reparació amb la data i l'import indicats.

Per la implementació de la història d'usuari de la gestió de personal, s'han implementat els procediments `contractació` i `acomiadament` i s'han modificat els procediments `compra`, `venda`, `entraRevisió`, `surtRevisió`, `entraReparacio` i `surtReparació`. Val a dir que l'atribut `enActiu` s'implementa amb una restricció de valors (`CHECK`) sobre un tipus predefinit, en aquest cas, un caràcter.

Per la contractació, senzillament, s'insereix un registre a la taula d'empleats i un altre a la taula de comercials o de mecànics, segons el paràmetre `tipus_empleat`. El procediment d'acomiadament és encara més senzill, ja

que senzillament, actualitza el registre de l'empleat, posant el valor `'N'` a l'atribut `enActiu`.

Pel que fa a la refactorització de la resta de procediments, ha consistit en recuperar el registre de l'empleat que fa l'operació de la taula d'empleats i comprovar si el camp `enActiu` conté `'N'`, aleshores, es llança l'error. Aquesta comprovació en primer lloc, abans de cap altre part del procediment corresponent.

Pel que respecta a l'ETL del magatzem de dades, s'han creat o modificat els següents procediments i funcions:

Procediment `etl`: El procediment s'ha modificat des de la versió anterior per identificar les funcions d'`entraReparacio`, `surtReparacio`, `contractacio` i `acomiadament`.

Procediment `etlEntraReparacio`: Aquest procediment registra al magatzem les funcions d'`entraReparacio`, a partir de les dades que hi ha al registre. Per cada atribut relacionat, en recupera el registre dimensional corresponent, menys pel camp `motiu`, que guarda la cadena de caràcters en el registre en memòria que, un cop recollits tots els atributs, fa servir per inserir un nou registre a la taula de fets de reparacions.

Procediment `etlSurtReparacio`: Aquest procediment funciona de forma molt similar al de les sortides de revisions, però amb les reparacions, que és el que actualitza, amb les dades noves. Així, recull totes les dades dels atributs relacionats amb el registre en qüestió, guardant-los en un registre en memòria, convertint-los a la informació corresponent a les dimensions que pertoquin, menys per l'import que, simplement, converteix en un nombre. Un cop ha recuperat tots els atributs, recupera el registre de la reparació que pertoca i n'actualitza l'hora de sortida, l'import cobrat i el temps empleat.

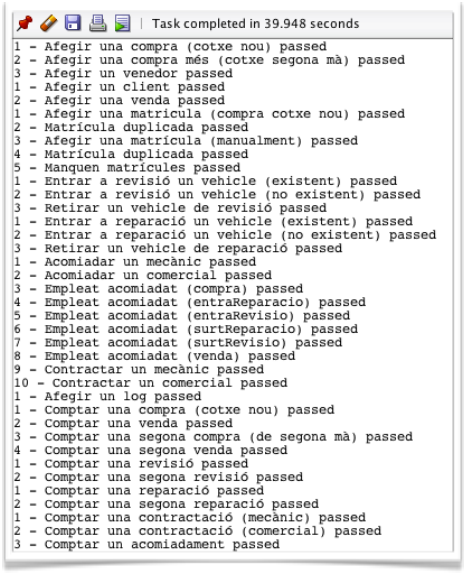
Procediment `etlContractacio`: Aquest procediment permet gestionar les aparicions de crides a `contractacio` que es puguin trobar al registre. El seu funcionament es basa, com tots els anteriors, en recollir les dades dels atributs que hi ha al registre, aplicant-ne les transformacions pertinents i assegurant-ne la presència a les dimensions corresponents, però en aquest cas, com només actualitza dimensions i amb funcions ja existents que ho fan (`etlNouEmpleat`,

`etlNouMecanic` i `etlNouComercial`), no cal que acabi fent cap inserció.

Procediment `etlAcomiadament`: Aquest procediment actualitza les dades de l'empleat, registrant-ne la data de baixa en la dimensió adient. Per fer-ho, el que fa és guardar-se l'empleat quan en recupera el NIF del registre, per actualitzar-ne la data de baixa.

Validacions

Com en les anteriors iteracions, utilitzant el JDeveloper, es comprova que la implementació feta satisfà tots els tests implementats:



Retrospectiva

Les històries d'usuari que s'han completat en aquesta iteració són; una, molt similar a les anteriors, i l'altra força diferent. La principal diferència radica en que les modificacions necessàries per implementar la contractació i l'acomiadament, tal i com s'havien definit, inclouen alterar les estructures de dades i el codi dels procediments, sense alterar-ne els resultats. Gràcies al desenvolupament guiat per tests, aquests canvis s'han pogut fer sense introduir nous problemes.

Pel que fa referència al temps, en aquesta iteració s'han dedicat 28 punts, però només hi havia 26 punts estimats. Tot i que no és una desviació molt significativa, cal indicar que s'ha invertit força temps en investigar, comparar i provar quina implementació del camp booleà es faria i com es comportaria.

Iteració 5

En aquest capítol es detalla com ha anat la cinquena iteració, en la que s'ha implementat la gestió de botigues i el llistat de clients amb vehicles pendents de revisar, a més de resoldre un parell d'errors detectats i algunes modificacions en el format del codi.

Kick off

Aquesta iteració té els següents objectius:

- ➡ Completar les tasques assignades.
- ➡ A partir de la línia de base actual del projecte, finalitzar la iteració en una situació prou definitiva del projecte.

Sobre les tasques del *backlog*, s'han triat totes les tasques que corresponen a la US de la gestió de les botigues i al llistat de revisió, també s'han inclòs les tasques de resolució d'errors detectats i les que s'han afegit. A continuació, s'inclou una llista de les tasques seleccionades:

#	Titol	Actualitzada	Queue Number	Estimation
106	Identificar i prototipar els processos implicats en la gestió de les botigues.	12/18/2012 22:07	270	1
107	Implementar els tests per la gestió de les botigues.	12/18/2012 22:08	280	2
108	Implementar la gestió de les botigues.	12/18/2012 22:09	290	2
109	Identificar canvis als procediments d'ETL per les gestions de les botigues.	12/18/2012 22:09	300	1
110	Verificar els tests existents de l'ETL involucrats en la gestió de les botigues.	12/18/2012 22:10	310	2
111	Implementar els canvis a l'ETL per la gestió de les botigues.	12/18/2012 22:10	320	2
112	Separar el codi del fitxer create.sql.	12/30/2012 12:15	323	1
113	Revisar els comentaris del codi.	12/30/2012 12:15	325	1
114	Possible incoherència al tractament de la matrícula en el procediment de venda.	12/31/2012 15:58	-	-
115	A surtRevisio i surtReparacio, no s'actualitza correctament el paràmetre de l'hora de sortida.	12/31/2012 16:00	-	-
96	Identificar canvis estructurals pels avisos de revisió.	12/18/2012 17:24	330	1

97	Identificar canvis procedurals pels avisos de revisió.	12/18/2012 19:13	340	1
98	Implementar els tests necessaris pels avisos.	12/18/2012 19:24	350	3
99	Implementar els avisos.	12/18/2012 19:30	360	3

Casos d'ús

Pel que fa a la implementació de la gestió de botigues, s'han identificat els següents casos d'ús:

- **Inauguració**: És el cas d'ús pel qual s'inaugura una nova botiga. Caldrà guardar-ne el nom, NIF, botiga al que se l'assigna, i si és comercial o mecànic.
- **Clausura**: La clausura d'una botiga implica que els treballadors associats són assignats a una altra botiga, per tant, caldrà indicar-ne els codis d'ambdues botigues. El funcionament haurà d'incloure el trasllat dels empleats corresponents.

Respecte al llistat d'avisos, aquest ha de generar un llistat de clients, amb nom, telèfon i correu electrònic, amb vehicles, identificats per matrícula, que hagin estat venuts fa més de 4 anys, i que faci més d'un any que no han estat revisats. Cal tenir present que s'han de considerar els vehicles venuts, revisats i reparats que compleixin aquestes condicions.

Pel que fa a les tasques de separació del codi en diferents fitxers i de comentat del codi, no hi ha casos d'ús que apliquin.

Disseny

Pel cas d'ús de les inauguracions i clausures de les tendes, no és necessari realitzar cap canvi a l'estructura de dades, ja que la botiga pot ser completament eliminada. En el cas del magatzem de dades, la botiga, pel contrari, sí que cal afegir un nou atribut a la dimensió de botigues per tal que quedi reflectida la data de clausura de la tenda.

Pel que fa a procediments de la base de dades, caldrà crear els següents:

Inauguració

És el procediment que s'utilitzarà per afegir una nova botiga. El seu prototip seria:

```
inauguració(Nom, Adreça, Telèfon, Fax, Correu)
```

Clausura

Quan es clausuri una botiga, s'indicarà amb aquest procediment. El seu prototip és:

```
acomiadament(Codi Clausurada, Codi Substituta)
```

Respecte als procediments de l'ETL relacionats amb la inauguració i clausura de tendes, cal tenir present que consistiran exclusivament en crear les noves botigues i marcar la data en què queden clausurades les que es donin de baixa, però no s'eliminaran registres, per no perdre les referències.

Pel llistat d'avisos de revisions, es pot fer amb una vista que treballi sobre el magatzem de dades. Es podria fer també amb una vista sobre la base de dades OLTP, però donat que ja s'ha creat un magatzem de dades, que s'ha desenvolupat un procés ETL que pot ser executat en gairebé qualsevol moment, que aquest llistat es consultarà de forma diària, i, principalment, que les dades poden ser molt més antigues que a l'OLTP.

Pel que fa a les tasques de separació i comentat del codi, no cal fer gaire disseny, ja que no hi ha canvis, ni en les estructures ni en el codi en si mateix.

Creació dels tests

Per a fer les validacions de les operacions sobre les botigues, s'ha implementat un test que, en primer lloc, prova una inauguració, comprovant que el codi ha crescut entre abans i després de la inauguració; en segon lloc, es comprova que en clausurar, la quantitat de botigues a la taula ha disminuït en una unitat respecte abans de la clausura. El test de l'ETL d'aquestes funcions és molt similar, però es comproven les quantitats de botigues a la dimensió abans i després de cada execució de l'ETL, que es fa després d'haver fet cada una de les operacions.

Finalment, s'ha implementat un test que valida la vista del llistat d'avisos de revisions introduint dades a la base de dades principal, i executa alguns procediments per generar vehicles de tots els tipus, executa l'ETL, i compta el nombre de files que retorna el llistat. Després fa la revisió d'un dels vehicles que està pendent de fer-la i torna comptar el llistat, comparant totes dues quantitats per validar si és correcte.

Des del punt de vista del *testing*, les tasques de separació i comentat del codi, sí que tenen un valor força gran, però no per implicar validacions específiques o noves, sinó perquè totes les proves que s'han anat fent

mentre es desenvolupava la base de dades s'han de seguir validant després dels canvis, funció que és principal i raó de ser dels tests en sí mateixos.

Implementació

Per la implementació dels procediments `inauguració` i `clausura`, és molt simple:

Pel procediment d'`inauguració`, senzillament, insereix la nova botiga, amb les dades que es reben per paràmetre, i recupera el codi de la fila acabada d'inserir, per poder-lo afegir a les dades del registre.

Pel que fa al procediment de `clausura`, primer s'actualitza el camp `codiBotiga` de tots els treballadors de la botiga clausurada, canviant-ne el valor pel codi de la botiga que la substitueix. Després, elimina la botiga clausurada de la taula de botigues, i registra el canvi.

Respecte al llistat d'alertes de revisió, en tractar-se d'una vista, només consisteix en una sentència. Malgrat això, és una sentència força complexa. La forma més simple que he trobat d'implementar-la és unint-ne, amb `UNION`, tres sentències separades, una pels cotxes venuts fa més de 4 anys que no s'han revisat en l'últim any; una altra pels cotxes que han vingut a revisar-se fa més d'un any; i, finalment, una altra pels cotxes que s'han reparat, però que fa més d'un any que no es revisen. Per al primer i últim cas, s'ha fet una altra sentència subordinada, per tal de recuperar els identificadors dels vehicles que s'han revisat fa menys d'un any.

Pel que fa als comentaris, a més d'afegir-ne dins del codi, també s'han afegit missatges de sortida per poder veure com està progressant l'execució de la creació de la base de dades i l'execució dels tests.

Finalment, la separació s'ha realitzat dividint el fitxer `create` inicial en sis parts, mantenint l'original com a un guió que els va invocant. Els sis nous fitxers són els que segueixen:

- `create_schemes`: Aquest fitxer conté el codi que crea els `TABLESPACES` i usuaris necessaris, amb els permisos adients, tant per la base de dades OLTP com pel magatzem de dades.
- `create_schema`: En aquest fitxer està programada la creació de l'esquema de la base de dades OLTP, amb totes les taules, seqüències i disparadors necessaris.
- `create_procedures`: El codi que crea tots els procediments i tipus de dades amb la lògica de l'aplicació està en aquest fitxer.

- `create_dwh`: El codi que genera les dimensions, les taules de fets, i les seqüències i disparadors necessaris per al magatzem de dades.
- `create_etl`: Aquest fitxer conté el codi amb els procediments d'ETL.
- `create_views`: Finalment, el codi que crea les vistes està en aquest fitxer.

Pel que respecta a l'ETL del magatzem de dades, s'han creat o modificat els següents procediments i funcions:

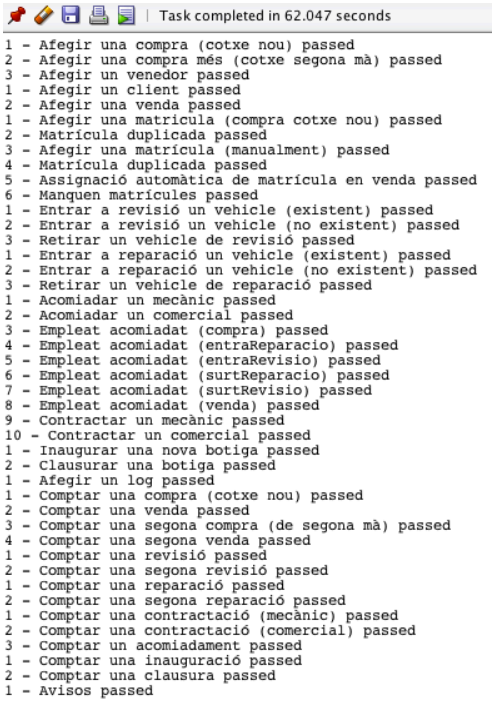
Procediment `etl`: El procediment s'ha modificat des de la versió anterior per identificar les funcions `inauguracio` i `clausura`.

Procediment `etlInauguracio`: Aquest procediment serveix per afegir una botiga que ha estat inaugurada. Recull tota la informació del registre, llegint atribut per atribut i guardant-ho en un registre en memòria, per, després, insertar les dades a la dimensió de Botigues.

Procediment `etlClausura`: Aquest procediment serveix per actualitzar les dades de la botiga que hi ha a la dimensió. Això ho fa mentre llegeix els atributs. Primer, en recuperar l'atribut que conté el codi de la botiga clausurada, recupera, de la dimensió de botigues, el seu registre, tan sols, per, quan recupera l'atribut amb la data de baixa, actualitzar-ne el camp `clausurada` amb la data correcta.

Validacions

Amb el JDeveloper es comproven que els tests es validen amb el codi desenvolupat, tant els nous com els antics, fet que valida que els canvis dels fitxers de codi no afecta al funcionament de l'aplicació:



Retrospectiva

Durant la implementació de la vista pel llistat d'avisos, s'ha observat que el procediment d'ETL de càrrega de vehicles era incorrecte degut a que filtrava el vehicle utilitzant el quilometratge. Aquesta situació no estava comprovada al test corresponent i s'ha hagut d'afegir.

Per la resta, cal mencionar que amb aquesta iteració es completen les funcionalitats indicades inicialment, així com les que han aparegut al llarg del projecte, tant per identificació posterior, com per errors de funcionalitats que no s'han implementat correctament en algun moment.

Pel que fa als punts, s'havien estimat 21 hores i s'han dedicat 22, per tant la relació és força ajustada i es pot considerar que l'estimació és correcta.

Valoració econòmica

En aquest apartat, s'han considerat dos tipus de valoracions: La que correspon al desenvolupament del projecte, tal i com s'ha realitzat; i la que correspon a tot el necessari per utilitzar-lo.

Costos del desenvolupament

El desenvolupament del projecte complet s'ha realitzat en 120 hores, el que, en una jornada laboral normal de 40 hores setmanals, implica que es podria haver realitzat completament en 3 setmanes. Tenint en compte el salari mig , obtingut d'un [Salaris] d'un administrador de bases de dades, el cost en recursos humans vindria a ser d'uns 4500 €. A això caldrà afegir-hi els costos dels recursos tecnològics (equipament, llum, costos diversos) i el marge de benefici. El preu final, doncs podria pujar a uns 5000€.

Costos de l'execució

S'han considerat dues possibilitats bàsiques: Per una banda, disposar de l'aplicació instal·lada en infraestructura pròpia; per l'altra, instal·lar el sistema creat en una infraestructura al núvol.

Infraestructura pròpia

Pel que fa a la infraestructura pròpia, es tindrà en compte un servei de *housing*, o hospedatge, per a ubicar un servidor de bases de dades. En això, doncs, cal comptar amb un cost inicial de 30 € i un cost mensual de 60 €.

Respecte al servidor, s'han considerat les recomanacions de maquinari d'Oracle [Oracle DB Hardware Requirements] pel que fa a la base de dades en Linux, i s'ha considerat el model de servidor més senzill que Oracle mateix ofereix, Sun Server X3-2 Small [Sun Server X3-2 Small], però amb un segon disc, per tal de poder configurar RAID 1, permetent tenir una mica de seguretat. Aquest model i configuració té un cost total aproximat de 6700 €.

Finalment, també cal considerar els costos de llicenciament i de suport de la base de dades, que en el cas proposat, tenint en compte el model de servidor i una llicència Standard Edition One, estaria al voltant de 23200 € per la llicència i 1276 € pel suport, segons el preu trobat [Oracle Price List],

el que deixaria el projecte en un preu inicial total aproximat de 29930 €, i un cost anual aproximats de 1996 €.

Evidentment, es podrien trobar configuracions més eficients, escalables o redundades, però els costos es dispararien considerablement. També es podria considerar altres models i fabricants de servidors, però sempre convindria ajustar-se a les polítiques de certificació d'Oracle [Oracle Certification], per tal de garantir-ne el suport.

Infraestructura núvol

Inicialment, s'ha considerat l'opció de serveis de *cloud* d'Oracle [Oracle Cloud], però, a banda de que el preu és força elevat, només permeten un sol esquema. Una altra opció, potser més adient, és Amazon AWS [Amazon AWS], que, amb una instància *micro* reservada, implica un cost inicial de 34'47 € o 68'94 € i un cost anual aproximat de 210'24 € o 420'48 €, ja que la documentació de preus d'Amazon RDS Oracle [Amazon RDS Oracle Pricing] no indica clarament si amb una sola instància es poden tenir múltiples esquemes. És molt probable que es puguin tenir, ja que el mateix servei en MySQL ho permet. Aquest preu inclou la llicència.

Si volem incloure el cost del suport d'AWS, es pot comptar atenció en 12 hores per correu electrònic per uns 36'72 € mensuals. Amb això, es fa un total màxim anual de 825'4 € i un cost inicial de 105'66 €.

Conclusions

Desenvolupament dirigit per proves

En cada iteració i per cada , i sempre sobre les especificacions derivades de les històries d'usuari corresponents, els tests han demostrat que el codi, canvi a canvi, continuava resolvent les necessitats tal i com s'havien indicat. El que és evident, és que els tests han d'estar ben dissenyats per provar totes les condicions i situacions que s'esperen.

Desenvolupament iteratiu

Les iteracions han estat clau per organitzar el progrés del projecte, donant visibilitat dels avanços i dels errors que s'aplicaven. En cas que hi hagués hagut un client real, s'haurien realitzat paquets d'entregables que es podrien, construïnt mòduls instal·lables, haver desplegat en un entorn en producció fent evolucionar l'aplicació i generant la satisfacció del client.

El que sí que s'ha pogut observar és com es comporta la metodologia en aparèixer nous i inesperats requeriments, o canvis en aquests.

Control de versions

El sistema de control, tal i com s'ha utilitzat, ha permès coordinar l'ordre, veure com era el codi en un moment anterior, comparar el codi en diferents fases del desenvolupament. És prou senzill d'utilitzar i, en cas de treballar amb grups de treball, sobretot si no comparteixen espai, horaris i demès, el fet de que sigui distribuït permet que hi hagi prou control sobre el codi que s'escriu.

Simplicitat i refactorització

Mantenint la simplicitat s'ha pogut tenir un codi simple, autoexplicatiu, funcional, que no fa res que no sigui necessari i, sobretot, permet fer refactoritzacions amb molta facilitat.

Fent refactorització, per exemple per canviar les condicions de la matriculació, en els moments que ha estat necessari, s'ha aconseguit flexibilitzar el disseny original per acomodar-se a les necessitats noves.

Resultat

El resultat ha estat un codi que satisfà tots els requeriments que s'han plantejat, tant inicialment, com durant el desenvolupament del projecte; del què es pot garantir que compleix les condicions establertes en els requeriments; del què en cas de continuar implementant-lo, es podrà comprovar que les condicions anteriors segueixen complint-se i que s'ha pogut crear en un total de 120 hores, que no és gaire desviat de l'apreciació inicial de 114.

Glossari

Metodologia àgil

Són cada un dels, i els diferents conjunts de, mètodes d'enginyeria basats en el desenvolupament iteratiu i incremental, que permeten que els requeriments d'un projecte evolucionin mentre es desenvolupa.

Scrum

És una metodologia àgil que es basa en un procés iteratiu i incremental concret, basat en *sprints* que s'alimenten d'un *backlog*, en la que l'equip de desenvolupament realitza una sèrie de reunions per cada iteració per planificar, revisar i fer-ne retrospectiva.

Extreme Programming

És una metodologia àgil que destaca més l'adaptabilitat que la previsió, basant-se en la simplicitat, la comunicació, la realimentació, coratge i humilitat.

Test driven development, TDD

És una tècnica que permet garantir que el codi que s'implementa basant-se en ella compleix els requeriments que s'especifiquin.

sprint

En la metodologia Scrum, és el nom que se li dóna a cada un de les iteracions que es realitzen.

backlog

En metodologia Scrum consisteix en el llistat complet de tasques unitàries que s'han de realitzar, completament desglossat, categoritzat, subdividit, estimat i prioritzat.

user story, US

És el nom amb el què es coneixen les definicions en llenguatge natural, és a dir, sense especificació formal, però prou ben detallades com per poder-se convertir en codi de validació.

kick off

És la reunió de llançament amb la que es comença cada iteració.

cloud

És el nom genèric que s'adjudica als serveis que es realitzen exclusivament des d'Internet, ocultant els serveis. Els serveis oferts

poden ser de molts tipus diferents, des de serveis d'emmagatzematge, com de correu electrònic o serveis de computació i càlcul.

Bibliografia

[Larman-Basili, 2003] 'Iterative and Incremental Development: A Brief History', Larman, C. and Basili, V.R., 2003, Computer, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1204375, a través de [Wikipedia, 1].

[Edmonds, 1974] 'A Process for the Development of Software for Nontechnical Users as an Adaptive System', General Systems, a través de [Wikipedia, 1].

[Wikipedia, 1] 'Agile software development', http://en.wikipedia.org/wiki/Agile_software_development, visitada el 12-10-2012.

[Takeuchi-Nonaka, 1986] 'New New Product Development Game', 1986, http://cb.hbsp.harvard.edu/cb/web/product_detail.seam;jsessionid=8D8BACDD4CC4F2F87B3CD58D2ED10332?R=86116-PDF-ENG&conversationId=22181&E=48834, a través de [Wikipedia, 2].

[Wikipedia, 2] 'Scrum (development)', http://en.wikipedia.org/wiki/Scrum_(development), visitada el 12-10-2012.

[Beck, 1999] 'Extreme Programming Explained', 1999, a través de [Wikipedia, 3]

[Wikipedia, 3] 'Extreme Programming', http://en.wikipedia.org/wiki/Extreme_Programming, visitada el 12-10-2012.

[Beck-et.al, 2001] 'Agile Manifesto', 2001, http://agilemanifesto.org/, visitada el 12-10-2012.

[Redmine] http://www.redmine.org, visitada el 29-09-2012.

[Redmine Y10K] http://redmine.y10k.ws, visitada el 29-09-2012.

[Tamrakar, 2012] 'Does The Use Of Fibonacci Numbers In Planning Poker Affect Effort Estimates?', Tamrakar, R. and Jørgensen, M., 2012, Simula research laboratory, http://simula.no/publications/Simula.simula.1282, visitada el 30-09-2012.

[Microsoft Community, 2010] 'Version control on Word 2010', http://answers.microsoft.com/en-us/office/forum/office_2010-word/version-control-on-word-2010/94c1b2dd-1e8b-4448-a23c-f9783b14be98?msgId=e4fc235a-4aac-42a7-9faa-302abb01de9f, visitada el 20-10-2012.

[Apple Support, 2009] 'How do I revert to a previously saved version of a document?', https://discussions.apple.com/thread/3821134?start=0&tstart=0, visitada el 20-10-2012.

[Novatech, 2012] 'Novatech MyXerver MX3684SN - Network Storage 1TB Black - Retail', http://www.novatech.co.uk/products/Components/HardDrives-External/NetworkAttached/Novatech/MX3686SN.html, visitada el 20-10-2012.

[SugarSync, 2012] 'Online backup - Online File Storage - Back Up Files - SugarSync', http://www.sugarsync.com/products/backup.html, visitada el 20-10-2012.

[Mercurial] http://mercurial.selenic.com, visitada el 29-09-2012.

[Mercurial Download] http://mercurial.selenic.com/downloads/, visitada el 20-10-2012.

[Mercurial WindowsInstall] http://mercurial.selenic.com/wiki/WindowsInstall, ciritada el 13-01-2013.

[TortoiseHg] http://tortoisehg.bitbucket.org/, visitada el 20-10-2012.

[TortoiseHg QuickStart] http://tortoisehg.bitbucket.org/manual/2.6/quick.html, visitada el 13-01-2013.

[VirtualBox] http://www.virtualbox.org, visitada el 20-10-2012.

[VirtualBox Install] http://www.virtualbox.org/manual/ch02.html, visitada el 13-01-2013.

[Vagrant] http://www.vagrantup.com, visitada el 29-09-2012.

[Vagrant Install] http://docs.vagrantup.com/v1/docs/getting-started/index.html, visitada el 13-01-2013.

[OIC Download] http://www.oracle.com/technetwork/database/features/instant-client/index-097480.html, visitada el 20-10-2012.

[Oracle Data Modeler] http://www.oracle.com/technetwork/developer-tools/datamodeler/downloads/datamodeler-087275.html, visitada el 20-10-2012.

[Oracle JDeveloper] http://www.oracle.com/technetwork/developer-tools/jdev/downloads/index.html, visitada el 20-10-2012.

[Chef] http://www.opscode.com/chef, visitada el 29-09-2012.

[VIN] http://es.wikipedia.org/wiki/N%C3%BAmero_de_chasis, visitada el 20-10-2012.

[Tablas PL/SQL] http://www.devjoker.com/contenidos/Tutorial-PLSQL/59/Tablas-PLSQL.aspx, visitada el 1-11-2012.

[Salaris] http://www.simplyhired.com/a/salary/search/q-oracle+database+designer, visitada el 10-1-2013.

[Oracle DB Hardware Requirements] http://docs.oracle.com/cd/B19306_01/install.102/b15667/pre_install.htm#i1011296, visitada el 10-1-2013.

[Sun Server X3-2 Small] https://shop.oracle.com/pls/ostore/cm_config_ui.config_frames?p_session=19754157182530 19&p_config_hdr_id=52328806&p_config_rev_nbr=1&p_install_country_id=1&p_store_app_id=6&p_return_apex_page=400&p_calling_apex_page=5&p_publication_mode=P&p_currency_code=USD, visitada el 10-1-2013.

[Oracle Price List] http://www.oracle.com/us/corporate/pricing/technology-price-list-070617.pdf, visitada el 10-1-2013.

[Oracle Certification] http://docs.oracle.com/cd/B19306_01/install.102/b15667/install_overview.htm#BGBEEGCB, visitada el 10-1-2013.

[Oracle Cloud] https://cloud.oracle.com/mycloud/f?p=service:database_pricing:0, visitada el 10-1-2013.

[Amazon AWS] http://aws.amazon.com/rds/oracle/, visitada el 10-1-2013.

[Amazon RDS Oracle Pricing] http://aws.amazon.com/rds/oracle/, visitada l'11-1-2013.