Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques
Universitat de Barcelona

PROCESSAT DE DADES I VISUALITZACIÓ EN DISPOSITIUS MÒBILS PER A EINES DE BI

Albert Gonzàlez Fonfria

Director: Enric Biosca Trias
Realitzat a: Departament de Matemàtica Aplicada i Anàlisi. UB

Barcelona, 14 de gener de 2013

Agraïments

Quan s'arriba al final d'un camí, agrair a totes aquelles persones que t'han ajudat arribar al final sol ser una tasca complicada perquè potser només amb la seva t'han animat a continuar un dia més fent feina. En especial, m'agradaria dedicar aquest projecte:

A Enric Biosca , el meu tutor, per la seva dedicació i sobretot paciència. El món que entres al endinsar-te al BI (Business Intelligence) a vegades pot fer-te desesperar al no saber trobar un camí o solució. En tot moment ha estat disposat a donar-me consells, no només per BI sinó també alhora de crear la interfície per l'usuari.

A Yaiza Gàlvez de la Hera per estar sempre al meu costat, animar-me i riure de les bromes sense sentit. No només ara sinó durant tota la carrera.

A tota la meva família, en especial als meus pares, al meu germà i a la meva gossa. Massa vegades hauria abandonat sinó fos per tots ells.

A tots els companys que m'han acompanyat en aquesta etapa i hem fet nostre una senzilla sala d'estudis: Carles Riera, Jordi Serral, Pau Almirall, Antoni Rocafort (Roka), Francisco Villalba (Paco), Pòlit Miró, Guillem Vila, Ernest Pastor, Alvaro Vidal, Cèsar Blesa i Xavier Sancho.

Als meus amics Anna Alverola, Irene Casals, Meri Girós, Roger Uterga, Carlota Esteller, Jordi Bosch, Roger Alvarez, Monica Richart, Patri Zarco, Maria Belza i molts més.

Sense vosaltres no hagués estat possible, moltíssimes gràcies.

Resum

Aquest projecte mostra el procés de desenvolupar una solució per presentar dades a usuaris de dispositius mòbils de forma genèrica tenint present en tot moment la portabilitat a altres plataformes tant de Business Intelligence com de dispositiu mòbil.

Aquest solució implementada utilitza eines Open Source de Business Inteligence per l'extracció i processat de dades, en particular Pentaho. En l'apartat de la visualització dades en els dispositius mòbils s'ha desenvolupat una aplicació sobre el sistema operatiu Android.

L'aplicació android dissenyada busca ser la interfície d'usuari de l'eina BI. Aquest fet porta a realitzar un disseny exhaustiu al voltant de l'usuari. A més a més, aquesta aplicació ha estat implementada seguint els patrons de responsive design per adaptació a tauletes i la nomenclatura Open Source d'Android per facilitar una futura continuïtat del projecte.

Abstract

This project demonstrates the process of developing a solution to present data to users of mobile devices taking into account future portability to other platforms of Business Intelligence or mobile device.

The implemented solution uses Open Source Business Intelligence tools for the extraction and processing of data, in this case Pentaho. In the section on data visualization in mobile devices has developed an application on the Android operating system.

The android application designed wants to be a UI BI tool. This leads to a user-centered design. In addition, this application has been implemented using responsive design patterns to adapt to tablets and Android Open Source nomenclature to facilitate future continuity of the project.

Índex de continguts

1 Introducció.

1.1 Àmbit del projecte

Per la realització d'aquest projecte s'ha d'aprofundir en:

- Com s'emmagatzemen grans quantitats de dades i com extreure informació d'aquestes.
- Com es poden extreure.
- Com es poden presentar a l'usuari adaptat als dispositius mòbils.

Per tant, aquest projecte intenta fer ús dels coneixements sobre bases de dades i interfícies d'usuari o factors humans, adquirits durant els estudis del grau en informàtica.

1.2 Motivació

La presentació de dades molt sovint es realitza sobre diferents plataformes amb tecnologies estàtiques o gens estandaritzades. El software Flash d'Adobe i tot el seu entorn de desenvolupament seria un dels grans exponents en aquest camp. S'ha de tenir en compte que aquest no és reproduïble als dispositius mòbils més habituals.

La idea d'aquest projecte es trobar una solució oberta, dinàmica i que pugui assolir el màxim de plataformes possibles amb els mínims canvis.

La meva motivació personal alhora de fer la proposta de projecte i realitzar-lo centrant-me en l'usuari és la meva intenció d'orientar el meu futur professional cap al desenvolupament mòbil i front-end.

1.3 Objectiu principal.

L'objectiu principal d'aquest projecte és crear una arquitectura bàsica que es pugui replicar en qualsevol plataforma alhora de extreure i presentar dades mitjançant

eines BI.

Aquesta arquitectura s'haurà de basar en procediments bàsics i comuns en la majoria de plataformes mòbils fent que la solució trobada no depengui d'una tecnologia exacte ja que aquestes queden obsoletes amb el pas del temps.

1.4 Objectius específics

- Entendre que és Business Inteligence (BI) i les estructures bàsiques que la configuren.
- Entendre la dualitat del desenvolupament d'aplicacions natives o genèrica.
- Construir una arquitectura base de comunicació entre la eina BI i el dispositiu mòbil.
- Crear una aplicació per dispositius mòbils de presentació de dades centrada en l'usuari.
- Consolidar els coneixements adquirits durant la carrera i aprendre a fer un projecte des de zero.

1.5 Organització de la memòria

La memòria està estructurada en sis blocs ben diferenciats:

1. Introducció i antecedents.

Aquest correspon a la part on ens trobem actualment, es realitza un breu resum del contingut de la memòria i es mostren els coneixements previs necessaris per la realització del projecte.

2. Anàlisi.

Tal com indica el nom del bloc es realitza l'anàlisi de la solució. Estructura bàsica, casos d'ús, anàlisi d'usuaris i tasques, i model de domini.

3. Disseny.

En aquest bloc es dissenyara l'arquitectura de la solució, els elements necessaris de l'eina BI per mostrar i extreure dades, i l'aplicació per a dispositius mòbils. Dins del disseny de l'aplicació s'inclou el disseny de la interfície de l'usuari amb els diferents prototips i la navegabilitat.

4. Implementació i valoració.

En aquest bloc s'explicaran les consideracions i solucions trobades alhora de realitzar la implementació del disseny, i també la valoració econòmica de la solució. Dins l'apartat de l'aplicació del dispositiu mòbil es diferenciarà la part de la interfície de l'usuari i el model.

5. Conclusions.

En aquest bloc es mostraran les conclusions i millores extretes durant l'elaboració d'aquest projecte.

6. Referencies bibliogràfiques i annexes.

En l'últim bloc s'esmentaran les diferents fonts consultades i s'afegeix un recull de manuals d'usuari tant de l'eina BI com del dispositiu mòbil.

2 Antecedents.

En aquest projecte necessitem entendre com es genera la informació i com es pot portar als dispositius mòbils per mostrar-la als usuaris. Per dur a terme aquesta tasca és necessari entendre que és Business Intelligence, que són les dades, el coneixement i la informació, on hem d'anar a buscar les dades que volem presentar al usuari.

També s'ha d'entendre les limitacions dels dispositius mòbils, i les complicacions que poden sorgir al llarg del temps.

Per acabar s'ha d'esbrinar com està la situació actual en la presentació de dades. Les grans empreses són les pioneres en aquest camp, grans consultores especialitzades en BI .

2.1 Intel·ligència empresarial o Business Inteligence.

La intel·ligència empresarial té tres grans pilars: la informació, l'emmagatzemament i l'anàlisi.

2.1.1 Dades, informació i coneixement.

El primer problema que ens trobem quan entrem al món de l'anàlisi de dades és la mala diferenciació que es fa entre els termes dades, informació i coneixement. És molt habitual en converses trobar-se barrejats aquests termes sense que realment s'utilitzin correctament.

Una forma senzilla de diferenciar-ho es donar context a cada element. Per exemple, les dades estan localitzades al món, el coneixement està situat en els agents, ja siguin éssers vius, empreses, màquines... i la informació adopta un paper de mitjancer entre tots dos.

Dades.

Les dades són la mínima unitat semàntica i es corresponen amb elements primaris d'informació que per si sols són irrellevants per la presa de decisions. També es pot definir com un conjunt discret de factors objectius sobre un fet real. Si portem aquesta definició a un context empresarial, una dada seria un registre de transaccions de qualsevol tipus. Llavors podem veure que una dada no diu res del perquè de les coses, i per si sola té poca o cap rellevància o propòsit.

Tots els sectors generen dades, ja siguin bancs, la seguretat social, el govern, petites empreses... el problema sorgeix quan no es gestionen bé les dades, sobretot en grans empreses ja que operen amb milions de transaccions diàries. Si aquestes organitzacions guarden masses dades sense sentit es complicarà la tasca d'identificar quines són rellevants i com ja s'ha explicat, elles soles no tenen cap significat.

Per exemple, un número de telefon, el nom d'una persona, un import d'una compra... són dades que sense un context, una utilitat o un propòsit no serveixen per prendre una decisió. Les tecnologies de la informació han ajudat molt a la recopilació d'informació, només el transit en un lloc web comercial poden donar lloc a milions de dades.

Les dades poden venir de moltes fonts externes o internes a la organització, aquestes poden ser objectives, subjectives, quantitatives... però només descriuen únicament una part de la realitat però no proporcionen interpretacions. Per tant, la presa de decisions es basarà en dades, però aquestes no diran que fer. Això si, sempre s'ha de recordar que la informació és poder i les dades són la base de la informació.

Informació.

El concepte informació es podria descriure com un missatge i com qualsevol missatge, té emissor i receptor. En aquest cas qui decideix si el missatge que rep l'està informant és el receptor i no l'emissor. Si es genera un informe amb taules

inconnexes, el receptor pot considerar que el que se li està entregant és soroll.

La informació també es pot definir com un conjunt de dades processades que tenen un significat (rellevància, context i propòsit), i que per tant tenen valor per qui ha de prendre decisions. Les dades es poden transformar en informació afegint valor:

- **Contextualitzant:** es sap en quin context i per quin propòsit es van generar.
- **Categoritzant:** es coneixen les unitats de mesura que ajuden a interpretar-los.
- **Calculant:** les dades poden haver estat calculades matemàticament o estadísticament.
- **Corregint:** s'han eliminat errors i inconsistències de les dades.
- **Condensant:** les dades s'han pogut resumir de forma més concreta.

Per tant, la informació és la comunicació de coneixements o intel·ligència, i és capaç de canviar la forma en que el receptor percep alguna cosa, impactant sobre els judicis de valor i el seu comportament.

Informació = Dades + Context +Utilitat

Coneixement.

Intuïtivament es té la sensació que el coneixement es alguna cosa més amplia més profunda i rica que les dades i la informació.

El coneixement és una barreja d'experiència, valors, informació i saber fer que serveix com a marc per la incorporació de noves experiències i informació. Sovint el coneixement no només es troba dins de documents o magatzems de dades, sinó que també està en rutines organitzatives, processos, pràctiques i normes.

El coneixement deriva de la informació, així com la informació deriva de les dades. Per tant, per aconseguir que la informació es tradueixi en coneixement s'han de realitzar certes accions:

- **Comparació amb altres elements.**
- **Predicció de conseqüències.**
- **Cerca de connexions.**
- **Conversa amb altres portadors de coneixement.**

2.1.2 Datawarehouse.

Un datawarehouse és una base de dades, normalment corporativa o orientada a un determinat àmbit, que es caracteritza per integrar i depurar informació d'una o més fonts per després processar-la i així poder analitzar-la des d'infinitat de perspectives i amb grans velocitats de resposta. Aquesta col·lecció de dades està integrat, no és volàtil ni variable en el temps i ajuda a la presa de decisions.

Es pot imaginar com un expedient complet d'una organització, més enllà d'informació transaccional i operacional, emmagatzemat en una base de dades dissenyada per afavorir l'anàlisi i la divulgació eficient de dades.

Segons un dels primers autors en escriure sobre datawarehouse, Bill Inmon, va definir-lo en termes de les característiques del repositori de dades:

- **orientat a temes.** Les dades s'organitzen per temes per facilitar el seu accés i enteniment per part de l'usuari final. Per exemple, totes les dades sobre clients poder ser consolidats en una única taula.
- **Variant en el temps.** Es carrega amb una variable de temps per poder fer comparacions i anàlisi de tendències.
- **No volàtil.** La informació es permanent, quan s'afegeixen nous valors mai es fa cap acció sobre els que ja existien.
- **Integrat.** Les dades han d'integrar-se en una estructura consistent, així que les inconsistències han de ser eliminades.

També s'han de tenir en compte en la definició els mitjans per obtenir i analitzar aquestes dades, extreure'ls, transforma-los i carregar-los així com les diferents formes per realitzar la gestió de dades.

Un datawarehouse es vol que contingui dades que són necessàries o útils per una entitat, es a dir, que es vol utilitzar com un repositori de dades per posteriorment transformar-los en informació útil per l'usuari.

El funcionament d'aquest hi ha dos idees molt important:

- **Integració** de les dades de les diferents bases de dades de la entitat són sovint d'estructura heterogènia. S'ha de facilitar una descripció global i un anàlisis comprensiu de tota l'organització.
- **Separació** de les dades utilitzades en operacions diàries de les dades emmagatzemades.

Les principals aportacions d'un datawarehouse són:

- proporciona una eina per prendre decisions en qualsevol àrea funcional i basant-nos en informació integrada i global de l'entitat.
- Facilita l'aplicació de tècniques estadístiques d'anàlisis i modelització per trobar relacions amagades entre les dades del magatzem.
- Proporciona la capacitat d'aprendre de les dades del passat i preveure situacions futures.
- Suposa una optimització tecnològica i econòmica en entorns de centre d'informació, estadística o de generació d'informes amb un gran retorn de la inversió.

Cubs d'informació.

La unitat més important dels datawarehouse són els *cubs d'informació o cubs OLAP.* Funcionen com un cub de rubbik on s'ha de resoldre per veure un costat complet. En el cas dels cubs d'informació s'han d'organitzar les dades per taules o

relacions, les cubs OLAP tenen un número indefinit de dimension per això també se'ls anomena *hipercubs.* Un cub d'informació contindrà dades d'una determinada variable que es vol analitzar, proporcionen una vista lògica de les dades.

2.1.3 Data mining.

El data mining (mineria de dades), és el conjunt de tècniques i tecnologies que permeten explotar grans bases de dades amb l'objectiu de trobar patrons repetitius que expliquin el comportament de les dades en un determinat context.

La mineria de dades sorgeix per intentar ajudar a comprendre el contingut d'un datawarehouse utilitzant tècniques estadístiques i algoritmes de cerca propers a la intel·ligència artificial i les xarxes neuronals.

El procés bàsic de la mineria de dades sol composar-se de quatre etapes.

- Determinació dels objectius.
- Preprocessament de les dades.
- Determinació del model.
- Anàlisis dels resultats.

La etapa més important i costosa es el preprocessament de les dades que sol constituir més de la meitat del temps en la majoria de projectes de mineria de dades.

2.1.4 Conclusions.

Podem definir la intel·ligència empresarial com l'habilitat de transformar les dades i registres bàsics d'una organització analitzant-les un cop ben emmagatzemades.

Per tant, si volem mostrar a usuaris informació sobre una organització s'haurà de crear un procés previ de generació d'informació i coneixement perquè puguin ser interpretada per l'usuari.

L'usuari en cap moment ha d'accedir als registres bàsics d'una organització ja que no tenen cap context ni utilitat per ell.

Per facilitar l'obtenció d'informació es construeix un magatzem on "classificar" les dades segons els criteris esmentats anterior, per tal de facilitar l'anàlisi d'aquestes.

Les bases de consulta i generació d'informació són els cubs OLAP o hipercubs.

2.2 Aplicacions natives vs webapps en dispositius mòbils.

Quan decidim crea una aplicació per dispositius mòbils s'han de plantejar certes qüestions: quins i quants usuaris tindrem, com accediran, a quins usuaris va enfocat, quins recursos necessitem, quin cost suposarà disposar de diferents plataformes per accedir-hi.

Decantar-se per un camí o un altre pot tenir conseqüències i limitacions. Principalment hi ha tres punts de vista a l'hora de dissenyar una aplicació per un dispositiu mòbil: aplicació nativa, aplicació web o aplicació híbrida.

2.2.1 Situació actual.

Per intentar veure cap on hem d'anar amb la nostra intenció de fer una aplicació per dispositius mòbil hem de veure com està el mercat actual, quines plataformes hi ha, tendències dels usuaris...

Empreses que es dediquen analitzar mercats i tendències poden fallar amb les seves prediccions, per tant, no es poden fer afirmacions de com es transformarà el mercat de les aplicacions mòbils i web sense córrer el risc a cometre un error. Per això aquest projecte s'inicia pensant en una viabilitat una mica superior a un trimestre evitant haver de fer una predicció més exhaustiva amb la possibilitat de generar més errors.

La primera mètrica o filtre que s'aplica a la situació actual és descartar tots els

mòbils que no són *smartphones*. El motiu és que si volem comparar un aplicació web en HTML5 i l'aplicació nativa, el dispositiu ha de ser capaç de connectar-se a Internet mitjançant un navegador estandaritzat ja que sinó ens trobaríem amb un mercat dominat encara per el sistema operatiu de Nokia, Symbian OS.

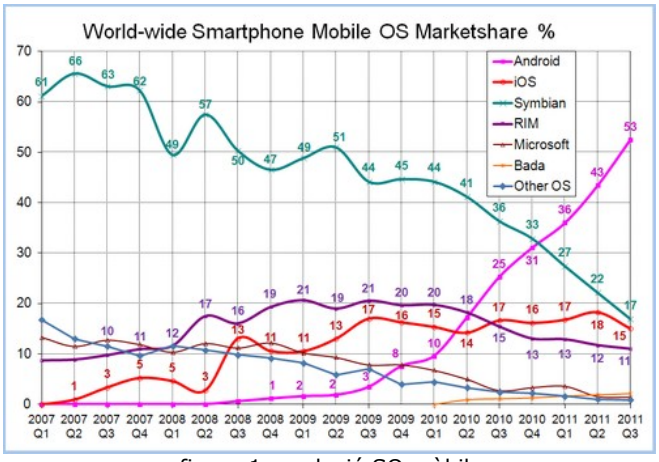


figura 1. evolució SO mòbils.

Hi ha molts estudis, anàlisis i gràfiques on es centren en la marca fabricant del dispositiu però el que ens interessa són els sistemes operatius que dominen el mercat.

Observant la gràfica ens adonem que estem davant un mercat molt canviant i fragmentat on les tendències varien ràpidament. Només es reflecteixen els quatre últims anys on el mercat estava dominat per les PDA dels sistemes operatius de Microsoft i Nokia i ha evolucionat al domini d'android OS i IOS que han aparegut en aquest lapse de temps.

Durant aquesta transició RIM i el seu SO van dominar certa part del mercat i sobretot el mercat més empresarial encara molt marcat per la empresa canadenca.

Veien aquesta evolució del mercat ja ens podem adonar que si pretenem fer una aplicació que duri 4 anys en una sola plataforma pot ser una decisió molt arriscada.

2.2.2 HTML5 als dispositius mòbils.

HTML5 és la cinquena versió del llenguatge de marques o "tags" bàsic a la WWW. El desenvolupament d'aquest llenguatge està regulat pel consorci W3C on és el primer cop en que les dues branques principals, HTML i XML, han avançat de forma paral·lela.

HTML5 avui en dia encara no està estandaritzat, s'espera per meitats de 2014, però les últimes versions dels principals navegadors són compatibles amb la gran majoria de innovacions que apareixen en aquesta nova versió.

Si volguéssim fer un breu resum de que significa la nova versió del llenguatge HTML podríem afirmar que *HTML5 = HTML + CSS + JAVASCRIPT.*

Aquesta nova versió és el primer suport a la web semàntica o web 3.0. Al afegir-se etiquetes com *header, footer, article, nav, time o link rel* es permetrà descriure quin és el significat del contingut ja sigui la seva importància, la seva finalitat o les relacions que existeixen. Aquestes novetats semàntiques no afecten a la visualització sinó que obren un nou món als cercadors.

Però HTML5 també dona suport:

- Geolocalització
- Inclinació mesurada per sensors.
- Càmera del dispositiu.
- Enregistrament i reproducció de so.
- Etiqueta "canvas" es poden afegir fàcilment gràfics vectorials.
- Connexions via socket donant la possibilitat d'accedir a arxius XML.
- Emmagatzemen local.

Donant un cop d'ull als principals dispositius mòbils, i en especial atenció als seus sistemes operatius ens adonem de la gran varietat que existeix i sota quins motors treballen. Per exemple:

- Android té Dalvik.
- IOS té Objective-C.
- Windows Mobile 7 té XNA/silverlight.
- Blackberry té java.

Tots quatre en donen suport a HTML5. Fins i tot, si busquem altres dispositius mòbils menys estesos en el mercat com Meego, WebOS, Kin OS, Zun OS, BB 10 tots ells donen també suport a HTML5. Si es vol fer una aplicació per diverses plataformes es evident que millor desenvolupar un cop és millor que quatre. Per tant, necessitarem menys recursos econòmics, coneixement i humans, i al cap i a la fi serà més ràpid.

2.2.3 Aplicació nativa.

Una aplicació nativa és una aplicació íntegrament desenvolupada per a un sistema operatiu utilitzant, si existeix, el SDK pertinent. Per utilitzar aquest SDK o publicar l'aplicació perquè estigui disponible pels usuaris del sistema operatiu pot tenir un cost associat segons en la plataforma que treballem.

Aquesta aplicació només estarà disponible per aquest sistema operatiu. Existeixen aplicacions externes que permeten exportar aplicacions d'un sistema operatiu a un altre però totes tenen defectes i poden no donar suport a funcionalitats del dispositiu.

Alhora d'implementar una aplicació nativa es té accés a tot el hardware del dispositiu, i l'accés aquest estarà estandaritzat segons el sistema operatiu. Aquest hardware pot donar noves funcionalitats a l'aplicació com per exemple:

- Multi-touch.
- Acceleròmetres.
- Micròfons.

- Vídeos.
- Brúixola.
- Aplicacions en background.
- Bluetooth.
- suport a diferents pantalles.
- Giroscopis.
- càmeres frontals.
- Baròmetre.
- NFC.
- Accessoris USB.

Els dispositius evolucionen a gran velocitat i s'afegeixen noves funcionalitats. Per exemple, la tecnologia NFC va sorgir a principis de 2011 pels dispositius android. Un any després les aplicacions que l'utilitzen van sorgint cada cop de forma més regular fins que es determini que tots els dispositius amb aquest sistema operatiu han de tenir aquest estandart implementat en el seu hardware.

Una dels punts forts de les aplicacions natives són la generació de gràfics, les llibreries gràfiques de baix nivell estan presents a la gran majoria de dispositius i sinó es disposen de frameworks especialitats en aquesta tasca. Mica en mica sorgeixen propostes de aplicacions externes de generació de gràfics on l'usuari només ha de rebre un stream de dades amb la visualització i enviar les accions.

2.2.4 Aplicacions híbrides.

Una aplicació híbrida és un intent d'utilitzar els avantatges que tenen les aplicacions web i les aplicacions natives per construir una aplicació més òptima.

L'enfoc que es sol seguir al intentar construir-la és generar la navegabilitat de forma nativa i fer les crides dins l'aplicació HTML5 amb javascript.

Si agafem com a base d'exemple una aplicació android, s'implementaria la navegació i la presentació es realitzaria amb un element webview realitzant una crida HTTP. Amb l'ajuda d'una interfície gestionada des de l'aplicació mòbil es gestionaria l'aplicació web.

Llavors el resultat que tindríem seria una navegació fluida i nodrida en possibles esdeveniments a més de l'accés a totes les funcionalitats del dispositiu com inclinació, brúixola, bluetooth, etc. Funcionalitats que normalment, per seguretat, des d'un navegador no es poden accedir.

Alhora de desenvolupar l'aplicació en altres plataformes, l'únic que canvia és la navegabilitat estalviant-nos molts recursos en el procés de desenvolupament que podríem dirigir a la millora de l'aplicació en HTML5. Per exemple, tant IOS i BB també posseeixen un element idèntic al webview d'android.

2.2.5 Conclusions.

Alhora d'escollir un dels tres enfocaments, s'han d'avaluar el cost econòmic i temporal, la tecnologia necessària i els recursos disponibles per dur-ho a terme.

Les aplicacions natives són potents i eficients mentre que les aplicacions web són flexibles.

Les aplicacions híbrides sorgeixen com un intent d'aconseguir una millora respecte cada una sense estar a l'alçada dels punts forts.

Una de les frases que crec que millor descriuen el disseny d'aplicacions web i híbrides per dispositius mòbils és: *si no creus que podries millorar-la fent-la nativa es que no ho has pensat suficient. (google i/o 2011).*

2.3 Quadres de comandament o dashboards.

La presentació de dades en una solució d'intel·ligència empresarial sol venir expressada en informes bàsics o en dashboards

Dashboard en anglès es sol utilitzar per descriure el tauler de control d'un cotxe. Es va adoptar la terminologia dashboard per crear una metàfora comparativa amb el tauler de control dels cotxes on es recull tota la informació d'aquest. Aquesta metàfora facilita a l'usuari entendre el seu propòsit i funcionament.

Figura 2. Exemple dashboards de microstrategy.

Com es pot veure sobre aquestes línies, un dashboard no és res més que una pàgina web on es mostra, potser en temps real, informació de la entitat que representa on aquesta pot venir de diferents fonts. Per tant, la funcionalitat principal d'un quadre de comandament es mostrar informació sobre la situació de la empresa en un determinat instant. Un dashboard reuneix diferents elements o widgets per mostrar aquesta informació de forma ordenada per l'usuari.

Es sol anomenar quadre de comandament perquè sol ser habitual que qui només hi accedeixi siguin persones que formin part de la directiva o de l'executiu de l'entitat ja que la informació que es busca donar és general sobre l'estat i els objectius de l'entitat.

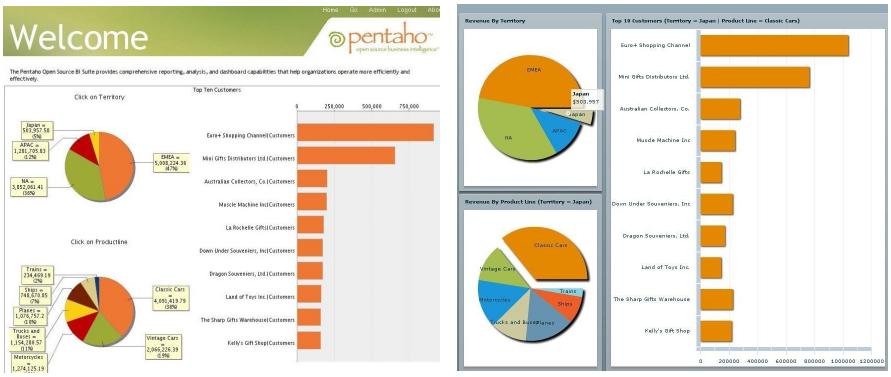


Figura 3. Exemple de dahsboards de pentaho.

Figura 4. Widgets presents a dashBoards.

2.4 Conclusions

Busines Intelligence es sol aplicar en grans entitats on la generació de dades bàsiques és enorme. Normalment es creu que com més dades millor, però com més dades més gran haurà de ser el preprocés d'aquestes fent que el termini d'implantació de nous elements sigui més llarg.

La solució de Business Intelligence per petites empreses pot semblar incoherent amb les definicions genèriques d'anàlisis de dades però si s'escala la tecnologia a la dimensió de l'entitat es poden aconseguir coneixements i informació que abans no sorgeixen en un primer cop d'ull.

La monitorització constant de l'estat de l'empresa i objectius afavoreix la presa de decisions podent reaccionar més aviat en situacions errònies.

Les aplicacions per dispositius mòbils s'ha de dissenyar pensant en els requisits

mínims que necessitem i l'amplitud de dispositius que volem utilitzar.

Les aplicacions natives repercuteixen sobretot en la qualitat i el cost final d'aquesta.

Si no pots millorar una aplicació web o híbrida fent-la nativa es que no ho has pensat suficientment bé.

En aquest projecte s'implementarà una aplicació híbrida per Android deixant la generació de dades al servidor de BI.

El servidor BI utilitzarà llibreries en HTML5 per la presentació de dades.

3 Anàlisi.

3.1 Estructura bàsica de la solució.

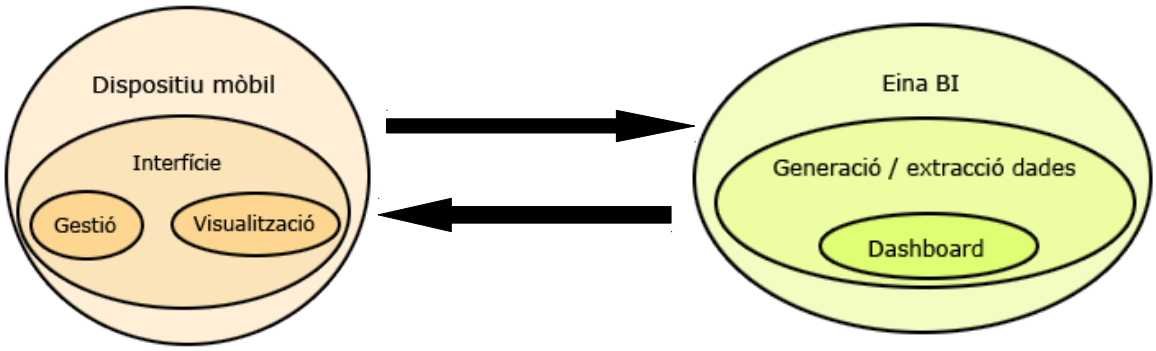


Figura 5. Estructura bàsica de la solució.

En aquest projecte s'implementarà una aplicació híbrida on el dispositiu mòbil haurà de gestionar la interfície de l'usuari i en el servidor es durà a terme la gestió de dades. Si agafem com a referencia el MVC, també anomenat model-view, la vista es portaria a terme en el dispositiu mòbil i el model estaria al servidor.

En el dispositiu mòbil s'haurà de gestionar totes les dades, usuaris i rols, i primordialment la visualització de dashboards. Accessibilitat i usabilitat seran el nucli sobre el que es construirà l'aplicació, ja que el que es busca és donar a l'usuari una experiència fàcil i intuïtiva

En el servidor es generaran les dades que es mostraran a l'usuari segons la seva demanda.

Un aspecte important de la solució que s'està cercant amb aquest projecte és intentar dependre el mínim possible de les tecnologies ja que aquestes evolucionen i també desapareixen. És a dir, l'accés a les dades d'un servidor BI es pot fer de moltes maneres. El mateix passa amb els dispositius, hi ha de molts tipus. El que es busca amb aquest projecte es utilitzar unes eines per trobar una solució per poder mostrar dashboards als usuaris.

3.2 Casos d'ús.

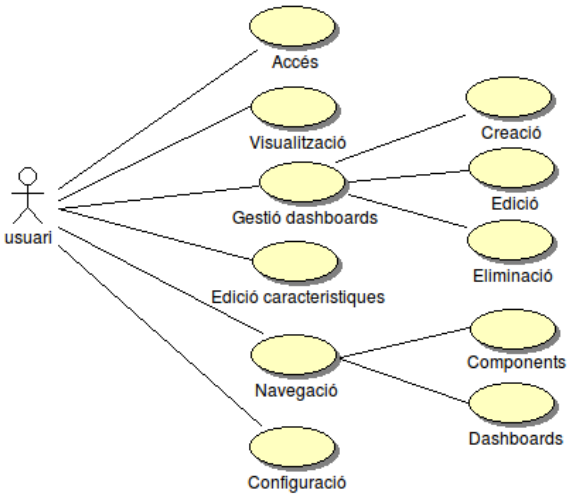


Figura 6. Diagrama de casos d'ús.

Un recull de casos d'ús d'aquesta aplicació seria:

- **accés restringit.** L'usuari haurà d'identificar-se per accedir a l'aplicació. Aquest accés haurà de ser persistent a petició de l'usuari. En aquest procés d'accés es gestionaran els rols dels usuaris ja que els permisos adjunts a cada un d'ells podran diferir.
- **Visualització de components.** L'usuari podrà visualitzar components per separat.
- **Edició de les característiques dels components.** L'usuari podrà personalitzar els components al seu gust. Aquesta personalització vindrà definida tal com visualització de títol, llegenda, canvi de color o gràfic.
- **Navegació de components.** L'usuari navegarà entre els diferents components preestablerts.
- **Creació, edició i eliminació de dashboards.** L'usuari podrà crear, editar i eliminar dashboards personalitzats d'un o més components sempre i quan tingui permisos.
- **Edició dels components d'un dashboard.** L'usuari podrà editar les

característiques dels components d'un dashboard fent que aquest siguin permanents cada cop que es visualitzi.

- **Navegació entre dashboards.** Tal i com amb els components l'usuari navegarà entre els diferent dashboards creats.
- **Configuració de les dades d'accés al servidor BI.** L'usuari podrà modificar les dades d'accés al servidor BI.

3.3 Anàlisi d'usuaris.

Observant els casos d'us es pot veure que existeix un únic usuari bàsic que tindrà més accés a funcionalitats segons el rol. Fent un paral·lelisme amb l'orientació a objectes tenim una classe d'usuari bàsic que es abstracte i que els diferents rols tindran accés a diferents funcionalitats però tots compartirien les bàsiques.

Característica	**Usuari bàsic abstracte.**
Edat	18-65
Gènere	Indiferent
Limitacions físiques	Pot presentar alguna limitació per escoltar o precisió, sent un cas aïllat. La limitació visual no ha de ser un impediment en la navegació normal d'un dispositiu mòbil.
Nivell educatiu	Irrellevant (normalment serà superior)
Experiència amb el dispositiu	Pot tenir poca experiència.
Experiència amb l'entorn de treball	Pot tenir poc o nul coneixement de la plataforma de Business Inteligence.
Motivació	Vol veure i accedir ràpidament a les dades, sense necessitar un processament elevat d'aquestes.
Actitud	Pot ser estressada.

Per fer una separació d'usuaris segons a les funcionalitats o tasques a les que podran accedir, es dividiran els usuaris en:

- **Usuari bàsic.**
- **Usuari administrador.**

3.4 Anàlisi de tasques.

Totes les tasques que pot realitzar l'usuari bàsic són extensibles a l'usuari administrador ja que els seus permisos estan per sobre dels del primer.

3.4.1 Usuari bàsic.

Tasca 1: accedir a l'aplicació.

Precondicions:

- Direcció al servidor pentaho.
- Nom d'usuari
- Paraula clau

Subtasques:

- Introduir la direcció del servidor ja sigui per ip o dns. (opcional)
- Introducció de les dades d'accés.
- Escollir si es vol accés persistent. (opcional)
- Acceptar.

Tasca 2. Visualització d'un component o dashboard.

Precondicions:

- Accedir a l'aplicació.
- Tenir accés algun component o dashboard segons el teu rol.

Subtasques:

- Seleccionar l'element a visualitzar.

Tasca 2b. Personalització component.

Precondicions:

- Mateixes que tasca 2.

Subtasques:

- Canviar color (opcional).
- Canviar gràfica (opcional).
- Canviar títol (opcional).
- Canviar llegenda (opcional).
- Moure llegenda (opcional).

Tasca 3. Navegar entre components o dashboards.

Precondicions:

- Accedir a l'aplicació.
- Seleccionar un component o dashboard.

Subtasques:

- Seleccionar o canviar de component o dashboard.

Tasca 4. Eliminar/Carregar totes les dades.

Precondicions:

- Accedir a l'aplicació.

Subtasques:

- Confirmar la eliminació de totes les dades actuals.

Tasca 5. Canviar d'usuari.

Precondicions:

- Accedir a l'aplicació.

Subtasques:

- realitzar logoff manual.

Tasca 6. Carregar Dashboard.

Precondicions:

- Accedir a l'aplicació.
- Tenir accés a tots els components que forma el dashboard.

Subtasques:

- Seleccionar el dashboard a carregar.
- Modificar nom dashboard (opcional).

Tasca 7. Ordenar dashboards.

Precondicions:

- Accedir a l'aplicació.
- Disposar de dashboards.

Subtasques:

- Arrossegar el dashboard que desitgis a la posició que vulguis.
- Confirmar la modificació.

3.4.2 Tasques usuari administrador.

Tasca 1. Creació de dashboards personalitzats.

Precondicions:

- Accedir a l'aplicació.
- Tenir el rol adient.

Subtasques:

- Introduir nom identificatiu.
- Seleccionar la estructura del dashboards.
- Seleccionar els components necessaris segons la estructura escollida.

Tasca 2. Edició dashboards personalitzats.

Precondicions:

- Accedir a l'aplicació.
- Crear o rebre algun dashboard personalitzat.

Subtasques:

- Seleccionar el dashboard que es vol modificar.
- Canviar nom dashboard. (opcional).
- Canviar components del dashboard segons estructura.
- Confirmar.

Tasca 3. Eliminar dashboards.

Precondicions:

- Accedir a l'aplicació.
- Crear o rebre algun dashboard personalitzat.

Subtasques.

- Seleccionar el dashboard que es vol eliminar.
- Confirmar.

Tasca 4. Compartir dashboards personalitzats.

Precondicions:

- Accedir a l'aplicació.
- Crear o rebre algun dashboard personalitzat.

Subtasques.

- Seleccionar el dashboard que es vol compartir.
- Introduir direcció de correu electrònic.

Tasca 5. Personalitzar component d'un dashboard.

Precondicions:

- Accedir a l'aplicació.
- Crear o rebre algun dashboard personalitzat.

Subtasques.

- Seleccionar el dashboard que es vol compartir.
- Seleccionar el component que es vol editar.

3.5 Model de domini.

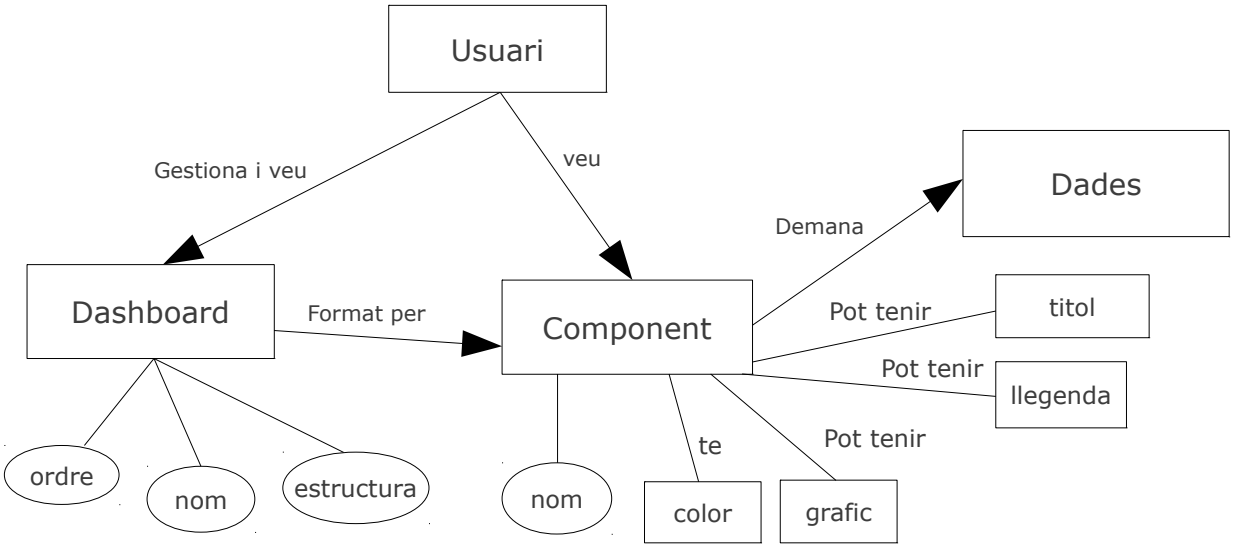


Figura 7. Diagrama de domini.

3.6 Requeriments.

3.6.1 interfície.

Després d'analitzar els possibles usuaris de l'aplicació els dos únics punts que s'estableixen com a requeriments són:

- **Possible desconeixement de la plataforma de BI.** Centrar la interfície en una reproducció de l'actual per mantenir una coherència i estandarització no seria adient ja que l'usuari molt probablement no treballi amb ella o només la utilitzi per veure els informes i dashboards ja subscrits.
- **Possible actitud estressada i necessitat d'accedir i navegar ràpidament.** La interfície ha de tenir pocs nivells alhora d'accedir a les dades i la navegació entre aquestes ha de ser immediata, intentant eliminar la repetició d'accions.

3.6.2 Tasques usuari bàsic.

Tasca 1: accedir a l'aplicació.

L'usuari haurà d'introduir les seves dades d'accés. En tot moment se li haurà de donar la possibilitat de modificar les dades d'accés al servidor bi, com poder escollir si vol o no que el seu accés sigui permanent.

- *Learnability:* s'haurà d'utilitzar la estructura habitual en qualsevol login d'una aplicació, ja sigui mòbil o web.

- *Error prevention*: s'haurà avisar a l'usuari en cas que falti alguna de les dades o en cas que siguin errònies. També s'haurà d'avisar a l'usuari en cas de no tenir accés a internet.

- *Security*: la paraula clau ha d'estar codificada perquè una persona al seu voltant no la pugui veure.

Tasca 4. Eliminar/Carregar totes les dades.

L'usuari ha de poder poder restablir totes les dades bàsiques, podent-li dir netejar la cache de la aplicació. S'ha de tenir en compte que es una acció que no es reversible.

- *Error prevention:* l'usuari haurà de confirmar aquesta tasca ja que al no ser

reversible totes les dades modificades o generades es perdran.

Tasca 7. Ordenar dashboards.

L'usuari haurà de poder ordenar al seu gust els dashboards al que tingui accés o hagi creat.

- *Error prevention*: l'usuari haurà de confirmar els canvis realitzats a l'ordre i també tindrà la possibilitat de deixar-los com estaven inicial-ment.

3.6.3 Tasques usuari administrador.

Tasca 1 i 2. Creació/edició de dashboards

L'usuari podrà crear o editar dashboards amb els components que ell vulgui seguint una estructura definida. Haurà de poder accedir a tots els seus components.

- *Learnability*: l'estructuració de com aniran els components ha de ser la mateixa que quan visualitzi el dashboard sencer.

- *Visibility*: l'usuari ha de veure en tot moment quin component o components a escollit.

- *Error prevention*: l'usuari podrà desfer tots els canvis que vulgui, des de canviar el nom del dahboard, els components i la estructura, aquest últim només en la creació, tants cops com vulgui sense haver de tornar de començar de nou.

Tasca 3. Eliminar un dashboard.

L'usuari podrà eliminar de forma permanent qualsevol dashboard ja sigui dels inicials o creats per ell.

- *Error prevention*: per evitar errades l'usuari haurà de confirmar aquesta tasca ja que no és reversible.

4 Disseny.

4.1 Arquitectura de la solució.

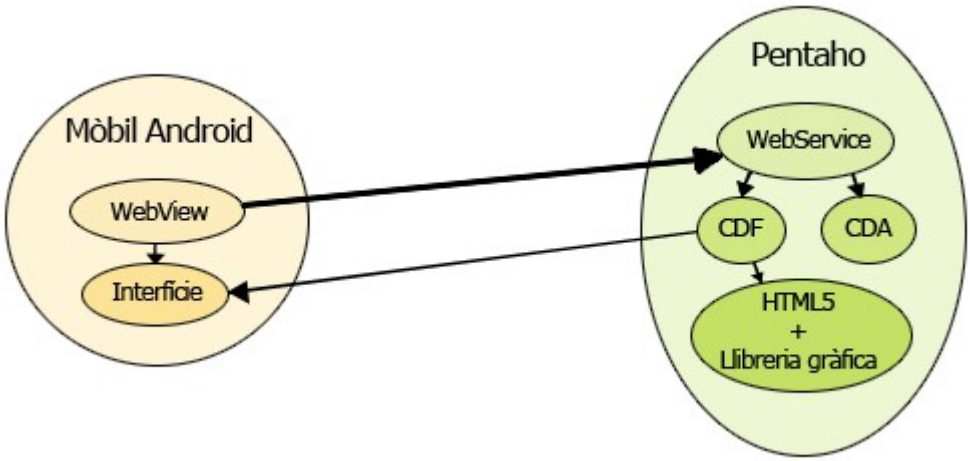


Figura 8. Recreació de la arquitectura de la solució.

Tal com s'apunta en el primer apartat de l'anàlisi del projecte, aquest es basa en separar el model i la vista, servidor Pentaho i l'aplicació android respectivament. Perquè aquest dos elements es comuniquin s'ha de crear una arquitectura entre ells per obtenir un de l'altre el que desitgen.

Per exemple, l'aplicació android necessitarà aconseguir totes les dades dels components i dashboards disponibles. en canvi, el servidor pentaho necessitarà la personalització del gràfic per poder generar-lo.

Pentaho no deixa de ser un servidor tomcat que rep peticions HTTP i els seus servlets reaccionen a aquestes peticions. Per tant, en un primer instant podríem dissenyar un servlet amb els paràmetres que desitgéssim i accedir a les funcionalitats que volguéssim de pentaho. Aquest solució bàsica té la problemàtica que no és extensible a qualsevol servidor pentaho. S'hauria crear el servlet en tots els servidors que es volgués integrar la solució, actualitzar-los, vigilar amb les versions per canvis de fitxers. Tot aquest afegit suposa un cost afegit que hauria d'assumir el client i un

cost temporal i de complexitat d'instal·lació de la solució que recauria sobre nosaltres. També sorgeix la problemàtica d'accés aquest servidors, la seguretat acreditativa que comporta una entitat financera faria molt feixuga la tasca de verificar tot el codi ja que estaríem modificant el seu servidor internament afegint novament un cost afegit temporal a la implementació de la solució.

Per intentar evitar aquestes problemàtiques i crear una solució a més alt nivell i que no depengui del servidor particular sinó de la aplicació en general, pentaho implementa de forma nativa un seguit de webservice on es poden executar certes tasques de forma preestablerta. Si s'enfoca en aquest nivell ens trobem que si canviem de plataforma BI a una altre, aquesta també es nodreix de webservice per resoldre el mateix tipus de problemàtiques fent que la implementació pugui diferir una mica però no pas el disseny.

4.1.1 Autentificació.

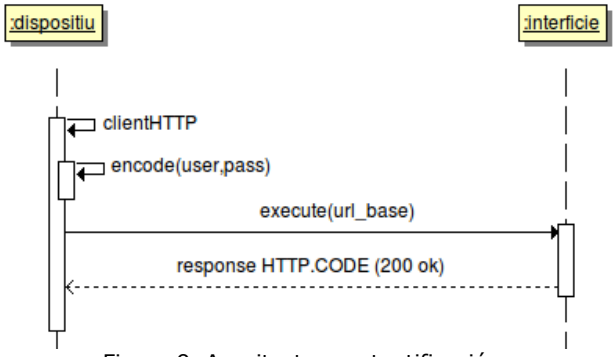


Figura 9. Arquitectura autentificació.

Per dur a terme la identificació, punt inicial de l'aplicació android per accedir a les dades, es realitzarà una petició bàsica HTTPOST al servidor pentaho amb l'usuari i contrasenya que l'usuari hagi introduït i s'esperarà la resposta adient per confirmar si existeix. Es pot comprovar que no és res més que el típic procediment que es realitza en qualsevol servidor.

4.1.2 Llistar elements d'un repositori.

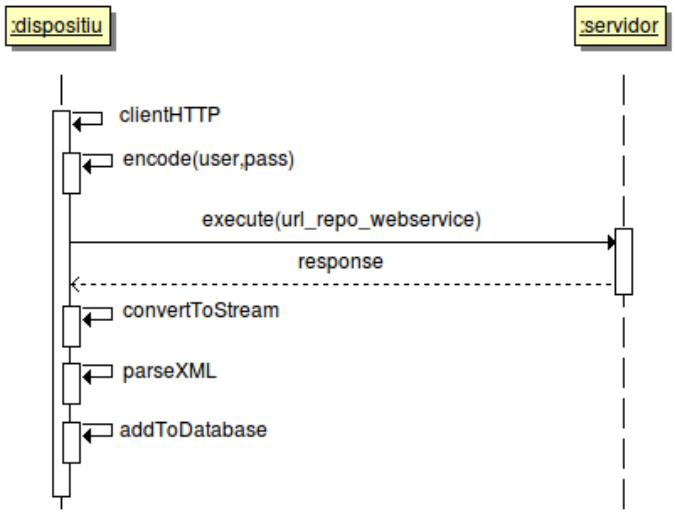


Figura 10. Arquitectura llistar elements repositori.

Pentaho dona accés a un webservice, que a través d'una petició HTTP parametritzada retorna un llistat de tot el contingut estructurat en un fitxer XML. A partir d'aquí l'aplicació android haurà de gestionar aquest fitxer i ensenyar-ho al usuari.

4.1.3 Visualitzar component CDF/CDA

Un component CDF(Community Dashboard Framework) o CDA(Component Data Acces) és un plugin de pentaho que permet realitzar accions estructurades. En el cas del primer, es el component bàsic en la creació de dashboards ja que es l'encarregat de generar el gràfic a partir d'unes dades en una plantilla de HTML i en el CDA és un component per accedir a les dades tenint la possibilitat d'utilitzar diferents protocols. En el següent apartat s'explica el funcionament dels dos components de forma més detallada.

On ens hem de centrar és en el fet que la finalitat d'un és l'obtenció de dades (CDA) i l'altre (CDF) de la possible obtenció de dades i la presentació del gràfic. S'utilitza el terme possible ja que els components CDF predefinits requereixen de la obtenció de dades dins del component, però si definim un component que no obtingui les dades sinó que li passem des de l'aplicació android podem alliberar l'analista BI de la tasca de generar la plantilla per mostrar la informació deixant a l'usuari aquesta

tasca com ell prefereixi. En el cas que es desitgi una plantilla exacte es podrà fer ús dels components CDF.

Per tant si volem visualitzar un component CDF o CDA l'arquitectura que estarà per sobre haurà de ser bàsicament la mateixa afegint la captura de dades amb el component CDA i la crida a un component CDF de suport sense accés a dades

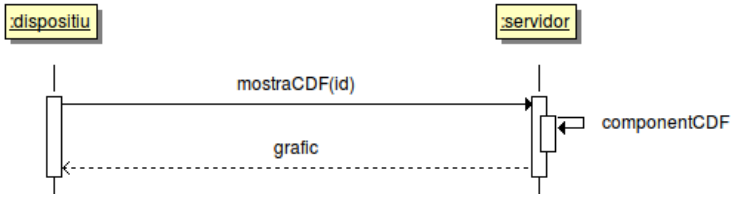


Figura 11. Arquitectura cdf normal.

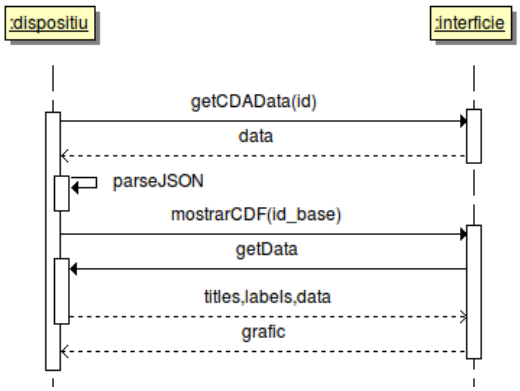


Figura 12. Arquitectura cdf amb dades cda.

4.1.4 Compartir dashboards.

Una de les problemàtiques més evidents és trobar una solució alhora de compartir els dashboards generats pels usuaris.

S'ha de tenir en compte que aquest projecte es basa en un dispositiu mòbil i un servidor BI intentant que aquest dos siguin fàcilment substituïbles.

La solució que s'ha arribat perquè els usuaris puguin compartir dashboards és utilitzar el correu electrònic, imprescindible en els dispositius mòbils actuals. Hem de tenir en compte que tots els fitxers adjunts d'un correu electrònic es poden descarregar al dispositiu mòbil i una aplicació pot accedir aquests sense problemes.

Llavors per compartir es generarà un fitxer utilitzant una estructura JSON i una extensió única i aquest s'adjuntarà per correu electrònic.

4.2 Pentaho.

Com s'ha esmentat en l'apartat anterior pentaho té un conjunt de tecnologies Open Source que es poden associar a la seva aplicació. Aquests plugins són els ja esmentats CDF (Community Dashboard Framework) i CDA (Community Data Acces). El primer és el plugin específic de pentaho per construir dashboards i el segon accessos a dades.

Community Dashboard Framework.

Els dashboards que es creen utilitzant CDF són pàgines web que utilitzen la tecnologia Ajax per combinar informes, gràfics, taules OLAP i mapes.

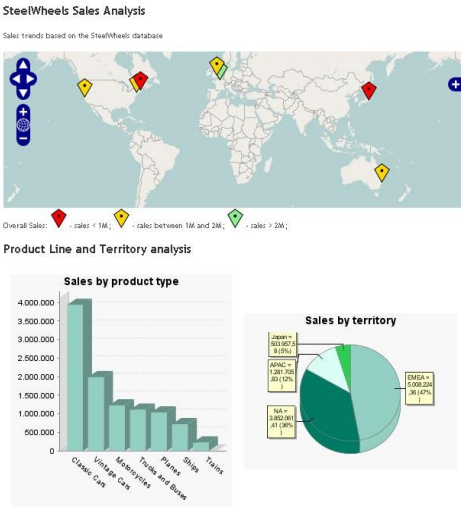


figura 13. Exemple de dashboard de cdf.

Com ja s'ha deixat entreveure Pentaho no està involucrada en el desenvolupament d'aquest projecte, sinó que inclou un plugin per utilitzar-ho i són els mateixos desenvolupador qui contribueixen a millorar el CDF.

Community Data Acces.

Les dades a les que s'accedeixen de pentaho es fa mitjançant urls, com s'ha explicat a l'apartat anterior. Es poden accedir a tots els formats d'emmagatzemament que disposa Pentaho (SQL, MDX, Metadata, kettle, etc). A l'usuari se li pot donar en

forma de JSON, XML, CVS, XLS o HTML bàsic com a l'aplicació web de pentaho.

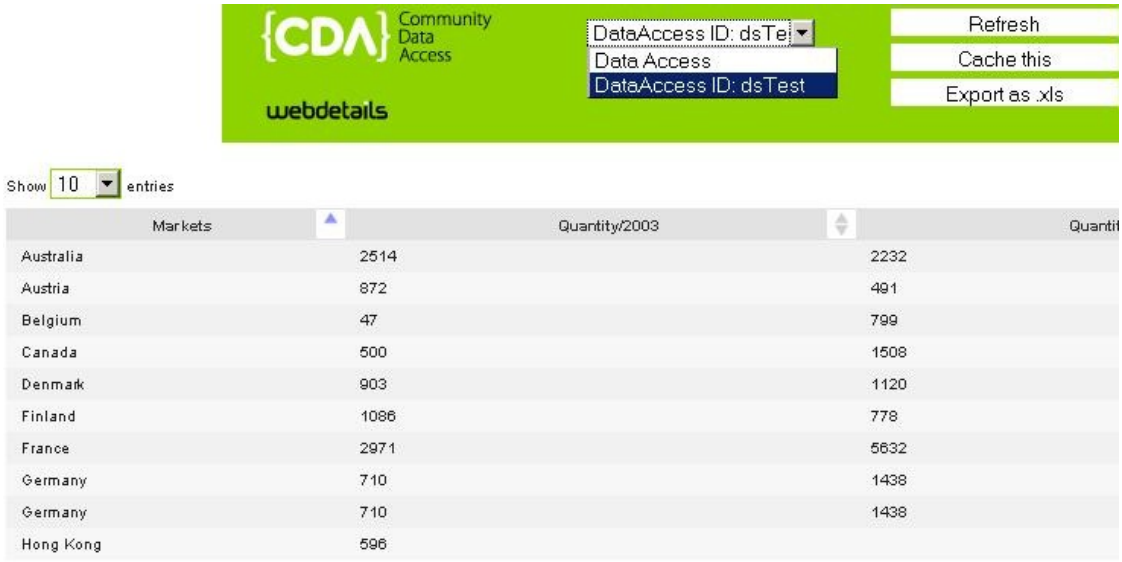


Figura 14. Component cda HTML

4.2.1 Arquitectura CDF.

Per executar un dashboard a pentaho es produeix un seguit d'accions creant una seqüència per executar-la en el servidor.

1. L'usuari utilitza un navegador web per obrir un dashboard. Això genera una petició HTTP que s'envia al servidor de pentaho.
2. El servidor reconeix una petició de Dashboard e intenta localitzar el fitxer .xcdf associat.
3. El fitxer .xcdf determina el template d'un dashboard. Aquest és un fitxer HTML parcial (no té cap cap clau de contingut reservada com html, head, body) que conté l'estructura posicional dels components i les instruccions javascript per generar aquest components.
4. El navegador rep una pàgina que conté el HTML parcial i s'inicialitza el dashboard i s'executen les instruccions javascript generant el contingut dels components.

5. Després de la inicialització es llança l'actualització dels components amb requests al servidor.
6. El servidor Pentaho rep les sol·licituds, que normalment corresponen a l'execució de seqüències d'accions (xactions).
7. El servidor executa la seqüència d'accions.
8. El contingut generat per la seqüència d'accions s'envia com a resultat per ser inclòs a la pàgina web.

Per tant, el resultat final que té l'usuari no és res més que un document HTML generat pel servidor de pentaho i el plugin CDF. Això vol dir es poden personalitzar alguna d'aquestes passes per obtenir els resultats que vulguem.

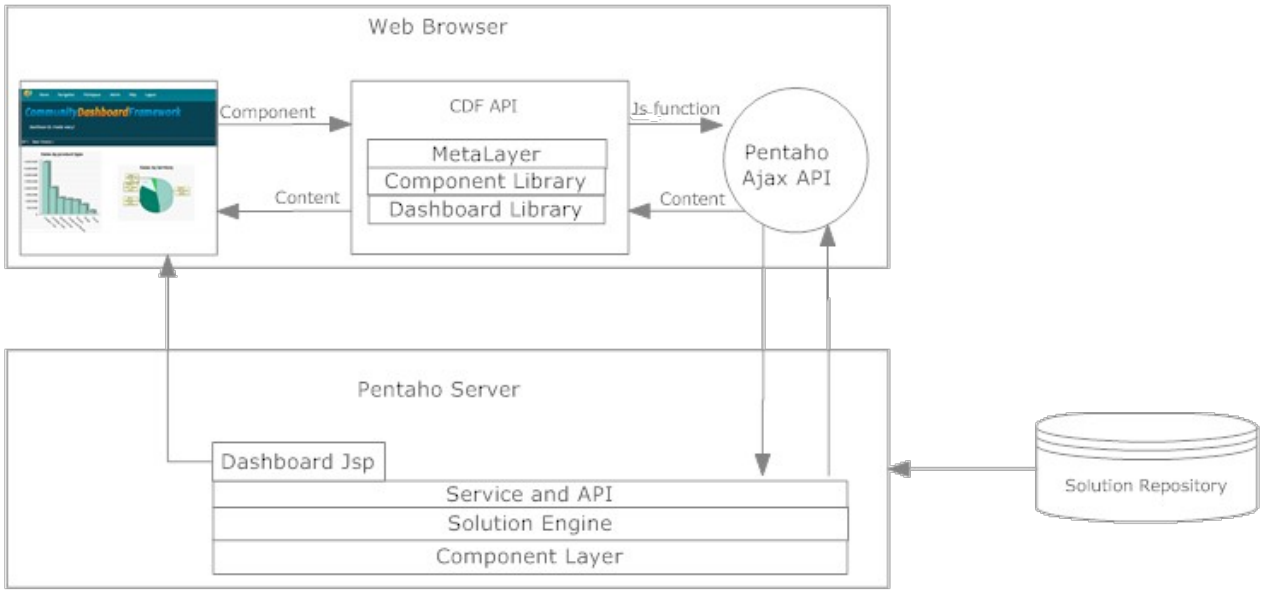


Figura 15. Arquitectura del plugin CDF.

4.2.2 Component CDF.

El plugin CDF es basa en la utilització de components ja definits (check, date imput, xaction, map,...) o en la creació de nous. Aquest components siguin existents o creats de zero tenen unes propietats bàsiques.

Propietat	*Descripció*	*Valor per defecte*	*Tipus*	*Requisit per a*
name	Nom del component		String	tots
type	Tipus del component		String	tots
solution	Solució de la xaction		String	
path	Cami a la xaction dins la solució		String	
action	Nom de la xaction		String	
listeners	Triggers del component	[]	Array of Strings	
valuesArray	Array de valors pel selector	[]	Array of Strings	
parameters	Parametres per executar per la xaction	[]	Array of String arrays	
parameter	La variable del dashboard que es modificada per aquesta		String	
htmlObject	La id del tag html (sol ser un div)		String	tots
expression	Funció per retornar un string parametritzat	function(){};		text
executeAtStart	S'ha d'executar el component a l'inici del dashboard	function(){};	boolean	tots
preExecution	Funció javascript a executar abans que s'executi el component.		function	
postExecution	Funcio java script a executar després de l'execucio		function	
preChange	Funcio javascript abans que s'executi un canvi		function	
postChange	Funcio javascript després que s'executi un canvi		function	

4.2.3 Xaction.

Les seqüències d'accions de pentaho o xaction s'estructuren de la següent forma.

1. **Entrades del procés (Process input).** Són els paràmetres que necessitarà la xaction del món exterior.
2. **Recursos (Resources).** Són els arxius necessaris per la xaction per realitzar un treball. Per exemple, si s'utilitza la llibreria jfreechart per mostrar els gràfics inclosos a la instal·lació bàsica de pentaho.

3. **Sortides del procés (Process output).** Aquestes sortides poden ser re-aprofitades per altres xactions.
4. **Accions de procés (Process actions).** Les accions a ser executades per la xaction.

L'ordre a la xaction és important ja que aquesta s'executa de dalt a baix. La construcció de les xactions es basa en arxius acabats en .xaction i aquest mantenen una estructura de "tags" compartida amb l'xml.

4.2.4 Component CDA.

Els components CDA es constitueixen amb dos elements.

Connexió. Definició de la base de dades o font de pentaho que s'utilitzarà.

Dades d'accés. Consulta que utilitzarà la connexió definida.

Tot això es constitueix en un fitxer XML. Poden haver-hi més d'una consulta, es poden parametritzar i afegir valors calculats.

4.2.5 Diagrames de seqüència.

4.2.5.1 Xaction.

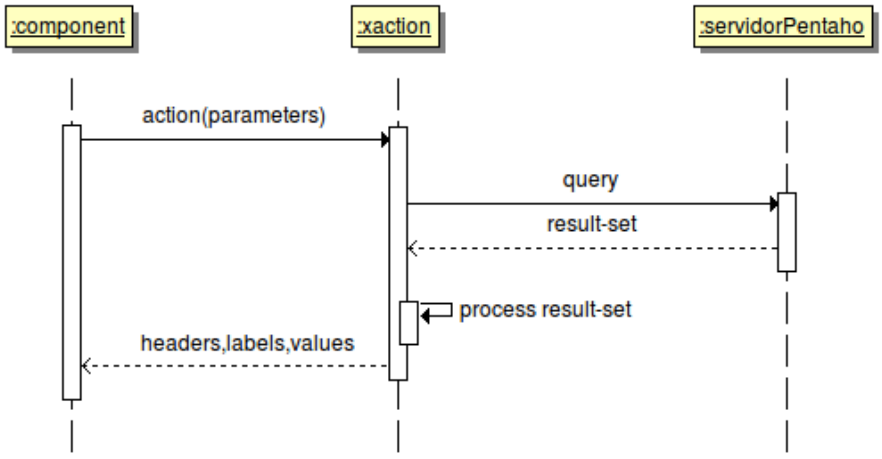


Figura 16. Diagrama de seqüència de la xaction.

Els paràmetres que es volen rebre s'hauran de definir, tant el tipus com en alguns casos el valor per defecte. La consulta al servidor pentaho s'ha de condicionar als paràmetres d'entrada, en especial atenció al tipus de query que es vol realitzar.

Un cop rebut el return en forma de result-set es processarà per generar la sortida de la xaction.

4.2.5.2 Component CDF.

L'estructura bàsica d'un component de CDF ja està definida en el mateix plugin amb una serie de crides pre-establertes. S'ha de tenir clar que un component es crea primer una instància i després s'omple amb la crida update().

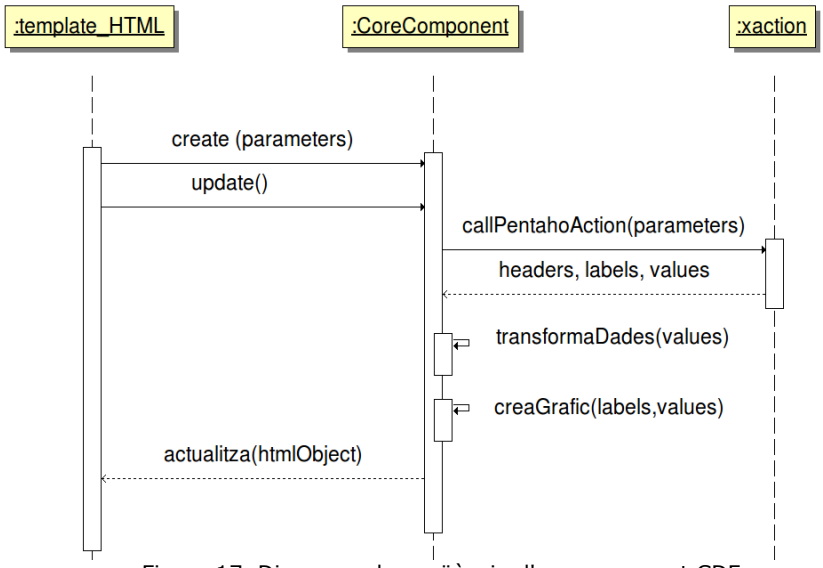


Figura 17. Diagrama de seqüència d'un component CDF

Un cop iniciada la cria d'actualització per "omplir" el component es procedirà a un seguit de crides seqüencials.

S'utilitzarà la xaction ja definida per aconseguir les dades necessàries per la representació gràfica.

Un cop rebudes es procedeix al pre-procés d'aquestes per poder ser utilitzades per la llibreria gràfica per construir el gràfic i a la parametrització d'aquest últim per visualitzar-lo correctament.

Alhora de mostrar-ho per pantalla es modifica l'objecte HTML definit al template.

4.2.5.3 Component CDF amb dades CDA.

En aquest cas i seguint la diferenciació que ja s'ha vist en l'apartat de l'arquitectura de la solució afegim l'obtenció de dades de forma externa, no com amb un component CDF normal.

Per obtenir aquestes dades utilitzarem una interfície per rebre-les des del dispositiu android.

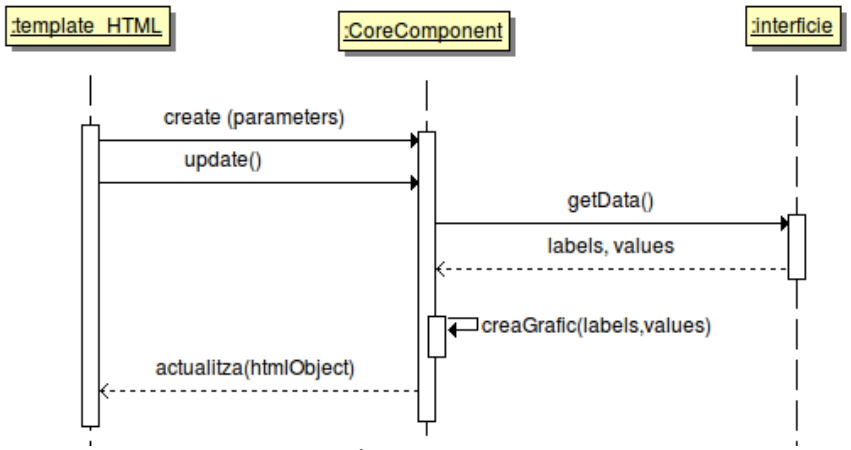


Figura 18. Diagrama de seqüència d'un component CDF amb dades CDA

4.3 Aplicació android

Aquest aplicació s'intenta seguir el patró de disseny User Center Design (UCD) ja que la idea bàsica d'aquesta aplicació es crear una nova interfície per dispositius mòbils per a servidors BI.

Utilitzant UCD es busca, tal com diu el seu nom, fer que l'usuari sigui el centre de tot. Per això aquest mètode és el millor alhora de dissenyar interfícies per l'usuari ja que aquest està present en totes les etapes del procés i facilita la possibilitat d'estudiar dissenys alternatius nodrint el resultat final.

Al inici faig esment de l'intent d'utilitzar l'UCD ja que alhora s'ha de coexistir

amb un disseny més estil en cascada com és l'apartat del servidor pentaho és fa difícil de compaginar-los.

A partir de l'apartat d'anàlisi es poden veure tres parts ben diferenciades dins l'aplicació android, l'autentificació, la gestió de components i dashboards, i la visualització i navegació entre ells.

Autentificació.

En aquest apartat de l'aplicació hem de dissenyar com configurar el servidor al que volem accedir, autentificar-nos al servidor pentaho i finalment obtenir tot el contingut del repositori al que es tingui accés segons el rol de l'usuari.

En la part de configuració només caldrà emmagatzemar les dades bàsiques d'una connexió HTTP, protocol,ip i port. Hem d'emmagatzemar-les perquè estiguin disponibles en tot moment un cop haguérem accedit als altres apartats de l'aplicació.

Tant a l'autentificació com a l'obtenció de tot el repositori s'ha de seguir l'arquitectura de la solució fent les peticions HTTP adients al webservices. En el cas de l'obtenció del repositori replicarem el contingut en una base de dades per tenir-la en tot moment disponible i així disminuir el numero de peticions al servidor.

Gestió.

En aquest apartat ens limitarem a gestionar els components i dashboards de l'aplicació. Aquesta gestió es bassa en la creació, edició, eliminació i organització dels dashboards. Com tenim tots els components replicats dins l'aplicació gestionarem la base de dades on guardarem la configuració de cada dashboard i els seus components. També s'afegeix una forma bàsica de compartir dashboard entre usuaris.

Navegació.

En aquest apartat de l'aplicació ens centrem en la visualització i navegació entre components i dashboards. Dins de la visualització dels components s'inclou la

personalització de cada component del dashboard, ja que la idea es que l'usuari pugui veure els canvis en temps real i no sigui una simple configuració en un formulari. Això es deu a un dels requeriments de l'aplicació a l'apartat d'anàlisi on es vol que tot sigui el més ràpid i fluid possible.

4.3.1 Diagrama general de l'aplicació.

Com ja s'ha explicat, l'aplicació queda dividida en tres subgrups autentificació, gestió i navegació. Tots tres subgrups accediran a la base de dades integrada dins l'aplicació i a la configuració d'usuari i servidor, i en el cas de l'autentificació i navegació accediran al webservice del servidor pentaho. Per tant, aquestes dos classes, base de dades i webservices, quedaran externes als subgrups fent que siguin comunes a l'aplicació.

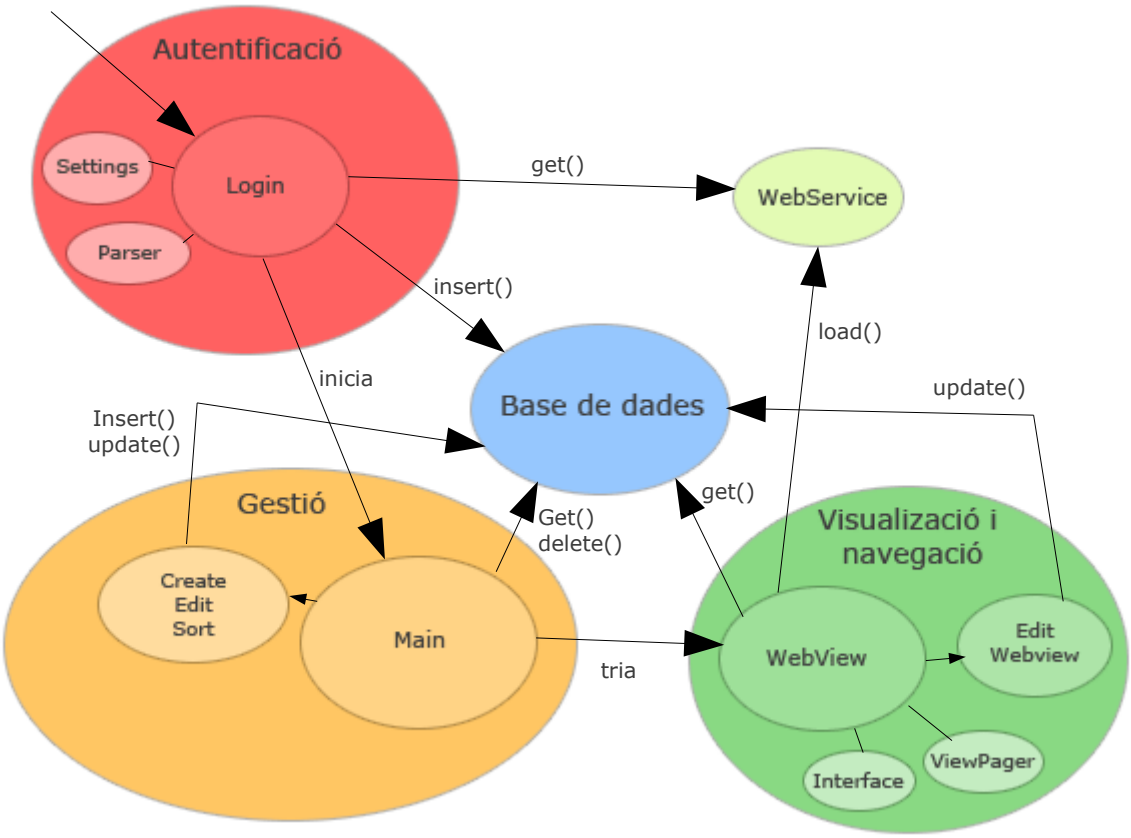


Figura 19. Diagrama general de l'aplicació.

L'aplicació s'iniciarà sempre per l'autentificació inclòs si l'usuari decideix que aquest es persistent. Un cop accedit i copiat el contingut del repositori s'accedeix a l'apartat de la gestió representada amb la classe main des d'on obtindran tots els components i dashboards de la base de dades. Des de la gestió es pot accedir a ordenar o crear/editar els dashboards. Tots dos tindran accions per actualitzar la base de dades i en el cas de la creació es podran introduir nous registres. Per accedir a la navegació sempre es farà des de la gestió d'on es rebran els paràmetres per saber quin element s'ha de visualitzar. La representació bàsica de com es visualitzaran els components és amb una classe webview base des d'on s'accedirà al webservice. Aquesta classe serà la encarregada de crear la interfície pel servidor pentaho i la navegació. La única forma d'editar els components dels dashboards serà iniciar aquest procés des del webview base. Des de la classe on s'editarà el component no deixarà de ser d'una extensió de la webview base, reduïda a un sol component. Aquesta nova classe serà la única dins la navegació que podrà modificar la base de dades.

4.3.2 Base de dades

La idea de base de dades que ha d'utilitzar l'aplicació és d'intentar replicar la resposta XML que rebem quan es s'enumeren els elements d'un directori utilitzant el webservice de pentaho. Aquesta resposta XML, que no deixa de ser un llistat d'entrades, està formada per un identificador, un nom, una descripció i una url. Ja tenim una estructura inicial pels components on afegim un atribut d'identificador d'usuari per gestionar a quins components es tenen accés.

Un altre camí que faria que el nombre d'entrades es reduís en el cas que molts usuaris diferents utilitzessin el mateix dispositiu seria un identificador de rol. El problema sorgeix que cada servidor pentaho pot tenir els seus propis rols definits i hauríem de codificar l'aplicació per cada un d'aquests servidor ja que l'accés al webservice es fa per usuari i no per rol.

Els dashboards que creem vindran definits pel nom i l'usuari que el crea. Per evitar que l'usuari confongui dashboards i per fer un replica metafòrica d'una carpeta

de fitxers, on aquests no es poden repetir l'identificador, s'inclou que el nom sigui únic en cada registre. Un parell d'atributs que s'afegeixen als bàsic són la estructuració utilitzada i l'ordre del dashboard dins del llistat d'aquests.

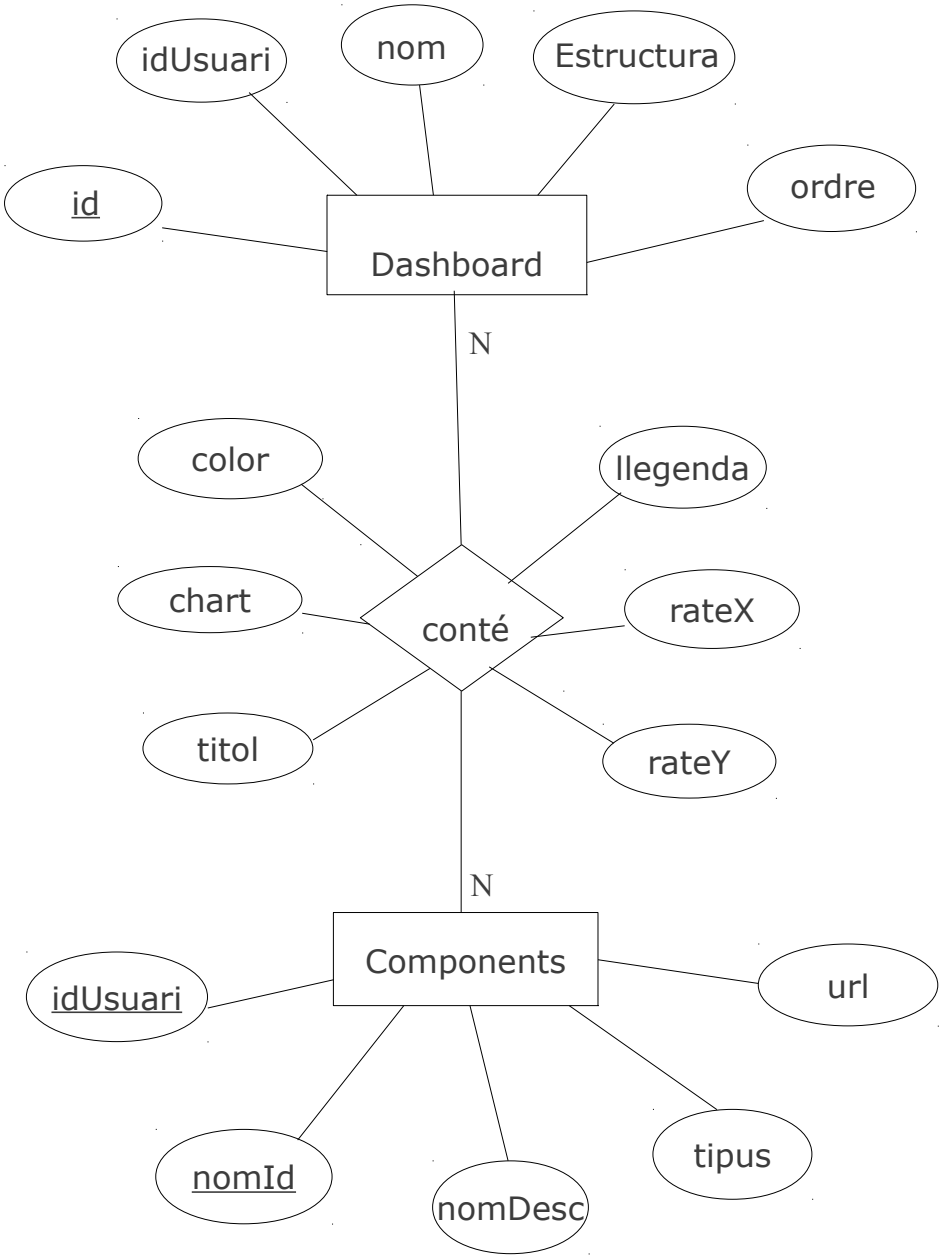


Figura 20. Diagrama Entitat-Relació de la base de dades

Finalment com tenim que un dashboard no deixa de ser un conjunt de components tenim la relació que es produeix entre ells. Aquesta relació té atributs diferencials en cada registre i que defineixen la representació gràfica que es mostrarà

a l'usuari. Aquest atributs són el color, si existeix la presencia de títol i llegenda, la posició relativa d'aquesta dins la pantalla i el tipus de gràfic.

Taula Dashboard	
Id **PK**	int
nom	String
idUsuari	String
ordre	int
estructura	int

Taula Component	
idUsuari **PK**	String
nomId **PK**	String
nomDesc	String
tipus	int
url	String

Taula Relació	
idUsuari *FK* **PK**	String
nomId *FK* **PK**	String
idDashboard *FK* **PK**	int
color	int
chart	int
titol	boolean
llegenda	boolean
rateX	int
rateY	int

Aquestes són les taules generades a partir del diagrama Entitat-Relació anterior amb tots els tipus associats a cada atribut. Es poden observar les claus foranies generades de la relació N a N.

4.3.3 Diagrames de classes.

4.3.3.1 Gestió components i dashboards.

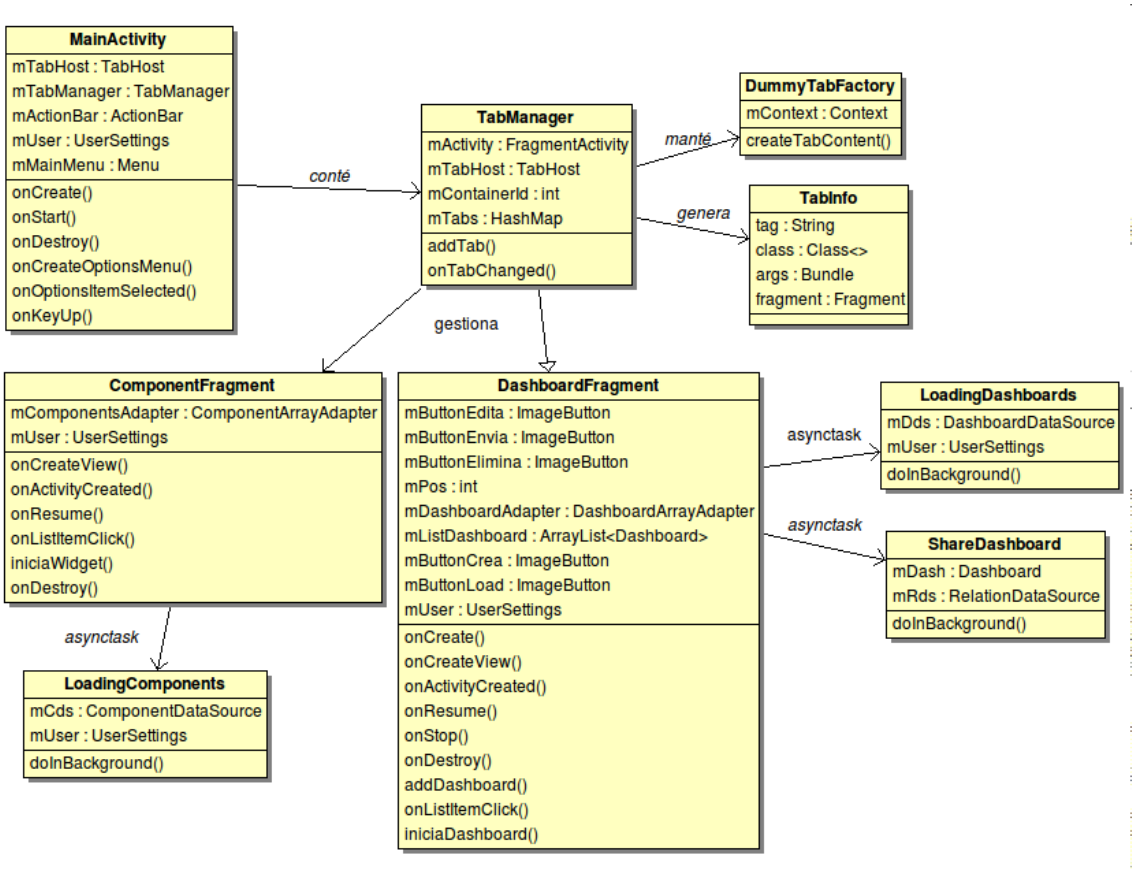


Figura 21. Diagrama de classes de la gestió.

L'apartat de gestió es realitza a l'activity principal. Aquesta estarà formada per dos fragments, un pels components i l'altre pels dashboards.

A l'activity principal, apart dels mètodes habituals de creació, resum, stop o destrucció, els mètodes principals són per la gestió de l'ActionBar. Per gestionar els fragments que formaran l'activity en forma de pestanyes s'utilitza un mànager de pestanya o TabManager que s'encarregarà de la gestió de canvi de fragment.

El comportament per part dels dos fragments es molt similar. Dos llistats amb el control sobre l'acció "click" de cada element per iniciar-lo. En el cas del fragment dels dashboards, s'afegirà el control sobre els botons de creació, càrrega i ordenació. A més, també es gestionarà l'acció de click de llarga durada per donar accés a la edició, enviament i eliminació del dashboard en qüestió.

El fragment de components utilitzarà una asynctask per carregar les dades de components existents a la base de dades mentre que el fragment de dashboard farà el mateix amb un de dashboards.

El procés de compartir dashboards amb altres persones mitjançant el correu electrònic s'utilitzarà una altre asynctask per alliberar el thread UI.

4.3.3.2 Visualització d'un component o dashboard.

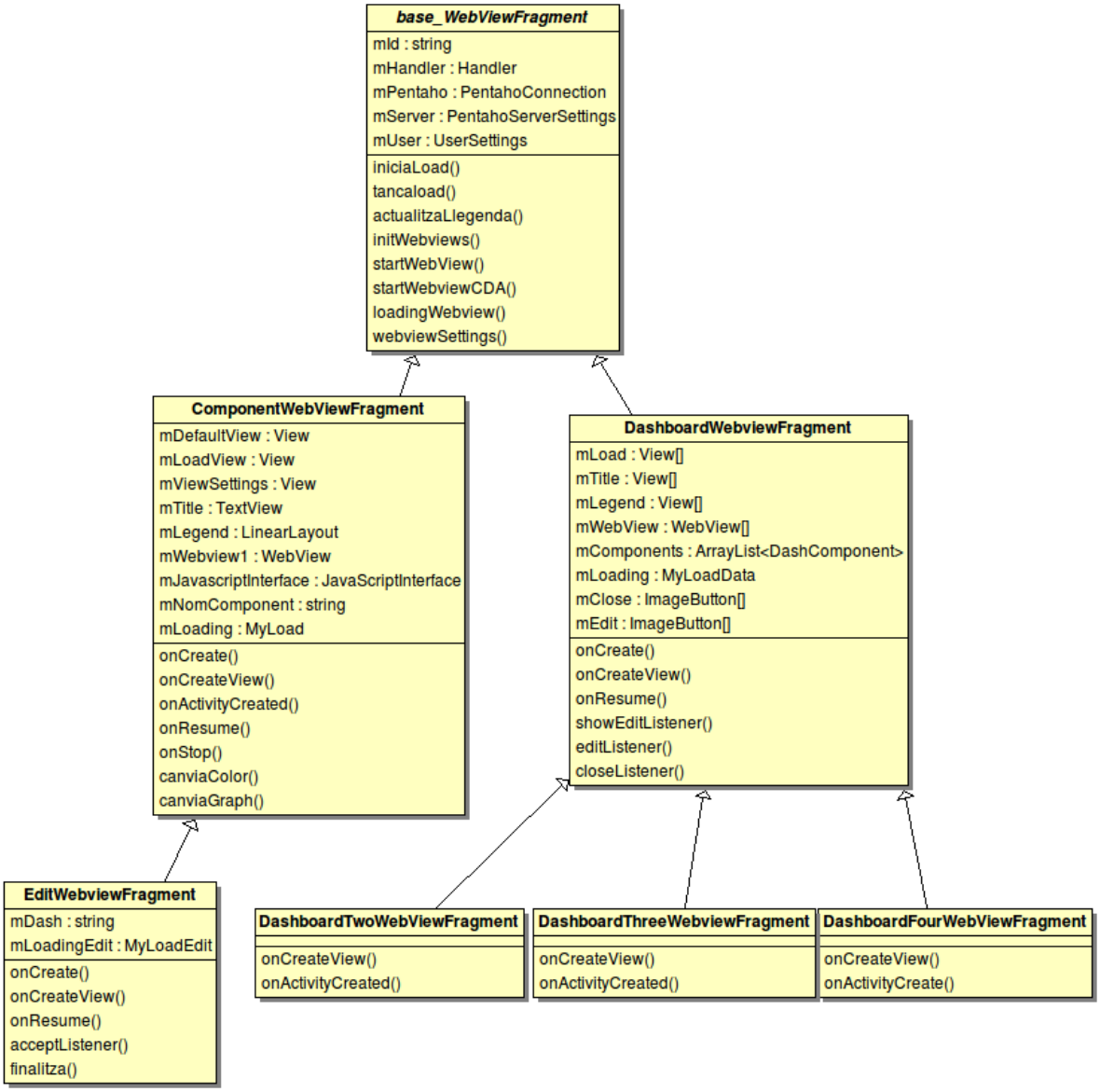


Figura 22. Diagrama de classes de visualització.

Alhora de dissenyar la visualització de dades, s'ha optat per crear una classe base abstracta on es produeix l'herència.

S'ha de tenir en compte que totes les visualitzacions ja sigui, component, dashboard o edició d'un component d'un dashboard. Tots parteixen que han de tenir com a mínim un webview on visualitzar la crida al servidor BI.

A la classe abstracta es defineixen atributs bàsics com la id, el handler que permetrà modificar coses dins el thread UI, la connexió a pentaho, la configuració de

pentaho i l'usuari actiu. Els mètodes descrits seran els que les classes que heretin hauran d'implementar.

La diferenciació principal entre components i dashboard és que el primer només tindrà un component, com el nom indica, i el podrà modificar com vulgui. En el cas del segon podrà tenir un o més elements segons la estructura escollida i veurà restringida la possibilitat d'editar-lo havent d'accedir a l'edició de forma particular en cada component del dashboard que s'estigui mostrant.

L'edició de components d'un dashboard hereta de la classe ComponentFragment perquè l'única cosa que s'ha d'afegir és la modificació de la base de dades per fer permanents aquests canvis.

L'herència expressada en el cas de dos, tres o quatre components és per la modificació del layout que carregaran en cada cas, sent diferent en cada cas i els components que es permetrà l'accés.

4.3.3.3 Creació dashboard.

En aquest cas, només tenim una única activity que ho controla tot ajudant-se de fragments per les diverses accions que haurà de realitzar l'usuari. Per controlar les tres fases de la creació, escollir nombre de components (NumComponentsFragment), afegir els components (Edit_base_Fragment) i escollir el component (PickComponentFragment), l'Activity implementarà una interfície de cada un dels fragments perquè es pugui comunicar amb ella amb el resultat de l'acció. Per això, tots tres fragments tenen un atribut *mCallback* on es guardarà la crida a la classe pare.

Escollir nombre de components només implicarà, comunicar a l'activity la tria escollida. En aquest fragment és on es controlarà si és una tauleta o un mòbil per restringir el nombre d'elements. En el cas que sigui de dos a quatre components, hi haurà classes que estendran de la base modificant únicament el layout i els listeners dels botons.

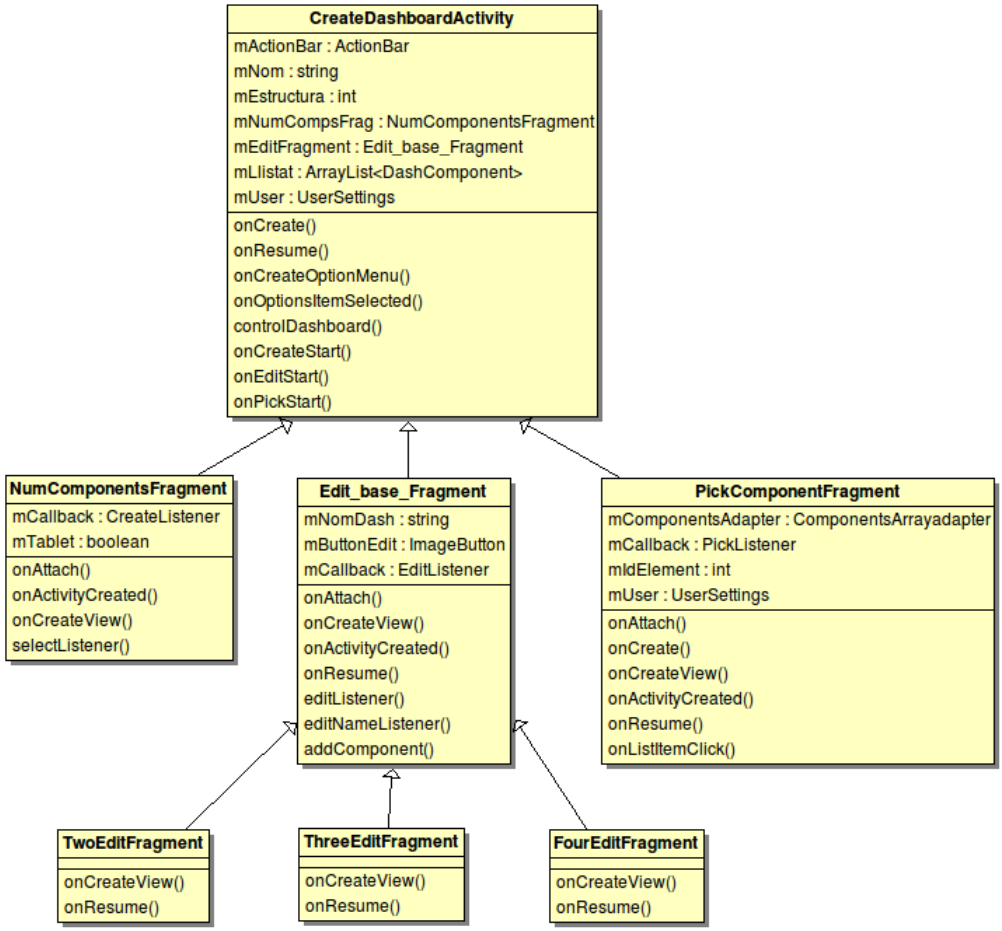


Figura 23. Diagrama de classes de la tasca de crear un dashboard

En el cas d'afegir els components, es controlarà la modificació del nom del dashboard i l'inici de la tasca d'escollir un component en concret.

En el cas d'escollir un component concret, es presentarà una llista idèntica a la de gestió de components perquè l'usuari pitgi un. Un cop efectuat s'avisarà a l'activity pare mitjançant el listener implementat en aquesta.

4.3.3.4 Càrrega dashboard extern.

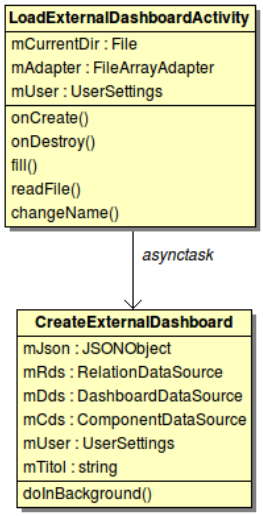


Figura 24. Diagrama de classe de càrrega de dashboards externs.

Alhora de carregar un dashboard extern tenim una activity que s'encarrega de gestionar un llistat amb tots els fitxers disponibles i una asynctask que s'encarregarà del procés de creació a la base de dades.

L'activity tindra accés al directori actual, al adaptador del llistat i l'usuari actual. Els mètodes es divideixen en procés d'omplir el llistat, lectura del fitxer seleccionat i en cas de necessitat de canvi la presentació del diàleg de solicitut.

L'asynctask contindrà l'objecte JSON i els connectors a la base de dades. L'únic mètode és el procés que realitzarà de creació o no del dashboard sinó existeix cap problema.

4.3.4 Diagrames de seqüència

4.3.4.1 Petició a pentaho.

Pentaho ofereix un webservices molt bàsic per accedir des de terceres aplicacions al contingut del servidor. Una és l'autentificació Basic utilitzant un client HTTP i l'altre es la modificació del servlet per separar l'entrada amb navegador web i webservice. Aquest segon mètode el que realment faria seria millorar el filtratge en l'accés per codi. Com aquest projecte es centra en la interfície amb l'usuari, deixem de banda aquesta configuració ja que realment, des del punt de vista de qui desenvolupa

l'aplicació del dispositiu mòbil, es realitza el mateix procediment d'autentificació.

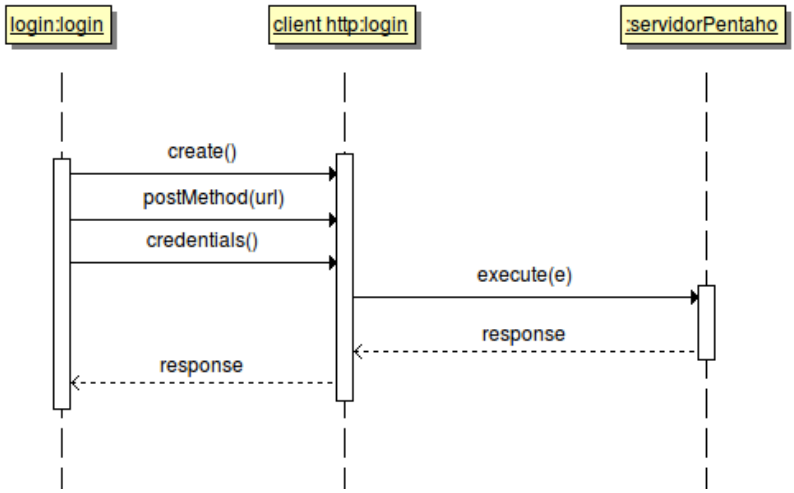


Figura 25. Diagrama de seqüència d'una connexió al webservice de pentaho

Un cop creat el client HTTP, s'utilitzarà el mètode post per incloure les credencials dins la petició. La resposta variarà segons quin webservice s'utilitzi.

4.3.4.2 Creació i/o edició d'un dashboard.

El procés d'edició i creació de dashboards és molt similar l'únic que variarà serà l'inici i final però el cos del procediment és el mateix, per això s'ha ajuntat en un sol diagrama.

A l'inici, en el cas de la creació rebem només el nom que hagi introduït l'usuari i escollirà l'estructura del dashboard. Aquesta estructura també conté associada el nombre de components que tindrà el dashboard. En el cas de la edició, el nom i l'estructura ja venen definits al igual que els components que estan inclosos en aquesta última. A partir d'aquí es podran dur a terme les mateixes accions en tots dos casos.

Al final, la diferenciació ve donada pel fet que en la creació s'haurà de realitzar un insert a la base de dades i en el cas de la edició un simple update. Alhora de fer aquest update s'ha de tenir en compte la configuració actual del component guardant-la en cas de no modificar el component. En el cas de la creació la configuració del

component serà la bàsica deixant que l'usuari faci la modificació en la visualització del dashboard.

El cos de la seqüència és una repetició d'afegir els elements necessaris o modificar-los utilitzant la llista amb tots els components disponibles per l'usuari. En tots dos casos la modificació del nom del dashboard es realitzarà de la mateixa forma tenint en compte el disseny de la base de dades que especifica que el nom ha de ser únic.

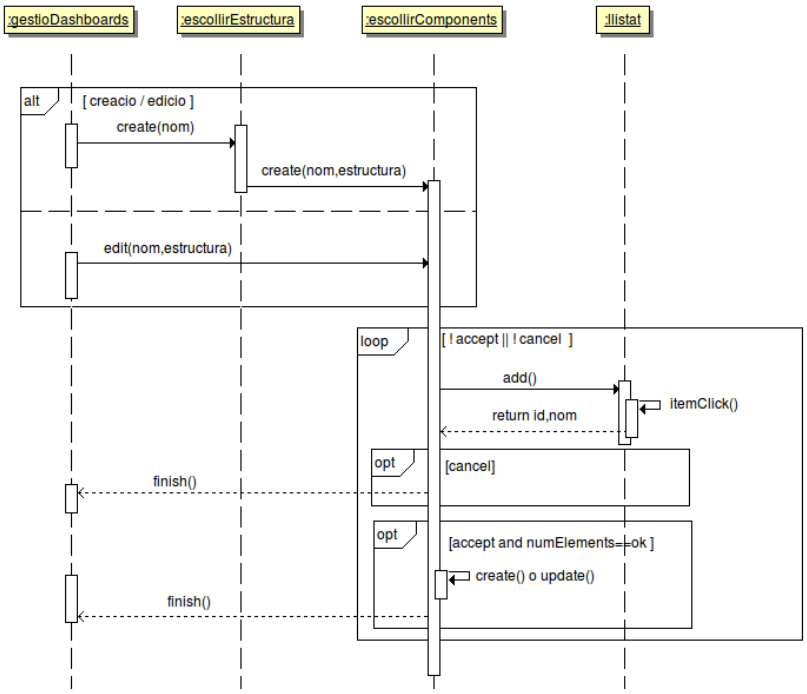


Figura 26. Diagrama de seqüència de la creació/edició d'un dashboard

4.3.4.3 Visualització d'un component o dashboard.

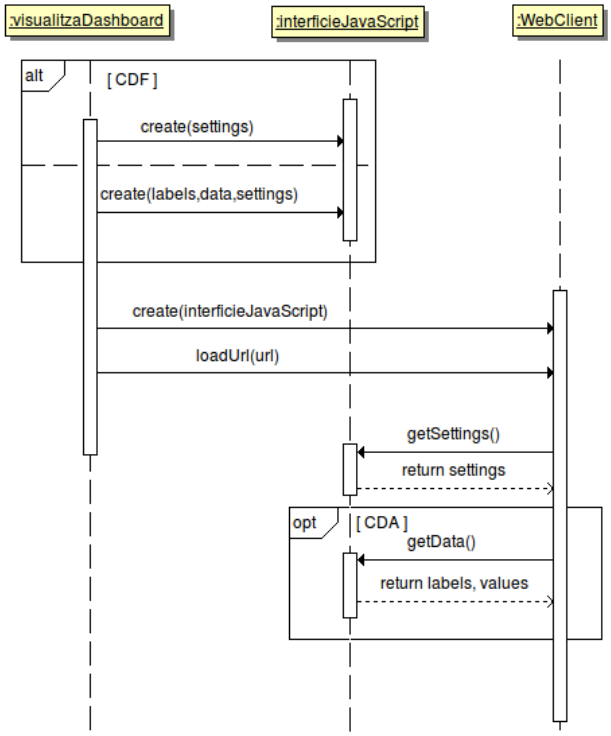


Figura 27. Diagrama de seqüència de la visualització d'un component o dashboard

A l'hora de visualitzar un component des de l'aplicació es segueix l'arquitectura de la solució que s'ha dissenyat amb anterioritat. Abans de realitzar cap petició al webservice declarem una interfície. Aquesta és amb la que es comunicarà el webservice quan hagi de demanar les dades de configuració del gràfic i les dades en si en cas que sigui un component CDA. Remarcar que aquesta interfície no pot estar associada directament al thread principal de l'aplicació perquè sinó deixaríem a l'usuari a l'espera de la resposta quan aquesta, pot variar molt depenent del processat necessari per generar les dades.

Un cop establerta aquesta interfície, l'aplicació es limitarà a fer una petició al client Web i rebrà una web base, tal com s'explica a l'arquitectura CDF. Llavors el Client Web, mitjançant la interfície, obtindrà totes les dades necessàries per generar el gràfic i l'inclourà a la web base.

4.3.4.4 Edició component dashboard.

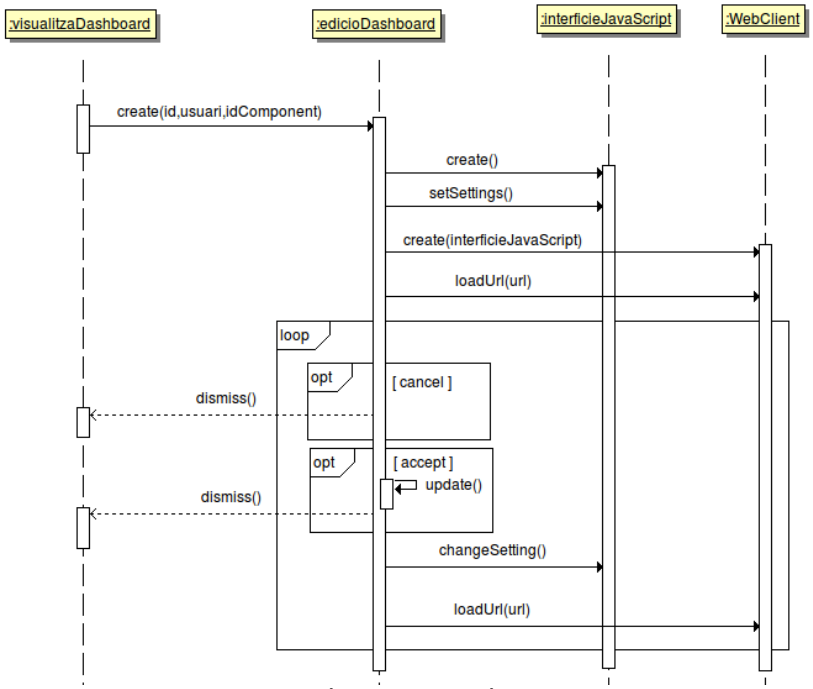


Figura 28. Diagrama de seqüència de l'edició d'un component dashboard.

Partim des del punt que estem visualitzant un dashboard i l'usuari decideix editar un component que forma aquest. Llavors com a entrada rebem l'identificador tant del component com del dashboard tal com defineix la clau primària de la relació definida en el disseny de la base de dades.

Com es vol fer que l'usuari interactuï amb el gràfic i no sigui un simple formulari on no es veuen els canvis, es fa una primera petició al webservice per obtenir l'estat actual del component. A partir d'aquí es produeix un bucle, sempre i quan l'usuari no es faci enrere o accepti els canvis, d'escollir un dels canvis a realitzar, ja sigui títol, llegenda, color o gràfic i es torna a carregar el gràfic.

Quan l'usuari accepti els canvis es realitza un update a la base de dades. Si descartes els canvis simplement tancaríem el procés ja que tots els canvis que produeix l'usuari es guardaran en una estructura temporal.

4.3.4.5 Carregar dashboard extern.

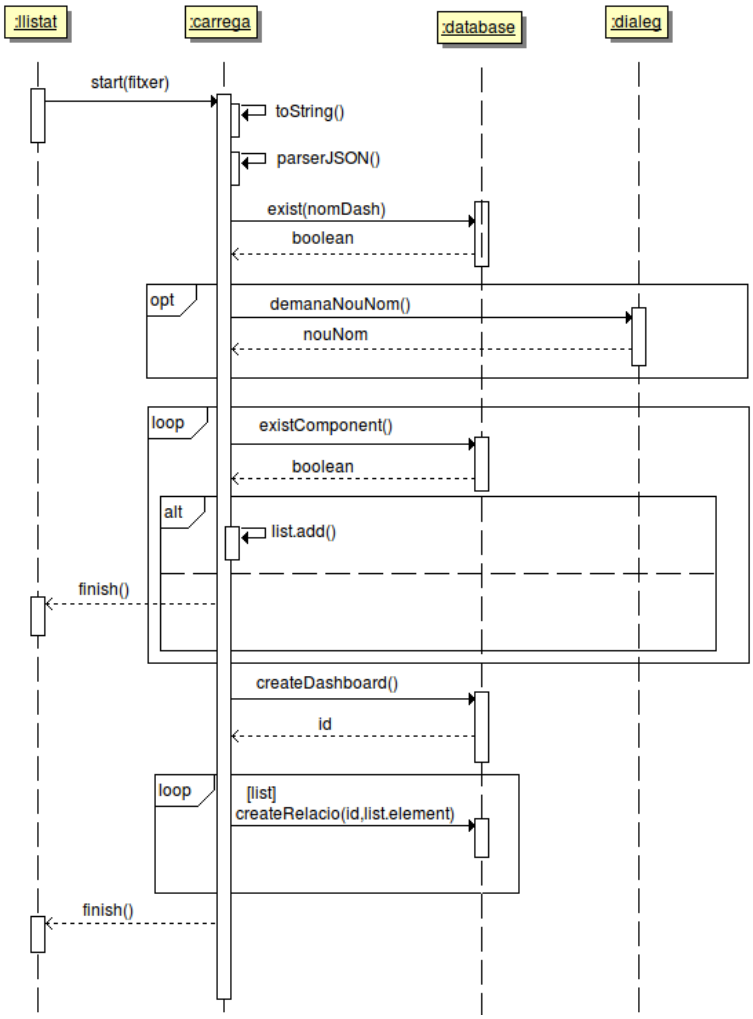


Figura 29. Diagrama de seqüència de la carrega d'un dashboard extern

El primer que s'haurà de fer amb el fitxer que conté el dashboard extern serà transformar-lo sencer a string i transformar-lo a un objecte JSON. El motiu es que tots els dashboards compartits es transformen a un objecte JSON per emmagatzemar totes les característiques.

Després es comprova l'existència d'un dashboard amb el mateix nom, si fos el cas es demanarà a l'usuari amb un diàleg que introdueixi un nom nou. El control que el nom nou sigui únic es produeix dins del diàleg.

Un cop tenim el nom del dashboard a crear, comprovem els elements que el formen. Només que hi hagi un que l'usuari no hi tingui accés, la creació del dashboard

s'anul·larà mostrant a l'usuari un missatge d'avís. Tots els elements s'introdueixen en un llistat.

Per finalitzar es produeixen els inserts a la base de dades. Primer el dashboard i després els components que el formen per garantir les claus foranies necessàries.

4.3.5 Prototips en “paper”

Alhora de realitzar els prototips i els diferents dissenys que s'han pensat per dur a terme les tasques que ha d'implementar l'aplicació android, s'ha tingut molt en compte les heurístiques presents en el mètode d'inspecció de l'usabilitat de Jakob Nielsen.

Aquest mètode es basa en analitzar la interfície segons la conformitat amb uns principis reconeguts d'usabilitat que són les heurístiques.

Heurístiques de Nielsen.

- H1: Visibility of system status.
- H2: Match between system and the real world.
- H3: User control and freedom.
- H4: Consistency and standarts.
- H5: Error prevention.
- H6: Recognition rather than recall.
- H7: Flexibility and efficiency of use.
- H8: Aesthetic and minimalist design.
- H9: Help users recognize, diagnose and recovers from errors
- H10: Help and documentation.

Intentar seguir aquestes heurístiques no garanteix aconseguir una bona interfície per l'usuari, però si assolir conductes que en un test d'usuari no es veurien com errades.

Actionbar.

A partir de la versió 11 d'android entra en funcionament un widget, que potser replicat en versions anteriors, que consisteix en una barra superior on introduir certa informació present i modificable en tota l'aplicació, l'ActionBar.

Aquest widget facilita complir amb la primera de les heurístiques perquè podem informar en tot moment d'on estem i quines accions podem realitzar de forma genèrica.

A més a més, facilita la consistència i els estàndarts ja que és una de les principals bones pràctiques recomanades per Google en la seva guia de disseny d'aplicacions.

Pantalla inicial o login.

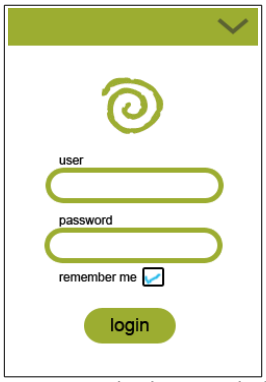


Figura 30. Proposta de disseny de la pantalla inicial.

En la pantalla inicial, no s'han generat dissenys alternatius amb els que l'usuari o un expert pogués valorar en temes d'usabilitat. Pràcticament es una estructura estandard en totes les UI.

Es solen afegir elements modificadors com el "check" per memoritzar la contrasenya o el botó de registre per poder-hi accedir. El botó de registre no s'ha inclòs ja que l'usuari ja hauria de tenir accés a pentaho o l'administrador li hauria de facilitar un. Així s'intenta evitar un problema de seguretat en la pèrdua del dispositiu.

En canvi, si que s'ha inclòs l'accés permanent ja que en cas de pèrdua només caldria canviar la contrasenya en el servidor per bloquejar la entrada automàtica de l'usuari.

Per seguir amb els patrons de disseny que recomana google, totes les accions de configuració del servidor queden dins de l'accionbar.

Figura 31. Propostar de disseny de la pantalla de inicial.
Reconeixement, diagnosi i recuperació d'un error.

Intentant seguir les heurístiques de la recuperació dels errors i la de consistència i estandarts. S'ha afegit una modificació en la variable del color per evidenciar l'error al introduir una paraula clau errònia. També s'utilitzarà la comunicació bàsica de missatges d'android per informar a l'usuari de l'existència de l'error i quina ha estat la causa. S'afegeix aquesta notificació per evitar problemes amb els usuaris amb problemes visuals com el daltonisme.

Gestió de components i dashboards.

En el cas de la gestió de components s'utilitzarà el widget de “tabs” o pestanyes per diferenciar els components dels dashboards i facilitar el canvi d'uns als altres el mes ràpid possible.

Pantalla gestió components.

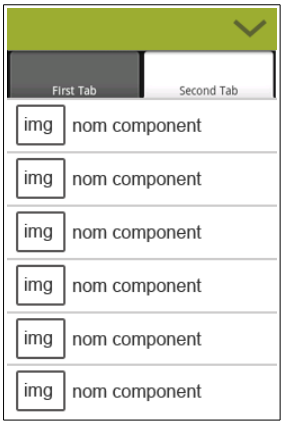


Figura 32. Disseny A d'un llistat de widgets

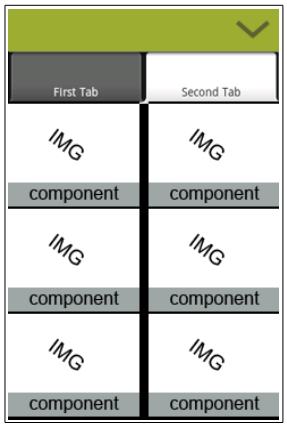


Figura 33. Disseny B d'un llistat de widgets

Les dos alternatives de disseny representen les dos tendències alhora de la representació d'un llistat d'informació. Interfícies d'usuari on són habituals són els reproductors de música on sovint es troben les dos vessant implementades.

El disseny A és més senzill, cada línia correspon a un sol element. És un estil més tradicional però fortament estructurat. La imatge present pot donar informació addicional però no es la primera font d'informació.

El disseny B és una aproximació a les icones, present a la UI central dels dispositius i a la UI central dels sistemes operatius dels ordinadors. S'intenta donar més familiaritat a l'usuari. Els ítems són més grans i la font d'informació principal passar a ser la imatge en comptes del text com en el disseny A.

Analitzant la informació que es rep del Model es descarta el disseny B. Pot ser és més elaborat i l'usuari es podria trobar més familiaritzat però com les imatges que es mostraran no són discriminatòries sinó que són estandaritzades segons el tipus de gràfic, l'usuari li costaria més navegar entre els components i recuperar-ne un d'específic.

Un altre motiu per la tria del disseny A és la heurística de la consistència i els estandarts. Si es vol que tota l'aplicació presenti el mateix aspecte per facilitar l'aprenentatge a l'usuari, la problemàtica amb el disseny B torna apareix a la tria de

components per confeccionar dashboards personalitzats.

Els esdeveniments que es registren en cada un dels ítems són la pulsació curta o click per accedir a visualitzar el component.

Pantalla gestió dashboards.

Figura 34. Disseny A gestió de dashboards

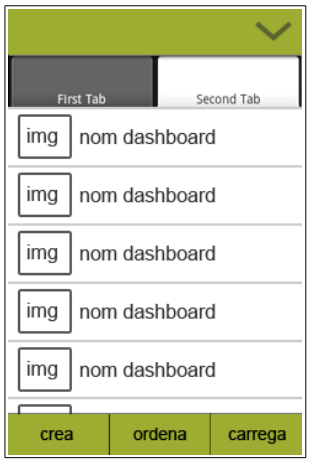


Figura 35. Disseny B gestió de dashboards

Un cop dissenyat la gestió de components, el factor de representació d'un llistat d'informació queda exclòs en aquest disseny pel mateix motiu que amb els components, la imatge no dona suficient informació diferenciadora com agafar aquell patró per diferenciar entre dashboards.

El factor determinant en les alternatives de disseny és la gestió dels dashboards, centrant-ho en la selecció de tasques incloses en aquesta. Crear, ordenar i carregar són les tres principals tasques de la gestió general de dashboards i són úniques ja que els components no les tenen. Les dues solucions que s'han trobat es afegir-les al desplegable de l'actionbar com en el cas A o fent una extensió de l'actionbar a la part inferior com en el cas B.

En aquest cas la tria del disseny s'ha decantat pel cas B. El motiu ha estat seguir amb l'heurística de la consistència i els estandarts. Els patrons de disseny de google recomana que les accions especifiques que no càpiguen dins la actionbar sigui exposats a la part inferior de la pantalla per facilitar l'accés a l'usuari. En el disseny de

la edició d'un component del dashboard també es seguirà aquesta recomanació fent que sigui extensa a tota l'aplicació i que l'usuari interpreti que totes les accions especifiques del que esta veient estan sempre situades al mateix lloc.

Els esdeveniments que es registraran en aquest llistat seran la pulsació curta i la llarga. En el cas de la primera s'accedeix a la visualització del dashboard seleccionat, en el cas de la segona s'accedeix a les tasques específiques del dashboard seleccionat.

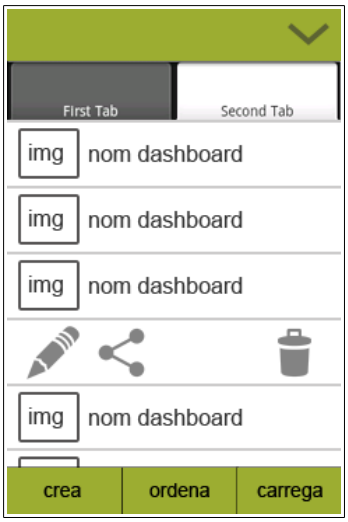


Figura 36. Detall pulsació llarga llistat dashboard.

En el cas de les tasques específiques s'ha intentat seguir metàfores visuals per realitzar aquestes tasques. La separació intencionada entre eliminar i les altres dos tasques es per evitar errades en la selecció de la tasca.

Pantalla diàleg eliminació dashboard.

Sumat a la separació de la tasca d'eliminar un dashboard s'afegeix un diàleg a l'usuari per avisar-lo que aquesta tasca té efectes irreversibles. Amb aquest diàleg s'intenta enfortir l'heurística de prevenció d'errors per part de l'usuari.

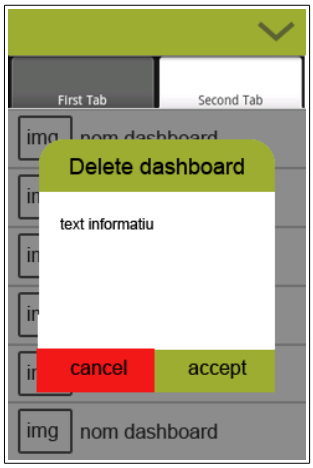


Figura 37. Diàleg d'avís al eliminar.

La utilització del vermell en el botó de cancel·lar és per cridar l'atenció de l'usuari a l'acció no destructiva. També utilitzar el vermell com metàfora de parar aquesta tasca. El text informatiu serveix per reforçar les conseqüències de dur a terme aquesta tasca per part de l'usuari.

Pantalla diàleg creació dashboard.

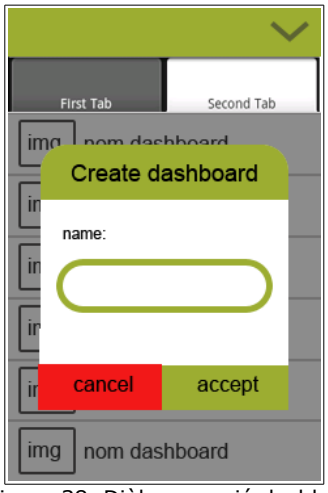


Figura 38. Diàleg creació dashboard.

Per continuar amb una consistència de l'aplicació, en la sol·licitud del nom del dashboard a crear es repeteix la mateixa estructura que amb l'eliminació d'un dashboard específic. Per reforçar l'heurística de prevenció d'errors i repetint el mateix procediment que amb el login, en cas que el nom de dashboard ja existeixi l'input es posarà vermell i apareixerà un missatge d'avís amb l'error.

Pantalla de visualització d'un component.

Figura 39. Disseny A de la visualització.

Figura 40. Disseny B de la visualització.

Les alternatives que es presenten en aquest disseny es centren en la disposició dels elements a la actionbar principal i la secundaria. En el disseny A, es situa els desplegables de color i gràfica a l'actionbar ja que són accions més desenvolupades que requereixen tornar a carregar el gràfic, en canvi, la visualització o no de la llegenda i el títol es tracten de simples botons alternatius. El disseny B es basa en crear una forta consistència de l'aplicació com ja s'ha utilitzat en el cas de la gestió de dashboards fent que l'usuari estigui habituat aquesta disposició. A més a més, dona la oportunitat de facilitar l'ajuda a l'usuari si no entén alguna cosa posant clarament una icona d'ajuda a la seva disposició.

S'ha escollit el disseny B perquè té consistència, respecta els estandarts, és minimalista i te una ajuda o documentació disponible.

Pantalla edició component Dashboard.

Figura 41. Dissent edició component dashboard.

Aquest disseny no té alternatives ja que sorgeix de la visualització dels components. L'únic canvi que es produeix és el canvi a l'actionbar on només apareix un botó d'acceptar. Tots els canvis que es produeixen no són emmagatzemats fins que s'accepten pitjant aquest botó, fent que l'usuari els pugui descartar sense cap problema.

Pantalla de visualització d'un dashboard.

Alhora de visualitzar un dashboard s'haurà de tenir en compte que no és el mateix un mòbil que una tauleta. La diferencia recau en les dimensions de cada pantalla, més que la resolució les polsades, ja que no és el mateix set o deu que quatre.

Els dashboard generats a pentaho per visualitzar a l'aplicació web solen tenir entre tres i quatre components. Aquests dashboards es generen pensant que els veuràs en una pantalla d'ordinador de com a mínim tretze polsades fins a vint-i-quatre o més. El problema sorgeix quan els vols veure en un mòbil de quatre polsades, ja no només per les gràfiques sinó per les dades, llegenda i títols. A l'usuari se li pot fer impossible saber que està veient. Per tant s'haurà de reduir el nombre de components per dashboard a dos. En canvi, a les tauletes es podran seguir veient estructures de tres o quatre components sense problemes.

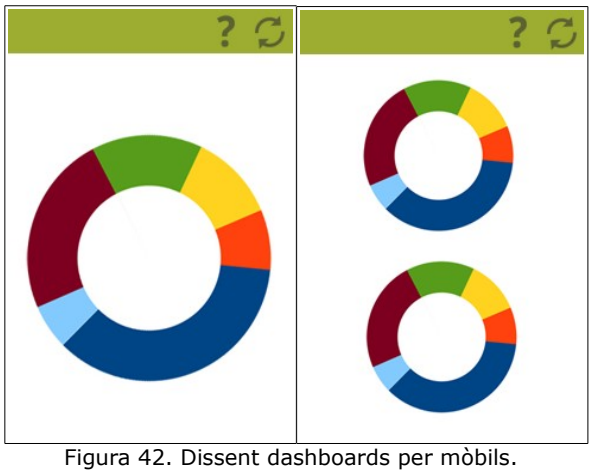


Figura 42. Dissent dashboards per mòbils.

En aquest disseny és molt important la senzillesa. Si donem un cop d'ull a la

visualització d'un component veurem que només s'ha eliminat la actionbar inferior i així donar-li encara més consistència al disseny.

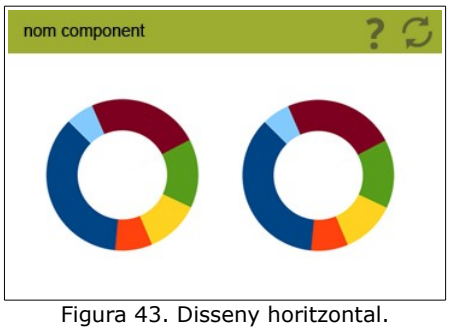


Figura 43. Disseny horitzontal.

També hem de tenir en compte que l'usuari pot voler veure els dashboard en horitzontal. En el cas que només tingués un element no seria cap problema però si en tenim dos s'ha de variar el repartiment d'aquest dins de la pantalla del dispositiu.

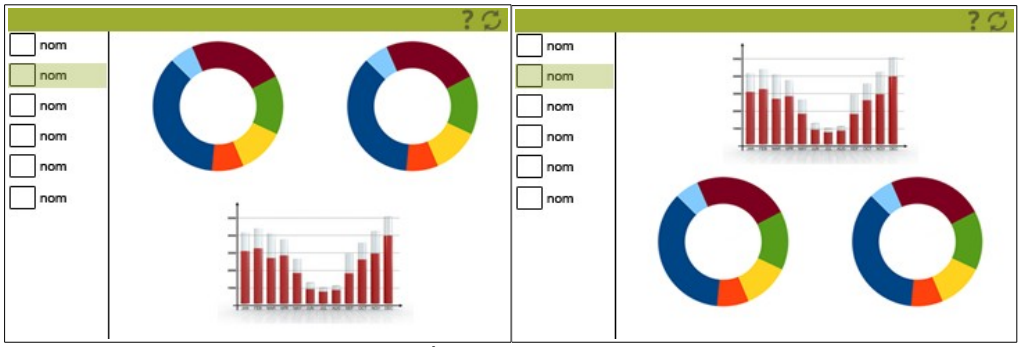


Figura 44. Disseny visualització dashbord en tauleta amb tres components.

Un canvi important que s'inclou amb les tauletes és una llista amb tots els dashboards disponibles fent encara més fàcil a l'usuari navegar entre ells i saber on està del llistat.

Figura 45. Disseny dashboard amb quatre compoents.

Ja sigui un mòbil o una tauleta, tots dos es comportaran de la mateixa forma a l'hora d'editar els components del dashboard. Fent ús de prémer llarg, l'usuari podrà accedir a la modificació del component.

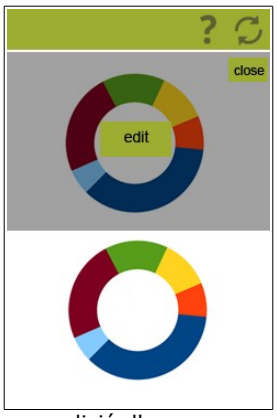


Figura 46. Disseny edició d'un component de dashboard.

A l'usuari se li donarà accés a editar o cancel·lar la edició deixant clar quin és l'element a modificar.

Disseny creació dashboard.

La creació d'un nou dashboard s'ha dividit en tres passes en el següent ordre, escollir estructura, visualització estructura i elecció. L'usuari sempre podrà desfer el camí fins descartar la creació del dashboard.

L'apartat d'elecció es parteix duplicant el disseny de la gestió de components així l'usuari ja està familiaritzat i només ha de pitjar sobre el que vol, en canvi les altres dos passes del procés si que requereixen d'un detall més precís.

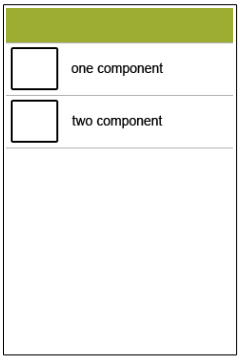


Figura 47. Disseny A escollir estructura dashboard.

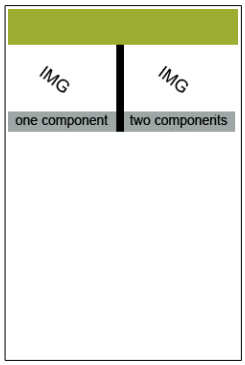


Figura48. Disseny B escollir estructura dashboard.

Si fem la mateixa reflexió que hem fet amb el llistat de components la conclusió en aquest cas varia completament ja que hem d'escollir el Disseny B ja que la imatge

si que és molt descriptiva i el text és un suport a l'accessibilitat.

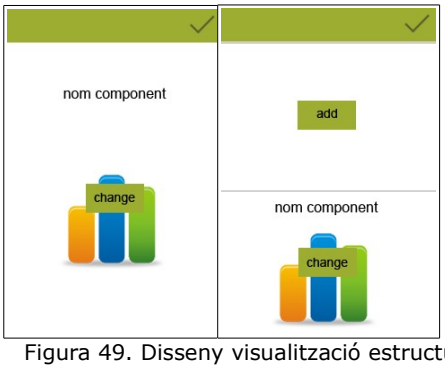
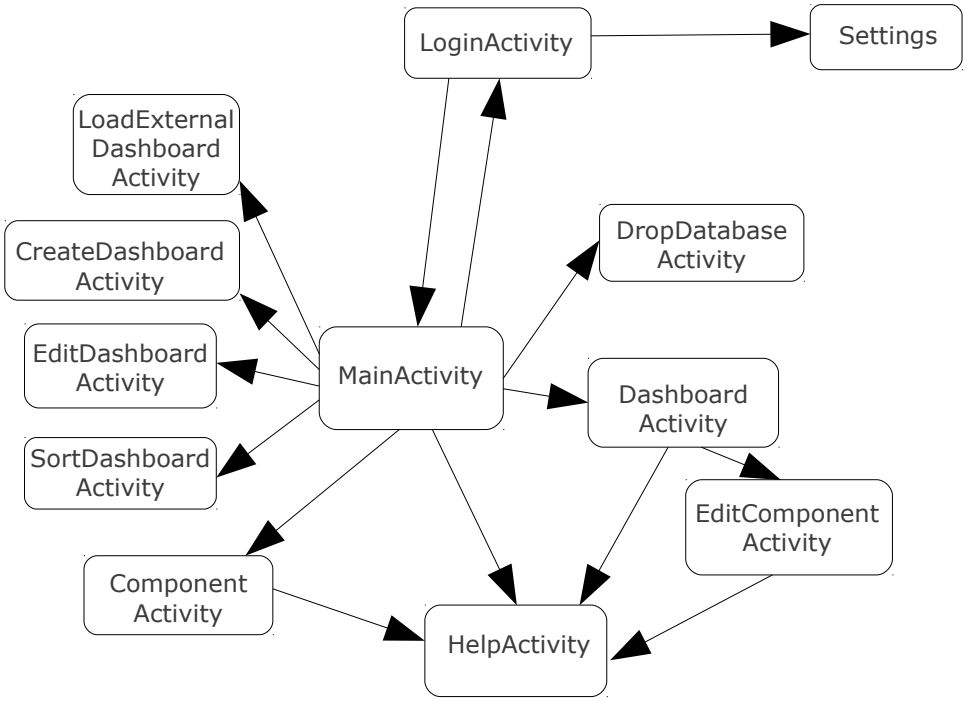


Figura 49. Disseny visualització estructura.

Per tal que l'usuari vegi com quedarien repartits els components del seu dashboard s'ha creat una replica d'aquest de forma informativa. La imatge no deixarà de ser predeterminada i el nom si que correspondrà amb el del component afegit, així l'usuari es podrà fer una idea de com quedarà el seu nou dashboard.

En el cas de la edició d'un dashboard, la part d'escollir estructura quedara eliminada totalment ja que es una definició bàsica del dashboard però a partir d'aquí compartirà els mateixos dissenys.

4.3.6 Navegabilitat per l'aplicació.



L'aplicació s'iniciarà amb el login, tal com s'ha comentat al diagrama general de l'aplicació. Un cop efectuat, es passa a l'activity principal. Des d'aquesta es podrà accedir a la gran majoria a excepció de la edició de components d'un dashboard.

S'ha de tenir en compte que un cop efectuat el login, només es podrà accedir a ell realitzant un “logoff” a l'activity principal o, en cas de no haver marcat el fet de recordar l'usuari i contrasenya, tornant a iniciar l'aplicació. Això es deu a que el login no serà emmagatzemat a la pila d'activitys.

La resta d'activitys no tenen un retorn exprés a l'activitat principal perquè aquestes finalitzaran l'acció que estan duen a terme, ja sigui pitjar un boto de confirmació o tornant enrere amb el botó inclòs en el dispositiu.

5 Implementació.

5.1 Pentaho.

5.1.1 Estructura repositori.

Per tal que tots els components CDF o CDA que es creïn utilitzin la mateixa configuració interna s'ha creat una estructura de fitxer on emmagatzemar-los. Aquesta estructura conté:

- **Colors.** Fitxer javascript que conté totes les possibilitats de colors que poden adquirir els gràfics generats.
- **Dades.** Fitxer javascript que conté les possibles configuracions de dades pels components CDF.
- **Gràfics.** Fitxer javascript que conté les configuracions dels gràfics. Tipus, delimitacions, efectes, etc.
- **Suport CDA.** Fitxer XCDF i fitxer HTML amb l'estructura del component que rebrà les dades del component CDA.
- **Llibreries externes.** Conjunt de fitxers amb totes les llibreries externes que s'utilitzaran.

5.1.2 Nou component CDF per visualitzar gràfics HTML5.

Els components CDF estan emmagatzemats en el fitxer de javascript del CDF "corecomponents.js". Per poder visualitzar els gràfics utilitzant una llibreria en HTML5 s'ha de crear un nou component. Aquest nova classe javascript s'extens de BaseComponent, tal com fa evidencia el nom, el component base de CDF.

El mètode principal serà **update** on es farà la crida de recuperar les dades del servidor pentaho mitjançant una xaction especifica.

Per fer que aquest component sigui també executable a l'aplicació base de pentaho es fa ús de "try/except" de javascript per intentar la captura de configuració de la interfície de l'aplicació mòbil. En cas que no sigui possible es fan servir uns paràmetres establerts en el fitxer HTML del component específic que s'està creant.

Per fer que el gràfic sigui adaptable a la pantalla del dispositiu o del navegador que el crida s'utilitza una crida javascript per obtenir les dimensions d'aquesta en píxels, així es poden definir les dimensions exactes del gràfic.

5.1.3 Nova xaction per obtenir dades.

Per poder retornar les dades de la consulta en un format òptim per poder gestionar-les, s'ha creat una xaction propia. Aquesta retorna les capçaleres, les etiquetes i els valors per separat segons si és un dataset de categoria o temporal.

Com l'objecte que es retorna és SEAP, es processa el resultat per obtenir les dades en brut. Els separadors de valors són especificats per evitar possibles errades amb els caràcters punt i coma.

5.1.4 Nou Component CDF per visualitzar gràfics HTML5 provinents de CDA.

En el cas que la font de dades sigui un component CDA, el component CDF creat no utilitzarà una xaction per obtenir-les, sinó que utilitzarà la interfície de l'aplicació mòbil per obtenir les dades. A partir d'aquí el procés és molt similar a l'anterior component CDF.

5.2 llibreria gràfica rgraph

S'ha triat aquesta llibreria perquè està implementada en HTML5 i JavaScript, i utilitza el tag canvas d'HTML5 en comptes de centrar-ho tot en representacions SVG que dificulten la càrrega en navegadors poc potents com els dels dispositius mòbils.

A més, aquesta llibreria te llicència Creative Commons amb el codi obert i lliure, fent possible optimitzacions particulars si es desitgés.

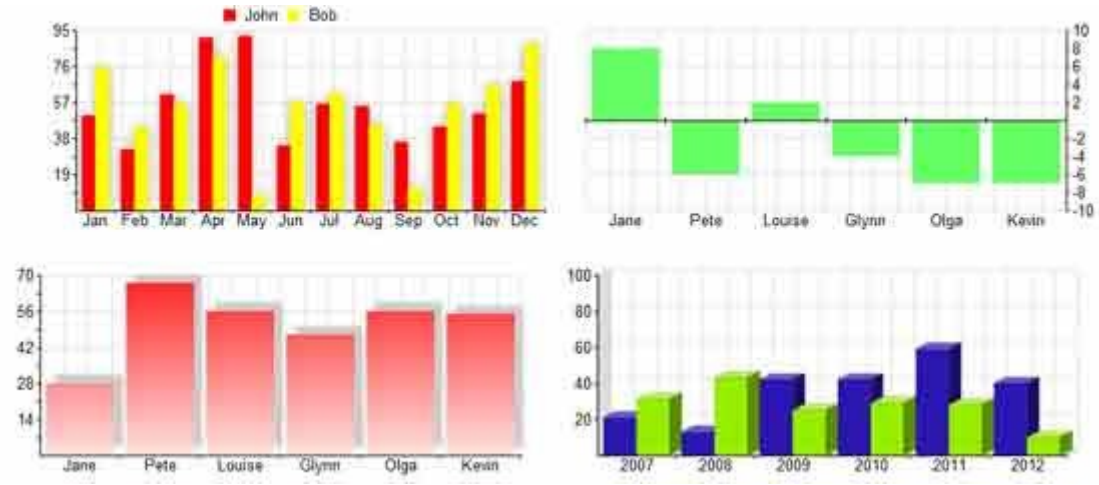


Figura 50. Gràfics utilitzant la llibreria Rgraph.

Un dels altres motius alhora d'escollir aquesta llibreria és la introducció de dades ja que es produeix de forma separada tal i com es reben tant de la xaction o dels components CDA.

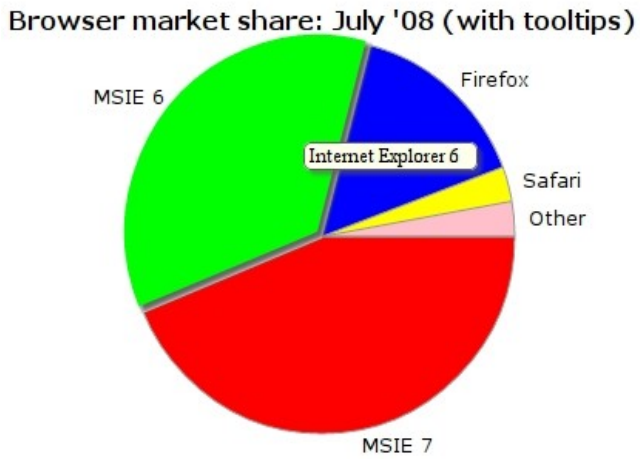


Figura 51. Gràfic utilitzant la llibreria Rgraph.

5.3 Android.

S'ha intentat seguir tots els patrons d'implementació que recomana google des del seu web de desenvolupament d'android. Aquestes recomanacions van des de la gestió de les Activity, els llistats, l'accés a dades, etc. La motivació al intentar ser el més estricte possible en aquesta idea és aconseguir que l'aplicació necessiti mínims canvis quan s'introdueixin noves versions del SO en el futur.

L'aplicació ha estat desenvolupada per la versió 4.0 del sistema operatiu o Ice Cream Sandwich, però és operativa en versions anteriors i molt esteses com Gingerbread.

La implementació s'ha dividit en dos apartats clarament diferenciats, la interfície de l'usuari i el model dins l'aplicació.

5.3.1 Interficie Usuari.

5.3.1.1 Responsive design.

Una de les principals problemàtiques que es presenten al implementar una aplicació d'aquestes característiques és que tot el seu disseny gràfic sigui totalment adaptable. Com exemple tenim les dimensions de les pantalles dels principals dispositius mòbils.

- **Nexus S** 4 polsades.
- **HTC One S** 4.3 polsades.
- **Sony Xperia S** 4.3 polsades.
- **Gañaxy S3** 4.7 polsades
- **Galaxy S2** 4.3 polsades.
- **Galaxy S** 4 polsades.
- **Iphone 4S** 3.5 polsades.
- **Blackberry Torch** 3.2 polsades.

I no només les mides si afegíssim la resolució de cada una d'aquestes veuríem que tampoc hi ha una estandarització.

Alhora d'implementar les pantalles segons els disseny, s'ha de jugar amb si les agrupacions d'elements són lineals com un llistat o relatius, ja que aquest últims permeten ocupar la pantalla percentual-ment, afavorint l'adaptabilitat de la implementació. Les dimensions han d'utilitzar unitats relatives com density-independent points (dp) o scale-independent ponts (sp) per les dimensions de les fonts.

La problemàtica apareix en el cas de mostrar la llegenda i el títol d'un gràfic a l'usuari. Aquests dos elements són creats a nivell de programació java i no amb estructures XML. la definició de la mida de la font ha de ser totalment relativa ja que no només depèn del dispositiu, sinó que també de la quantitat de components que tindrà un dashboard ja que no es tindrà el mateix espai disponible un cas que l'altre. Per tant, s'ha de definir una constant dimensional que comparada amb les dimensions del component que es mostra en el webview es pugui escalar a la dimensió adequada.

5.3.1.2 Fragments.

Una altre de les bones pràctiques promocionades per google és la utilització dels Fragments dins l'aplicació ja que permeten gestionar més fàcilment un disseny adaptable a dispositius mòbils i tauletes digitals.

No significa que totes les activitys hagin de passar a ser administradores de fragments però encoratgen la utilització perquè facilita enormement nodrir amb noves funcionalitats a les versions per tauletes que per qüestions de mida no poden ser aplicades als mòbils.

Un dels inconvenients alhora d'utilitzar fragments és la inicialització de valors en cas que el fragment sigui l'inicial i rebi valors importants en la seva creació. La bona pràctica que recomanen des de Mountain View és la creació d'un Fragment en blanc i gestionar la creació i els arguments del fragment amb el mànager de fragments que incorpora les llibreries de suport.

5.3.1.3 Drag and drop.

Hi ha dos tasques on l'usuari veurà millorada la seva llibertat i eficiència alhora de realitzar-les si s'implementa l'habilitat drag and drop. Aquestes dos tasques són la d'organitzar els dashboard en el llistat inicial i la situació de la llegenda dins el component.

Alhora d'implementar un drag and drop s'ha de tenir en compte que a partir de la versió del SO android Honeycomb s'implementen uns listeners que ho faciliten però el problema està en que no és una View, sinó que és aquesta més els fills que conté i en el cas de la llegenda el nombre de fills és totalment indeterminat.

Per dur a terme el drag and drop primer s'implementa un TouchListener sobre la vista general. Quan l'esdeveniment que es produeixi sigui la pulsació ininterrompuda s'ha de controlar que sigui sobre la View que desitgem moure. Llavors es crea una copia a baix nivell que serà la que es mourà per la pantalla i l'original la tornem invisible.

En aquest moment es crea l'esdeveniment que es controlarà en el DragListener. Aquest gestió els tipus d'esdeveniments ja sigui l'inici, final, entrada i sortida d'una zona on es pot fer el drop. Com no hi ha diferents zones on realitzar el drop, aquests dos últims es deixen sense implementar. En canvi, a l'inici d'arrossegar guardarem la posició inicial i al final obtindrem la posició final. Per realitzar la translació efectiva de la vista, ja que el que arrossegàvem era una copia bàsica, calcularem la nova posició relativa al centre de la vista ja que es on l'usuari sol pitjar. Un cop deixat anar la copia es destrueix i la View es torna visible a la posició que ha desitjat l'usuari.

El problema apareix si es vol utilitzar drag and drop a versions inferiors a Honeycomb perquè el drag and drop no està implementat i s'ha de crear des de zero. Per implementar-ho, com s'ha fet en aquesta aplicació, s'han de crear interfícies que representin el drag, el drop i el procés de cancel·lar si es fes fora de zona. Però la tasca més feixuga apareix alhora de realitzar la copia base de la view. Aquesta copia s'ha d'incloure en un bitmap, que serà el que es mourà i s'han de reproduir tots els moviments del dit per la pantalla.

5.3.1.4 Navegació i ViewPager.

Com es tindran diversos elements s'ha d'implementar la navegació entre aquests. Ja siguin components o dashboards, la navegació implementada ha de ser la mateixa. Les dos opcions que es plantegen són que l'usuari esculli cada cop l'element que vol veure o que esculli una i després a partir d'aquesta pugui accedir a les més properes i així recursivament. Per seguir el precepte de fer el més ràpid possible l'accés a l'usuari l'opció adient serà la segona.

Google subministra un element per gestionar Fragments que s'anomena ViewPager. Aquest element conté un adaptador on emmagatzemar tots els fragments que vulguis i navegar entre ells lateralment, lliscant el dit per la pantalla i canviant al següent. Google Play utilitza aquest element per passar entre les diferents categories d'aplicacions. Com l'aplicació està enfocada als fragments l'adaptació d'aquesta aquest tipus de navegació és immediata.

Per implementar el viewpager primer s'haurà de recuperar totes les dades necessàries per crear els fragments que es desitgin introduir en aquest. Aquestes dades estaran a la base de dades, per tant, s'haurà d'accedir per una asynctask ja explicada anteriorment. Un cop cop creats tots els fragments, es genera un adaptador que serà el que gestionarà el viewpager. Per saber a quin fragment s'ha de situar, es rebrà com argument en la creació de la navegació l'element inicial i només caldrà enfocar l'adaptador en aquella posició.

S'ha de tenir en compte que per mostrar l'estat del sistema, és a dir, el nom del component o del dashboard que s'està visualitzant en aquell moment i hauria d'aparèixer a l'actionbar tal com s'ha dissenyat, s'haurà de controlar el canvi de fragment i actualitzar el nom disponible.

L'avantatge d'implementar aquesta solució és que el ViewPager no carrega només un de fragment sinó que també els seus dos contigus, sempre que existeixin. Per tant, l'espera de càrrega de l'usuari disminuirà oferint un resultat molt més fluït fins i tot semblant que només hi ha càrrega a l'inici depenen de la velocitat en que passi pels elements.

5.3.1.5 Adaptació tauletes.

La implementació de l'aplicació per tauletes té dos vessants.

El primer de tots són els layouts XML. S'han d'implementar de nou tots aquells que es diferenciïn de la versió mòbil. Perquè de forma automàtica l'aplicació sàpiga quin layout ha de triar es crea una carpeta *layout-xlarge-land* o *layout-xlarge*, segons la versió vertical o apaïsada, dins els recursos de l'aplicació. Els fitxers XML han de respectar els noms dels seus homogenis de la versió normal.

El segon és el codi. Per saber si es tracta o no d'una tauleta s'ha implementat un codi de suport que retorna un booleà amb la resposta. Aquest s'utilitza per afegir els controls de les diferencies existents, ja sigui la inclusió d'un segon Fragment o de nous botons.

El principal problema que s'ha trobat és alhora d'implementar el disseny de visualització de dashboards i components en tauletes. Aquest disseny afegeix un llistat amb tots els elements disponibles, per exemple els dashboards que tingui l'usuari. Teòricament seguin les bones pràctiques de Google amb Fragment no hi hauria cap problema però no és així. Els Fragments tenen una limitació important i és que no poden tenir altres Fragments dins seu. Això és important perquè qui estan encarregades de gestionar tots els Fragments són les Activity. Aquestes només donen suport a un únic gestor de Fragments, per tant, s'ha de triar entre un gestor bàsic per tenir el llistat i un fragment on es visualitzarà el dashboard o el gestor avançat de fragments que ja hem explicat el ViewPager. Teòricament, això vol dir que s'ha de triar entre la navegació gestual o lateral. però es vol que l'aplicació sigui ràpida i consistent per l'usuari i si tenim una navegació pel mòbil que hem trobat que és molt més eficient no s'ha de renunciar per les tauletes.

Per poder replicar la navegació del mòbil es deixa com està la part del View pager i s'afegeix el llistat però no utilitzant un FragmentList, sinó creant el llistat des de zero amb un ScrollView, introduint tots els elements i els seus corresponents listeners en cas que l'usuari pitgi sobre ells. El resultat per l'usuari és exactament el mateix que amb la construcció d'un FragmentList però afegim la possibilitat que

l'usuari pugui fent servir un moviment de swipe per passar entre dashboards i accelerar la seva càrrega.

5.3.1.6 Actionbar.

Ja s'ha esmentat l'aparició d'aquest element a partir de Honeycomb. Per implementar-ho s'ha de fer a les activity i no pas en els fragments ja que aquest element depèn de les primeres.

Si es vol que una de les accions incloses dins de l'Actionbar afecti al fragment que s'està visualitzant s'haurà de aconseguir el fragment actual mitjançant el gestor de fragments de l'activity i fer una crida directa aquest amb el mètode que es requereixi.

La problemàtica apareix si es vol implementar aquest element en versions anteriors del sistema operatiu android ja que a les llibreries de suport d'android no apareixen implementats. Perquè sigui possible s'ha afegit a l'aplicació la llibreria SherlockActionbar. Aquesta, no deixa de ser una extensió de les llibreries de suport realitzada per Jake Wharton i OpenSource, per incloure una simulació d'ActionBar a l'aplicació. Perquè funcioni en comptes d'estendre de Fragment la classe passa a estendre de SherlockFragment i així successivament amb tots els tipus de Fragment i d'Activitys.

5.3.1.7 Icones.

Una de les últimes propostes de bones pràctiques a l'hora d'implementar aplicacions en android és la utilització d'unes icones predeterminades en forma. Aquestes icones estan disponibles a l'anomenat Android Assets Studio dins de googlecode.com.

La implementació d'aquesta bona pràctica ofereix una forta consistència amb tot l'entorn android ja que les metàfores gràfiques associades són fàcils d'entendre. S'ha dut a terme perquè al repassar aplicacions tant populars com Gmail, Facebook,

Instagram i Evernote.

A més a més, al utilitzar l'assets studio, els fitxers que genera ja estan preparats per suportar les diferents mides de pantalla.

5.3.2 Model.

5.3.2.1 Accés base de dades.

Tal com s'ha vist en l'apartat de disseny, aquesta aplicació android té una base de dades SQLite implementada. Qualsevol activity pot accedir a una base de dades directament, només cal el nom i la versió d'aquesta. En canvi, els patrons d'implementació de google demana eliminar aquesta pràctica. El motiu es per encapsular la comunicació amb la base de dades i estandaritzar les crides aquesta.

Les recomanacions s'inicien amb la creació d'una classe java que extengui de SQLiteOpenHelper, classe nativa d'android per gestionar bases de dades SQLite. Aquesta nova classe és la que s'encarregarà de la creació, actualització i eliminació de les taules de la base de dades i també de l'obertura i tancament de la base de dades. Una recomanació o bona pràctica de java amb les connexions a bases de dades és la creació d'una classe on es guardin totes les configuracions de la base de dades en constants com el nom, la versió, nom de les taules,etc.

Un cop tenim definit la classe d'accés s'ha d'implementar una classe per on es farà la connexió a la base de dades per inserir, actualitzar, eliminar o consultar registres a les taules que conformen aquesta. Exactament, el que s'ha creat ha estat una interfície base on es recullen els mètodes base de totes les taules, i cada una d'aquestes tindrà la seva classe per gestionar en concret cada cas.

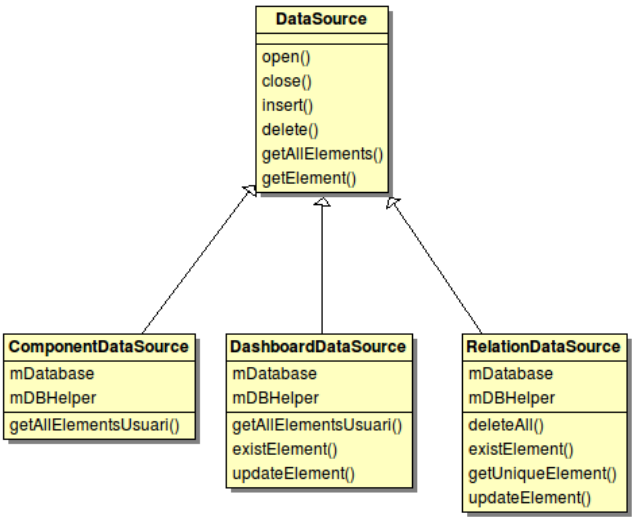


Figura 52. Herència accés a la base de dades.

Amb aquesta implementació s'aconsegueix un accés a la base de dades robust i igual per totes les activity fent que aquest sigui encapsulat com en el disseny base de l'aplicació.

A més a més, l'aplicació fa que la memòria necessària sigui menor ja que no dupliques la base de dades en memòria de l'activity, sinó només quan la necessites.

5.3.2.2 Asynctask.

Una altre de les recomanacions de google en el disseny de interfícies és la utilització d'Asynctask (tasques asíncrones) que permeten un ús correcte i fàcil del thread UI de l'activity.

El thread UI d'una activity és el thread principal on es reben les interaccions de l'usuari com les pulsacions de botons o de la pantalla. Si fem operacions amb força pes de processament o operacions amb accés lents directament en aquest thread totes les accions de l'usuari s'apilen i es produeix el típic comportament de l'aplicació que es tanca sense saber perquè si només volia tirar enrere d'un activity. Per exemple, l'usuari prem X vegades el botó de tornar enrere i s'apilen. Quan s'acaba de dur a terme l'operació que ha bloquejat el thread principal s'executen totes les accions de l'usuari i no és una sinó que són varies aquestes es van executant fins i tot produint la sortida de l'aplicació. Un comportament típic d'un desenvolupador seria

donar la culpa a l'usuari ja que ha premut massa cops el botó, però realment si en el primer cop que pitja el botó sortís de l'activity les altres no s'haurien produït.

Les asynctask sorgeixen per solucionar aquest problema creant un thread que corre en background per alliberar el thread UI i l'exemple anterior no es produeixi. Aquest thread especial es produeix en el mètode *doInBackground().* Per tant, totes aquestes operacions que ocupin molt de temps s'hauran d'executar dins d'aquest mètode.

Quan aquesta classe s'implementa dins l'Activity sovint comporta cometre errors de concepte. Com estem dins la classe de l'Activity podem accedir a tots els mètodes i variables des del mètode doInBackground(), fins i tot podem accedir a les Views que s'inclouen en aquesta. Si aquest és el thread que corre en background no pot tenir accés directe a cap d'aquests elements ja que es podrien produir errors de memòria amb el thread UI. Si volem accedir algun d'aquests elements s'ha d'utilitzar un Handler o esperar a la finalització del thread en background i fer les crides des de el mètode onPostExecute() de la classe asynctask que si que té l'accés al thread UI.

En l'aplicació s'utilitzen les asynctask per realitzar les consultes, insercions, actualitzacions i eliminacions de la base de dades. També s'utilitzen per les peticions http al webservice de pentaho per l'autentificació i la recuperació dels components. En el cas de la base de dades es fa ús de les classes d'accés aquesta i es tanca la connexió dins el thread en background per alliberar abans memòria i evitar errors de duplicitat d'accesos a la base de dades si es tornes enrere en l'activity.

S'ha de tenir en compte que es pot caure en l'error d'executar les peticions web de representació gràfica també en aquest tipus de thread però en aquest cas, les peticions http dels Webview ja creen un altre thread connectat directament amb el thread principal i la view que omplirà amb la seva petició. Des de google s'avisa que realitzar aquesta tasca en asynctask no es donarà suport en futures versions del sistema operatiu.

Per facilitar no cometre l'error de cridar mètodes o variables fora del thread UI

les asyntask es guarden en un altre paquet de l'aplicació i a l'activity es sobreescriuen els mètodes que si tenen accés al thread UI.

S'ha de tenir en compte que el mètode PostExecute es cridarà sempre que acabi el thread en background, si l'activity ja ha finalitzat i accedim a parts d'aquesta saltarà una NullPointerException i finalitzarà l'aplicació. Per tant, s'ha tractat aquest cas amb les clausules try/catch.

5.3.2.3 ListAdapters.

Les ListActivity presenten un problema exponencial si no s'implementa una de les bones pràctiques més promocionades des de google. Aquesta bona pràctica sorgeix en una conferència de Google IO 2010.

Els llistat no deixen de ser una acumulació de views organitzats de forma vertical per poder fer scroll entre ells. Hem de tenir en compte que si tenim deu entrades a un llistat el thread UI d'un mòbil normal no presentarà cap problema, però si en comptes de deu són cent o mil la cosa canvia. El motiu és que el thread UI passa a gestionar tot i cada un dels elements, totes les Views que formin (text, buttons, imatges, etc...) cada element. El thread gestiona posició, visibilitat, dibuix. En deu o 20 entrades no es nota el problema que s'esta generant, però amb un centenar la suavitat de moviments es degrada molt perceptiblement per l'usuari.

La bona pràctica consisteix en modificar adaptador del llistat per només visualitzar els elements visibles, la resta no s'arriben ni a crear sinó es mostren per pantalla. Això fa que la suavitat de moviments sigui perfecte tot i que sigui un llistat de deu mil elements.

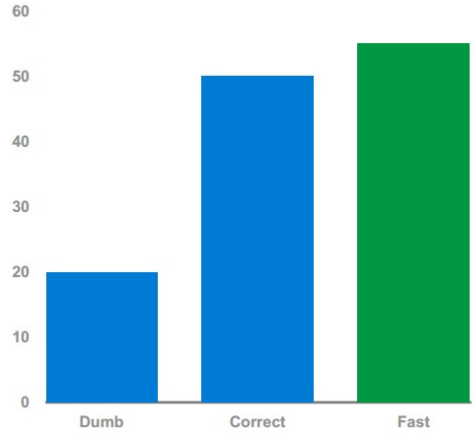


Figura 53. Comparativa extreta de google IO 2010, world of listviews

A més a més, s'ha implementat la versió més ràpida de totes que consisteix en declarar de forma estàtica els elements comuns en un ViewHolder.

5.3.2.4 WebView.

Els Webview que s'han implementat a l'aplicació han de modificar tant el client base com el client Chrome.

El primer, el client base o InsideWebViewClient ha de forçar que la petició web s'executi dins l'aplicació. Per defecte, aquesta petició passa a un dels navegadors del dispositu però es vol que es visualitzi dins l'aplicació amb l'estructura desitjada.

En el cas del client chrome o InsideWebChromeClient s'ha modificar per evitar consumir massa recursos en la base de dades de cache que es crea quan es fa servir un webview. Si no es controla aquest fet el consum de memòria es dispara alentint tot el procés.

5.3.2.5 JavaScript Interface.

Als webviews es pot associar una interfície JavaScript que pot rebre peticions a nivell client incloses a HTML o en fitxers JavaScript.

Donada l'arquitectura base de la solució, aquesta interfície és la que contestarà

a les peticions dels components de pentaho.

S'ha de tenir en compte que aquesta interfície serà consultada fora del thread UI de l'aplicació així que tota la configuració del gràfic, i les dades en el cas que el component sigui CDA, ha d'estar inclòs dins d'aquesta. Un altre solució seria utilitzar handlers per accedir a les dades però s'ha descartat per alliberar totalment el thread UI d'aquesta tasca.

Aquesta interfície implementa un inici i finals simbòlics que si que accedeixen al thread UI mostrant o no un missatge de càrrega a l'usuari per saber si evoluciona el procés.

A més a més també s'accedeix al thread UI per visualitzar la llegenda en cas que la configuració del gràfic així ho digui. Es realitza per separat de la fi simbòlica de la càrrega del gràfic perquè es genera a nivell de programació aquesta View i així no provocar una demora a l'usuari.

A l'aplicació existeixen dos tipus d'interfície segons el component sigui CDF o CDA. El motiu és que amb la captura de dades del component CDA ja disposem de la llegenda a l'activity, en canvi en el cas del CDF la llegenda s'obté per mitja de la interfície.

5.3.3 Captures de l'aplicació.

Captura login.

Figura 54. Captura de la pantalla de login de l'aplicació.

En aquestes dues captures es veu l'estructuració de la pantalla de login i l'avís visual i escrit alhora d'accedir-hi. També es pot observar la icona d'accés a la configuració a la barra superior.

Captura pantalla inicial.

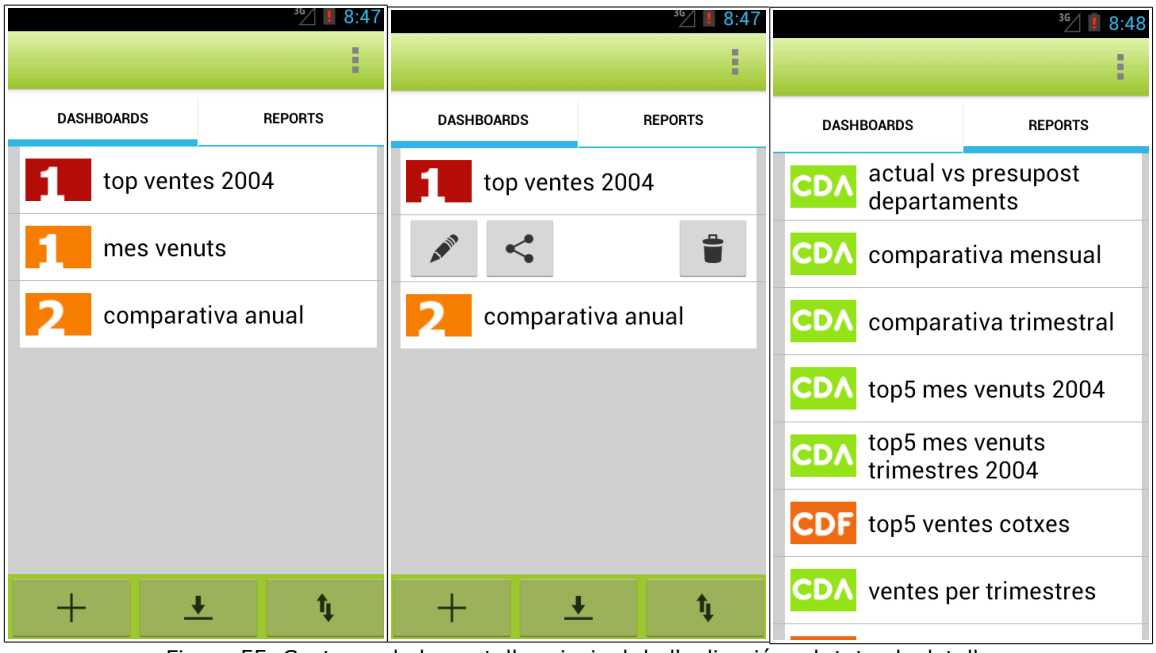


Figura 55. Captures de la pantalla principal de l'aplicació amb tots els detalls.

En aquestes tres captures es mostren els detalls més importants de la pantalla inicial un cop accedit a l'aplicació. La dreta tenim el llistat de components disponibles diferenciats segons la procedència de les dades ja sigui CDF o CDA. Les dos de l'esquerra tenim el llistat de dashboards. A la part inferior tenim disponibles les opcions de crear, descarregar i ordenar dashboards. A la captura del mig es pot veure el canvi que es produeix al seleccionar un element del llistat amb les opcions d'editar, compartir o eliminar el dashboard seleccionat.

Creació dashboard.

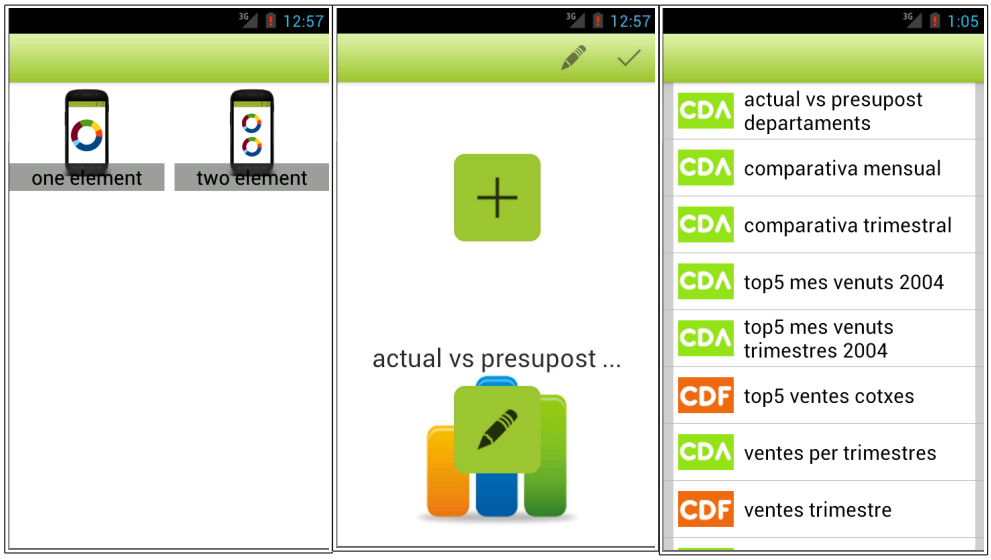


Figura 56. Captura del procés de creació de dashboards.

En aquestes captures es poden veure les tres etapes de creació d'un dashboard. En la primera s'escull l'estructura, en la segona es selecciona quin és el lloc on es vol afegir un component i en la tercera s'escull quin és el component.

Si ens fixem en la captura del mig podem apreciar la modificació de l'actionbar per poder accedir a l'edició del nom del dashboard i acceptar el procés de creació.

Visualització i edició de dashboards.

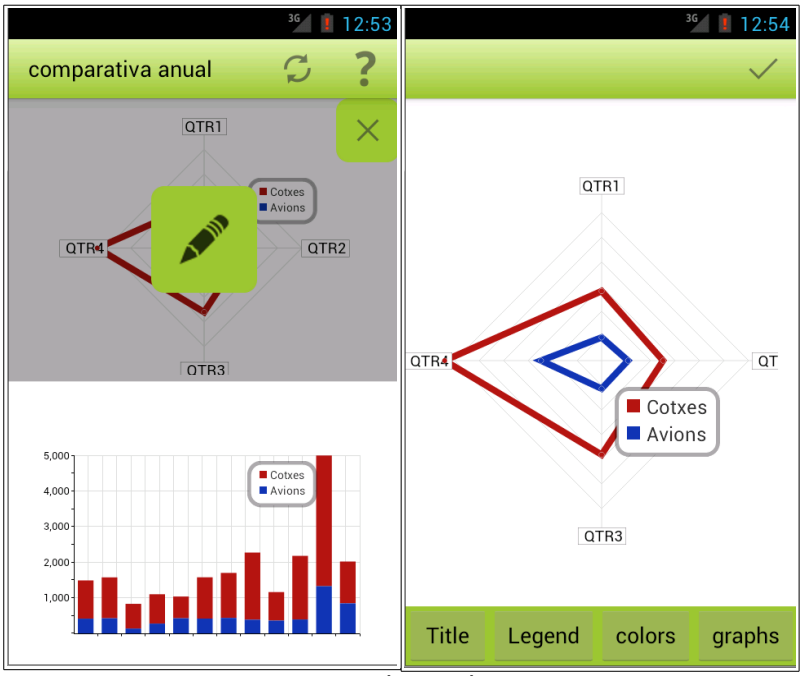


Figura 57. Captures de la visualització i edició de components d'un dashboard.

En aquesta captura es pot veure com es presenta a l'usuari un dashboard format per dos components i l'edició d'un d'aquests. Si es realitza la selecció d'un dels components s'afegeix l'opció d'iniciar l'edició amb un botó evident i de fàcil pulsació.

En la edició es pot apreciar les modificacions possibles al peu de la pantalla i l'acceptació d'aquests canvis premen el botó present a l'actionbar.

Ajuda o documentació.

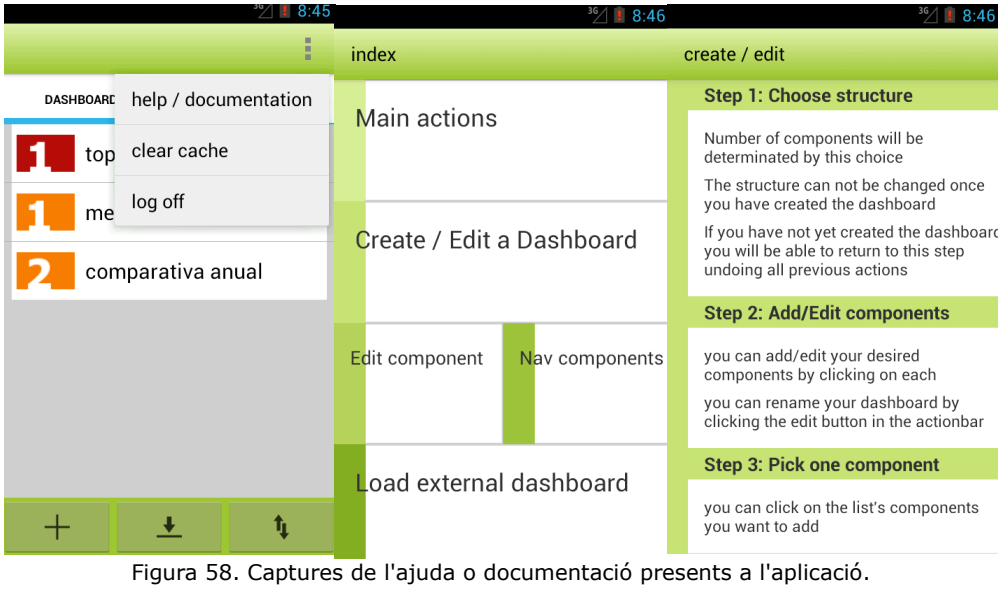


Figura 58. Captures de l'ajuda o documentació presents a l'aplicació.

En aquestes captures es mostra l'accés i forma d'ajuda present a l'aplicació. Per accedir-hi s'utilitza el desplegable present a l'actionbar de la pantalla inicial.

L'ajuda presenta un index de fàcil accés i tota l'ajuda de cada un dels temes està present en una sola pantalla.

Versió tauletes.

Figura 59. captura de la versió tauletes.

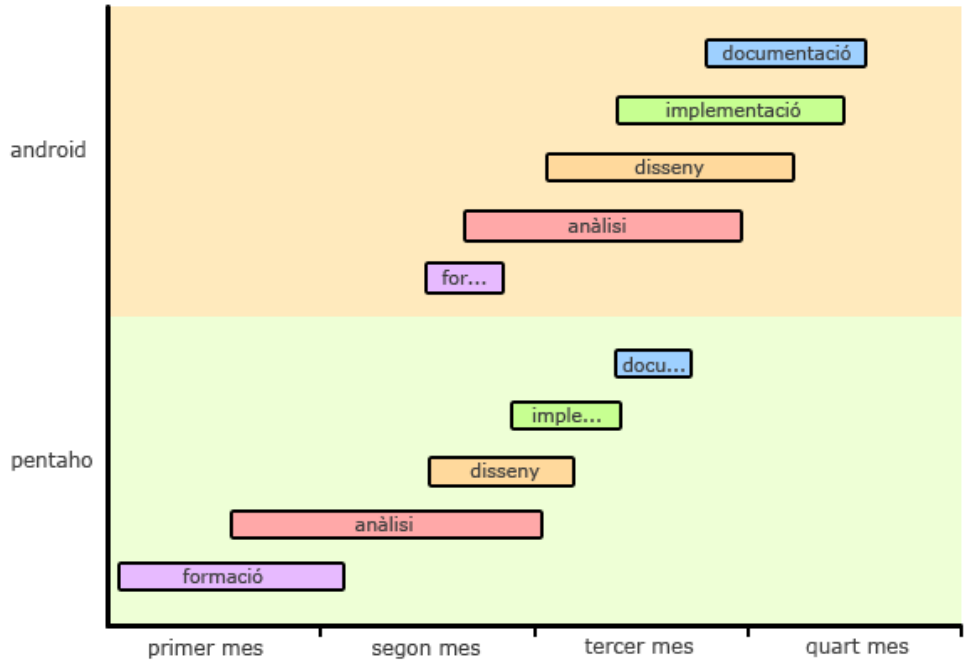
En aquesta captura es mostra la diferenciació amb la versió mòbil, introduint el llistat esquerra de navegavilitat i l'augment de número de components del dashboard.

5.3.4 Demostració de l'aplicació.

http://youtu.be/rNUJuKf1XYc

6 Valoració econòmica.

6.1 Anàlisis del temps de realització del projecte.



Aquesta gràfica on es mostren els temps de realització del projecte s'ha dividit en dos sèries per evidenciar la separació evident entre els dos apartats, el servidor BI i l'aplicació mòbil. Com s'ha vist a l'anàlisi de l'aplicació, la comunicació entre totes dues parts es limita a peticions HTML i crides a una interfície. Un cop definides les estructures de totes dues parts el disseny i la implementació van per separats. Aquest detall fa que el projecte podria ser dividit en dos equips diferents per agilitzar el procés.

Centrant-nos en la sèrie de pentaho es pot veure que la realització del projecte s'ha centrat molt en l'anàlisi. El motiu és la complexitat interna que té el servidor de BI alhora de realitzar algunes de les tasques i entendre el seu funcionament. També perquè hi ha molt desconeixement de part meva en l'apartat d'MDX i gestió de consultes, i perquè a pesar que hi ha documentació disponible per la plataforma

aquesta està estructura de forma caòtica i com la corba d'aprenentatge és molt marcada a l'inici, costa acostumar-s'hi. Aquesta sèrie també reflexa l'estil del model de l'aplicació, un estil en cascada on una etapa d'anàlisi forta afavoreix el disseny i la implementació.

Si ens centrem en la sèrie de l'aplicació mòbil es pot veure que, al contrari de com passa amb el servidor BI, reflexa l'estil del disseny centrat en l'usuari fent que les etapes estiguin molt més sobreposades. El motiu de l'endarreriment de l'inici de l'apartat d'anàlisi es deu a que primer s'havia d'analitzar l'estructura bàsica de l'aplicació per poder encarar l'aplicació cap un camí o un altre. Explícitament no hi ha iteracions en la creació de l'aplicació sota el model UCD però la presencia de les tres etapes casi fins el final demostra el constant anàlisi i disseny implícit en la creació.

6.2 Valoració del cost econòmic del projecte

La mitja de dedicació del projecte ha estat de 20 hores setmanals com a mínim. si s'exclouen les hores de formació, ja que no comptabilitzen alhora de realitzar un pressupost, el període de realització del projecte ha estat unes 12 setmanes.

La realització del projecte es pot separar en tres tipus de qualificacions ja siguin d'anàlisi, programació i documentació.

tipus	cost unitari	quantitat	acumulat
analisi	60,00 €	110	6.600,00 €
programació	40,00 €	90	3.600,00 €
documentació	40,00 €	20	800,00 €
		total	11.000,00 €

El cost unitari és aproximat segons les estimacions que es faciliten en la realització d'aquest projecte.

6.3 Viabilitat econòmica del producte

Roambi, es una aplicació de visualització de dades per Iphone. Aquesta aplicació té un cost de 99 euros per any i usuari.

Si partim d'aquest cost aproximat on serien els nostres competidors ja establerts i el nostre projecte seria de nova implementació el preu per any i usuari es veuria limitat a 80 euros. Si establim els següents supòsits inicials.

1. Inversió inicial. **11.000 €**
2. Preu producte. **80 €**

El punt mort d'una empresa és el punt que s'ha d'assolir per obtenir beneficis i no generar pèrdues. Si realitzem una simple regla de tres, ometen els costos de manteniment de l'empresa, obtenim que s'haurien de crear 138 vendes d'aquest producte per assolir-lo.

Es pot arribar a la conclusió, que la viabilitat de la solució implementada en aquest projecte és molt elevada si tenim en compte que cada client no es limitarà a adquirir una llicencia i a més, els clients a les que sol anar dirigit aquest producte són empreses suficientment gran com per adquirir un volum important de llicencies per donar accés als seus directius.

6.4 Preu implantació del producte

El preu d'implantació del producte variarà segons l'empresa. Tal com es comentava en els requisits és necessitava un servidor i un dispositiu mòbil. En el cas que l'empresa ja disposi d'aquest dos elements el preu d'implantació només dependrà de la configuració del servidor.

Servidor.

En el cas del servidor s'ha agafat com a referencia l'empresa Dell.

- Intel Xeon 3.10 Ghz 4 nuclis.
- 16 gbs de RAM.
- Linux Red Hat enterprises.
- RAID 1 TB.

Preu aproximat de 1.200 €

Dispositiu mòbil.

En el dispositiu mòbil s'ha agafat com a referencia l'ultim model presentat per la mateixa Google, Nexus 4.

preu aproximat de 300 €.

Instal·lació o configuració.

Per la tasca d'instal·lació o configuració del producte es requerirà la presència d'un tècnic informàtica durant una jornada laboral completa. La tarifa del qual serà de 40 €/hora.

7 Conclusions.

Primer de tot vull realitzar una petita valoració del projecte en general. Aquesta, ha estat bastant positiva, més en la apartat de l'aplicació mòbil que en l'eina BI. Ja he comentat amb anterioritat que la principal dificultat al intentar aconseguir els objectius que m'he marcat en aquest aspecte és la forta corba d'aprenentatge que he pogut apreciar en l'eina BI.

Tècniques.

Respecte als objectius d'entendre el funcionament de Business Intelligence com crear l'arquitectura de comunicació entre l'eina BI i el dispositiu mòbil han estat completament assolit. Crec que la solució trobada ha estat aïllada a suficient alt nivell com perquè pugui ser adaptada a qualsevol tipus de dispositiu, fins i tot un altre servidor com explicaré millor en una de les possibles millores o futures ampliacions.

Respecte als objectius referents a l'aplicació mòbils des de el meu punt de vista també s'han assolit. El resultat s'hagués pogut millorar encara si s'hagués realitzat un estudi heurística amb un expert i també un test d'usuaris però la falta de temps i altres inconvenients m'han impedit afegir aquests elements del disseny d'aplicacions centrades en l'usuari.

Tenint en compte la motivació personal referent al desenvolupament front-end s'ha intentat realitzar un disseny detallat tant a nivell gràfic com d'usabilitat.

La principal problemàtica ha estat les versions del Sistema Operatiu Android i les seves versions de client web amb les llibreries d'HTML5 a les que tenen accés. La sort en aquest aspecte és que des de que es va començar el projecte fins ara les actualitzacions només han produït millores, fins i tot WC3 ha publicat la pròxima estandarització d'HTML5 fent més robusta l'aplicació i millorant els temps de resposta.

Una altre problemàtica són les velocitats de connexió a Internet dels dispositius mòbils, però al igual que amb l'anterior problemàtica, aquesta només pot anar a millor amb el temps.

Personals.

Respecte a la corba d'aprenentatge ha estat frustrant la dificultat que es presenta alhora de realitzar consultes MDX ja que no hi ha cap tipus de compilador que mostri errors, o funciona o no i el motiu del error no es presenta en cap moment.

Una altre de les frustracions va ser iniciar el projecte pensant en la versió Gingerbread d'android i descobrir que no hi havia drag and drop quan una de les primeres idees de l'aplicació era la ordenació i la situació de la llegenda del gràfic. Per sort, vaig trobar la implementació del drag and drop de forma manual.

En moments ha estat desesperant la lentitud del emulador del SDK d'android alhora de carregar elements al webview. Jo disposo d'un dispositius mòbil, però és una sola versió i s'ha de comprovar que funcioni correctament en les altres. A més a més, s'ha d'afegir les tauletes que no en disposo de cap i la quantitat de components és més gran per tant el temps d'execució augmentava.

Un de les coses que més satisfacció m'ha produït ha estat l'adaptabilitat assolida personalment alhora de dissenyar i implementar l'aplicació android amb el disseny centrat a l'usuari.

També em satisfà la implementació de la versió per tauletes i sobretot la visualització de dashboards on he aconseguit un combinació d'elements a priori no possibles com és el ViewAdapter i un llistat.

7.1 Futures millores o ampliacions.

Les futures millores o ampliacions d'aquest projecte es basarien en extremar la idea base d'una solució híbrida, tirant més cap a la nativitat de l'aplicació o generalitzar-la per crear un aplicació web.

7.1.1 Aplicació nativa.

La representació gràfica de dades s'ha realitzat amb una llibreria gràfica en HTML5. Un dels punts més febles es la manipulació d'aquest gràfics per part de l'usuari, no només les característiques bàsiques com el color, forma o mida, sinó la interactuació.

Per potenciar això i veient que l'aplicació pot accedir a les dades amb les que es construeixen els gràfics amb components CDA, es podria realitzar la representació gràfica de forma nativa amb llibreries gràfiques d'android SDK.

Aquesta proposta de millora podria tirar enrere ja que openGL per android no deixar de ser bastant caòtic i el temps de desenvolupament augmentaria molt si es volgués realitzar en una altre plataforma mòbil. Però existeixen motors gràfics aptes per diverses plataformes, per exemple unity, reduint el temps de desenvolupament. A més, la riquesa en representacions gràfiques seria enorme i els temps de resposta també agilitzant el procés per l'usuari.

7.1.2 Aplicació web.

Una altre possibilitat és l'altre extrem, orientar-ho tot a una aplicació web i deixar al mòbil només la interfície.

Les avantatges d'extremar-ho cap a l'aplicació web és una gestió molt més acurada dels usuaris. Es podria incloure un mètode de subscripcions i compartir dashboards molt més efectiu i segur.

Per una altre banda les comunicacions es podrien assegurar molt més, ja que totes les peticions des de el mòbil anirien a un servidor intermedi que garantitzaria la seguretat de les dades en tot moment

Aquesta millora implicaria un augment del cost d'implantació ja que s'hauria d'utilitzar un nou servidor però els beneficis superen clarament als costos.

8 Referències bibliogràfiques.

- Guia d'estil de codi d'Open Source d'Android. http://source.android.com/source/code-style.html
- Guia d'estils i patrons d'interfícies d'usuaris d'Android http://developer.android.com/design/index.html
- Referencia de desenvolupament en Android http://developer.android.com/develop/index.html
- Stack Overflow. A language-independent collaboratively edited question and answer site for programmers http://stackoverflow.com
- Vogella. Java, Eclipse, Android and Web programming tutorials. http://www.vogella.com/
- World of ListView. Google IO 2010. http://www.google.com/events/io/2010/sessions/world-of-listview-android.html
- HTML5 versus Android. Google IO 2011. http://www.google.com/events/io/2011/sessions/html5-versus-android-apps-or-web-for-mobile-development.html
- Pentaho. Business analytics and business intelligence leaders – Pentaho http://www.pentaho.com
- Webdetails. Webdetails.Lead your business http://webdetails.pt
- Wiki Pentaho. Pentaho community wiki. http://wiki.pentaho.com

Annex A. Manual d'usuari servidor pentaho.

1 Instal·lació nou servidor pentaho BI + Ctools (opcional).

En cas de no disposar d'un servidor pentaho disponible o si es desitja instal·lar un de nou es pot seguir el tutorial escrit per Pedro Alves un dels Guru del desenvolupament sobre Pentaho i creador de CDA i CDF d'entre d'altres plugins disponibles.

http://pedroalves-bi.blogspot.com.es/2011/12/back-to-basics-step-by-step-pentaho.html

Dins del tutorial es pot escollir la opció d'instal·lar-ho en un entorn Windows.

2 Instal·lació del repositori base i el components CDF.

Per instal·lar el repositori base i els components es recomana tenir aturat el servidor de pentaho. En cas que no fos possible, un cop finalitzat el procés caldrà força al servidor a refrescar la caché dels repositoris. Per fer aquesta tasca s'haurà d'entrar a l'aplicació web de pentaho, clickar a "tools/refresh/repository cache".

Per instal·lar el repositori:

1. Copiar el contingut de la carpeta "*pentahoUB_pentaho*" al directori dins on estigui instal·lat el servidor "*biserver-ce/pentaho-solutions*"

Per instal·lar els components CDF:

1. Accedir al directori "biserver-ce/pentaho-solutions/system/pentaho-cdf/js".
2. Opció A (substituir tot el fitxer).
    1. Agafar el fitxer "./corecomponents/CoreComponents.js" del projecte i substituir el del servidor.

3. Opció B (afegir a fitxer existent).
    1. Amb un editor de text pla obrir el fitxer "*CoreComponents.js*".
    2. Afegir al final tot el contingut del fitxer "*componentsCDF.js*".

Per instal·lar la xaction perquè funcioni el component CDF:

1. Accedir al directori "*biserver-ce/pentaho-solutions/cdf/components/*"
2. afegir el fitxer "./xaction/*dadesarrayPFG.xaction*"

3 Creació fitxer de dades.

Per crear un fitxer de dades amb els paràmetres de la consulta que es realitzarà a través de la xaction s'ha de seguir aquestes passes.

1. Crear un fitxer javascript a "*pentahoUB_pentaho/data*"
2. Crear una variable buida.
3. Afegir un objecte JSON com mostra Annex C.1.

4 Creació template CDF.

Per crear el fitxer HTML que servirà com a plantilla s'ha de seguir aquestes passes.

1. Crear el fitxer .html dins el directori base del repositori.
2. Afegir les crides als fitxers javascript amb la configuració base.
3. Definir element/s Canvas on es realitzarà el dibuix de la gràfica/ques.
4. Afegir script amb la definició del component/s CDF (exemple Annex C.2).
5. Afegir script amb la inicialització dels components.

Dins el repositori a la carpeta "*help/documentation"* existeix un fitxer de mostra de template.

Annex B Manual d'usuari aplicació android.

1 Instal·lació de l'aplicació.

Per instal·lar l'aplicació en un dispositiu mòbil, s'haurà de copiar en aquest el fitxer ".apk" dins del dispositiu i procedir a la instal·lació com qualsevol altre aplicació.

Per defecte, un servidor pentaho utilitza la parella usuari/clau:

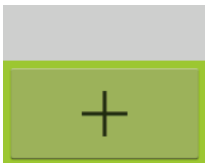
usuari: joe **clau:** password

2 Configuració paràmetres servidor pentaho.

Per configurar el paràmetres del servidor pentaho com la IP, port o protocol, s'haurà d'accedir a través del botó de configuració de l'actionbar de l'aplicació en la pantalla de login.

3 Creació d'un dashboard.

Per crear un dashboard, s'ha de clicar en el botó disponible en el llistat d'aquest.



La primera acció serà introduir un nom al dashboard, aquest no podrà existir amb anterioritat.

Després s'ha de triar l'estructura desitjada pel dashboard prement sobre la

miniatura corresponent.

Un cop escollida s'introduiran els components necessaris prement el botó d'afegir i seleccionant-lo en un llistat.

4 Edició d'un component de dashboard.

Per editar un component d'un dashboard es premerà llarg sobre aquest en la visualització i després sobre la icona d'edició que apareixerà.

Un cop iniciada l'edició del component, aquest es carregarà i es podra modificar utilitzant els botons inferiors i movent la llegenda si s'escau.

Un cop finalitzades totes les tries, es premeria sobre el boto d'acceptar per actualitzar-lo. Si es volgués descartar tots els canvis fets només s'hauria de pitjar el boto del dispositiu per tirar enrere.

5 Ordenació de dashboards.

Per ordenar els dashboards només caldrà pitjar sobre el botó al llistat de dashboards.

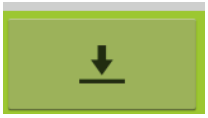
Un cop iniciat es podran arrossegar els elements per ordenar-los com es

prefereixi. Per acceptar l'ordenació s'haurà de prémer el botó de l'actionbar o pitjar el botó enrere del dispositiu.

6 Enviar/Carregar dashboard.

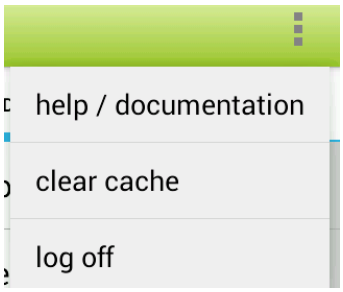
Per enviar un dashboard només s'haurà de pitjar sobre el botó de compartir d'un dashboard. Aquest botó si pot accedir fent una pulsació llarga sobre aquest. Només s'haurà d'introduir el correu electrònic de qui vols que el rebi.

Per carregar un dashboard, només s'ha de pitjar el botó disponible en el llistat de dashboards i seleccionar el fitxer desitjat. Si el fitxer que desitgeu no està disponible assegureu-vos que està al directori on es dipositen totes les descarregues del mòbil.



7 Neteja cache de l'aplicació.

Per netejar la cache de l'aplicació, s'ha d'accedir pel desplegable del l'activitat principal.



Un cop accedit un missatge us informarà que no us en podreu fer enrere si procediu a netejar la cache de l'aplicació.

Per dur a terme la tasca haureu de marcar la casella d'acceptació de les condicions i pitjar el botó de l'actionbar.