Panell de control del negoci

Sistema per prendre decisions amb arquitectura SOA i basat en tecnologia Android

Raúl Díaz Jerez

Enginyeria Superior Informàtica

Consultor: Jordi Ceballos Villach

07/01/2013

Resum

Els dispositius mòbils tàctils, des del primer telèfon a l'actual boom de les tablets, han suposat un abans i un després en la manera d'utilitzar la tecnologia. Han apropat aquesta a un gran número de persones, gracies en part a la facilitat d'us i a la mobilitat que ofereixen. No només el sector d'oci o personal ha acollit amb molta força aquests dispositius, si no que el sector empresarial també ho està fent degut a les gran avantatges que ofereixen. De fet, a dia d'avui, moltes empreses acostumen a utilitzar smartphones o tablets per accedir a la informació. També han sorgit moltes empreses de consultoria dedicades a aquests nous dispositius.

Aquests dispositius s'adapten perfectament a una arquitectura orientada al servei (SOA) que es troba en moltes empreses, per tant, a les que ja utilitzaven l'arquitectura SOA els ha estat molt fàcil adaptar els seus serveis per integrar-los amb aquets dispositius i les empreses que no tenien arquitectures SOA s'estan modernitzant cap a projectes orientats al servei ón aquets dispositius s'adapten a la perfecció.

També s'ha produït un canvi en termes de tecnologia a utilitzar. Si be en ordinadors tradicionals el predominant és el sistema operatiu Windows i últimament la tecnologia .Net de Microsoft està força ficada en el mon empresarial, en aquests dispositius la tecnologia predominant és el sistema operatiu Android o IOS la qual cosa provoca que llenguatges com objective-c o JAVA hagin crescut molt. Amb aquesta gran barreja de tecnologies queda un dubte en l'aire: si tinc una arquitectura SOA en una tecnologia i uns dispositius que utilitzen una altre tecnologia, com es comuniquen entre ells?.

Aquest projecte de final de carrera està destinat a aquest escenari que s'està vivint en la actualitat. Per tant s'ofereix una aplicació per tablet Android per veure en temps real el teu negoci i veure com aquests dispositius poden ajudar a millorar el teu negoci. Juntament amb l'aplicació tablet s'ofereix una capa de serveis sota la tecnologia .Net de Microsoft i tot aquet conjunt sota una arquitectura SOA.

Índex

1. Introducció

1.1 Justificació i context del projecte

Aquest projecte de final de carrera es realitza en el context de l'àrea de xarxes. L'objectiu és la realització d'una aplicació de mobilitat per a la plataforma Android.

L'escenari de partida d'aquest projecte ha estat una proposta de l'empresa de consultoria STS per a la creació d'una Aplicació de mobilitat amb tecnologia Android. Aquesta empresa volia fer un projecte en Android que utilitzés serveis WCF de la plataforma. NET, perquè ells ja treballaven amb aquesta tecnologia.

Aquesta proposta va ser presentada i acceptada per la UOC dins de l'àrea de xarxes. L'acceptació del projecte per part meva, va ser simplement perquè es tractava d'una aplicació de mobilitat en tecnologia Android, que encaixava a la perfecció amb el tipus de projecte que jo volia fer.

Per donar vida al projecte, l'empresa de consultoria, em va presentar l'escenari sobre el qual es basaria el projecte, perquè pogués donar una solució tecnològica adequada a l'escenari presentat.

Les decisions tecnològiques preses per la meva part durant el desenvolupament del projecte, s'han basat en complir els objectius per tenir una aplicació Android que es comuniqui amb serveis amb tecnologia. Net.

1.2 Descripció del projecte

Aquest projecte té com a objectiu realitzar l'anàlisi, disseny i implementació d'una primera versió de l'aplicació de mobilitat descrita per l'empresa STS. Aquesta primera versió serà un producte de demostració per als seus clients. Aquest projecte es realitzarà fent ús d'una arquitectura orientada als serveis (SOA) utilitzant la tecnologia Android per a l'aplicació client per tablet, .Net per desenvolupar la capa de serveis i REST per comunicar l'aplicació tablet amb els serveis.

L'arquitectura SOA, és una arquitectura per la creació de sistemes distribuïts, on s'exposen un conjunt de serveis amb el negoci del nostre domini perquè els diferents components els consumeixin. Aquesta arquitectura és altament escalable i permet el desacoblament entre el negoci i la resta d'elements.

Els serveis comentats anteriorment exposen les seves funcionalitats perquè puguin ser cridades utilitzant els estàndards en missatgeria. L'únic requisit és que el missatge compleixi el contracte del servei.

Durant el desenvolupament del projecte, s'ha creat la capa de serveis que exposa el negoci o funcionalitats necessàries per a les diferents aplicacions, en aquest cas únicament tenim l'aplicació tablet però podrien ser cridats per qualsevol tipus d'aplicació. Aquests serveis s'han realitzat complint els principis bàsics de l'arquitectura SOA que són les següents:

- Contractes estandarditzats: Cada servei ha de definir un contracte perquè el client pugui cridar-lo. El servei per la seva part ha d'estar implementat d'acord al seu contracte.
- Serveis poc acoblats: Els serveis han d'estar poc acoblats per evitar que si es produeix un canvi afecti molt a les aplicacions client. Per tant l'acoblament ha de ser únicament a nivell de contracte del servei.
- Reusabilitat de Serveis: El servei pot ser utilitzat per diferents clients.
- Autonomia de Serveis: El servei ha de ser autònom i oferir la seva funcionalitat de manera independent.
- Serveis sense Estat: Els serveis no han de dependre de l'estat d'altres serveis.
- Descobriment de Serveis: Els serveis s'han de poder descobrir i cridar de manera senzilla.
- Composició de Serveis: Els serveis han de suportar la composició de serveis per crear serveis de més alt nivell.

Amb una arquitectura d'aquest tipus, es facilita la creació d'aplicacions clients de qualsevol tipus com per exemple en aquest cas una aplicació tablet en tecnologia Android que consumeix serveis .Net.

La plataforma .Net ofereix un conjunt d'eines excel·lents per desenvolupar la capa de serveis i dades. Per la seva banda Android ofereix eines molt potents per crear una aplicació de tablet. Gràcies a aquest tipus d'arquitectura podem aprofitar-nos de les eines disponibles per a cadascuna de les tecnologies per desenvolupar els diferents components.

1.3 Objectius

Els objectius que es pretenen assolir amb aquest projecte són els següents:

- Conèixer la plataforma Android.
- Construir amb èxit una aplicació distribuïda sota el paradigma de l'orientació als serveis.
- Veure com es comuniquen dues tecnologies diferents com. NET i Android.
- Aprofundir els meus coneixements de. NET per a desenvolupar la capa de serveis i dades.
- Treballar amb dispositius mòbils actuals com és el cas del tablet.

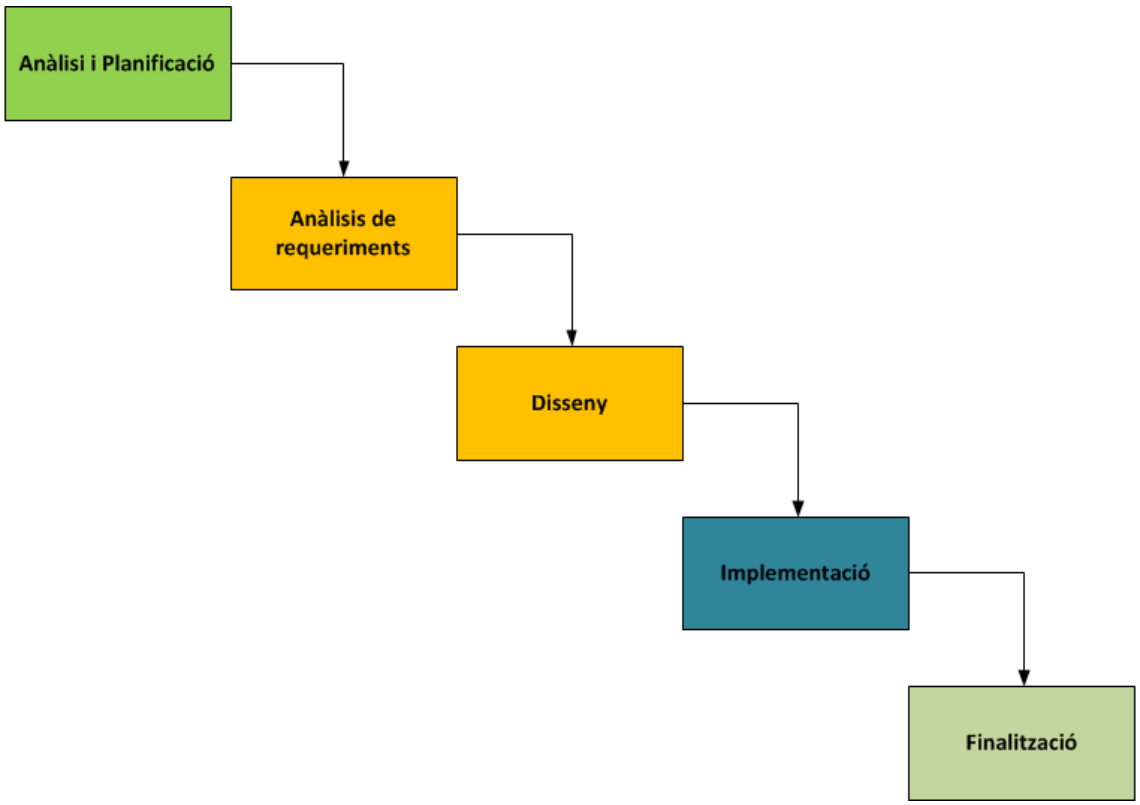
- Utilitzar tecnologia actual com. Net o Android.
- Aprofundir en el llenguatge JAVA.
- Conèixer en profunditat eines de desenvolupament d'aplicacions com Eclipse o Visual Studio.
- Posar en pràctica els coneixements adquirits durant la carrera en el camp de l'enginyeria del programari.

1.4 Planificació

1.4.1 Cicle de vida

El projecte ha estat desenvolupat utilitzant el cicle de vida en cascada, ja que era el cicle que millor s'adaptava al tipus de projecte.

Les etapes que constitueixen aquest cicle de vida són les següents:



Cadascuna de les fases correspon amb els lliuraments establerts del projecte, de manera que al final de cada etapa, s'ha generat i lliurat la documentació pertinent.

Les etapes d'anàlisi funcional i disseny, es van lliurar conjuntament d'acord a les fites establertes.

1.4.2 Detall d'activitats

Anàlisi i planificació

En aquesta etapa s'ha pretès obtenir una visió global del projecte i establir una temporització.

Activitat	Descripció
Anàlisi del projecte	Descripció del projecte a nivell de requisits i tecnologia que utilitzarà.
Temporització del projecte	Temporització del projecte d'acord amb les dates establertes.
Preparar el document	Creació del document resultant d'aquesta etapa.
Entrega de la PAC1	Lliurament del document.

Anàlisi de requeriments i disseny

En l'etapa d'anàlisi de requeriments i disseny, s'ha definit per una banda les necessitats funcionals com no funcionals que ha de cobrir el projecte.

En l'etapa de disseny, s'ha definit com s'hauria implementar el sistema per complir els objectius. També s'ha dissenyat el prototip del projecte.

Activitat	Descripció
Anàlisi funcional	Creació dels casos d'us.
Anàlisi no funcional	Definició dels requeriments no funcionals del projecte.
Disseny de l'arquitectura	Disseny de l'arquitectura del projecte.
Disseny de classes	Disseny dels diagrames de classes.
Disseny de persistència	Disseny del model de base de dades.
Disseny de la interfície	Disseny del prototipus.
Preparar el document	Creació del document resultant d'aquesta etapa.
Entrega PAC2	Lliurament del document.

Implementació

En l'etapa d'implementació, s'ha desenvolupat l'aplicació tablet i la capa de serveis i dades.

Activitat	Descripció
Creació de la base de dades	Desenvolupar el model de base de dades i inserció de les dades de prova.
Creació de la capa de serveis	Creació dels serveis que recullen les necessitats del projecte.
Creació de l'aplicació tablet	Creació de l'aplicació tablet.

Proves d'integració	Proves d'integració del projecte.
Entrega PAC3	Creació del paquet amb el codi i la documentació.

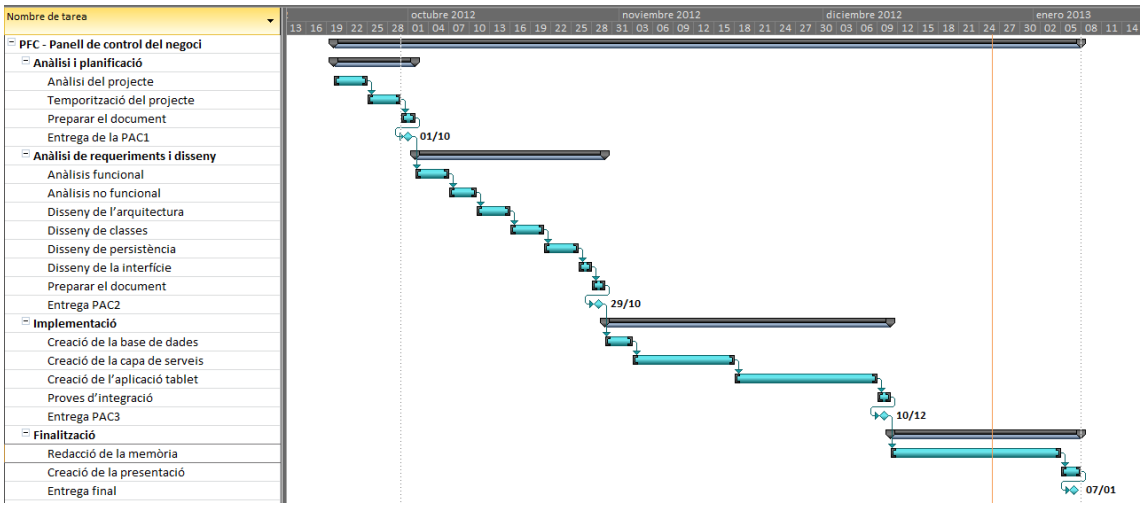
Finalització

Finalment, en aquesta etapa, s'ha generat tota la documentació necessària per al lliurament i presentació del projecte final de carrera.

Activitat	Descripció
Redacció de la memòria	Redacció de la memòria que recull tot el projecte.
Creació de la presentació	Creació d'un PowerPoint amb la presentació del projecte així com la composició del vídeo de presentació.
Entrega final	Entrega de la memòria i del vídeo de presentació.

1.4.3 Temporització del projecte

Descripció	Durada	Inici	Fi
Anàlisi i planificació	**12 dies**	**20/09/2012**	**01/10/2012**
Anàlisi del projecte	5 dies	20/09/2012	24/09/2012
Temporització del projecte	5 dies	25/09/2012	29/09/2012
Preparar el document	2 dies	30/10/2012	01/10/2012
Entrega de la PAC1	0 dies		
Anàlisi de requeriments i disseny	**28 dies**	**02/10/2012**	**29/10/2012**
Anàlisis funcional	5 dies	02/10/2012	06/10/2012
Anàlisis no funcional	4 dies	07/10/2012	10/10/2012
Disseny de l'arquitectura	5 dies	11/10/2012	15/10/2012
Disseny de classes	5 dies	16/10/2012	20/10/2012
Disseny de persistència	5 dies	21/10/2012	25/10/2012
Disseny de la interfície	2 dies	26/10/2012	27/10/2012
Preparar el document	2 dies	28/10/2012	29/10/2012
Entrega PAC2	0 dies		
Implementació	**42 dies**	**30/10/2012**	**10/12/2012**
Creació de la base de dades	3 dies	30/10/2012	02/11/2012
Creació de la capa de serveis	15 dies	03/11/2012	17/11/2012
Creació de l'aplicació tablet	22 dies	18/11/2012	08/12/2012
Proves d'integració	2 dies	09/12/2012	10/12/2012
Entrega PAC3	0 dies		
Finalització	**28 dies**	**11/12/2012**	**07/01/2013**
Redacció de la memòria	25 dies	11/12/2012	04/01/2013
Creació de la presentació	3 dies	05/01/2013	07/01/2013
Entrega final	0 dies		



1.5 Eines utilitzades

Per la realització del treball de final de carrera s'han utilitzat les següents eines i dispositius:

Hardware

- Ordinador portàtil HP Pavilion dv6: Intel Core I7, 6 GB de memòria RAM amb sistema operatiu Windows 7 Ultimate de 64 bits.
- Tablet Asus Tranformer Prime Tf201 amb sistema operatiu Android 4.1.1 Jelly Bean.
- Servidor IIS 7.0 per allotjar els serveis.
- Servidor SQL Server 2008 r2 per allotjar la base de dades.

Software

- Microsoft Visual Studio Ultimate 2010 SP1 entorn de desenvolupament per realitzar la capa de serveis.
- SQL Server Management Studio 2008 R2 per dissenyar la base de dades.
- Entorn de desenvolupament Eclipse per realitzar el projecte tablet.
- Microsoft Office 2010 per redactar la memòria.
- Microsoft Visio 2010 per la realització de diagrames per la memòria.
- Microsoft Project 2010 per la planificació del projecte.
- Microsoft PowerPoint 2010 per crear la presentació del projecte.
- Camtasia Studio 8 per realitzar el muntatge del vídeo de presentació.
- Google Drawing per dissenyar el prototipus.
- Google Chrome amb l'extensió REST Console per provar els serveis.

1.6 Productes obtinguts

Durant el desenvolupament del projecte s'han generat els següents lliurables:

- Pla de treball.
- Document d'anàlisi i disseny.
- Document d'implementació.
- Memòria final del projecte.
- Presentació virtual.
- Projecte de mobilitat per tablet Android.

1.7 Estructura del document

La memòria presenta els apartats més destacables de les activitats realitzades. El primer apartat de la memòria, són els requisits del projecte, on es descriu l'escenari i les funcionalitats que el componen. A continuació es presenta la part de disseny on s'exposa el projecte des del punt de vista arquitectònic. També es presenten les tecnologies seleccionades, els dissenys de classes i el prototip de les pantalles. En el següent apartat ens fiquem de ple en la implementació on es presenten els aspectes i decisions més important que s'han dut a terme durant el desenvolupament del projecte. En l'ultima part de la memòria trobem les conclusions, les línies de obertes, on s'exposa com continuaríem el projecte i finalment el glossari de termes i les fonts utilitzades.

2. Requisits inicials

En aquest apartat, es descriu l'escenari real en que es basarà i l'abast del projecte de final de carrera, així com els requisits i funcionalitats necessàries.

2.1 Escenari de partida

L'empresa de consultoria STS ha detectat la possibilitat de millorar els processos de presa de decisions d'un dels seus clients mitjançant una aplicació de mobilitat. Amb aquest projecte final de carrera es vol construir des de zero una aplicació de mobilitat que consisteix en una aplicació per tablet Android, una capa de serveis que exposin la informació i el disseny d'una base de dades que alimenti l'aplicació mòbil. El projecte doncs serà un projecte pilot o de demostració per tal de presentar al client de STS. L'escenari sobre el que es basarà el projecte es descriu a continuació.

L'empresa client, te la seva estructura comercial dividida en zones geogràfiques que inclouen moltes botigues. Aquestes zones estan controlades per un mànager que s'encarrega de gestionar-les. Com que les zones tenen una gran amplitud geogràfica i dins de cada zona existeixen moltes botigues, és difícil fer un seguiment en temps real dels objectius i vendes de cada botiga. Actualment els managers han de extreure informes de diferents aplicacions cada cert temps. Tot i que aquests informes es poden consultar en qualsevol moment és un sistema poc àgil ja que els managers es troben sempre en moviment i per consultar l'estat de la seva zona han de poder accedir a l'aplicació. Per tal de donar més versatilitat i millorar el seu treball diari, s'ha pensat en una aplicació de mobilitat que permeti al manager consultar la informació de les seves zones en temps real, sense necessitat d'accedir a diferents aplicacions per extreure informes. D'aquesta manera s'agilitza l'accés a la informació i al ser en temps real es poden detectar ràpidament mancances en les seves zones.

Per tal de millorar els processos descrits anteriorment, es crearà una aplicació de mobilitat per a tablet Android amb la qual, s'oferirà al manager un panell de control dividit en components independents, on cada component li oferirà indicadors en temps real de diferents processos de negoci i li permetrà accedir fàcilment a la informació. D'aquesta manera el mànager podrà accedir a la informació de manera senzilla i des de qualsevol lloc. El panell de control estarà dividit en components independents. Aquest concepte l'anomenaré a partir d'ara amb la paraula widget. Aquest widget és un component autònom que mostra informació concreta sobre un domini d'informació, l'usuari pot visualitzar les dades del widget en temps real i interactuar amb ell. Cada widget és un component independent i cadascun d'ells mostra informació sobre el seu domini. El panell de control està format per una

composició de widgets que mostren en temps real informació de diferents dominis d'un mateix negoci.

2.2 Abast de projecte

Tal i com s'ha comentat anteriorment, el projecte consisteix en crear els següents components:

- Disseny de la base de dades pilot d'on s'extraurà la informació de les botigues i de les vendes.
- Disseny de la base de dades on es guardarà la configuració dels widgets.
- Creació dels serveis WCF per tal de poder consultar aquesta informació.
- Creació de l'aplicació per a tablet.

La informació que contindrà la base de dades de les botigues i vendes seran dades de demostració, no seran dades reals del client. Es carregaran inicialment aquestes dades i el projecte es realitzarà basant-nos en aquestes dades.

2.3 Usuaris a considerar

La primera versió del sistema només tindrà en compte un tipus d'usuari, el Manager de Zona.

- Manager de zona: Accedirà al sistema i es carregaran els seus widgets. El Manager podrà consultar la informació que ofereixen els seus widgets, ordenar la posició dels seus widgets i interactuar amb ells.

Encara que aquesta versió només es basa en un usuari si que es contempla la configuració en funció del rol d'usuari. Per tant, quan l'usuari s'identifiqui al sistema aquest verificarà quins components pot carregar en funció del rol.

2.4 Requisits funcionals

Com s'ha comentat anteriorment, l'objectiu principal del projecte es crear un sistema àgil i en temps real de presa de decisions, composat per un panell de control i una sèrie de widgets que proporcionen al usuari informació en temps real del negoci. L' usuari podrà interaccionar amb el panell per dur a terme les tasques que consideri oportunes.

Les funcionalitats s'han agrupat en dos grans blocs:

- Funcionalitats de seguretat: En aquest apartat es detallen les necessitats d'accés a l'aplicació.
- Funcionalitats del panell de control: En aquest apartat es detallen les funcionalitats del panell, dels widgets i els requisits de la informació que es mostrarà.

2.4.1 Funcionalitats de seguretat

Pel que fa referencia a la seguretat, el sistema només oferirà les opcions d'iniciar sessió i finalitzar sessió. L'administració d'accés recau sobre una l'aplicació que utilitza el manager en la actualitat i que queda fora del abast d'aquest projecte. Per tal d'accedir a l'aplicació el manager utilitzarà un nom d'usuari i contrasenya.

Inici de sessió

Permetrà als usuaris indicar el seu usuari i contrasenya per tal d'iniciar sessió i poder accedir al seu panell de control. En aquesta primera versió només es te un tipus d'usuari i components limitats, per tant es carregarà el panell d'usuari d'acord a la seva configuració. En properes versions en funció del rol del usuari es podran carregar diferents configuracions i funcionalitats.

Finalitzar sessió

Finalitza la sessió actual, de manera que pot iniciar la sessió amb un nou usuari.

2.4.2 Funcionalitats del panell de control

La funcionalitat més important del panell de control és mostrar informació rellevant del negoci en temps real de manera clara i concisa. A més també permet al usuari interactuar amb els diferents widgets del panel per tal de prendre decisions. Per fer-ho en aquesta primera versió es mostraran quatre widgets:

- Nombre de comercials que han anat a treballar avui: Els comercials han d'accedir a un sistema intern de l'empresa per tal d'enregistrar la seva presencia en el seu lloc de treball. Aquest widget indica quants comercials han realitzat el login i quants no. L'usuari podrà consultar el llistat d'usuaris que han fet el login i els que no han fet el login.

- Cercador de persones: Aquest component li permetrà cercar un comercial per contactar amb ell. El sistema li mostrarà la informació bàsica de l'usuari per tal que el manager pugui contactar amb ell.

- Nombre de botigues amb/sense vendes: Aquest component, mostrarà la quantitat de botigues que han realitzat vendes i quantes no. El manager podrà accedir al llistat de botigues per tal de poder contactar amb elles.

- Vendes per mesos: Es mostrarà un gràfic de línies que mostrarà l'evolutiu de vendes total de les zones per cada més.

2.5 Requisits no funcionals

2.5.1 Requisits de la interfície

La interfície estarà dissenyada per una tablet de 10.1 polzades en posició horitzontal amb una resolució de 1280 x 800 pixels, aquesta interfície està pensada per ser utilitzada de manera tàctil. Per tant l'objectiu és aconseguir una interfície senzilla i que es pugui utilitzar amb un únic dit.

2.5.2 Requisits de seguretat

El sistema haurà d'identificar a l'usuari per tal de poder accedir al panell i carregar la seva configuració, encara que en aquesta versió només hi ha un tipus d'usuari, es validarà l'usuari i rol. L'autenticació es farà mitjançant un usuari i contrasenya que hauran d'estar subministrades per l'usuari.

2.5.3 Requisits de informació

L'objectiu del panell de control és mostrar la informació de manera clara i concisa a l'usuari, per tal que aquest pugui prendre decisions en base a la informació. Per tant, s'ha de mostrar la informació necessària i evitar la sobrecàrrega d'informació. La informació que mostrarà en cadascun dels quatre widgets serà la següent:

- Nombre de comercials que han anat a treballar avui: Es mostraran dos comptadors amb la quantitat de comercials que ha fitxat i la quantitat que no han fitxat. Prement sobre qualsevol dels comptadors es veurà el llistat d'usuaris que han fitxat o que no en funció d'on toqui l'usuari. Es mostrarà un llistat amb el nom i cognoms de les persones, la botiga, el telèfon i el e-mail de contacte.

- Cercador de persones: Nom, Cognoms, Botiga, direcció de e-mail i telèfon. En cap cas es mostrarà informació relacionada amb la identificació de l'usuari en el sistema.

- Nombre de botigues amb/sense vendes: Es mostraran dos comptadors amb la quantitat de botigues que han venut productes o que no hagin venut cap. Prement sobre qualsevol dels comptadors es veurà el nom de la botiga, nom del responsable de la botiga, e-mail i el seu telèfon.

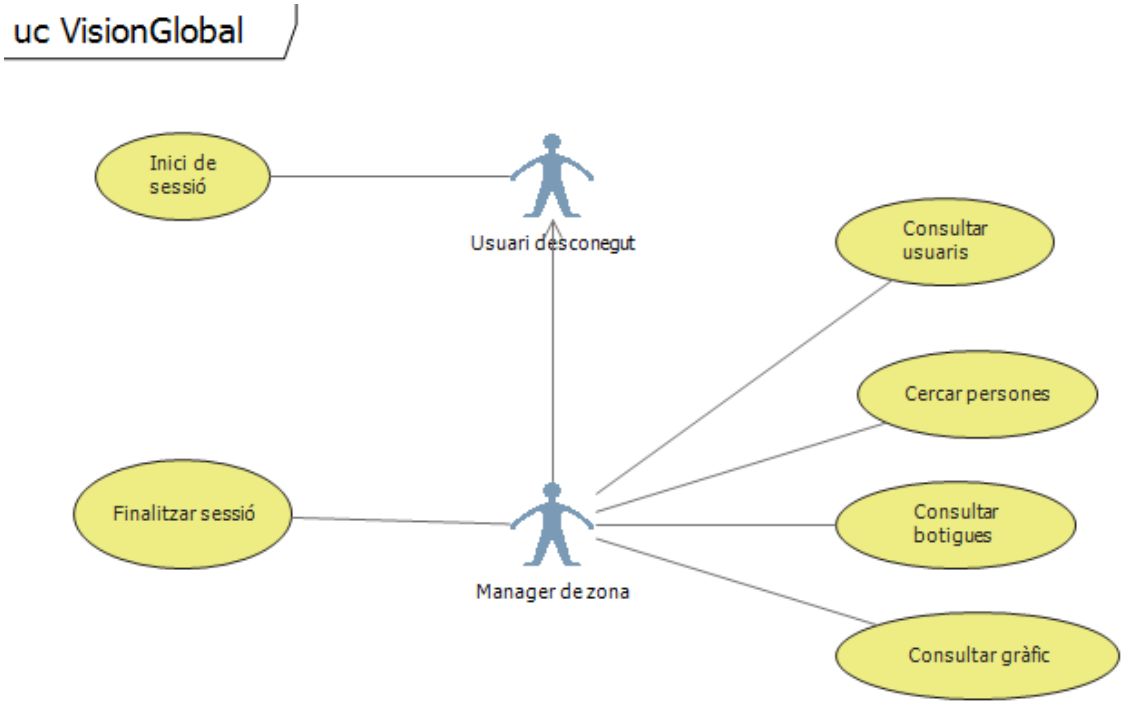
- Vendes per mesos: Es mostrarà un gràfic de línies que mostrarà l'evolutiu de vendes total de la zona per cada més.

3. Anàlisis del sistema

En aquest apartat es recull el resultat de la fase d'anàlisi del sistema. Per tal de dur a terme aquesta tasca, es definiran i detallaran els diferents casos d'ús dels que estarà compost el sistema.

3.1 Diagrames de casos d'ús

A continuació es presenta una visió global dels casos d'us de l'aplicació que descriuen els requisits funcionals de l'aplicació:



Com podem veure, el sistema contempla dos actors:

- Usuari desconegut: Representa l'usuari que vol accedir al sistema. Només pot fer l'acció d'inici de sessió.

- Manager de zona: Usuari identificat en el sistema que interactuarà amb l'aplicació. Serà l'usuari principal de l'aplicació i qui pot fer més accions. És una especialització del primer usuari que inclou aquells usuaris que han passat el procés de identificació.

La següent taula resumeix els cassos d'ús del sistema, aquests cassos d'ús seran detallats en la següent secció del document.

Codi	Descripció	Actor
CU01	Inici de sessió	Usuari desconegut
CU02	Finalitzar sessió	Manager de zona
CU03	Consultar usuaris	Manager de zona
CU04	Cercar persones	Manager de zona
CU05	Consultar botigues	Manager de zona
CU06	Consultar gràfic	Manager de zona

3.2 Descripció textual de casos d'us

3.2.1 CU01 – Inici de sessió

Identificador	CU01
Nom	Inici de sessió
Autor	Raúl Díaz Jerez
Resum	Aquest cas d'ús indica com s'identifica en el sistema un usuari
Actor(s)	Usuari desconegut
Pre-condicions	No hi ha sessió activa
Post-condicions	L'usuari a iniciat sessió o l'usuari no ha pogut iniciar la sessió
Flux normal	• El cas d'ús comença quan l'actor accedeix al sistema per primer cop o bé quan ha finalitzat una sessió anterior. • El sistema demana les credencials a l'usuari. En aquest cas nom d'usuari i contrasenya. • L'usuari indica el seu nom d'usuari, contrasenya i envia la seva informació. • El sistema valida la informació facilitada per l'usuari. • Si la informació és correcta, el sistema inicia la sessió i es mostra el panell de control del usuari amb la seva. configuració.
Fluxos alternatius	• Si les dades indicades per l'usuari en el punt 3 no són correctes, el sistema mostrarà un error i el flux torna al punt anterior.

3.2.2 CU02 – Finalitzar sessió

Identificador	CU02
Nom	Finalitzar sessió
Autor	Raúl Díaz Jerez
Resum	Aquest cas d'ús indica com finalitzar la sessió d'usuari.
Actor(s)	Manager de zona
Pre-condicions	L'usuari a iniciat la sessió
Post-condicions	No hi ha cap sessió activa
Flux normal	• El cas d'ús comença quan l'actor selecciona tancar la sessió. • El sistema mostra un missatge de confirmació. • L'actor confirma el missatge. • El sistema finalitza la sessió.
Fluxos alternatius	• L'actor no confirma el missatge del sistema en el punt 3. • L'actor elimina l'aplicació del llistat d'aplicacions que estan en funcionament en el sistema. Pot fer-ho en qualsevol moment.

3.2.3 CU03 – Consultar usuaris

Identificador	CU03
Nom	Consultar usuaris
Autor	Raúl Díaz Jerez
Resum	Aquest cas d'ús indica com es consulta quins usuaris han fixat i quins no.
Actor(s)	Manager de zona
Pre-condicions	L'usuari a iniciat la sessió
Post-condicions	-
Flux normal	• L'usuari prem sobre l'indicador d'usuaris connectats o usuaris de baixa del widget "comercials que ha anat a treballar avui". • El sistema li mostra una finestra amb el llistat d'usuaris. • L'usuari prem fora de la finestra de consulta per tornar al panell.
Fluxos alternatius	-

3.2.4 CU04 – Cercar persones

Identificador	CU04
Nom	Cercar persones
Autor	Raúl Díaz Jerez

Resum	Aquest cas d'ús indica com es consulta un usuari.
Actor(s)	Manager de zona
Pre-condicions	L'usuari a iniciat la sessió
Post-condicions	-
Flux normal	• L'usuari accedeix a la opció del widget de cercar persones. • El sistema li mostra una finestra per tal que introdueixi el nom de l'usuari. • El sistema li mostra els resultats d'acord a la informació facilitada. • L'usuari selecciona l'usuari. • En el widget de cercar usuaris, es mostra la informació de l'usuari.
Fluxos alternatius	• Si la informació que introdueix l'usuari en el punt 3 no coincideix amb les dades de cap persona, l'usuari tindrà que tornar a fer la cerca de persones.

3.2.5 CU05 – Consultar botigues

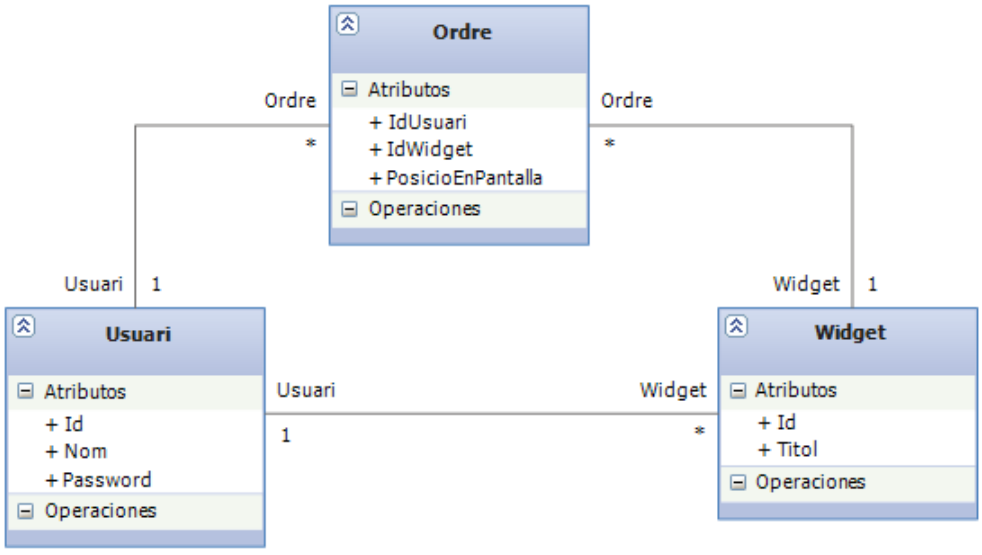
Identificador	CU03
Nom	Consultar botigues
Autor	Raúl Díaz Jerez
Resum	Aquest cas d'ús indica com es consulten les botigues que han realitzat vendes i les que no han fet durant el dia d'avui.
Actor(s)	Manager de zona
Pre-condicions	L'usuari a iniciat la sessió
Post-condicions	
Flux normal	• L'usuari prem sobre l'indicador de botigues que ha realitzat vendes o sobre les que no han realitzat cap venda el dia d'avui. Aquests indicadors estan en el widget de nombre de botigues amb/sense vendes. • El sistema li mostra una finestra amb el llistat botigues.
Fluxos alternatius	-

3.2.6 CU06 – Consultar gràfic

Identificador	CU03
Nom	Consultar gràfic
Autor	Raúl Díaz Jerez
Resum	Aquest cas d'ús indica consulta amb detall el gràfic de vendes mensuals.
Actor(s)	Manager de zona
Pre-condicions	L'usuari ha iniciat la sessió
Post-condicions	-
Flux normal	• L'usuari prem sobre el widget del gràfic de evolució de vendes mensuals. • El sistema li mostra una finestra amb el gràfic amb més detall.
Fluxos alternatius	-

3.3 Entitats del domini

El model del domini representa les entitats identificades en el domini del problema a solucionar, en el nostre cas el domini és molt senzill ja que només es tracta de mostrar a l'usuari certa informació configurada.



Les entitats detectades son:

- Usuari: Representa l'usuari que te configurats una sèrie de widgets.
- Widget: Representa el diferents widgets que veurà l'usuari en el seu panell.
- Ordre: Representa la configuració de widgets de l'usuari.

4. Disseny

4.1 Arquitectura global

L'arquitectura global del projecte està pensada per una aplicació distribuïda orientada als serveis (SOA) i per a l'ús de dispositius mòbils, com és el cas del tablet en el que funcionarà el projecte. El projecte a més està dividit en dos blocs clarament diferenciats. Per una banda està el front-end que seria l'aplicació tablet i per l'altra tenim el back-end de l'aplicació que seria la capa de serveis i base de dades.

Per tal de profunditzar en l'arquitectura, a continuació es detallen un conjunt de vistes del sistema:

- Vista física: En aquesta vista es mostren com estan distribuïts físicament els diferents components del sistema.

- Vista lògica: En aquesta vista es mostren els components conceptuals necessaris per complir les necessitats del projecte.

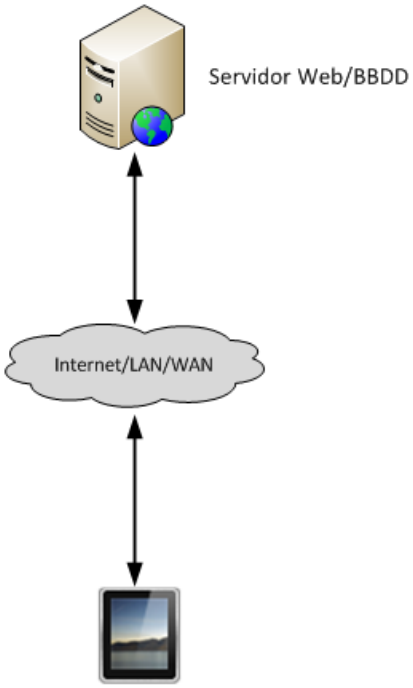
- Vista de components: Aquesta vista intenta il·lustrar la relació entre els components realitzats i la seva relació amb la vista física i lògica.

4.1.1 Vista física

L'estructura que s'utilitzarà en el disseny del sistema és un model molt flexible i amb un alt grau d'escalabilitat. Un model així permet adaptar-nos fàcilment a les necessitats de carrega d'usuaris que poden existir.

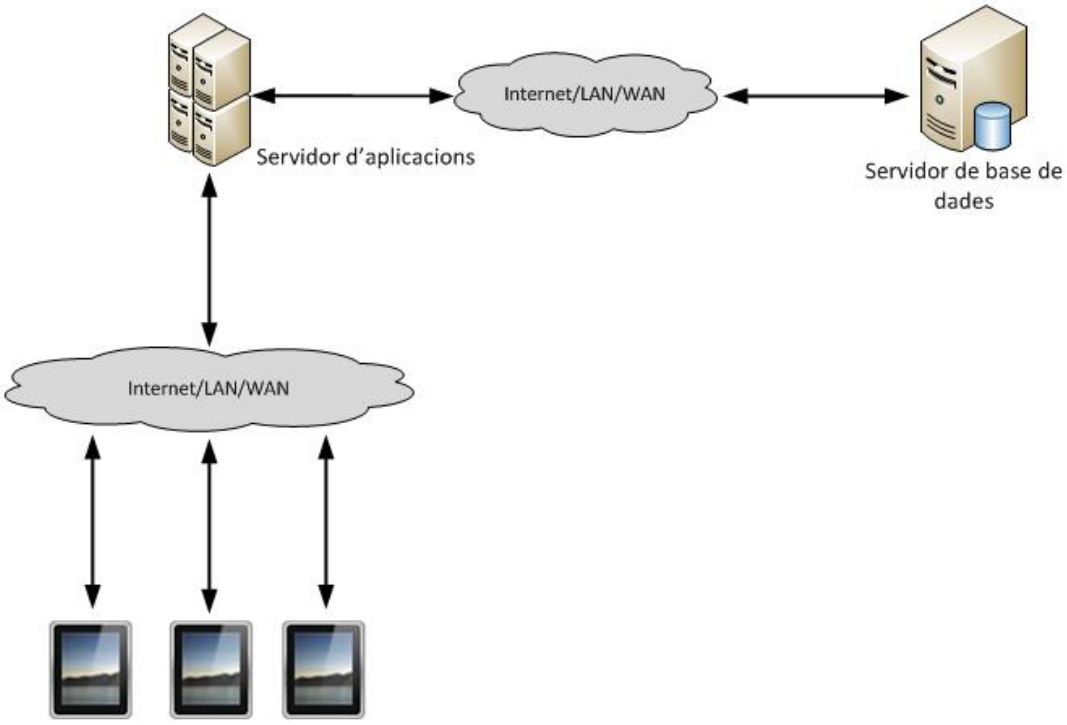
Escenari de partida

En l'escenari inicial del projecte només hi haurà un usuari que utilitzi el panell de control. Això fa que amb un únic servidor on estiguin els serveis i la BBDD en tenim suficient.



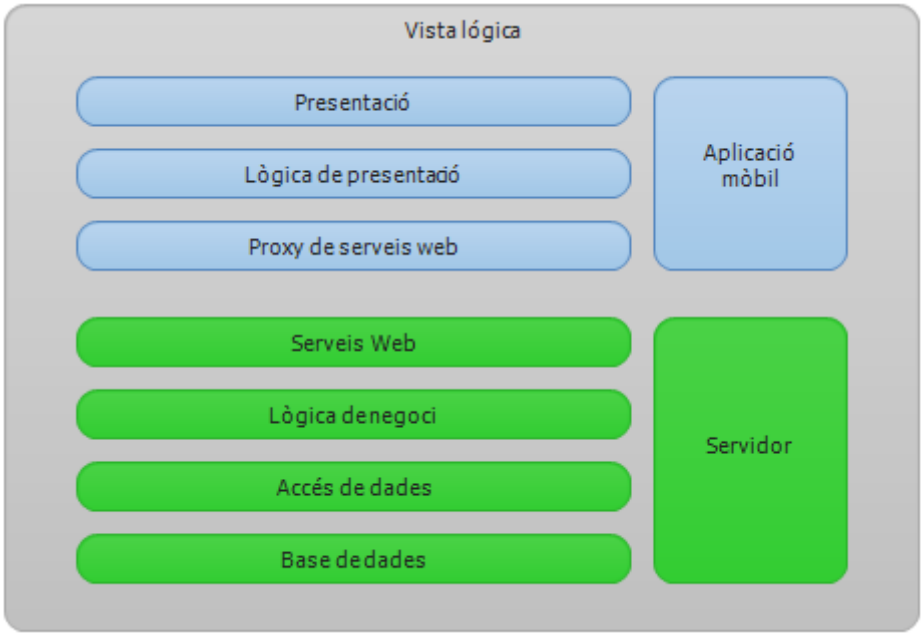
Escenari amb múltiples usuaris utilitzant el panell

En aquest escenari amb possible càrrega d'usuaris es podrien utilitzar n servidors d'aplicacions amb balanceig de càrrega i un servidor dedicat de base de dades.



4.1.2 Vista lògica

A continuació es mostren les capes que formen el projecte i es descriu amb detall en que consisteix cada capa.



Com es pot observar en l'esquema, no totes les capes es troben en el mateix lloc, si no que es troben repartides entre l'aplicació mòbil i el servidor. Les tres primeres capes, que tenen a veure amb la presentació de la informació i amb la comunicació amb el servidor, es troben a l'aplicació mòbil, en canvi els serveis web, components de negoci i base de dades es troben al servidor.

- Presentació: Aquesta capa, és la capa amb la qual l'usuari interactuarà. Aquesta capa mostrarà a l'usuari la informació i capturarà les accions del usuari per comunicar-les a la lògica de presentació.

- Lògica de presentació: És la capa que captura les accions de l'usuari o del sistema per tal de traduir-les a les diferents necessitats de l'aplicació.

- Proxy de serveis web: Son els elements que ens permeten consumir els serveis web.

- Serveis Web: Son els elements que capturen les peticions que es fan des de l'aplicació, consumeixen la lògica de negoci i retornen un resultat.

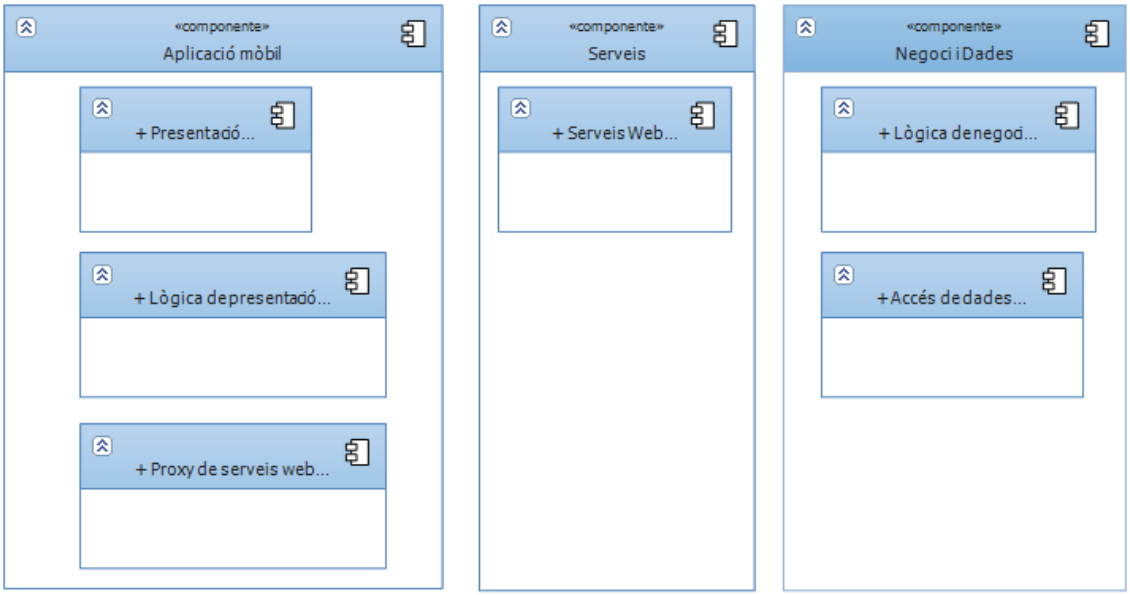
- Lògica de negoci: És on es troba el negoci de cada component, és la capa que interactua amb la base de dades i els serveis.

- Accés de dades: Capa que permet accedir als repositoris d'informació necessaris per l'aplicació.

- Base de dades: Repositori d'informació on es troben les dades materialitzades del sistema.

4.1.3 Vista de components

A continuació veiem un diagrama que resumeix l'aplicació des del punt de vista dels components pels quals està format.



Com podem comprovar el sistema està composat pels següents components:

- Aplicació mòbil: Es tracta del client que s'instal·la en el dispositiu mòbil.
- Serveis: Es tracta del component on es troben els serveis web.
- Negoci i dades: Es troba la lògica de negoci i l'accés de la base de dades.

4.2 Decisions tecnològiques

Aquest es l'apartat on es descriuen i justifiquen totes les decisions tecnològiques preses per al desenvolupament del projecte. Com vam comentar anteriorment l'abast del projecte consisteix en realitzar l'aplicació de tablet amb tecnologia Android i crear un conjunt de serveis en tecnologia .Net. Aquests serveis seran consumits per l'aplicació client. Per tal d'entrar en detall, he dividit aquesta secció en tres blocs de decisions, les decisions preses a l'hora de desenvolupar els components de servidor i base de dades, les decisions a l'hora de realitzat l'aplicació de client i les decisions preses en interoperabilitat de les diferents tecnologies.

4.2.1 Servidor i base de dades

Aquesta part del projecte utilitza tecnologia .NET i constitueix la capa de serveis i base de dades que utilitzarà l'aplicació. A continuació es descriuen els diferents aspectes d'aquest projecte.

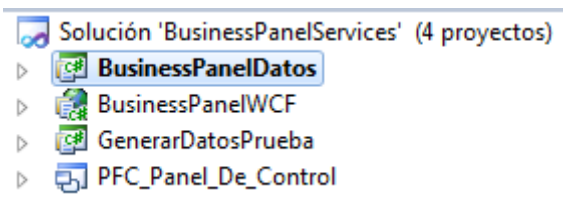
Entorn de desenvolupament

L'entorn de desenvolupament triat per realitzar aquesta part és Visual Studio 2010, aquest entorn és l'oficial per tal de desenvolupar projectes en .Net i ofereix moltes ajudes als programadors per tal de millorar la seva productivitat, a més ofereix integració amb el motor de base de dades SQL Server i permet realitzar consultes en el mateix entorn de desenvolupament. Visual Studio 2010 ha estat facilitat per l'empresa STS, on aquest IDE també és l'entorn amb el que acostumen a treballar.

Estructura del projecte

El projecte es dividirà principalment, en dues capes, la capa de serveis i la capa d'accés a dades, la capa de serveis està formada per un conjunt de serveis WCF que seran consumits per l'aplicació client. La capa d'accés a dades s'encarregarà d'obtenir la informació de la base de dades. No s'ha inclòs la capa de negoci, ja que es tracta de serveis de consultes on les dades ja es troben processats per tal que la resposta sigui ràpida i no es tinguin que aplicar regles de negoci.

L'estructura resultant és la següent:



Com podem veure a més a més de la capa de serveis i dades, existeixen més projectes. A continuació es detalla que és cada projecte.

- "BusinessPanelDatos" → És la capa d'accés a les dades.
- "BusinessPanelWCF" → Capa de serveis.
- "GenerarDatosPrueba" → Petit executable per generar dades per a la demostració del projecte.
- "PFC_Panel_De_Control" → Projecte de modelatge del projecte.

Accés a les dades

Per tal d'accedir a la BBDD s'utilitzarà LINQ to SQL, amb aquesta tecnologia podem accedir a la BBDD i procediments emmagatzemats de manera molt senzilla. Es tracta d'una tecnologia amb la que estic acostumat a treballar amb bons resultats. El projecte estarà composat tant per procediments emmagatzemats com per consultes amb LINQ.

Serveis web

Per tal de dissenyar els serveis web s'utilitzarà Windows Communication Foundation (WCF). WCF és una tecnologia de Microsoft derivada dels serveis Web tradicionals que permet obtenir una millora de rendiment a l'hora de consumir els serveis sempre i quan utilitzis consumidors sota el Framework de Microsoft. Aquesta capa de serveis difereix una mica dels serveis WCF tradicionals, ja que els he configurat com a REST per tal de realitzar la comunicació entre l'aplicació client i l'aplicació servidor. Això s'explicarà amb més detall a l'apartat interoperabilitat.

Llenguatge de programació

El llenguatge que s'ha utilitzat per escriure els diferents components ha estat C# 4.0. S'ha triat aquest llenguatge per diversos motius, per una banda es tracta d'un llenguatge amb el que tinc molta experiència i per altra banda és el llenguatge amb el que acostuma a treballar l'empresa per a la qual estic realitzant el projecte.

SGBD

El SGBD que s'ha utilitzat és el SQL Server 2008 R2, s'ha triat aquest servidor perquè proporciona una bona relació qualitat – preu a part de la integració amb la tecnologia de Microsoft que permet el fàcil accés. De igual manera que el llenguatge de programació, l'empresa també utilitza aquests servidors en els seus projectes i jo estic acostumant a utilitzar aquest servidor de BBDD.

Servidor d'aplicacions

El servidor d'aplicacions que s'utilitzarà serà el Internet Information Services 7 (IIS7) que és el servidor d'aplicacions de Microsoft. En aquest servidor estaran els serveis WCF. S'ha utilitzat aquest servidor degut a la tecnologia que utilitza el back-end del projecte, ja que els serveis estan construïts mitjançant la plataforma .Net i s'ha d'allotjar en un servidor IIS.

4.2.2 Aplicació client

Entorn de desenvolupament

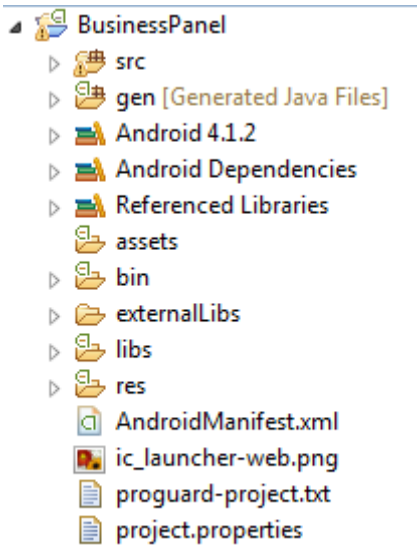
L'entorn de desenvolupament utilitzat per realitzar l´aplicació client ha estat Eclipse Indigo, que és l'entorn recomanat per Google per realitzar solucions Android. Aquest entorn de desenvolupament és gratuït i permet la fàcil integració amb la plataforma Android.

Llenguatge de programació

El llenguatge utilitzat per desenvolupar l'aplicació client en Android ha estat Java amb la versió del JDK 1.6.0.37. Aquest llenguatge de programació és l'estàndard per realitzar aplicacions Android.

Estructura del projecte

Tot projecte Android te una estructura definida per defecte. Aquesta estructura està molt acoblada a la tecnologia i no s'ha de modificar si es vol que el projecte funcioni correctament. La estructura del projecte serà la següent:



D'aquesta estructura de projecte, em centraré en la carpeta "src" i en la carpeta "res" ja que són les carpetes principals que s'utilitzaran en el desenvolupament del projecte. En el sistema Android cada "pantalla" s'anomena activitat, aquesta activitat està formada per una classe JAVA que es troba a la carpeta "src" i una plantilla o "layout" que es troba a la carpeta "res". Respecte a la carpeta "src" l'he dividit en quatre paquets.

- "com.sts.businesspanel" → Conté les classes de les activitats.
- "com.sts.businesspanel.arquitectura.rest" → En aquest paquet es troba la API dissenyada per comunicar-nos amb els serveis.
- "com.sts.businesspanel.customcontrols" → En aquest paquet es troben tots els controls personalitzats que faré, per exemple els quatre widgets es trobaran aquí.
- "com.sts.businesspanel.entities" → Les entitats que utilitzaré durant el desenvolupament del projecte.

Android

Per tal de realitzar l'aplicació mòbil i la capa de presentació del projecte, s'ha triat la tecnologia Android de Google. S'ha triat aquesta tecnologia per diversos motius:

- Es una tecnologia amb molt de temps en el mercat i es tracta d'una tecnologia molt evolucionada.
- Es volia fer una aplicació per a tablets i en el mercat hi ha dos tipus principals de dispositius: els d'Android i els d'Apple. En aquest cas el dispositiu Android és més obert i es pot programar des de qualsevol ordinador. Apple és un sistema molt tancat i Microsoft ara comença en el món del tablet amb els primers productes i es prefereix utilitzar una tecnologia més provada com Android que no pas la primera versió de tablets Windows 8.
- Es una tecnologia molt oberta i amb un gran suport de la comunitat per tant es troba molt documentada.
- Els sistemes Android estan molt estesos a nivell mundial i per tant el nivell d'usuaris potencials a utilitzar el teu sistema és molt elevat.
- Amb el projecte, també es vol mostrar la resolució d'un projecte amb arquitectura SOA i la interoperabilitat de dues tecnologies diferents com Android i .NET.

- A nivell personal volia aprendre Android i Java perquè trobo que son dues tecnologies molt interesants.

4.3 Interoperabilitat

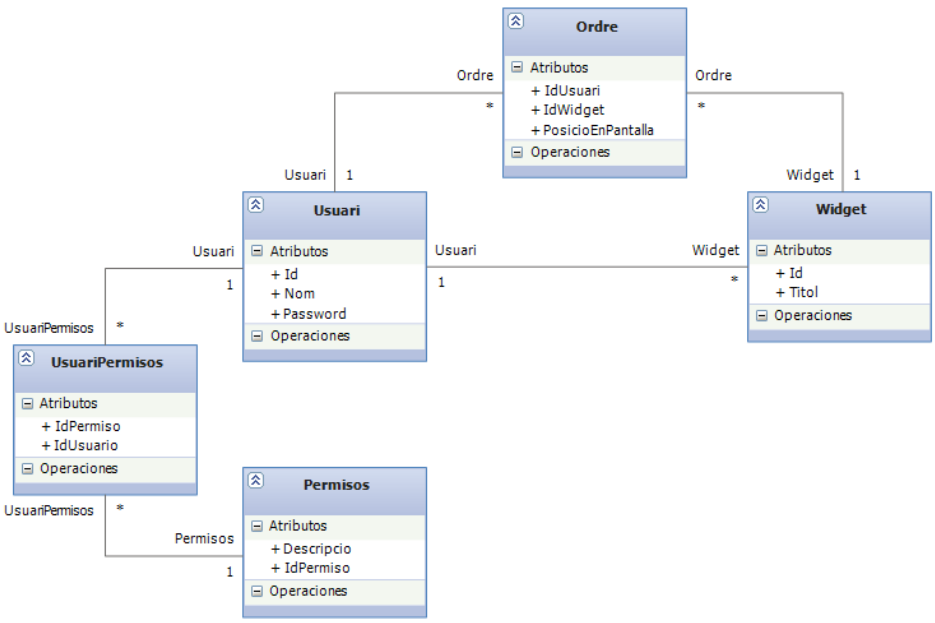
Una part important del projecte és la interoperabilitat entre tecnologies, com s'ha comentat prèviament aquest projecte està dividit en dues parts, l'aplicació de tablet i la capa de serveis i dades. Aquestes dues parts utilitzen tecnologies molt diferents, l'aplicació tablet utilitza tecnologia Android i la part de serveis tecnologia .Net. A més a més, el client al tractar-se d'una aplicació mòbil hem de tenir cura de la quantitat de dades que s'envien ja que normalment es tenen tarifes de dades contractades amb límits de transferència a màxima velocitat.

La elecció triada ha estat la tecnologia REST, aquesta tecnologia te la avantatge de ser molt eficient a l'hora de transmetre informació, també te la peculiaritat de que és fàcil de llegir la informació ja que s'accedeix per URI. Finalment és fàcil de desenvolupar ja que utilitza les operacions del protocol HTTP per comunicar-se (POST, GET, PUT i DELETE) d'aquesta manera no necessiten eines per comunicar-se ni fer acoblaments de cap tipus entre els projectes.

D'aquesta manera podem comunicar-nos de manera senzilla i òptima entre dues tecnologies totalment diferents.

4.4 Diagrama estàtic de disseny

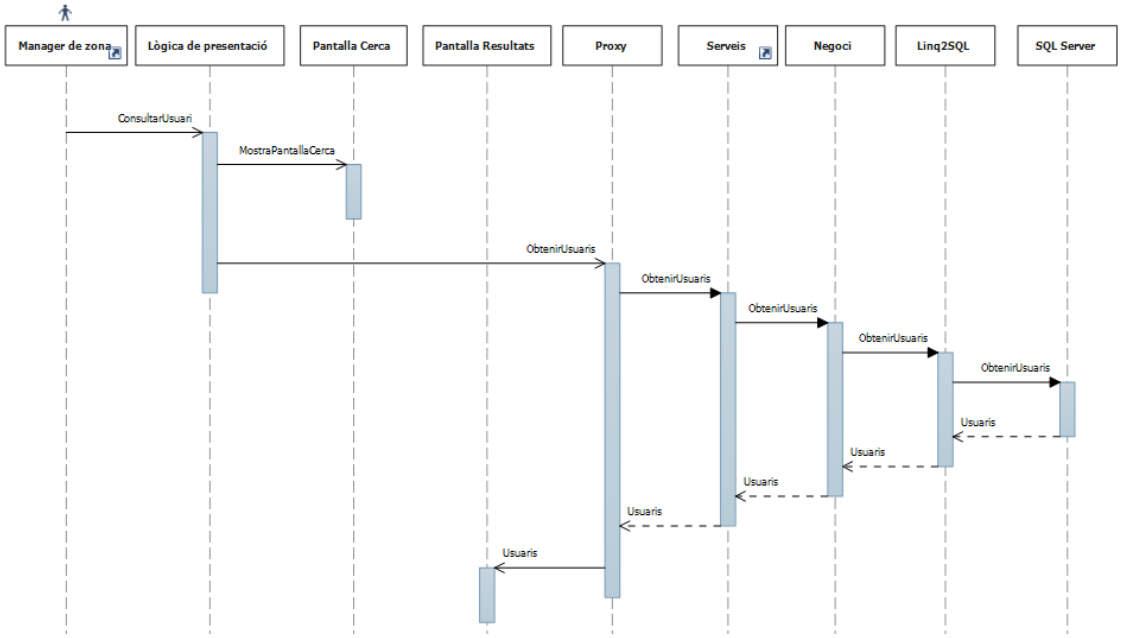
A continuació el diagrama estàtic del sistema que recull les entitats utilitzades pel sistema:



A part de les entitats del model de domini, s'han afegit les necessàries per representar els usuaris del sistema.

4.5 Diagrama de seqüencia

En el diagrama de seqüencia es mostra la interacció entre els diferents objectes del sistema, com que en tots els casos la interacció és molt similar, em centraré en el cas mes complet del sistema per tal d'il·lustrar la comunicació. A continuació es mostra el diagrama de seqüencia del CU03 Consultar Usuaris.



La seqüencia que mostra el diagrama es la següent:

1. El manager de zona comunica a la lògica de presentació que vol consultar els usuaris.
2. La lògica de presentació mostra la pantalla de cerca. L'usuari introdueix el nom del usuari a cercar.
3. El Proxy es comunica amb el mètode del servei que conte la informació.
4. El servei captura el missatge i li demana a la lògica de negoci els usuaris.
5. La lògica de negoci li demana la informació d'usuaris a la capa d'accés a dades.
6. La capa d'accés a dades obté els usuaris del repositori.
7. Es retorna la informació fins la pantalla de resultats.

4.6 Disseny de la persistència

S'ha dissenyat una base de dades tant per emmagatzemar la informació de configuració de l'usuari com per emmagatzemar la informació de les botigues o ventes. En aquest apartat es diferenciarà entre aquests dos conceptes encara que comparteixin la mateixa base de dades.

4.6.1 Model relacional de la base de dades

Model de la configuració del panell de usuari:

C_Aplicactiones(**Id**, Descripcion, Estado)

C_Permisos(**Id**, IdAplicacion, Descripcion, Estado)
- IdAplicacion és una clau foranea cap a C_Aplicaciones

C_TipoWidget(**Id**, Descripcion, Estado)

D_UsuarioConfiguracion(**IdUsuario, IdWidget**, **IdAplicacion**, Orden)
- IdUsuari és una clau foranea cap a D_Usuarios
- IdWidget és una clau foranea cap a D_Widget
- IdAplicacion és una clau foranea cap a C_Aplicaciones

D_Usuarios(**Id**, Login, Password, Nombre, Apellido1, Apellido2, DataAlta, Empresa, DataUpdate)

D_Widget(**Id**, IdAplicacion, Tipo, NombreControl, FechaInicio, FechaFin, Titulo, Descripcion)
- IdAplicacion és una clau foranea cap a C_Aplicaciones

Model de botigues i ventes

D_Empleados(**Id,** Nombre, Apellido1, Apellido2, Email, Movil, Encargado)

D_EmpleadosTienda(**Id**, IdEmpleado, IdTienda)
- IdEmpleado és una clau foranea cap a D_Empleados
- IdTienda és una clau foranea cap a C_Tiendas

C_Tiendas(**Id,** Descripcion)

D_RegistroEmpleados(**IdEmpleado, IdTienda, FechaRegistro**)
- IdEmpleado és una clau foranea cap a D_Empleados
- IdTienda és una clau foranea cap a C_Tiendas

D_TiendaResponsable(**IdTienda, IdResponsable**)

- IdResponsable és una clau foranea cap a D_Empleados
- IdTienda és una clau foranea cap a C_Tiendas

D_Tienda_Ventas(**Id, IdTienda, Fecha**, TotalVentas)

- IdTienda és una clau foranea cap a C_Tiendas

4.6.2 Diagrama de la base de dades

Diagrama de la configuració del panell de usuari

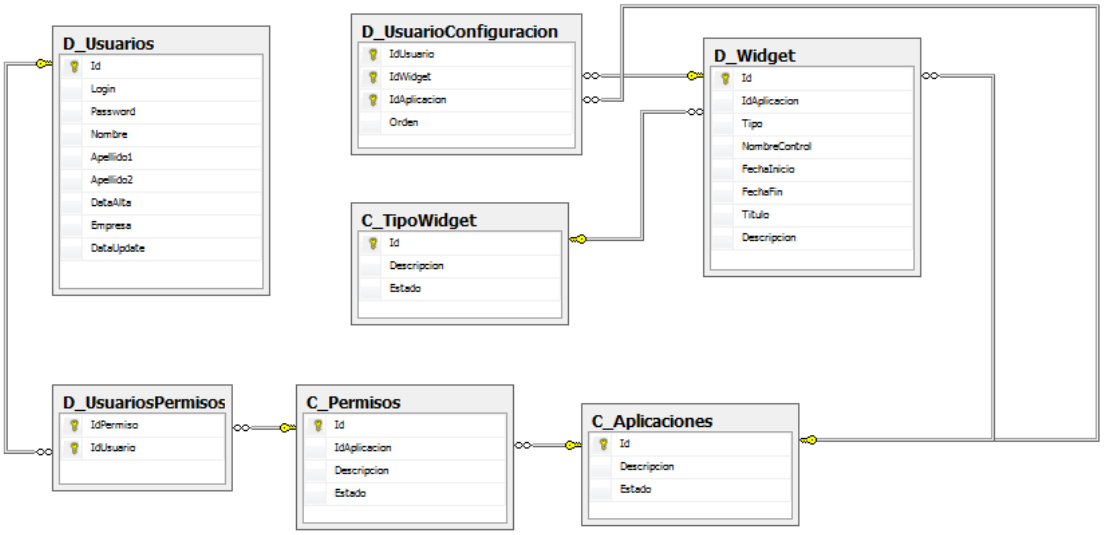
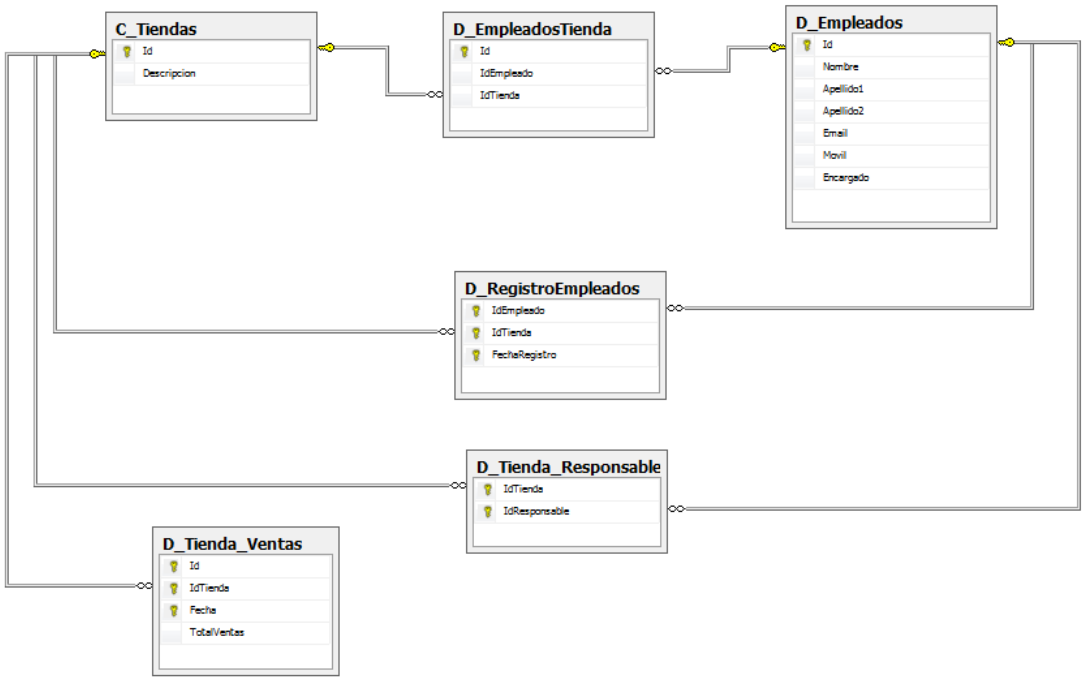


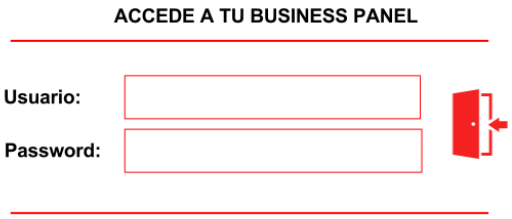
Diagrama de botigues i vendes



4.7 Prototipatge de la interfície d'usuari

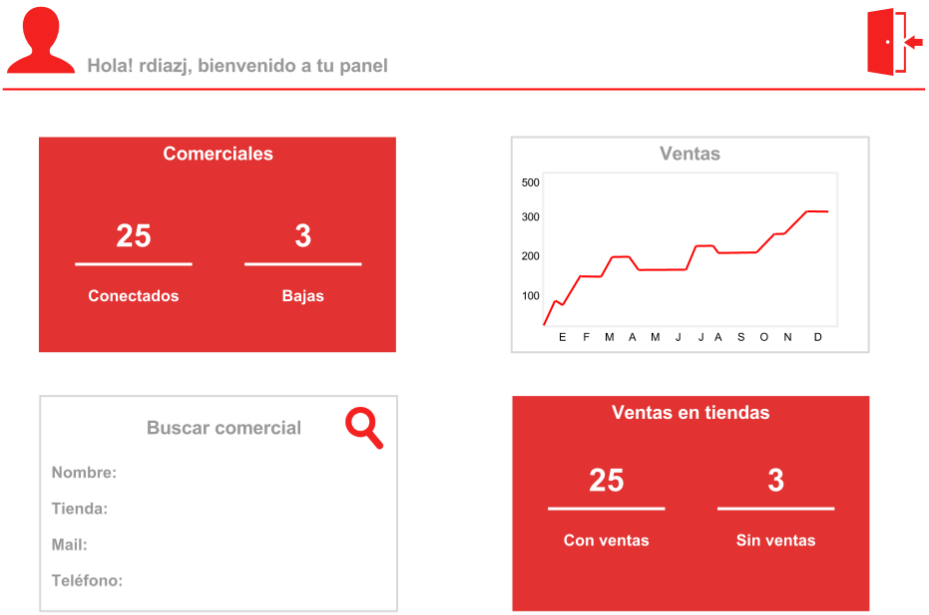
En la etapa de disseny s'ha desenvolupat el prototipus de l'aplicació on es recullen conceptualment les principals pantalles que utilitzarà l'usuari i com es farà la navegació dins de l'aplicació. A continuació es mostren les principals pantalles i es fa una breu descripció de cadascuna d'elles.

4.7.1 Pantalla de login



Aquest és el disseny de la pantalla de login, on l'usuari indicarà el seu nom i contrasenya i accedirà a l'aplicació.

4.7.2 Pantalla principal del Business Panel



Aquesta és la pantalla principal, on es veu el business panel i la informació s'actualitza en temps real. L'usuari pot interactuar amb els diferents widgets, tal com es reflexa en les següents captures de pantalla.

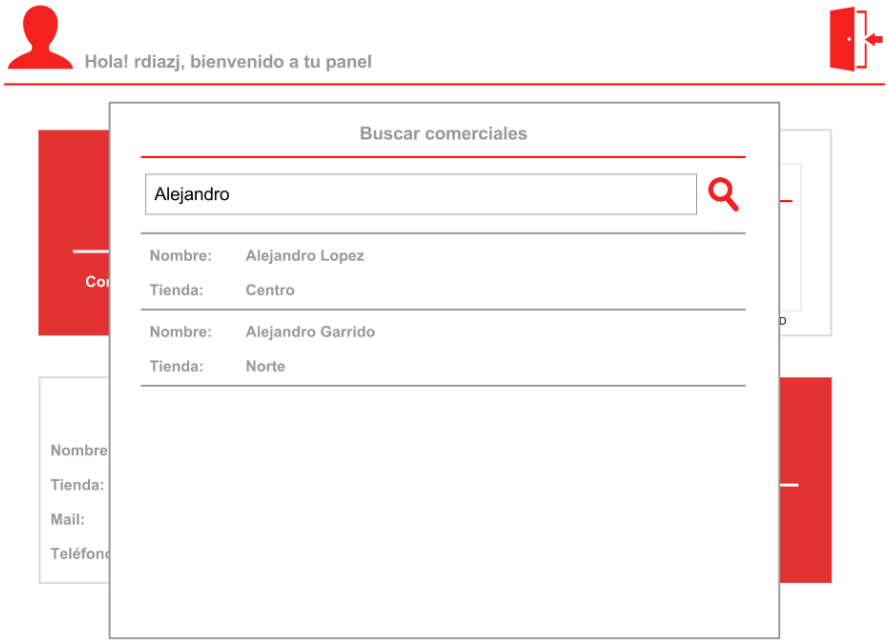
4.7.3 Detall dels comercials

Aquesta és la pantalla de detall del primer widget "Comercials" i s'accedeix prement en comercials connectats o comercials de baixa. Es mostra una pantalla amb la informació de contacte dels comercials.

4.7.4 Detall botigues

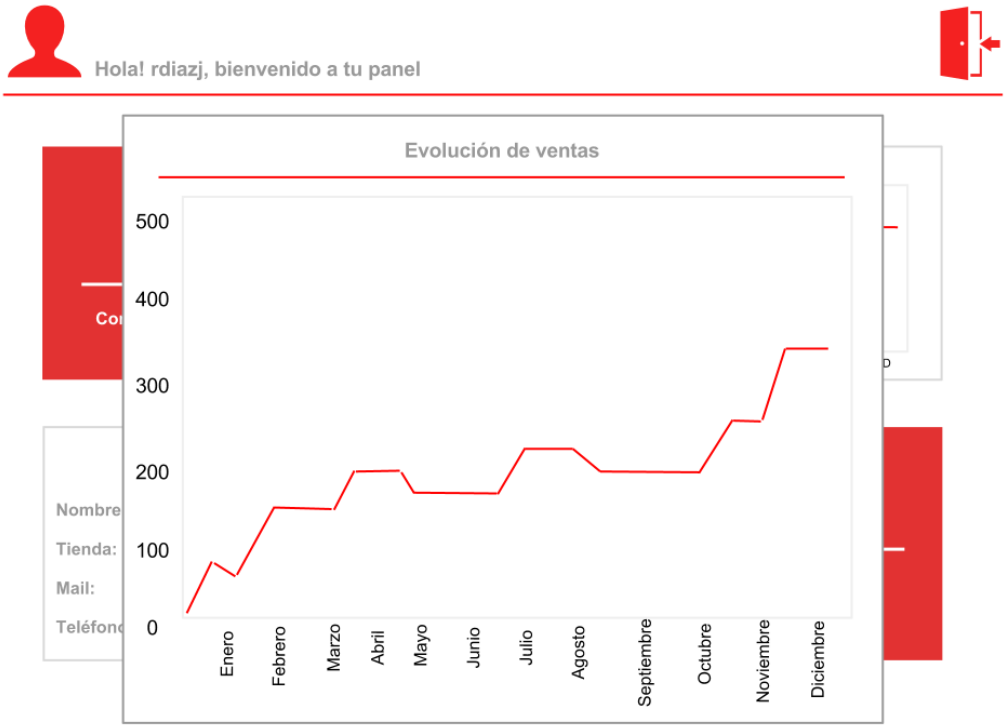
Pantalla de detall de les vendes de les botigues. S'accedeix al prémer sobre botigues amb vendes o sense vendes i visualitza la informació de contacte de les botigues.

4.7.5 Cerca de comercials



Pantalla per cercar Comerciales. Prement en el widget de cercar comercial, s'obrirà aquesta finestra perquè es puguin cercar els comercials.

4.7.6 Detall evolució de vendes



Aquesta pantalla mostra en detall el gràfic de punts de vendes mensuals. S'accedeix prement sobre el gràfic del widget vendes.

5. Implementació

En aquesta secció es descriu en detall el resultat de la implementació del projecte. En aquest projecte s'han fet dues implementacions una per a la part dels serveis i una altre que és l'aplicació client. En aquesta secció es veurà detallada cadascuna de les implementacions en dos punts clarament diferenciats: la capa de serveis i dades i l'aplicació per a tablet.

5.1 Capa de serveis i dades

5.1.1 Serveis

Tal i com he comentat anteriorment, la comunicació entre l'aplicació client i els serveis es farà mitjançant REST. Els serveis WCF, per defecte, no estan preparats per utilitzar aquest tipus de tecnologia, s'ha de fer una petita configuració per tal de poder utilitzar-los com a REST.

Els passos que he fet per configurar els serveis WCF com a serveis REST son els següents:

1. Dels mètodes que tenim disponibles definim quins d'ells seran GET i quins POST, PUT o DELETE d'acord a les operacions que accepta REST. Als mètodes que siguin GET s'ha d'afegir l'atribut *"WebGet"* i els que siguin POST, PUT o DELETE s'ha d'afegir el atribut *"WebInvoke"* i especificar el mètode.

Exemple per configurar un mètode com a GET.

```
[OperationContract]
[WebGet(UriTemplate = "ObtenerDatosUsuario?u={u}&p={p}", BodyStyle =
WebMessageBodyStyle.Bare, ResponseFormat = WebMessageFormat.Json)]
DatosUsuario ObtenerDatosUsuario(string u, string p);

[OperationContract]
[WebInvoke(UriTemplate = "GuardarConfiguracion", Method = "POST")]
void GuardarConfiguracion(ConfigurationEntity conf);
```

En aquest cas, el mètode te dos atributs, *"OperationContract"* que es crea per defecte amb els WCF i defineix el mètode com una operació que es pot consumir. L'altre atribut *"WebGet"* és el que defineix el mètode com un mètode per ser utilitzat mitjançant una petició GET. Com veiem l'atribut te les següents peculiaritats:

- *UriTemplate* → Ens permet definir una URI que identifica el mètode. En els serveis he decidit que la URI sigui el mateix nom que el mètode i que els noms del paràmetres també es mantinguin.
- *BodyStyle* → Permet configurar el format de la resposta. En aquest cas, *"WebMessageBodyStyle.Bare"* fa que només rebin les dades de la resposta i no es rebin les dades d'informació sobre el mètode.
- *ResponseFormat* → Permet dir-li al servei com ha de serialitzar la resposta. En el projecte únicament es treballa en format JSON, perquè d'aquesta manera els missatges que s'envien son el més petit possible i perquè existeixen objectes per poder manipular fàcilment aquest tipus de format.

Exemple configurar un mètode com a POST

```
[OperationContract]
[WebInvoke(UriTemplate = "GuardarConfiguracion", Method = "POST")]
void GuardarConfiguracion(ConfigurationEntity conf);
```

En aquest cas a diferencia del anterior s'utilitza l'atribut *"WebInvoke"* i de la mateixa manera es defineix l'URI del mètode i es defineix quin mètode accepta. En aquest cas accepta "POST".

2. Es modifica el fitxer de configuració dels serveis.

Es modifica el "System.servicemodel" per tal d'afegir un "end-point HTTP". Amb el tag "standardEndpoint" podrem obtenir l'ajuda dels serveis. Això ens és de molta utilitat per a les proves com veurem més endavant.

```
<system.serviceModel>
    <serviceHostingEnvironment aspNetCompatibilityEnabled="true">
    </serviceHostingEnvironment>
    <standardEndpoints>
      <webHttpEndpoint>
        <standardEndpoint name="" helpEnabled="true"
automaticFormatSelectionEnabled="true"></standardEndpoint>
      </webHttpEndpoint>
    </standardEndpoints>
  </system.serviceModel>
```

3. Modificar el fitxer global.asax.

Modifiquem aquest fitxer perquè quan l'aplicació de serveis s'inicia, es redirigeixin les peticions al meu servei.

```
protected void Application_Start(object sender, EventArgs e)
{
  RouteTable.Routes.Add(new ServiceRoute("BusinessPanelService", new
WebServiceHostFactory(), typeof(BusinessPanelService)));
}
```

A continuació si ens dirigim a la següent URL podem veure tots els serveis que he fet i els mètodes per consumir-los.

URL → http://217.76.135.234:81/BusinessPanelService/help

Operations at http://217.76.135.234:81/BusinessPanelService

This page describes the service operations at this endpoint.

Uri	Method	Description
BuscarComercial	GET	Service at http://217.76.135.234:81/BusinessPanelService/BuscarComercial?u={U}&p={P}&pattern={PATTERN}
GuardarConfiguracion	POST	Service at http://217.76.135.234:81/BusinessPanelService/GuardarConfiguracion
ObtenerComercialesBaja	GET	Service at http://217.76.135.234:81/BusinessPanelService/ObtenerComercialesBaja?u={U}&p={P}
ObtenerComercialesConectados	GET	Service at http://217.76.135.234:81/BusinessPanelService/ObtenerComercialesConectados?u={U}&p={P}
ObtenerConfiguracionWidgets	GET	Service at http://217.76.135.234:81/BusinessPanelService/ObtenerConfiguracionWidgets?u={U}&p={P}
ObtenerDatosUsuario	GET	Service at http://217.76.135.234:81/BusinessPanelService/ObtenerDatosUsuario?u={U}&p={P}
ObtenerDisponibilidadComerciales	GET	Service at http://217.76.135.234:81/BusinessPanelService/ObtenerDisponibilidadComerciales?u={U}&p={P}
ObtenerTiendasConVentas	GET	Service at http://217.76.135.234:81/BusinessPanelService/ObtenerTiendasConVentas?u={U}&p={P}
ObtenerTiendasSinVentas	GET	Service at http://217.76.135.234:81/BusinessPanelService/ObtenerTiendasSinVentas?u={U}&p={P}
ObtenerVentasMes	GET	Service at http://217.76.135.234:81/BusinessPanelService/ObtenerVentasMes?u={U}&p={P}
ObtenerVentasTotalesTiendas	GET	Service at http://217.76.135.234:81/BusinessPanelService/ObtenerVentasTotalesTiendas?u={U}&p={P}

Si fem clic al camp *"Method"* d'un dels mètodes anem a la pàgina d'informació on tenim ajuda per saber com podem cridar-los.

Exemple del mètode *"ObtenerDatosUsuario"*:

http://217.76.135.234:81/BusinessPanelService/help/operations/ObtenerDatosUsuario

Reference for http://217.76.135.234:81/BusinessPanelService/ObtenerDatosUsuario?u={U}&p={P}

Url: http://217.76.135.234:81/BusinessPanelService/ObtenerDatosUsuario?u={U}&p={P}

HTTP Method: GET

Message direction	Format	Body
Request	N/A	The Request body is empty.
Response	Xml	Example,Schema
Response	Json	Example

The following is an example response Xml body:

```
<DatosUsuario xmlns="http://schemas.datacontract.org/2004/07/BusinessPanelDatos.Entities">
  <Apellido1>String content</Apellido1>
  <Apellido2>String content</Apellido2>
  <Empresa>String content</Empresa>
  <Id>2147483647</Id>
  <Nombre>String content</Nombre>
</DatosUsuario>
```

The following is an example response Json body:

```
{
        "Apellido1":"String content",
        "Apellido2":"String content",
        "Empresa":"String content",
        "Id":2147483647,
        "Nombre":"String content"
}
```

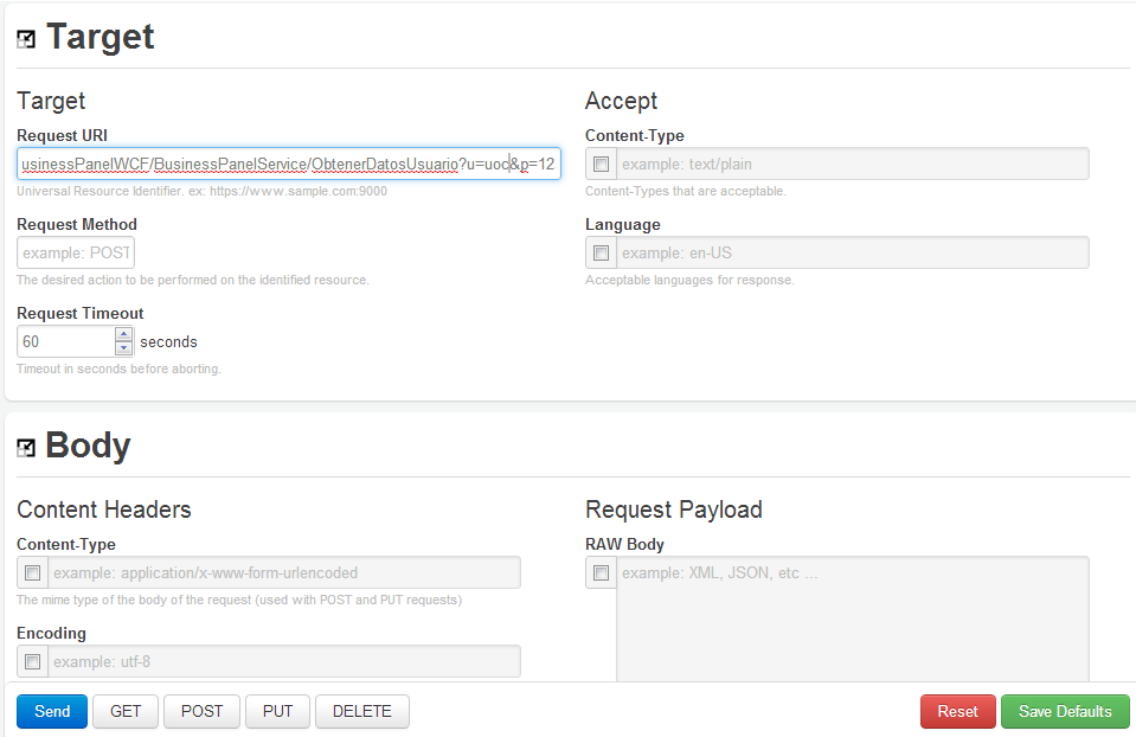
The following is the response Xml Schema:

```
<xs:schema xmlns:tns="http://schemas.datacontract.org/2004/07/BusinessPanelDatos.Entities" elementFormDefault="qualified"
targetNamespace="http://schemas.datacontract.org/2004/07/BusinessPanelDatos.Entities" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="DatosUsuario">
    <xs:sequence>
      <xs:element minOccurs="0" name="Apellido1" nillable="true" type="xs:string" />
      <xs:element minOccurs="0" name="Apellido2" nillable="true" type="xs:string" />
```

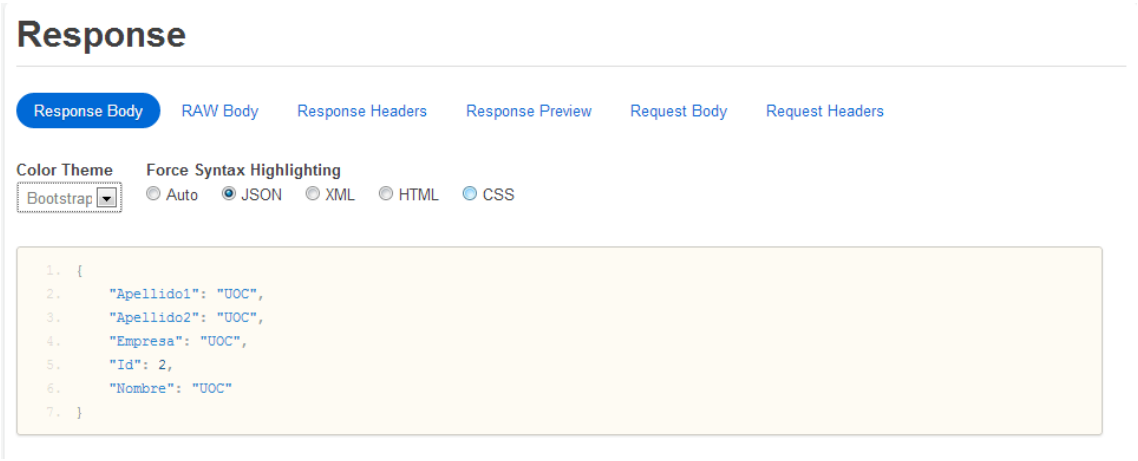
Per poder provar-los fàcilment he utilitzat un complement anomenat *"REST Console"* que es troba disponible a la botiga d'aplicacions del navegador Google Chrome.

https://chrome.google.com/webstore/detail/rest-console/cokgbflfommojglbmbpenpphppikmonn?utm_source=chrome-ntp-icon

En la consola rest posem el mètode amb els paràmetres i fem la petició GET.



A continuació la resposta:



5.1.2 Entitats

S'han desenvolupat entitats de domini per tal de representar la informació que viatja entre l'aplicació client i el servidor. Aquestes entitats s'han de serialitzar quan s'envien del servidor a l'aplicació client. Per realitzar la serialització, s'han d'afegir els atributs *"DataContract"* a la classe i *"DataMember"* als atributs de la classe.

A continuació podem veure un exemple de classe serialitzable que s'utilitza:

```
[DataContract]
public class DatosComercial
{
    [DataMember]
    public string Nombre { get; set; }

    [DataMember]
    public string Tienda { get; set; }

    [DataMember]
    public string Mail { get; set; }

    [DataMember]
    public string Telefono { get; set; }
}
```

Aquesta entitat s'utilitza per emmagatzemar les dades de contacte d'un comercial, perquè el dispositiu pugui obtenir la informació és necessari que contingui els atributs *"DataContract"* i *"DataMember"* per tal que es serialitzin correctament.

A part de les entitats de dades, també existeixen un conjunt de enumerats per tal de representar dades mestres a l'hora de realitzar consultes LINQ.

A continuació podem veure un enumerat que recull els permisos existents en l'aplicació.

```
public enum Permisos
{
   AccesoPFC = 1
}
```

5.1.3 Seguretat

Al tractar-se d'un producte pilot, s'ha utilitzat un sistema de seguretat bàsic, que utilitza les dades d'identificació de l'usuari per validar-lo. A cada petició que es fa des de l'aplicació client, s'envia l'usuari i la contrasenya, els serveis verifiquen aquestes dades amb la taula *"D_Usuarios"*, en cas que l'usuari sigui vàlid es guarda en la sessió del servidor d'aplicacions, de manera que la següent vegada no ha d'accedir a la base de dades, si que es verifica a la sessió que el usuari és vàlid.

A continuació es mostra un exemple del funcionament:

```
public DatosUsuario ObtenerDatosUsuario(string u, string p)
{
    if (Seguridad.Autenticar(u, p))
    {
        _repostory = new BusinessPanelRepostory();
        return _repostory.ObtenerDatosUsuario(u, p);
    }

    return null;
```

```
}
```

```
public class Seguridad
{
     public static bool Autenticar(string usuario, string password)
     {
         if (HttpContext.Current.Session["Autenticado"] == null)
         {
             var repository = new BusinessPanelRepostory();
             if(repository.ValidarUsuario(usuario, password))
             {
                 HttpContext.Current.Session["Autenticado"] = true;
                 return true;
             }
             return false;
         }
         return (bool)(HttpContext.Current.Session["Autenticado"]);
     }
}
```

En cada servei trobem la mateixa estructura, es crida a la funció *"Autenticar"* aquesta funció verifica si aquest usuari ja ha estat prèviament autenticat, en cas que no ho estigui, es valida amb el repositori i en cas que siguin correctes, es guarda en sessió que l'usuari es vàlid per posteriors peticions.

Com es pot veure es tracta d'un sistema de validació senzill que compleix les necessitats actuals del projecte.

5.1.4 Persistència

Per a la capa de persistència de dades s'ha utilitzat la tecnologia LINQ per la facilitat i rapidesa a l'hora de realitzar consultes. Per tal d'utilitzar aquesta tecnologia s'ha tingut que crear un fitxer de classes LINQ que proporciona el mitja per comunicar-nos amb el repositori de dades. En aquest fitxer tenim que posar totes les taules i procediments emmagatzemats que utilitzem per tal de poder accedir des del codi. Aquest fitxer un cop configurat genera una classe auto-generada *"DataContext"* que és la classe amb la que interactuarem per accedir a les dades.

LINQ ens permet accedir i manipular de diverses maneres el repositori. Durant la implementació he utilitzat consultes LINQ i procediments emmagatzemats d'acord a un criteri de claredat en la consulta, es a dir, LINQ encara que és molt potent te una sintaxi una mica complicada i no és tan senzilla com la de SQL tradicional. Per llegibilitat de les consultes LINQ quan es complicaven en excés les he implementat com a procediment emmagatzemat. A continuació poso un exemple de LINQ per accedir a un procediment emmagatzemat i un altre exemple de realitzar la consulta directament.

Codi LINQ per accedir a un procediment emmagatzemat:

```
public DatosComercial[] ObtenerComercialesConectados()
{
   return context.D_Obtener_Comerciales_Conectados(DateTime.Now)
          .Select(x => new DatosComercial
          {
            Mail = x.Email,
            Nombre = String.Format("{0} {1} {2}",
                           x.Nombre, x.Apellido1, x.Apellido2),
            Telefono = x.Movil.HasValue ?
x.Movil.Value.ToString(CultureInfo.InvariantCulture)
: String.Empty,
            Tienda = x.Descripcion
           }).ToArray();
}
```

En aquest cas, cridem al procediment emmagatzemat *"D_Obtener_Comerciales_Conectados".* Com a paràmetre d'entrada del procediment tenim la data actual. El resultat del procediment emmagatzemat el projectem a un *"array"* de *"DatosComercial"* que és l'entitat de sortida.

Codi de consulta LINQ:

```
public bool ValidarUsuario(string u, string p)
{
   var query = (from usuario in context.D_Usuarios
                join pu in context.D_UsuariosPermisos on usuario.Id
                     equals pu.IdUsuario
                join permiso in context.C_Permisos on pu.IdPermiso
                     equals permiso.Id
                where usuario.Login.ToLower() == u.ToLower() &&
                      usuario.Password.ToLower() == p.ToLower() &&
                      pu.IdPermiso == (int)Permisos.AccesoPFC &&
                      permiso.IdAplicacion == (int)Aplicaciones.PFC
                select usuario);
            return query.Any();
}
```

En aquest cas executem una consulta directament sobre el nostre model. Podem accedir a aquest model gracies al objecte *"context"*. La sintaxi es semblant al SQL tradicional, encara que en la consulta LINQ es construeix seguint el patró "from.... in...where...select" i una consulta SQL el patró és "select....from....where". Com veiem es força similar.

5.1.5 Creació de dades de prova

Amb previsió de la demostració del projecte, s'ha creat una aplicació de consola de Windows per inserir dades de prova i veure el refresc automàtic de les dades en l'aplicació client. En la classe del repositori, s'ha creat un mètode anomenat

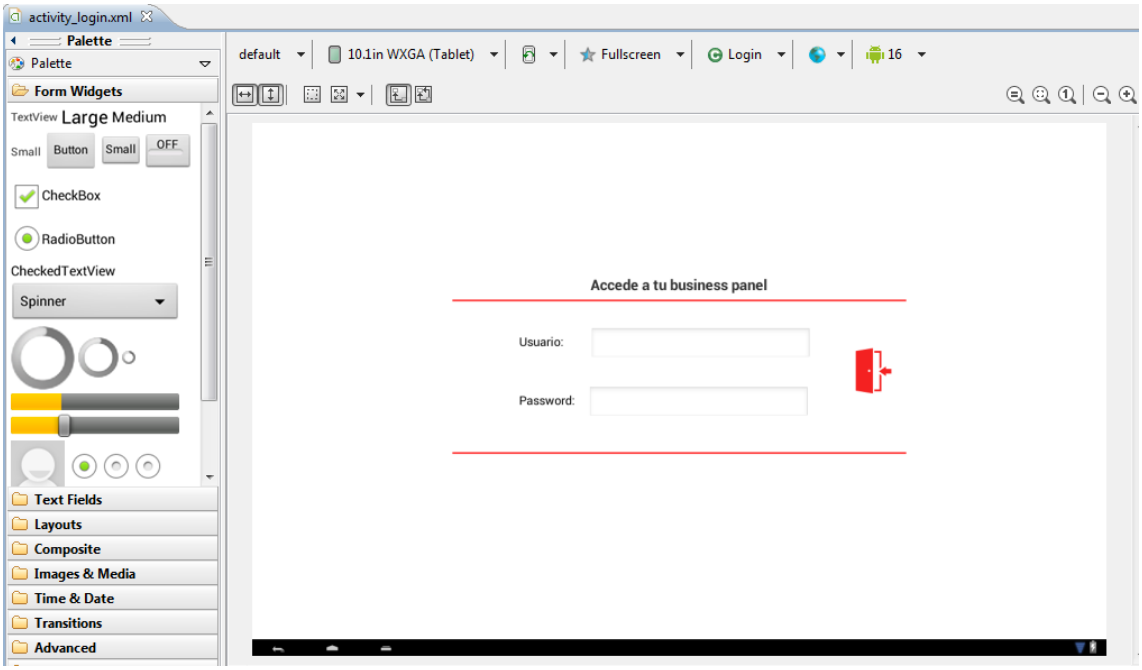
"CargarDatosDemo" que insereix dades de prova. Per inserir les dades, només s'ha d'executar el *"GenerarDatosPrueba.exe"* que es troba en el directori bin del projecte de serveis.

5.2 Aplicació per a tablet

5.2.1 Interfície

Aquest projecte es un projecte principalment de visualització de dades per tal de prendre decisions del negoci. Per tant durant el desenvolupament de tota la interfície d'usuari he buscat com objectiu la claredat a l'hora de representar la informació. Un altre objectiu era que tota l'aplicació es pogués utilitzar amb un dit ja que es volia la màxima senzillesa possible.

Per tal de dissenyar les interfícies he utilitzat l'editor natiu d'Android.



Aquest editor bàsicament genera per darrere un XML amb la interfície d'usuari.

activity_login.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:descendantFocusability="beforeDescendants"
    android:focusableInTouchMode="true"
    >

    <TextView
        android:id="@+id/tvTituloLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="222dp"
        android:padding="@dimen/padding_medium"
        android:text="@string/titulo_login"
        android:textSize="22dp"
        android:textStyle="bold"
        tools:context=".Login" />

    <View
        android:id="@+id/vwLineaRojaInicio"
        android:layout_width="fill_parent"
        android:layout_height="3dp"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/tvTituloLogin"
        android:layout_marginLeft="300dp"
        android:layout_marginRight="300dp"
        android:background="@android:color/holo_red_light" />

    <TextView
        android:id="@+id/tvUsuario"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/vwLineaRojaInicio"
        android:layout_marginRight="40dp"
```

L'editor d'Android està força bé i ajuda a plantejar amb facilitat les pantalles, no obstant, en totes les pantalles he tingut que anar als fitxer XML que genera per acabar de quadrar certs aspectes de disseny com la posició dels controls, els estils, etc...

5.2.2 Entitats

Les entitats que he creat pel desenvolupament de l'aplicació son entitats estàndard en JAVA, aquestes entitats les he utilitzat per emmagatzemar informació. De les entitats vull destacar que aquelles que passen d'una activitat a una altra activitat han de ser serialitzables per tal que la activitat de destí pugui agafar l'entitat. Més endavant ho veurem amb detall.

5.2.3 Widgets i controls personalitzats

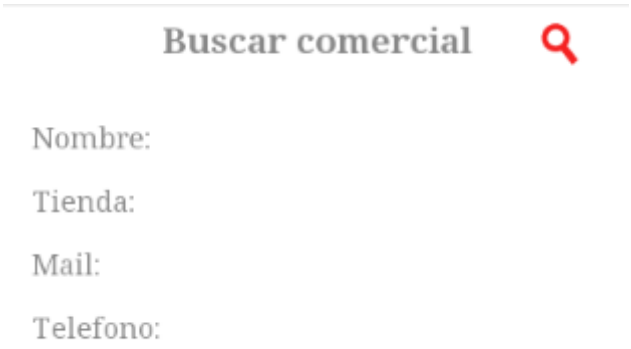
Una de les parts més important del disseny de l'aplicació han estat els widgets i els controls personalitzats. Per tal de donar el disseny desitjat he tingut que crear una gran quantitat de controls personalitzats. A continuació comento en detall cadascun dels controls.

Widgets

Es tracta del control principal del projecte, ja que tot gira entorn a aquest control. El panell està compost per quatre tipus de widgets: dos widgets que s'encarreguen de comptar, un widget que permet fer cerques i finalment un widget amb gràfics.

Widgets comptadors:

Widget per fer cerques:



Widget gràfic:

Cadascun d'aquests widgets es un component autònom, on a partir d'una configuració que es passa en el seu constructor ell ja s'encarrega de mostrar la informació i d'anar-la a buscar. La freqüència amb la que és realitza la recerca de la informació, es diferent per cadascun dels widgets, ja que depèn del propòsit de l'indicador. Per exemple els widgets comptadors actualitzen la informació cada minut ja que interessa que la informació estigui actualitzada de manera gairebé instantània. D'altra banda el widget per cercar els comercials, actualitza la informació a petició de l'usuari. Finalment el widget de gràfic actualitza la informació cada hora ja que es tracta de veure una evolució mensual i no és necessari tindre la informació gairebé en temps real per decidir. Alguns valors com que s'actualitzi cada minut pot semblar massa, l'he configurat d'aquesta manera perquè es tracta d'un producte de presentació i d'aquesta manera es pot veure la funcionalitat del panell amb facilitat. La freqüència d'actualització hauria de ser adaptada depenent del negoci al que va dirigit.

A continuació detallo en profunditat com he construït un dels widgets i després comentaré particularitats exclusives dels widgets.

Exemple Widget comptador Comercials:

Aquest és el control final que volem. A continuació detallo les principals tasques a nivell de desenvolupament que s'han de fer per obtenir aquest resultat.

Per crear aquest control hem de crear una classe Java que estengui la classe *"View"* d'Android.

```
public class Contador extends View
```

Per tal de configurar el widget, he creat un constructor que rep els paràmetres de configuració.

```
public Contador(Context ctx, ContadorEntity conf) {
	super(ctx);

	title = conf.getTitulo();
	contador_1_title = conf.getContador1Titulo();
	contador_2_title = conf.getContador2Titulo();
	usuario = conf.getUsuario();
	password = conf.getPassword();
	urlServicio = conf.getUrlServicio();
	metodo = conf.getMetodo();
	Valor1 = "0";
	Valor2 = "0";
	MiliseconsUpdateInterval = conf.getMiliseconsUpdateInterval();
	MiliseconsStartInterval = conf.getMiliseconsStartInterval();
	mHandler = new Handler();
	t = new Timer();

	Init();
}
```

El widget rep de la entitat de configuració el títol del widget, els títols de cadascun dels indicadors així com la informació necessària per comunicar-se amb el servei. Tota aquesta informació l'emmagatzema internament. Cal destacar que en el constructor és crea un *"Handler"* i un *"Timer"* que més endavant explicaré perquè s'utilitza. Cal destacar també, que al final del constructor es crida al mètode "Init".

El mètode *"Init"*:

```
private void Init(){
	//Definimos las dimensiones
	setMeasuredDimension(DEFAULT_WIDTH, DEFAULT_HEIGHT);

	rectanglePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
	rectanglePaint.setStyle(Paint.Style.FILL);
	rectanglePaint.setColor(Color.RED);

	lineaPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
	lineaPaint.setStyle(Paint.Style.FILL);
	lineaPaint.setColor(Color.WHITE);
	lineaPaint.setStrokeWidth(3);

	Typeface tf = Typeface.create(Typeface.SERIF,Typeface.BOLD);
	textTitlePaint = new Paint(Paint.ANTI_ALIAS_FLAG);
	textTitlePaint.setStyle(Paint.Style.FILL);
	textTitlePaint.setColor(Color.WHITE);
	textTitlePaint.setTextSize(22);
	textTitlePaint.setTypeface(tf);
	textTitlePaint.setTextAlign(Align.CENTER);

	textContadorPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
	textContadorPaint.setStyle(Paint.Style.FILL);
	textContadorPaint.setColor(Color.WHITE);
	textContadorPaint.setTextSize(16);
	textContadorPaint.setTextAlign(Align.CENTER);

	textValorPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
	textValorPaint.setStyle(Paint.Style.FILL);
	textValorPaint.setColor(Color.WHITE);
	textValorPaint.setTextSize(40);
	textValorPaint.setTypeface(tf);
	textValorPaint.setTextAlign(Align.CENTER);

	t.scheduleAtFixedRate(new UpdateTimeTask(),
MiliseconsStartInterval, MiliseconsUpdateInterval);
}
```

Aquest mètode és important perquè és on es defineixen els elements per pintar a la pantalla els diferents components com per exemple un rectangle, una línia, un tipus de lletra, etc...

Finalment en aquest mètode configuro el *"Timer"* per tal de que cada cert període de temps cridi al servei per obtenir la informació.

Com es pinta el control:

```
@Override
protected void onDraw(Canvas canvas) {
	super.onDraw(canvas);
	canvas.drawRect(0, 0, 400, 250, rectanglePaint);

	canvas.drawLine(70, 130, 170, 130, lineaPaint);
```

```
        canvas.drawLine(230, 130, 330, 130, lineaPaint);

        canvas.drawText(title, 200, 30, textTitlePaint);
        canvas.drawText(contador_1_title, 120, 170, textContadorPaint);
        canvas.drawText(contador_2_title, 280, 170, textContadorPaint);
        canvas.drawText(Valor1, 120, 110, textValorPaint);
        canvas.drawText(Valor2, 280, 110, textValorPaint);
}
```

Per tal de veure el control tal i com em vist en la captura, tenim que sobreescriure el mètode *"onDraw"* per tal de personalitzar com es dibuixarà en pantalla. Com que prèviament en el mètode *"Init"* hem creat els diferents objectes per tal de dibuixar els diferents elements, ara només cal utilitzar la classe "Canvas" d'Android i aquests objectes, per dibuixar-los en la pantalla. Per exemple:

```
canvas.drawLine(70, 130, 170, 130, lineaPaint);
```

Aquest codi pintarà una línia de color blanc (definida per l'objecte "lineaPaint") en funció de les coordenades que especifiquem, on el primer valor és l'inici de la coordenada x, el segon és l'inici de la coordenada y, el tercer valor és el final de la coordenada x i el quart valor és el fi de la coordenada y.

Com el control crida al servei:

```
final Runnable Refresh = new Runnable() {
        public void run() {
             invalidate();
             requestLayout();
        }
};

class UpdateTimeTask extends TimerTask {
   public void run() {

   RESTClient rClient = new RESTClient(urlServicio);
   Map<String,String> params = new HashMap<String, String>();
   params.put("u",usuario);
   params.put("p",password);

   String resultado = rClient.webGet(metodo, params);

   if(!resultado.isEmpty()){
         try {
                  JSONObject jobj = new JSONObject(resultado);
                  Valor1 = jobj.getString("Valor1");
                  Valor2 = jobj.getString("Valor2");
          } catch (JSONException e) {
             e.toString();
          }
     }
   mHandler.post(Refresh);
   }
}
```

Tal i com hem vist, en la funció *"Init",* es configura cada quant temps el *"timer"* executa una funcionalitat. La funcionalitat que executa el *"timer"* és una *"TimerTask"* aquesta tasca crea un nou fil i fa la petició al mètode corresponent. Un cop tenim la resposta li diem al fil principal que refresqui la vista (es crida de nou al mètode *"onDraw"*). Per dir-li això s'utilitza l'objecte *"Handler"* i el mètode que ha de cridar *"Refresh"*. Aquest invalida la vista i fa que es torni a cridar al mètode *"onDraw"*.

Com s'interacciona amb el widget:

Per a cada widget he definit una zona per arrossegar el control i una altre zona per interactuar amb el control tal i com es mostra en la següent figura:

Aleshores per captar la intenció d'usuari sobreescrivim el mètode *"onTouchEvent"* i definim uns *"listeners"* per tal de comunicar que s'han produït els esdeveniments.

```
private OnTouchContadorListener listener;
private OnDragContadorListener dragListener;

public interface OnTouchContadorListener {
      void onTouchContadorOccurred(View v, String valor);
}

public interface OnDragContadorListener{
      void onDragContadorOccurred(View v, int id);
}

public void setOnTouchContadorListener(OnTouchContadorListener l) {
       listener = l;
}

public void setOnDragContadorListener(OnDragContadorListener l) {
      dragListener = l;
}


@Override
public boolean onTouchEvent(MotionEvent event) {
      // Obtener la posicion relativa
```

```
        float x = event.getRawX() - this.getLeft();
        float y = event.getRawY() - this.getTop();

        if(y > 100 && y <= 250 && x <= 400){
                if(x < 250){
                        listener.onTouchContadorOccurred(this, "Valor1");
                }
                else{
                        listener.onTouchContadorOccurred(this, "Valor2");
                }
        }
        else if(y >= 0 && y <=99 && x <= 400){
                dragListener.onDragContadorOccurred(this, Id);
        }

        return super.onTouchEvent(event);
}
```

Els *"listeners"* serveixen per subscriure's als esdeveniments del control de manera que quan el control detecta un esdeveniment, tots els elements que es troben subscrits s'assabenten i poden interaccionar amb el control. En el mètode *"onTouchEvent"* primer és detecta en quina zona ha tocat l'usuari. En funció d'on ha tocat el control se sap si es troba en la zona d'arrossegament o interactiva. Un cop se quina és la zona, notifico els esdeveniments als *"listeners"* subscrits. Per tal d'enregistrar *"listeners"*, he habilitat un parell de mètodes d'acord als esdeveniments que permet el widget.

```
public void setOnTouchContadorListener(OnTouchContadorListener l) {
        listener = l;
}

public void setOnDragContadorListener(OnDragContadorListener l) {
       dragListener = l;
}
```

Fins ara em vist funcionar el control. Per acabar només queda com el ficarem en el *"layout"* o pantalla que desitgem. En el meu escenari tenim una *"RelativeLayout"* que conté tot el panell. Per inserir aquest control ho farem de la següent manera:

```
//Contador comerciales
ContadorEntity confComerciales = new ContadorEntity();
confComerciales.setTitulo("Comerciales");
confComerciales.setContador1Titulo("Conectados");
confComerciales.setContador2Titulo("Bajas");
confComerciales.setUrlServicio(SERVICE_URI);

confComerciales.setMetodo("ObtenerDisponibilidadComerciales");

confComerciales.setUsuario(datosUsuario.getUsuario());

confComerciales.setPassword(datosUsuario.getPassword());
```

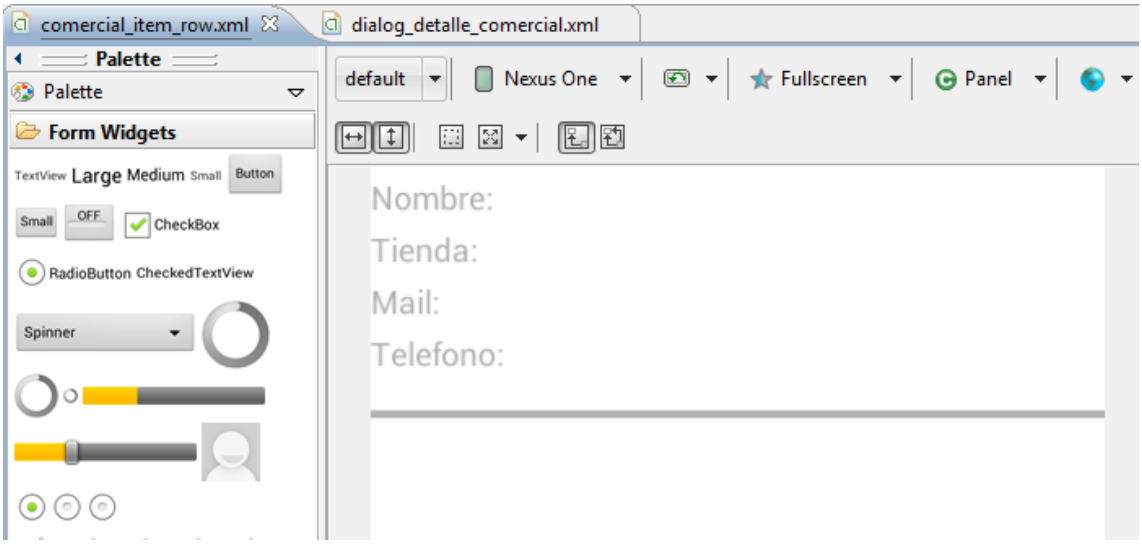
```
confComerciales.setMiliseconsUpdateInterval(60000);//cada minuto

confComerciales.setMiliseconsStartInterval(1000);//esperamos un segundo
antes de recuperar los datos

contadorComerciales = new Contador(this, confComerciales);

RelativeLayout.LayoutParams lpComerciales = new
RelativeLayout.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT,Relat
iveLayout.LayoutParams.WRAP_CONTENT);
lpComerciales.leftMargin = leftM;
lpComerciales.topMargin = topM;
lpComerciales.addRule(RelativeLayout.BELOW, R.id.vwLineaRojaTitulo);

contadorComerciales.setLayoutParams(lpComerciales);

contadorComerciales.setOnTouchContadorListener(contadorComercialesListener
);

contadorComerciales.setOnDragContadorListener(contadorComercialesLongClick
);

//contadorComerciales.setOnDragListener(contadorComercialesDragListener);

contadorComerciales.setId(conf.getId());
rl.addView(contadorComerciales);
```

Primer de tot es crea la entitat de configuració i es crea l'objecte. Després es creen uns paràmetres de *"layout"* que indiquen en quina posició de la pantalla es trobarà. Després ens subscrivim als esdeveniments afegint els *"listeners"* que volem i finalment fem *"addView"* al *"layout"* corresponent.

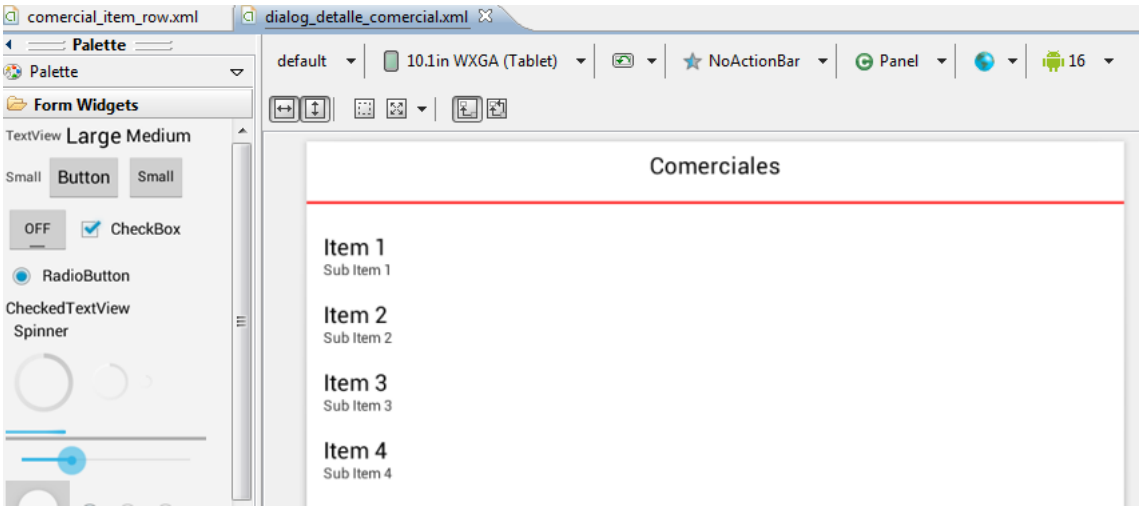
Llistat personalitzat

Els llistats que veiem al interaccionar amb els widgets, són llistats personalitzats basats en el control *"listView"* d'Android. El meu llistat personalitzat es el següent:

Per crear-lo, primer de tot vaig dissenyar la fila del llistat amb els camps que volia que apareguéssim.



Un cop dissenyada la fila, es dissenya la pantalla amb el títol i l'objecte "listview" que canviarem per la nostre fila personalitzada.



Per mostrar-lo amb el disseny final ho he fet de la següent manera:

```
private void MostrarComerciales(){
	DetalleComercialAdapter adapter = new DetalleComercialAdapter(this,
R.layout.comercial_item_row, comerciales);

	final Dialog dialog = new Dialog(this,
android.R.style.Theme_DeviceDefault_Light_Dialog_NoActionBar);
	dialog.setContentView(R.layout.dialog_detalle_comercial);
	ListView lv =
(ListView)dialog.findViewById(R.id.listViewDetalleComercial);

       lv.setAdapter(adapter);

	lv.setOnItemClickListener(new OnItemClickListener() {
		@Override
		public void onItemClick(AdapterView<?> arg0, View arg1,
int arg2, long arg3) {
				// TODO Auto-generated method stub

			}
		});

	dialog.show();
}
```

Primer de tot he creat una classe *"DetalleComercialAdapter"* que permet lligar unes dades amb cadascuna de les files personalitzades que he dissenyat anteriorment. Un cop fet això creem el *"dialog"* amb la peculiaritat que com a vista posem la vista que hem creat i al *"listview"* de la vista que hem creat li passem el *"dataAdapter"* amb la fila personalitzada per nosaltres.

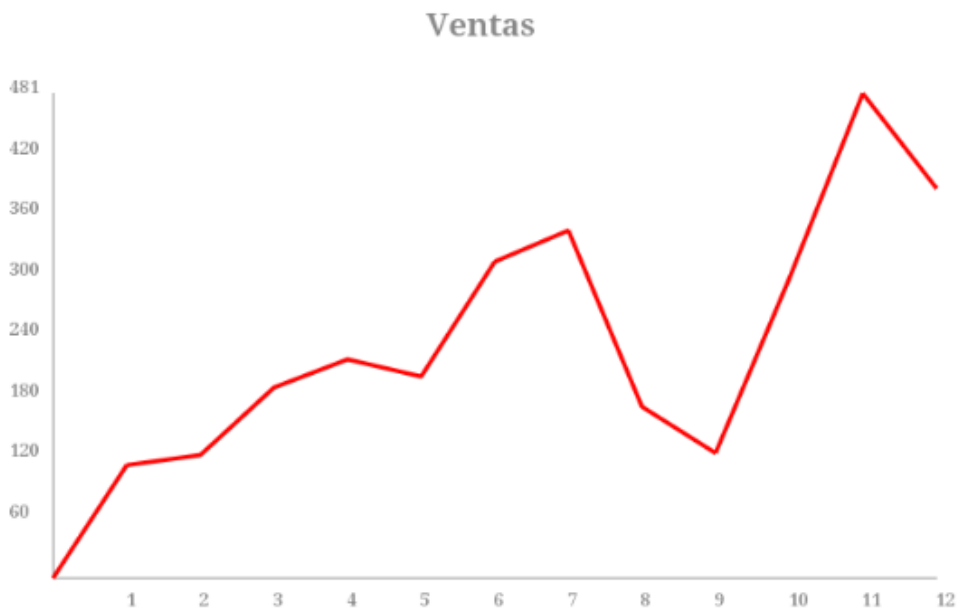
Gràfic

Un dels widgets, mostra un gràfic amb l'evolució de les vendes per mes. Per fer-ho, s'ha dissenyat un motor propi de renderitzat mitjançant l'objecte *"Canvas"* d'Android. Aquest motor permet re-escalar el gràfic x2, x3.... donant en cada escala

un major nivell de detall. El gràfic en el widget es mostra a escala 1 i quan accedim al detall es veu a escala 2 visualitzant aquest amb més nivell de detall.

Gràfic a escala 1

Gràfic a escala 2



L'algoritme per mostrar-lo es troba en el mètode *"onDraw"* del widget del gràfic i es basa en utilitzar primitives i calcular les posicions x i y. No entro en detall del codi degut a la seva extensió, perquè es pot consultar en el projecte i perquè es troba degudament comentat.

5.2.4 Altres funcionalitats importants

Arrossegar i ordenar widgets

Una de les funcionalitats implementades és ordenar els widgets arrossegant-los. Per realitzar aquesta funcionalitat, s'ha definit una zona d'arrossegament per a cada widget. Un cop l'usuari toca en aquesta zona el widget llença un esdeveniment indicant que s'inicia l'arrossegament de l'objecte i s'ha de crear una ombra del widget. El terme ombra és un objecte que es crea quan comença l'arrossegament i proporciona una imatge que l'usuari arrossegarà per la pantalla. Mentre dura l'arrossegament el sistema notifica en tot moment als *"listeners"* per on passa l'ombra. Un cop deixem de tocar la pantalla l'ombra es cau i tenim que detectar en quina posició es troba.

```
private OnDragGraficoListener dragListener;

public interface OnDragGraficoListener{
	void OnDragGraficoOccurred(View v, int id);
}

public void setOnDragGraficoListener(OnDragGraficoListener l) {
	dragListener = l;
}
@Override
public boolean onTouchEvent(MotionEvent event) {
	if(escala == 1){
	// Obtener la posicion relativa
	float x = event.getRawX() - this.getLeft();
	float y = event.getRawY() - this.getTop();

	if(y > 100 && y <= 250 && x <= 400){
		listener.onTouchGraficoOccurred(this);
	}
	else if(y >= 0 && y <=99 && x <= 400){
		dragListener.OnDragGraficoOccurred(this, Id);
	}
   }
   return super.onTouchEvent(event);
}
```

Per cada widget he creat un *"listener"* per l'esdeveniment d'arrossegar i un mètode per afegir subscriptors. Quan l'usuari toca el widget, el propi widget, detecta en quina zona es troba i llença l'esdeveniment corresponent.

```
contadorComerciales.setOnDragContadorListener(contadorComercialesLongClick
);

private OnDragContadorListener contadorComercialesLongClick = new
OnDragContadorListener() {

	public void onDragContadorOccurred(View v, int id) {
```

```
                idOrigen = id;
                IniciarDragDrop(v, id);
        }
};

private void IniciarDragDrop(View v, int id){
        MyDragShadowBuilder sb = new MyDragShadowBuilder(v);
        ClipData.Item item = new ClipData.Item(String.valueOf(id));
        String[] mime = new String[1];
        mime[0] = "text/plain";
        ClipData dragData = new ClipData(String.valueOf(id), mime, item);
        v.startDrag(dragData, sb, null, 0);
}
```

En l'activitat del panell primer de tot he afegit un subscriptor per l'esdeveniment d'arrossegament del widget. Aquest subscriptor permet capturar quan el widget llença l'esdeveniment corresponent. Finalment aquest esdeveniment crida a la funció *"IniciarDragDrop"*. Aquesta funció és l'encarregada de crear l'ombra i de iniciar l'arrossegament.

```
// Defines a variable to store the action type for the incoming event
final int action = event.getAction();
boolean rtv = true;

switch(action) {
        case DragEvent.ACTION_DRAG_STARTED:
                        break;

        case DragEvent.ACTION_DRAG_ENTERED:

                        break;

        case DragEvent.ACTION_DRAG_LOCATION:

                        break;

        case DragEvent.ACTION_DRAG_EXITED:

                        break;

        case DragEvent.ACTION_DROP:

                        break;

         case DragEvent.ACTION_DRAG_ENDED:

                        break;
}
```

A partir d'aquest moment ja ens troben plenament en l'acció d'arrossegament. El sistema operatiu per cada acció que fem, comunica als *"listeners"* que es troben subscrits a aquest esdeveniment. En el meu cas tinc el mètode *"DragDropManager"* que conté una estructura com la de l'exemple de mes amunt.

Per cada moviment que fem, es crida aquest mètode i podem veure quin tipus d'acció realitza l'usuari. En el meu cas jo únicament tinc en compte l'acció *"drop"* on identifico el destí, recarrego en pantalla els widgets i em comunico amb la capa de serveis per guardar la informació del nou ordre dels widgets.

5.2.5 Comunicació

En aquest projecte una part important és la comunicació entre els diferents components. En aquesta secció vull destacar l'intercanvi d'informació entre activitats i la comunicació amb el servidor.

Intercanvi d'informació entre activitats

L'aplicació te dues activitats principals, el login i el panell. Quan es realitza el login, si ha estat correcte, es guarda en una entitat les dades de l'usuari i en altre entitat l'ordre dels widgets. Aquestes entitats no son valors primitius com pot ser una cadena o un sencer, si no que són entitats més o menys complexes. Per poder intercanviar-les entre activitats, han d'implementar la interfície serialitzable de JAVA tal i com es mostra a continuació.

```
import java.io.Serializable;

public class DatosUsuario implements Serializable {
	/**
	 *
	 */
	private static final long serialVersionUID = 8735194711884619813L;

	private int id;
	private String Nombre;
	private String Apellido1;
	private String Apellido2;
	private String Empresa;
	private String Usuario;
	private String Password;
```

Un cop les entitats implementen *"Serialitzable"* ja es poden enviar a una altre activitat. Per fer-ho només cal fer el següent.

```
public final static String DATOS_USUARIO =
"com.sts.businesspanel.DATOS_USUARIO";

Intent intent = new Intent(this, Panel.class);
intent.putExtra(DATOS_USUARIO, datosUsuario);
startActivity(intent);
```

En la activitat d'origen declarem una clau del valor que tenim que passar a l'altre activitat, en aquest cas la clau és "DATOS_USUARIO". A continuació, utilitzem la

classe "Intent" d'Android per instanciar la nova activitat i amb el mètode "putExtra" inserim la entitat per poder-la agafar en la activitat de destí.

```
Intent i = getIntent();
datosUsuario = (DatosUsuario)i.getSerializableExtra(Login.DATOS_USUARIO);
```

En la activitat de destí, tenim que accedir a la classe "Intent" i cridar al mètode *"getSerialitzableExtra"* amb la clau que hem definit prèviament en la activitat d'origen(DATOS_USUARIO) .

Comunicació amb la capa de serveis

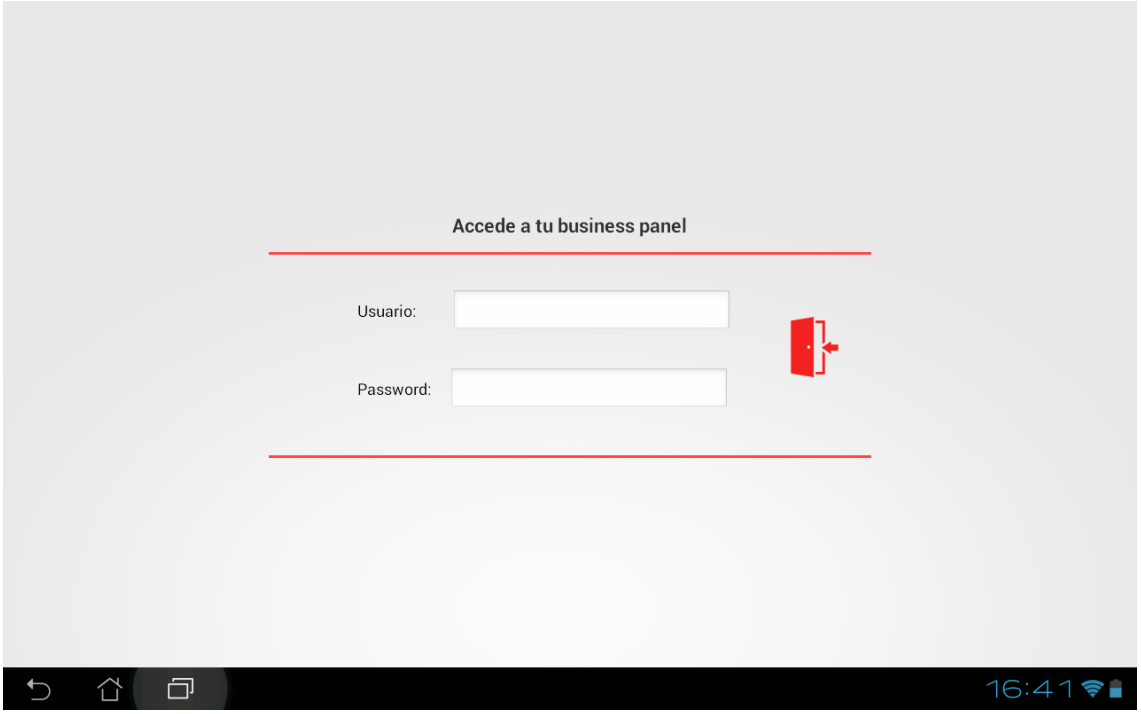
Per realitzar la comunicació entre l'aplicació client i els serveis, vaig investigar si existia una llibreria nativa per Android per fer la crida als serveis. Segons la documentació d' Android, et diuen que has d'utilitzar el paquet HTTP de Apache, però la llibreria l'has de crear des de zero. Com que crec que és un problema molt habitual vaig cercar API's lliures que hagués desenvolupat algun usuari i vaig trobar la següent, que complia amb les necessitats del meu projecte http://www.josecgomez.com/2010/04/30/android-accessing-restfull-web-services-using-json/

No obstant vaig realitzar canvis. Aquesta llibreria utilitza per retornar les entitats JSON una llibreria de Google GSON ,per fer la conversió del objecte de resposta a un objecte JSON. Vaig decidir no utilitzar aquesta llibreria i utilitzar la pròpia de JAVA, per fer la solució més estàndard i no dependre de tercers. El resultat ha estat molt bo i funciona perfectament. Un altre canvi que vaig fer, es augmentar el valor del *"timeout"* per evitar problemes de resposta en servidors petits o locals.

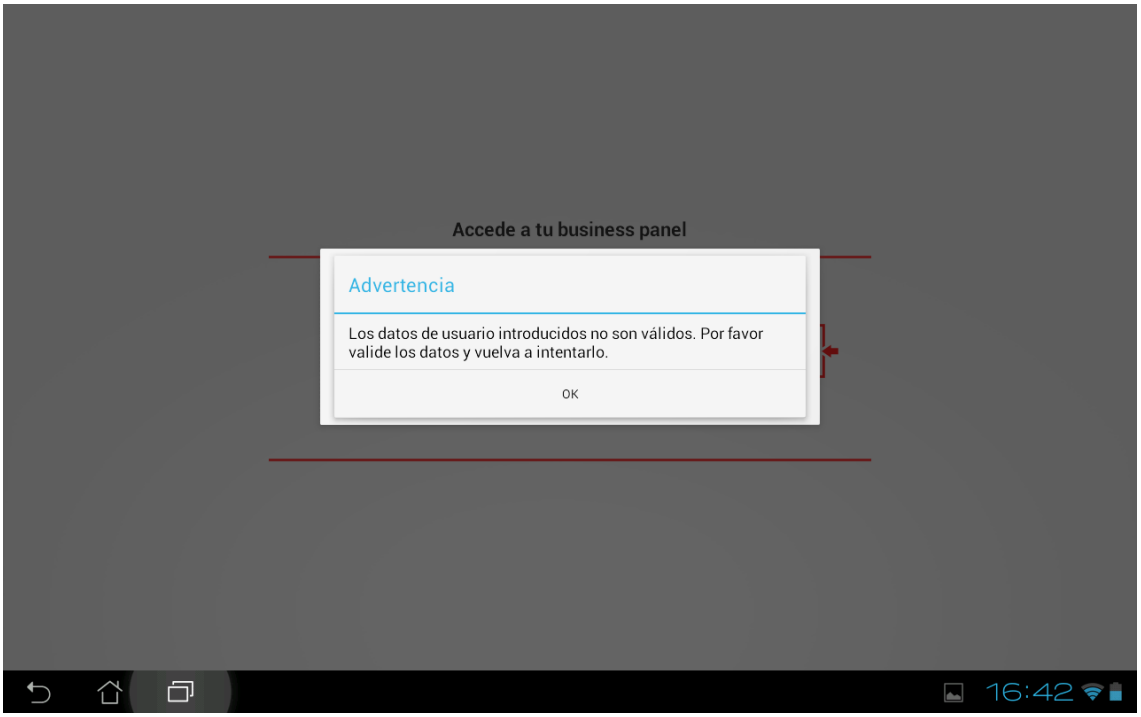
5.2.6 Funcionalitat i captures de pantalla

En aquest apartat, mostraré les captures de pantalla que he realitzat de l'aplicació i faré una petita descripció de cada escenari que mostren les captures per tal de facilitar la seva comprensió.

Login

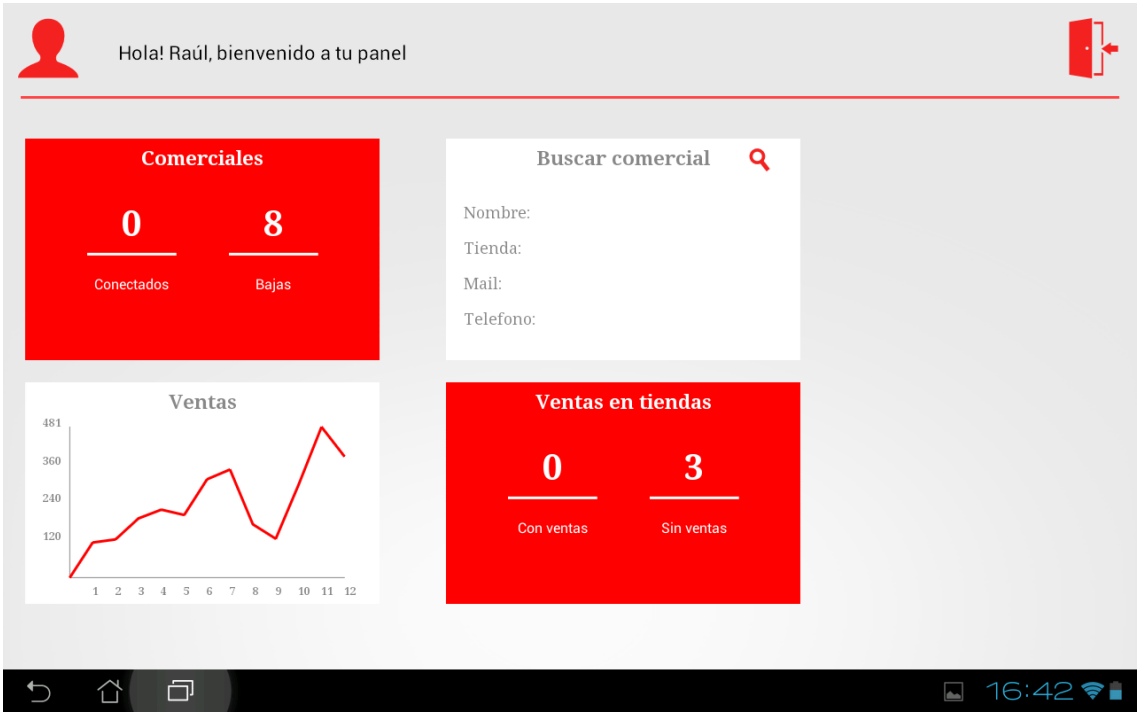


Aquesta en la pantalla d'accés al panell, l'usuari ha d'introduir les seves dades i tocar a la porta, aleshores es validen les seves dades i es carrega la seva configuració. A continuació una captura de l'aplicació quan les dades introduïdes son incorrectes.

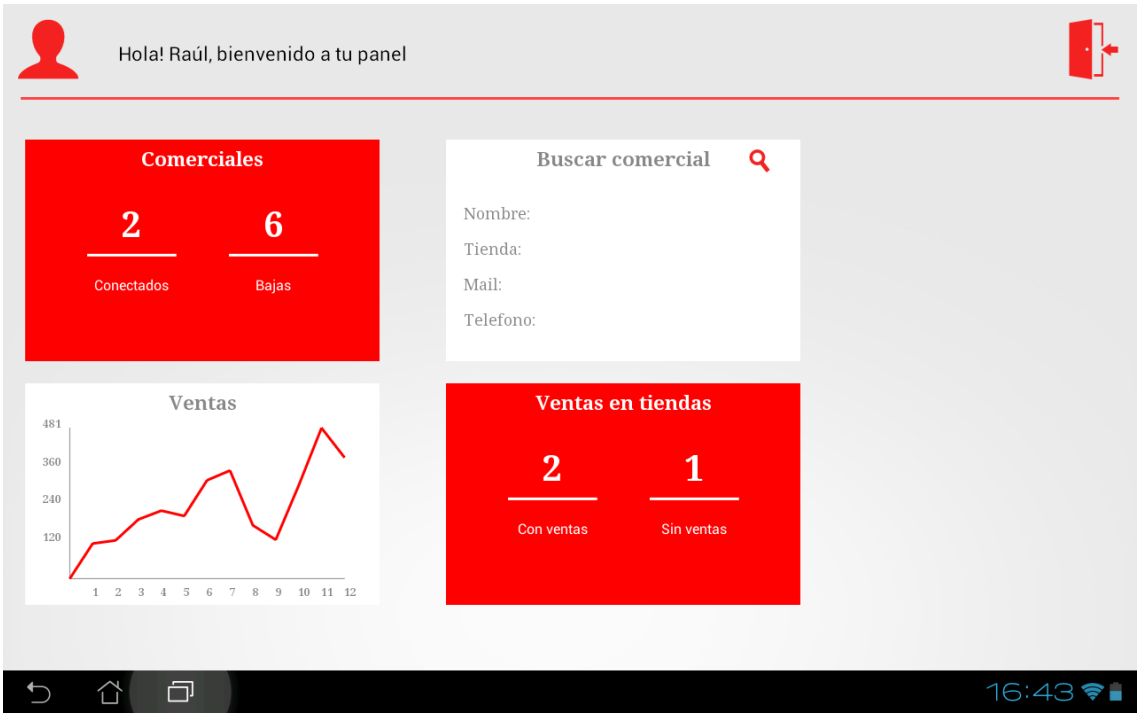


Panell de negoci

Un cop l'usuari es vàlid, es mostra el panell d'acord a la seva configuració. En el panell es carreguen els quatre widgets amb l'ultima informació relacionada. A continuació una captura de pantalla del panell de l'usuari.

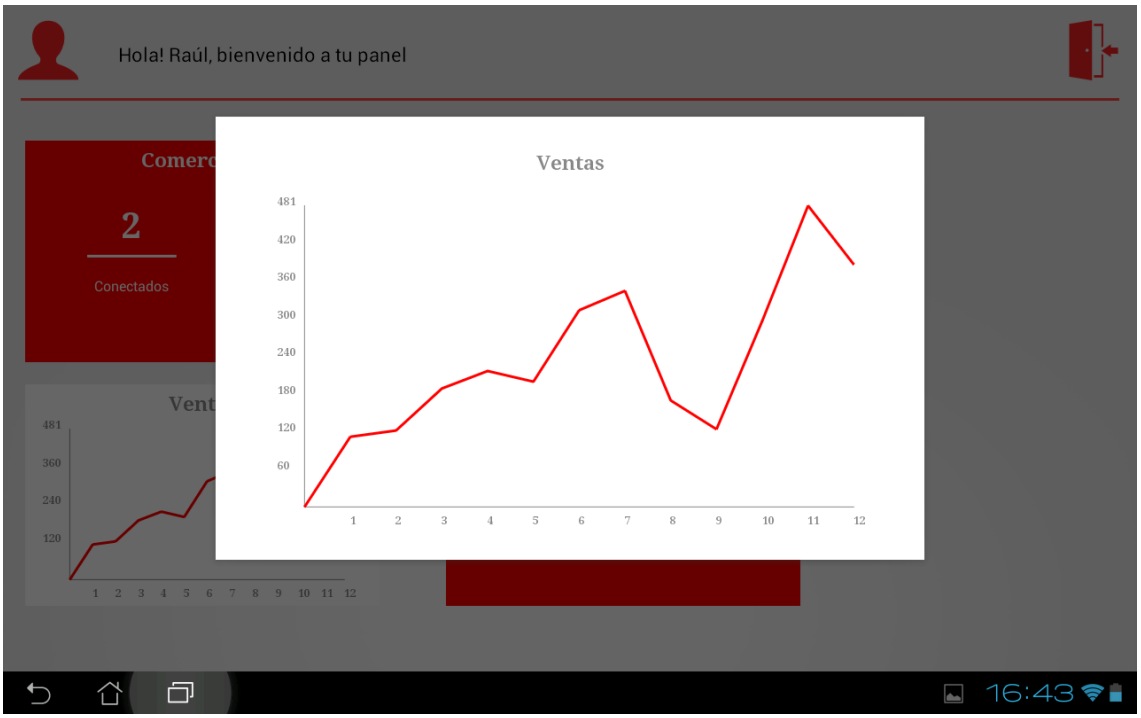


A continuació un captura de pantalla del panell amb dades actualitzades (el widget "comerciales" i "ventas en tiendas" s'han actualitzat amb la informació recent).

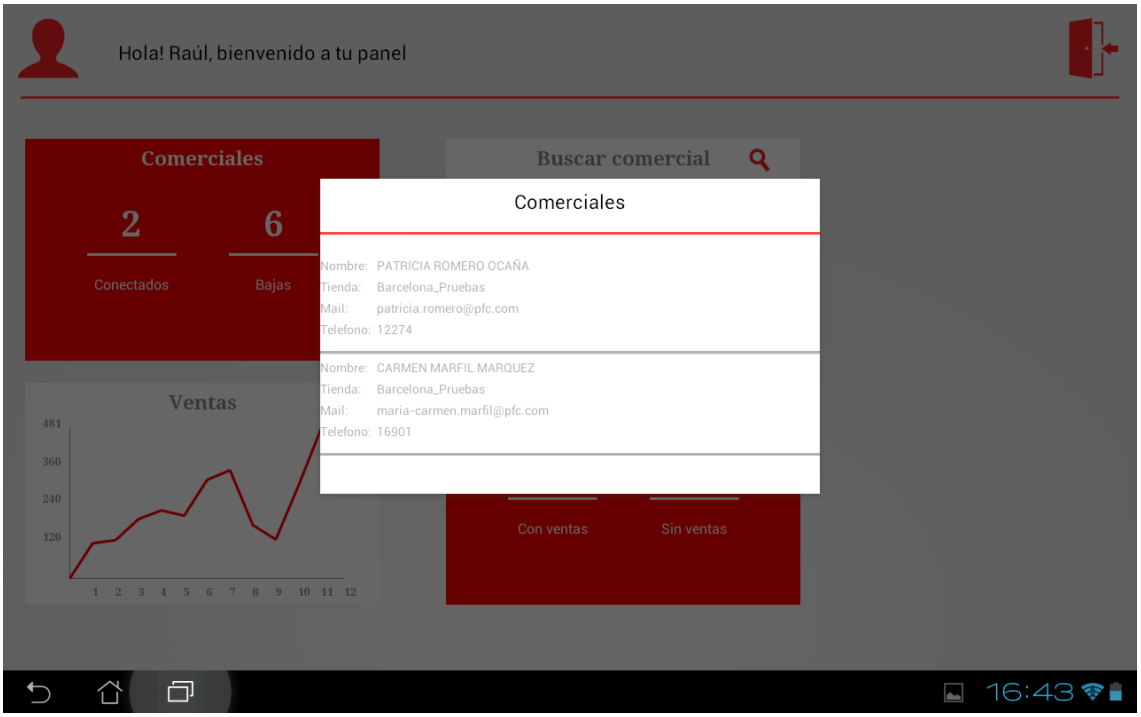


Tal i com s'ha comentat es tracta d'un panell interactiu. A continuació es mostren unes captures de la interacció entre l'usuari i el panell.

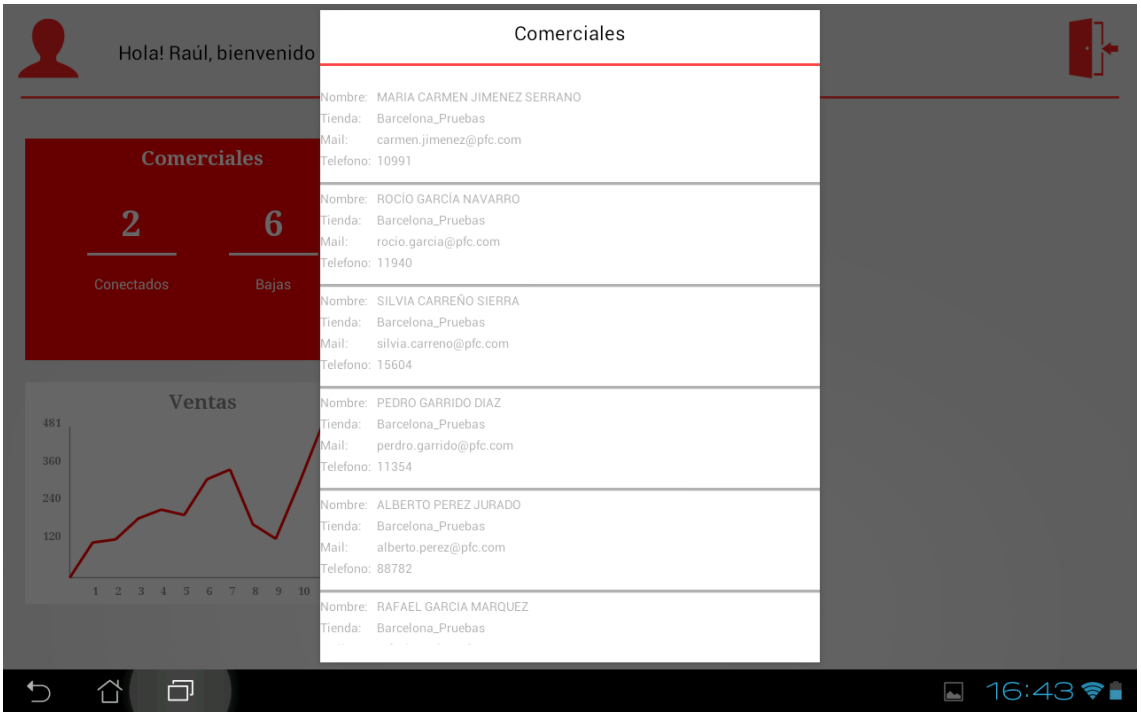
En aquest cas, l'usuari ha tocat la zona interactiva del widget del gràfic de ventes i el sistema li mostra el gràfic amb més detall.



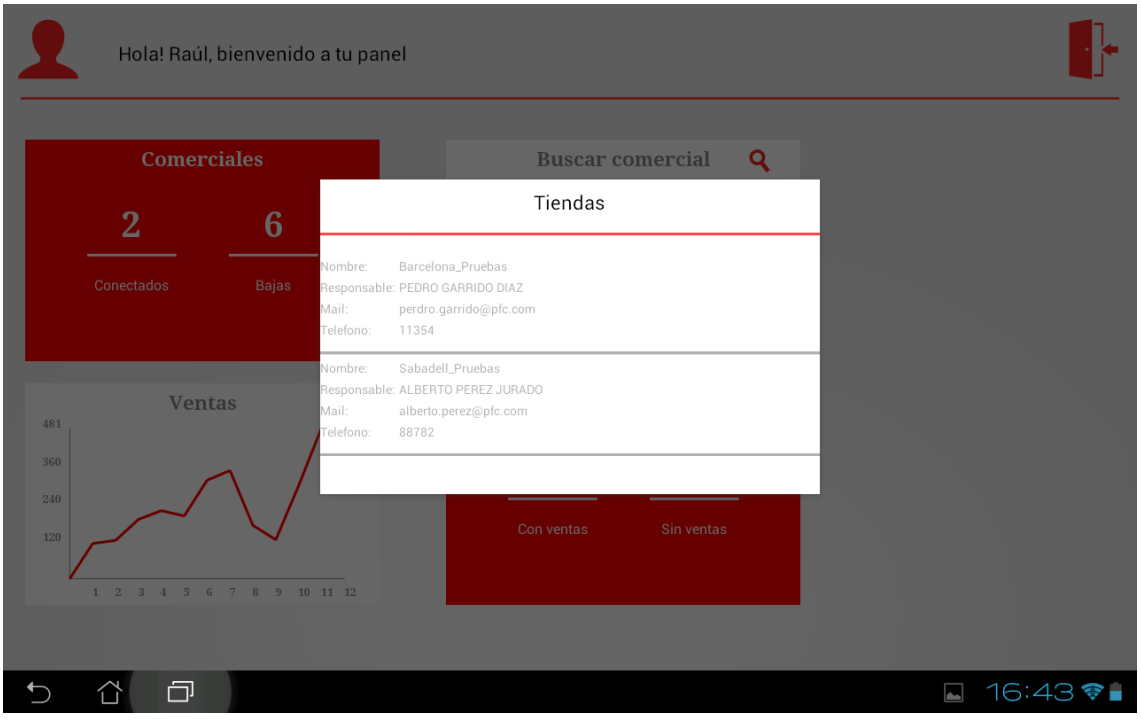
A continuació, l'usuari interacciona amb el widget "Comerciales" concretament està veien els dos comercials connectats.



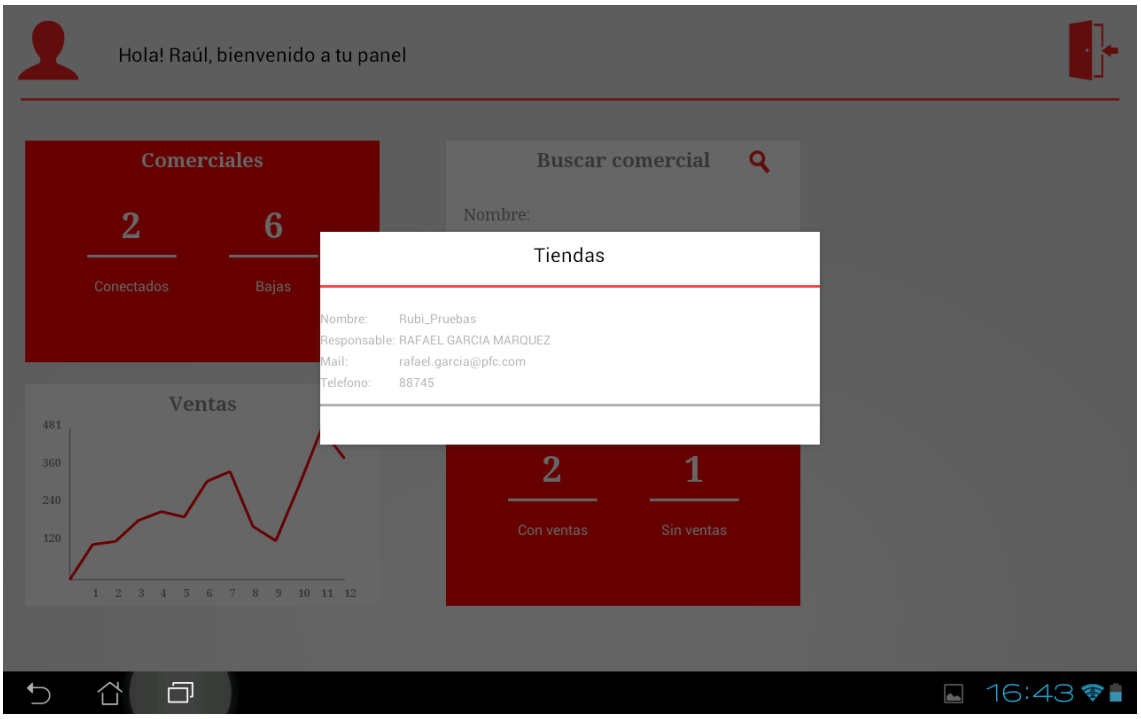
A continuació l'usuari mira els 6 comercials que són baixa de moment.



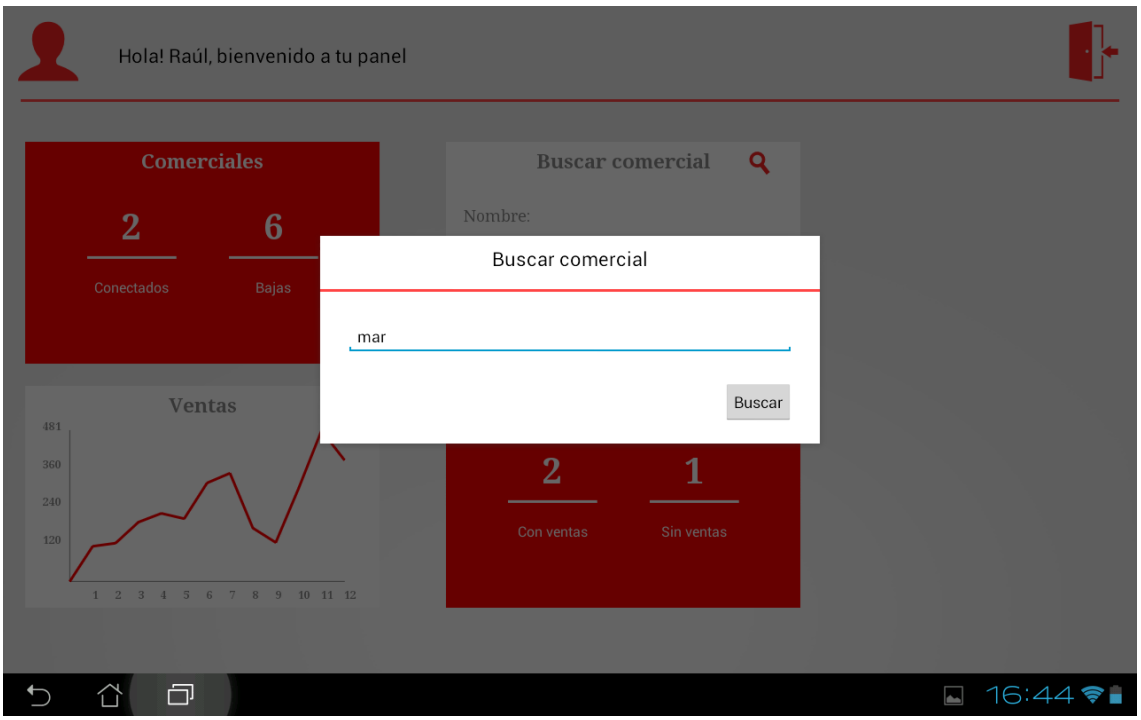
Passem al widget "Ventas en tiendas". A continuació es veu una captura de l'usuari visualitzant les dues botigues que ja han realitzat vendes.



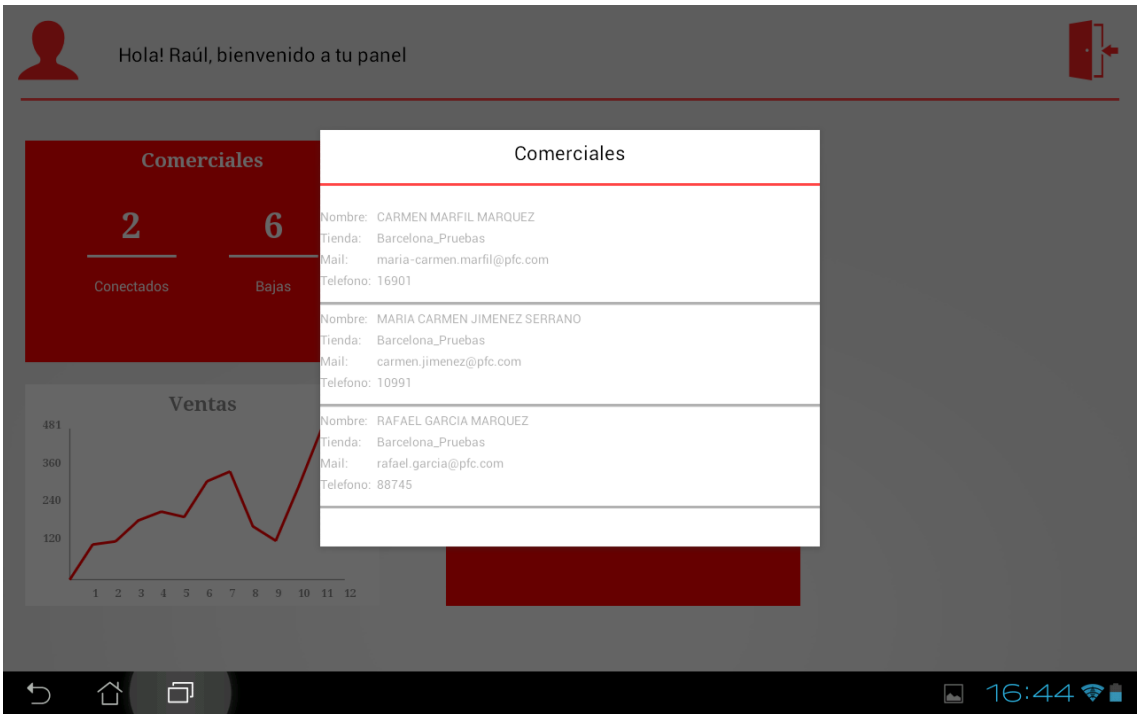
I ara l'usuari mirant la botiga que encara no ha fet cap venda.



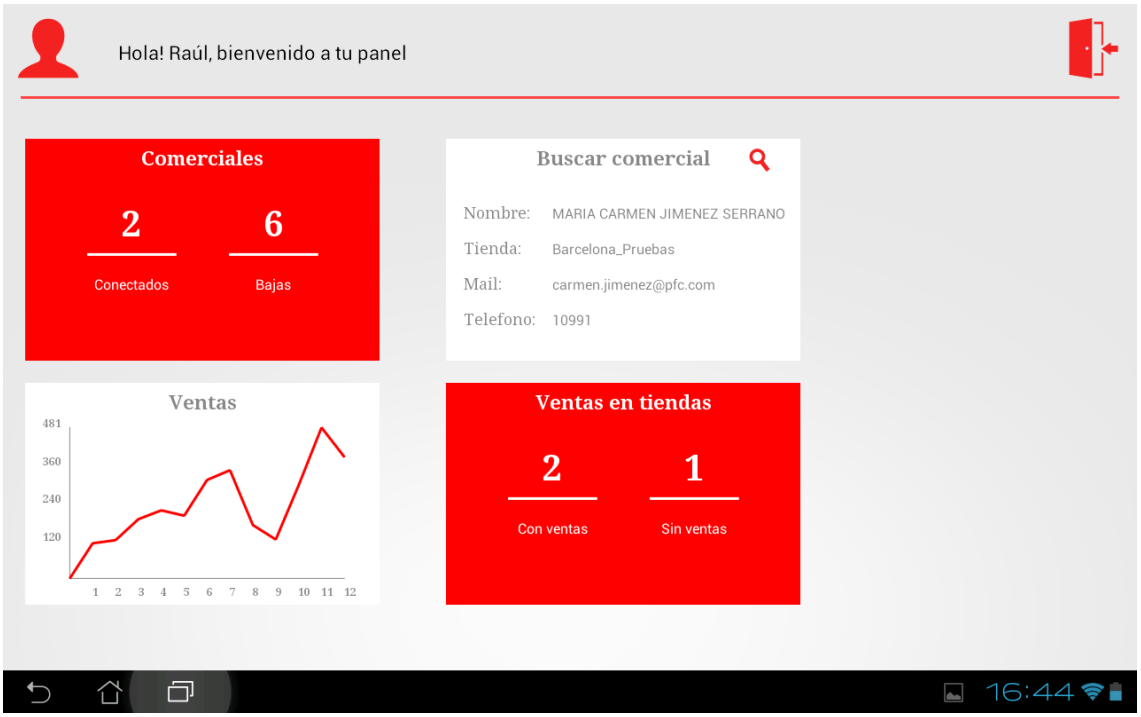
Finalment l'usuari vol cercar un comercial. Per fer-ho toca a la zona interactiva del widget "Buscar comercial" i li surt una pantalla perquè introdueixi els criteris de cerca.



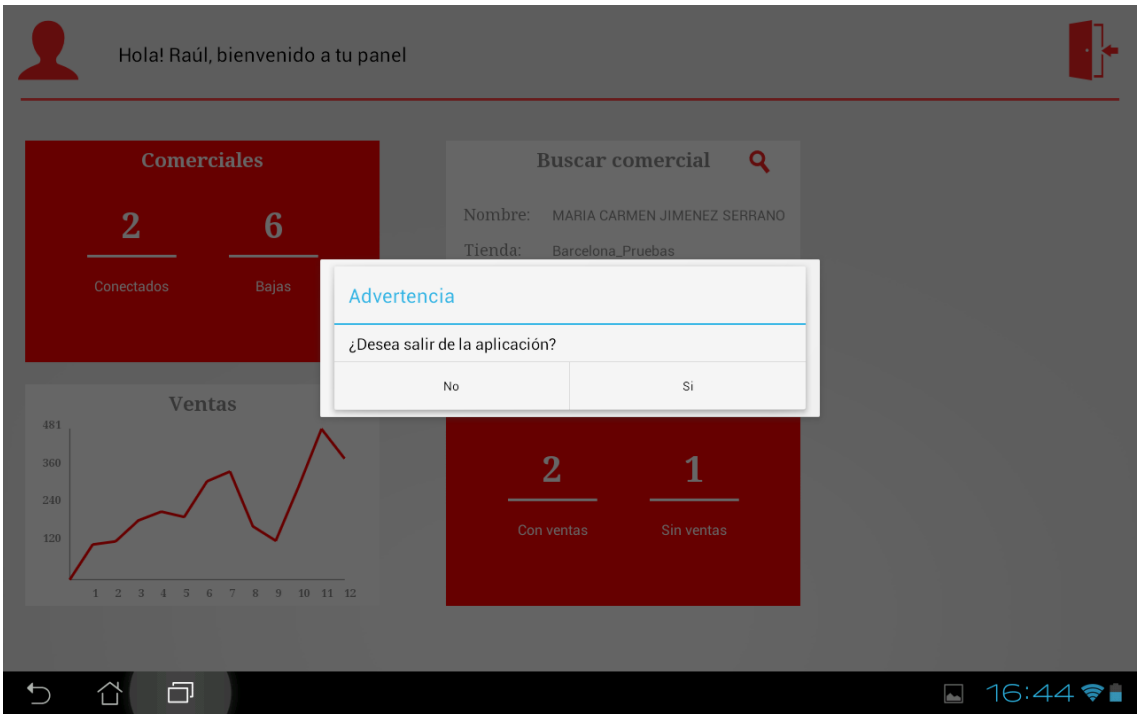
Un cop introduït, el sistema li mostra els comercials que coincideixen amb la paraula que ha cercat.



Finalment l'usuari selecciona el segon comercial i el widget mostra les dades associades a aquest comercial.



Per acabar, l'usuari vol sortir de l'aplicació i el sistema li demana confirmació.


Hola! Raúl, bienvenido a tu panel
Comerciales
2
Conectados
6
Bajas
Buscar comercial
Nombre:
MARIA CARMEN JIMENEZ SERRANO
Tienda:
Barcelona_Pruebas
Advertencia
¿Desea salir de la aplicación?
No
Si
Ventas
481
360
240
120
1 2 3 4 5 6 7 8 9 10 11 12
2
Con ventas
1
Sin ventas
16:44

6. Línies obertes del projecte

La primera línia amb la que continuaria el projecte és canviar la tecnologia per refrescar les dades. En el projecte defineixo un interval de temps per tal de refrescar les dades. Un cop passa aquest interval de temps es fa una petició i s'obtenen les noves dades. Aquesta manera de fer-ho és fàcil d'implementar, parametritzable i és una bona solució per la majoria d'escenaris. No obstant això, no ens permet veure el negoci realment en temps real, ja que hi ha un petit període de temps que no tenim les dades actualitzades. Això pot provocar que en algun tipus de negoci no sigui la millor solució, per exemple en l'entorn borsari ón es important tenir les dades al instant. Per tal de millorar el sistema utilitzaria notificacions PUSH. Aquesta tecnologia crea un canal entre el client i el servidor ón el servidor és el que genera la transmissió de manera que no cal que el client demani la informació al servidor, sinó que aquest se la pot donar en quant la té. Aquesta tecnologia necessita més infraestructura la qual no es disposava per realitzar el projecte a part es més complexa d'implementar.

L'altra línia que modificaria és canviar els serveis WCF per serveis ASP.net MVC. MVC s'està convertint en l'actualitat en la manera més estàndard per realitzar aplicacions web, ja que tot el projecte queda estructurat de manera més clara. MVC a més a més, és més fàcil de configurar com a REST que no pas WCF per tant utilitzaria aquesta solució per senzillesa i per seguir els estàndards del sector.

Un altre aspecte important per continuar és que, a apart de la versió tablet, també hauria d'existir una versió per smartphones adaptada a aquesta plataforma. Com que la pantalla és de menys polsades que en els tablets, el disseny de la interfície d'usuari hauria d'estar totalment redissenyat i pensat per aquets dispositius. D'aquesta manera tindríem en tot moment els nostre negoci en la nostra ma i en qualsevol lloc.

Respecte a noves funcionalitats, afegiria les següents:

- Utilitzar la càmera interna dels dispositius perquè l'usuari pugui fer-se una foto i posar-la com perfil o bé la possibilitat que seleccioni una imatge de la galeria per posar-la de perfil.
- També afegiria una nova opció de menú perquè l'usuari pugui canviar la seva configuració d'usuari, com per exemple el nom d'usuari i contrasenya.
- Com que poden existir una gran quantitat de widgets i diferents tipus d'usuaris que els utilitzin, crearia una galeria de widgets de manera que l'usuari pugui afegir o eliminar els widgets que vol veure en el seu panell. D'aquesta manera el panell s'adaptarà a les necessitats de cada usuari.

7. Conclusions

Durant el desenvolupament d'aquest projecte, s'ha creat amb èxit una aplicació per tablet Android i una capa de serveis funcionant com una arquitectura SOA. Aquestes fites han permès aconseguir els objectius que es van exposar al començament d'aquesta memòria de projecte.

El model d'arquitectura SOA ha donat molt bons resultats. Els serveis resultants son fàcilment re-aprofitables per qualsevol tipus d'aplicació no només per l'aplicació tablet. L'aplicació tablet ha complert completament els objectius marcats en quan usabilitat i disseny ja que mostra la informació de manera clara i concisa. Aquesta aplicació fa molt senzill poder accedir a la informació del nostre negoci.

La comunicació entre la tecnologia Java-Android i els serveis .NET s'ha realitzat amb èxit de manera molt òptima utilitzant la tecnologia REST. D'aquesta manera s'ha aconseguit l'objectiu de mostrar com poden conviure dues tecnologies totalment diferents.

En quant a la metodologia utilitzada s'ha adaptat perfectament a aquest tipus de projecte permetent acomplir les fites planificades inicialment.

Finalment i des del punt de vista personal, aquest projecte ha estat una experiència molt bona, permetent per una banda realitzar una aplicació de mobilitat per a tablet. També he pogut aprendre des de zero una tecnologia com Android i finalment m'ha agradat moltíssim veure el fàcil que era que es comuniquessin dues tecnologies tant diferents com Android i .NET. Finalment el resultat del conjunt m'ha encantat ja que l'aplicació resultant crec que és un gran producte pel món empresarial.

8. Glossari

A continuació tenim una taula que recull la definició dels diferents termes, abreviatures i acrònims que són necessaris per comprendre la memòria del projecte.

Terme	Definició
.Net	Framework de Microsoft pel desenvolupament d'aplicacions.
Android	Sistema operatiu per a dispositius mòbils, basat en Linux i desenvolupat per Google.
API	"Application Programming Interface" es tracta d'un conjunt de mètodes o funcions que ofereix una llibreria per ser utilitzada per un altre component de software abstraient-se de la funcionalitat de la llibreria.
ASP.net	Framework per aplicacions web desenvolupat per Microsoft.
Back-end	Part del software que processa l'entrada del front-end.
C#	Llenguatge de programació orientat a objectes de la plataforma .NET desenvolupat per Microsoft.
Canvas	Component de software utilitzat per dibuixar en la pantalla diferents elements com per exemple línies, rectangles o imatges.
Eclipse	Entorn de desenvolupament d'aplicacions multi plataforma.
Front-end	Part del software que interactua amb els usuaris.
GSON	Llibreria de codi obert desenvolupada per Google i que permet serialitzar i deserialitzar objectes JAVA i JSON.
HTTP	"Hypertext Transfer Protocol" protocol sense estat utilitzat en peticions web.
IDE	"Integrated Development Environment" component de software destinat al desenvolupament d'aplicacions.
IIS7	Servidor Web desenvolupat per Microsoft pel sistema operatiu Windows.
iOS	Sistema operatiu mòbil basat en UNIX creat per Apple.
JAVA	Llenguatge de programació multi

	plataforma i orientat a objectes desenvolupat per SUN Microsystems.
JDK	Java Development Kit" conjunt d'eines per realitzar aplicacions en JAVA.
JSON	"Javascript Object Notation" és un format per representar dades que te l'avantatge de ser lleuger a l'hora de transmetre les dades.
LINQ	"Language Integrated Query" tecnología de Microsoft per realitzar consultes semblants al estàndard SQL utilitzant llenguatges de la plataforma .Net.
MVC	"Model Vista Controlador" és un patró de disseny de software molt utilitzant en aplicacions web.
Objective-c	Llenguatge de programació orientat a objectes molt semblant a "C" i que en la actualitat s'utilitza per desenvolupar aplicacions per dispositius d'Apple com iPhone o iPad.
Pixels	"Picture Element" és la unitat mínima homogènia en color per la qual es troba composada una imatge digital
POST, GET, PUT i DELETE	Conjunt d'operacions de comunicació del protocol HTTP.
procediments emmagatzemats	Procediment o programa emmagatzemat directament en la base de dades. Normalment s'utilitza per encapsular en la base de dades un procediment complex o gran.
PUSH	Tecnologia de comunicació per internet on la transacció s'origina des del servidor envers al client.
REST	"Representational State Transfer" actualment s'utilitza per descriure una interfície web sense les abstraccions addicionals del protocol SOAP.
SOA	"Service Oriented Arquitecture" arquitectura on els serveis s'utilitzen per donar suport al negoci.
SOAP	"Simple Object Acces Protocol" protocol que defineix com es comuniquen dos objectes mitjançant XML.
SQL Server	Sistema de gestió de base de dades desenvolupat per Microsoft.
Visual Studio 2010	Entorn de desenvolupament de

	aplicacions en la plataforma .Net.
WCF	"Windows Communication Foundation" plataforma de missatgeria de .Net destinada a crear aplicacions distribuïdes de manera senzilla.
Windows 8	Última versió del sistema operatiu de Windows que destaca perquè unifica les interfícies de diferents dispositius (mòbils, tablet i ordinadors tradicionals).

9. Fonts d'informació

Per al desenvolupament d'aquest projecte, s'han consultat les següents fonts d'informació.

Microsoft Developer Network

http://msdn.microsoft.com/es-es/

Developer Android

http://developer.android.com/index.html

Altres enllaços d'interès

http://es.wikipedia.org/wiki/Wikipedia:Portada

http://www.codeproject.com/Articles/255684/Create-and-Consume-RESTFul-Service-in-NET-Framewor

http://fszlin.dymetis.com/post/2010/05/10/Comsuming-WCF-Services-With-Android.aspx

http://android-pro.blogspot.com.es/2010/01/understanding-string-resources.html

http://andmobidev.blogspot.com.es/2010/01/getting-relative-coordinates-from.html

http://www.ezzylearning.com/tutorial.aspx?tid=1763429

http://www.mkyong.com/android/android-custom-dialog-example/

http://www.mkyong.com/android/android-prompt-user-input-dialog-example/

http://www.josecgomez.com/2010/04/30/android-accessing-restfull-web-services-using-json/