



EPS

Escola Politècnica
Superior

Projecte/Treball Fi de Carrera

Estudi: Eng. Tècn. Informàtica de Sistemes. Pla 2001

Títol: ToIP: Implementació d'un identificador de trucades.

Document: Memòria

Alumne: Oriol Motger Albertí

Director/Tutor: Pere Vila

Departament: Arquitectura i Tecnologia de Computadors

Àrea: Arquitectura i Tecnologia de Computadors

Convocatòria (mes/any): 09/09

ÍNDIX DE CONTINGUTS

Índex de continguts	0
Índex de figures	3
1.INTRODUCCIÓ	5
1.1.Presentació	5
1.2.Objectius	6
1.3.Àmbit i abast	7
1.4.Temporització prevista	8
2.ANÀLISI DEL SISTEMA	9
2.1.Infraestructura de la que disposa l'empresa	9
2.2.El servidor de missatgeria instantània <i>Openfire</i>	11
2.2.1.Descripció	11
2.2.2.Com afecta al projecte	12
2.2.2.1.Autenticar usuaris mitjançant Directori Actiu	12
2.2.2.2. Execució de <i>plugins</i>	14
2.3.Els <i>plugins</i> (<i>PhonePlugin</i> i <i>LoginUnic</i>)	16
2.3.1.Descripció	16
2.3.2.Com afecta al projecte	16
2.3.2.1.El <i>PhonePlugin</i>	17
2.3.2.2.El <i>LoginUnicPlugin</i>	20
2.4.La centraleta CUCM	21
2.4.1.Descripció	21
2.4.2.Com afecta al projecte	22
3.DESENVOLUPAMENT DE L'APLICACIÓ	23
3.1.Metodologia de desenvolupament utilitzada	23
3.2.Anàlisi i disseny	24
3.2.1.Necessitats del client	24
3.2.2.Estudi de viabilitat	24
3.2.3.Anàlisi econòmic i tècnic	25
3.2.4.Requisits	26
3.2.4.1.Requisits tècnics	26
3.2.4.2.Requisits funcionals	26

3.3.Diagrames d'activitats	28
3.3.1.Procés de <i>login</i>	28
3.3.2.Procés d'arribada de trucades	28
3.3.3.Procés de mostrar un <i>popup</i> corresponent a una t. Externa	29
3.3.4. Procés de mostrar un <i>popup</i> corresponent a una t. Interna	31
3.3.5.Procés de client en espera	32
3.4.Diagrama de classes i fitxes CRC	33
3.4.1.La classe Client.cs	35
3.4.2.La classe Jabber.cs	35
3.4.3.La classe GestorHistorial.cs	36
3.4.4.La classe Registre.cs	37
3.4.5.La classe GestorPopups.cs	38
3.4.6.La classe GestorPopupTExt.cs	39
3.4.7.La classe GestorPopupTInt.cs	41
3.4.8.La classe Popup.cs	42
3.4.9.La classe TextPopup.cs	46
3.4.10.La classe GestorErrors.cs	47
3.4.11.La classe Error.cs	48
3.4.12.La classe GestorTrucades.cs	48
3.4.13.La classe GestorTExt.cs	49
3.4.14.La classe GestorTInt.cs	50
3.4.15.La classe Trucada.cs	50
3.5.Pantalles de l'aplicació	52
3.5.1.Els botons	52
3.5.2.Pantalla del <i>popup</i> d'una trucada externa	53
3.5.3.Pantalla del <i>popup</i> d'una trucada interna	55
3.5.4.Pantalla de l'historial de trucades	55
3.6.Com fer el popup i els seus elements	57
3.6.1.La finestra emergent senzilla	57
3.6.2.Afegint botons i imatges	60
3.6.3.Afegint texts	63
3.6.4.Resultat	65
4.PROVES, VERSIÓ FINAL I DISTRIBUCIÓ	66
4.1.Primeres proves	66

4.2.Dotant l'aplicació d'imatge corporativa	66
4.3.Paquetització de l'aplicació	67
4.4.Estudi del rendiment	68
4.5.Distribució de l'aplicació	68
4.6.Posta en marxa i aturada de l'aplicació	69
5.RESULTATS	70
5.1.Objectius assolits	70
5.2.Temporitzacio real	71
5.3.Conclusions	72
5.4.Treball futur	73
4.5.Coneixements adquirits	74
6.BIBIOGRAFIA I REFERÈNCIES	75
7.GLOSSARI	76
8.AGRAÏMENTS	77

1.INTRODUCCIÓ

Figura 1.1 Temporització prevista

2.ANÀLISI DEL SISTEMA

Figura 2.1 Infraestructura de l'empresa

Figura 2.2 Administració del servidor *Openfire*

Figura 2.3 Llistat d'usuaris del servidor *Openfire*

Figura 2.4 Detall d'un usuari del servidor *Openfire*

Figura 2.5 Llistat de *plugins* del servidor *Openfire*

Figura 2.6 Exemple del funcionament del servidor *Openfire*

Figura 2.7 Llistat d'usuaris del *plugin PhonePlugin*

Figura 2.8 Exemple de funcionament del *pluguin PhonePlugin*

Figura 2.9 Exemple de funcionament del *CallManager*

3.DESENVOLUPAMENT DE L'APLICACIÓ

Figura 3.1 Metodologia de desenvolupament

Figura 3.2 Diagrama d'activitats del procés de connexió

Figura 3.3 Diagrama d'activitats del procés d'arribada de trucades

Figura 3.4 Diagrama d'act. del procés d'arribada de trucada externa

Figura 3.5 Diagrama d'act. Del procés d'arribada de trucada interna

Figura 3.6 Diagrama d'activitats del procés de client en espera

Figura 3.7 Diagrama de classes de l'aplicació

Figura 3.8 Fitxa CRC de la classe Client.cs

Figura 3.9 Fitxa CRC de la classe GestorHistorial.cs

Figura 3.10 Detall del fitxer Hist.xml

Figura 3.11 Fitxa CRC de la classe Registre.cs

Figura 3.12 Fitxa CRC de la classe GestorPopups.cs

Figura 3.13 Fitxa CRC de la classe GestorPopupText.cs

Figura 3.14 Fitxa CRC de la classe GestorPopupTInt.cs

Figura 3.15 Fitxa CRC de la classe Popup.cs

Figura 3.16 Fitxa CRC de la classe TextPopup.cs

Figura 3.17 Fitxa CRC de la classe GestorErrors.cs

Figura 3.18 Detall del fitxer Err.xml

Figura 3.19	Fitxa CRC de la classe Error.cs
Figura 3.20	Fitxa CRC de la classe GestorTrucades.cs
Figura 3.21	Fitxa CRC de la classe GestorTExt.cs
Figura 3.22	Fitxa CRC de la classe GestorTInt.cs
Figura 3.23	Fitxa CRC de la classe Trucada.cs
Figura 3.24	Menú de l'aplicació
Figura 3.25	Exemple d'una imatge d'un botó
Figura 3.26	Imatge del botó utilitzat
Figura 3.27	Imatge d'un <i>popup</i> corresponent a una trucada externa
Figura 3.28	Imatge d'un <i>popup</i> corresponent a una trucada interna
Figura 3.29	<i>Form</i> de l'històric de trucades
Figura 3.30	Imatge base d'un <i>popup</i>
Figura 3.31	Estats del <i>popup</i>
Figura 3.32	Diagrama d'estats del <i>popup</i>
Figura 3.33	Imatge d'un botó
Figura 3.34	Imatge d'un botó en estat de repòs
Figura 3.35	Imatge d'un botó en estat seleccionat
Figura 3.36	Imatge d'un botó en estat premut
Figura 3.37	Esquema de <i>popup</i> senzill

4.PROVES, VERSIÓ FINAL I DISTRIBUCIÓ

Figura 4.1	Exemple de <i>Visual Basic Script</i> per instal·lar l'aplicació
------------	------------------------------------------------------------------

5.RESULTATS

Figura 5.1	Temporització real
------------	--------------------

1.INTRODUCCIÓ

1.1.Presentació

Aquest document pretén explicar detalladament com he realitzat el meu PFC. Primer de tot he de dir que el projecte en si consisteix en el desenvolupament d'una aplicació per a l'empresa Caixa Girona utilitzant específicament la tecnologia .NET.

Més endavant hi trobareu una sèrie d'apartats que descriuen com ho he fet per desenvolupar aquesta aplicació. Des d'un anàlisi de la infraestructura de la que disposa l'empresa fins a una detallada explicació de les diferents classes que componen el programa.

Al final de tot hi trobareu un apartat on defineixo a grans trets alguns conceptes que tenen a veure amb el projecte en general però que no cal que siguin explicats detalladament.

Trobareu també un apartat de bibliografia i referències per tal que pugueu contrastar i ampliar molta de la informació que conté aquesta memòria.

1.2.Objectius

El principal objectiu d'aquest projecte és desenvolupar una aplicació client en .NET que sigui capaç de processar les trucades IP entrants que pugui rebre un usuari al llarg de la seva jornada laboral.

Un cop assolit l'objectiu principal s'haurà de dotar de diverses funcionalitats l'aplicació.

La principal funcionalitat serà que cada vegada que l'usuari rebí una trucada IP es mostri una finestra emergent (*popup*) que contingui tota la informació disponible sobre la persona que realitza la trucada.

Aquesta informació ha de constar com a mínim de: la foto, el nom, el número de telèfon, el departament i el correu electrònic de la persona que realitza la trucada. Excepte el número de telèfon la resta d'informació només estarà disponible si prèviament s'ha pogut emmagatzemar a alguna base de dades (ja sigui de forma automàtica o manual).

Un cop mostrada la finestra emergent, l'usuari haurà de ser capaç de realitzar accions sobre la trucada que està rebent (rebutjar-la, posar-la en espera, desviar-la, etc...) .

Una altra funcionalitat amb la que s'haurà de dotar l'aplicació serà la gestió d'un historial diari de trucades. L'aplicació haurà de ser capaç d'emmagatzemar totes les trucades que rebí l'usuari així com la informació de les persones que realitzen aquestes trucades. També haurà de ser consultable i al mateix temps oferir altres funcionalitats com ara realitzar una trucada a una persona que prèviament hagi trucat a l'usuari, enviar-li un correu electrònic, consultar informació del seu departament, etc...

També s'haurà d'implementar un sistema per poder fer el tractament d'errors que es puguin produir durant el temps que l'aplicació estigui en funcionament.

Un cop finalitzada l'aplicació haurà de ser sotmesa a una bateria de proves per depurar-la i obtenir així una primera versió per ser distribuïda a tots els empleats de l'empresa.

1.3. Àmbit i abast

L'empresa per la qual es desenvoluparà aquest projecte és una empresa dedicada a la banca, fundada l'any 1940 per la Diputació de Girona i que disposa d'una plantilla d'uns 1300 empleats repartits entre unes 250 oficines disposades arreu del territori català i uns Serveis Centrals (SSCC) situats a la ciutat de Girona.

Pel que fa a l'abast aquest projecte es limita a:

- Realitzar una adquisició d'informació exhaustiva per tal de poder entendre bé el que es demana i com funciona l'empresa (estructura).
- Realitzar el desenvolupament d'una aplicació en .NET que assoleixi els objectius esmentats en l'apartat 1.2.
- Realitzar una bateria de proves sobre aquesta aplicació per tal d'ajustar-ne els paràmetres i depurar-la.
- Realitzar la distribució de l'aplicació a tots els empleats de l'empresa.

Aquesta aplicació s'haurà d'adaptar al sistema del qual disposi l'empresa. Per tant:

- Haurà de córrer sobre Windows XP.
- Haurà de consumir pocs recursos.
- Haurà d'interactuar amb els models de servidors de que disposi l'empresa.

Un cop distribuïda l'aplicació brindarà als seus usuaris tota una sèrie de funcionalitats destinades a millorar-ne el rendiment.

1.4. Temporització prevista

En aquest apartat podem veure com es va planificar el projecte. Per fer-ho mostro un gràfic on es pot veure el temps previst que invertiré en cada un dels apartats.

Number	Task	Start	End	Duration	2008	2009							
					December	January	February	March	April	May	June	July	
1	PRIMERA FASE	21/12/2008	13/3/2009	59									
1.1	ESTUDI DE LA INFRAESTRUCTURA	21/12/2008	13/3/2009	59									
1.2	ESTUDI DE L'ENTORN .NET	5/1/2009	12/1/2009	5									
1.3	ESTUDI DE LA PROGRAMACIÓ EN C#	5/1/2009	2/2/2009	20									
2	SEGONA FASE	9/2/2009	2/5/2009	60									
2.1	ESTUDI DELS REQUISITS	9/2/2009	28/2/2009	15									
2.2	DESENVOLUPAMENT DE L'APLICACIÓ	28/2/2009	30/4/2009	43									
2.3	PROVES	20/4/2009	2/5/2009	10									
3	TERCERA FASE	4/5/2009	3/7/2009	44									
3.1	ESTUDI RENDIMENTI	4/5/2009	20/6/2009	35									
3.2	DISTRIBUCIÓ	15/6/2009	3/7/2009	14									

Figura 1.1

S'ha dividit el projecte en tres fases.

La primera consisteix en entendre bé tota la infraestructura de la que disposa l'empresa així com entendre el funcionament de les eines que faré servir per desenvolupar l'aplicació.

La segona fase correspon al desenvolupament de l'aplicació i la realització de les proves corresponents.

Finalment la tercera fases consta de l'estudi de rendiment i la distribució de l'aplicació a tots els empleats de l'empresa.

El que hem vist aquí és el gràfic de Gantt ideal per a la realització del projecte. En finalitzar-lo afegiré a l'apartat de conclusions el gràfic real i podrem apreciar les diferències entre els dos gràfics. Aquestes diferències correspondran a problemes que molt probablement hem trobaré durant el desenvolupament de l'aplicació

2. ANÀLISI DEL SISTEMA

Aquest apartat pretén explicar quina infraestructura té actualment l'empresa i com afecta aquesta al desenvolupament de l'aplicació.

Primer de tot explicaré com estan disposats els diferents elements que formen aquesta infraestructura i més endavant entraré a explicar amb més detall cada un d'aquests elements, tant de forma general com la forma en que afecten a l'aplicació.

2.1. Infraestructura de la que disposa l'empresa

Per tal de desenvolupar el projecte cal conèixer a fons la infraestructura que te actualment muntada l'empresa.

En aquest apartat podrem veure a grans trets els components que formen aquesta infraestructura i en els apartats posteriors s'aprofundirà més en cada un dels components de forma individual.

Així dons l'empresa disposa de:

- Un **Directorí Actiu** de Microsoft. Necessari per controlar tots els usuaris del sistema, gestionar els recursos de xarxa, etc... funcionant amb Windows Server 2003.
- Un **sistema de gestió de telefonia IP** *Cisco Unified CallManager (CUCM)*. Necessari per processar el volum de trucades (tant internes com externes) que es produeix a través de l'entitat tant a nivell de Serveis Centrals com d'oficines. Funcionant amb una distribució *Linux* desenvolupada per CISCO i que només permet l'execució del CUCM.
- Un **servidor de missatgeria instantània** *Openfire*. Necessari per gestionar els usuaris de l'aplicació que desenvoluparé i per monitoritzar i gestionar tots els

events que es produeixen en els telèfons IP de l'entitat. Funcionant amb una distribució *Linux Redhat*.

A continuació podem veure un esquema que mostra la infraestructura de la que disposa l'empresa.

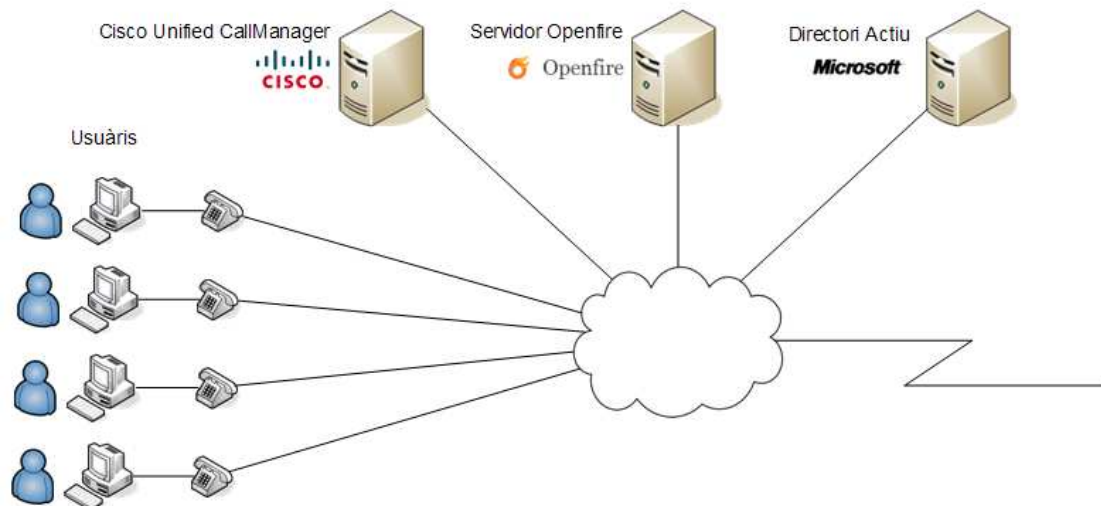


Figura 2.1

2.2.El servidor de missatgeria instantània Openfire

2.2.1.Descripció

El servidor *Openfire* és un servidor RTC (*Real Time Collaboration*) amb llicència *Open Source GPL* desenvolupat en Java per “*Ignite Realtime*”.

S'utilitza generalment per realitzar tasques de missatgeria instantània. Per fer-ho utilitza el protocol *Jabber*, anomenat també XMPP (*eXtensible Messaging and Presence Protocol*).

Les principals característiques d'aquest protocol son:

- És open. Tothom pot desenvolupar el seu software basat en *Jabber* si segueix les seves especificacions.
- És descentralitzat: Tothom pot fer funcionar el seu servidor *Jabber* i connectar-lo a altres servidors de la xarxa.
- És segur. Utilitza encriptació, autenticació, i està dotat de diverses característiques d'identitat per tal d'oferir la màxima confidencialitat i protecció als seus usuaris.
- És flexible. El seu sistema de missatgeria instantània es pot fer servir per transportar qualsevol tipus de dades.

Una de les característiques més importants d'aquest servidor es el fet de poder ampliar les seves funcionalitats mitjançant *plugins*. Els *plugins* poden ser descarregats de la mateixa web del servidor o be poden ser desenvolupats per l'usuari (en Java). Tant si s'obtenen d'una manera com de l'altre un cop tenim el *plugin* a la nostra disposició només és necessari copiar-lo a la carpeta corresponent del sistema de directoris del servidor per poder-lo posar en funcionament.

El servidor *Openfire* s'administra mitjançant un entorn web. Aquesta característica permet que alguns *plugins* més complexes puguin integrar en aquest entorn la seva pròpia consola d'administració.

Aquest tipus de consola ha d'estar desenvolupada en JSP i es mostra com una funcionalitat més dins el propi entorn d'administració del servidor *Openfire* tal i com es mostra a la figura 2.2.



Figura 2.2

2.2.2.Com afecta al projecte

El servidor *Openfire* és un dels components més importants del sistema. A grans trets les seves funcions són:

- Autenticar usuaris mitjançant el Directori Actiu.
- Execució de *plugins*.

2.2.2.1.Autenticar usuaris mitjançant Directori Actiu

Per autenticar els usuaris el servidor ho pot fer de dues maneres.

- La primera consisteix en la utilització d'una base de dades local (MySQL o pròpia del servidor) en la qual s'hi van afegir els usuaris que volen utilitzar el servei de missatgeria.

Per introduir els usuaris a la base de dades ho podem fer de dues maneres:

1. Mitjançant un usuari administrador que doni d'alta tots els usuaris que han de poder accedir al servidor.
2. Requerint a l'usuari les seves dades personals (usuari, contrasenya, correu electrònic, etc...) a través d'un formulari web per emmagatzemar-les posteriorment a la base de dades.

Un cop donat d'alta un usuari el servidor verificarà les seves credencials a la base de dades cada vegada que aquest usuari vulgui establir una nova connexió XMPP.

- L'altre manera d'autenticar usuaris es configurar el servidor de forma que obtingui les dades dels usuaris existents en el Directori Actiu de l'empresa mitjançant el protocol LDAP. Així només es podran connectar al servidor els empleats de l'entitat que estiguin donats d'alta al Directori Actiu.

Aquesta configuració ens ofereix també la possibilitat d'autenticar els usuaris mitjançant SSO (*Single Sign-On*). SSO utilitza les credencials d'inici de sessió de Windows de l'usuari per establir la connexió amb el servidor. Això facilita que l'usuari només s'hagi d'autenticar una sola vegada a l'entrar a la seva sessió de Windows.

Cal destacar que aquesta opció estalvia a l'administrador del servidor el manteniment dels usuaris ja que aquest queda lligat al manteniment del Directori Actiu.

Les següents figures mostren la consola d'administració de la base de dades d'usuaris i el contingut d'un usuari concret.

Total Users: 1.000 -- Showing 1-15, Sorted by Username -- Users per page: 15

Pages: [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 ... 67]

	Online	Username	Name	Created	Last Logout
1		1	...	May 28, 2009	
2		10	...	May 28, 2009	
3		100	...	May 28, 2009	

Figura 2.3

User Properties

Below is a summary of user properties.

User Properties	
Username:	101
Status:	(Offline)
Is Administrator?:	No
Name:	...
Email:	...
Registered:	May 28, 2009
Groups:	Interns, Personals

Figura 2.4

2.2.2.2. Execució de *plugins*












En el nostre cas disposem de dos *plugins* anomenats *PhonePlugin* i *LoginUnic*.

Aquests *plugins* es troben en el directori `\opt\openfire\plugins` (qualsevol *plugin* que es vulgui afegir s'ha de posar en aquest directori) i s'activen automàticament a l'iniciar el servidor.

A la figura 2.5 es pot apreciar la senzillesa que ofereix *Openfire* per gestionar-los.

Plugins

Plugins add new functionality to the server. The list of plugins currently installed is below. To download new plugins, please visit the [Available Plugins](#) page.

Plugins	Description	Version	Author	Restart	Delete
 Search	  Provides support for Jabber Search (XEP-0055)	1.4.3	Ryan Graham		
 loginunicplugin	Login Unic plugin	1.1.1	Tecocom		
 phoneplugin	CCM Phone plugin	1.1.1	Getronics		

Upload Plugin
Plugin files (.jar) can be uploaded directly by using the form below.

Figura 2.5

2.3.Els *plugins*

2.3.1.Descripció

Un *plugin* és una aplicació que interactua amb una aplicació principal per tal d'afegir-li alguna funcionalitat específica.

Els *plugins* són necessaris perquè desenvolupadors externs a l'aplicació principal hi afegixin funcionalitats sense necessitat de modificar-la. Els *plugins* interactuen amb l'aplicació principal mitjançant una API (*Application Programming Interface*).

A la web de *Ignite Realtime* hi podem trobar els *plugins* més populars que utilitza el servidor *Openfire* com ara l'*Asterisk-IM Openfire Plugin* que dota el servidor de la possibilitat d'interactuar amb *Asterisk*, el *Monitoring Service* que permet monitoritzar totes les conversacions i estadístiques que es produeixen a través del servidor, etc... No obstant això, els nostres *plugins* han estat dissenyats per cobrir les necessitats específiques de l'empresa.

2.3.2.Com afecta al projecte

Com hem pogut veure en l'apartat anterior, s'ha dotat el servidor *Openfire* amb dos *plugins* directament relacionats amb les funcionalitats de la telefonia IP.

- L'anomenat *PhonePlugin* és l'encarregat de monitoritzar tots els events dels telèfons IP associats als diferents llocs de treball.
- El *LoginUnicPlugin* permet la mobilitat dels empleats entre diferents llocs de treball.

2.3.2.1.El PhonePlugin

A grans trets, les funcionalitats d'aquest *plugin* són:

1. Establir un canal (*thread*) entre el servidor *Openfire* i el CUCM per cadascun dels usuaris.

Primer de tot el *plugin* obté les adreces MAC dels telèfon IP associats als terminals dels empleats. En el cas que un empleat no disposi de telèfon IP fa que no es pugui establir aquest canal de monitorització.

A continuació podem veure un exemple de com seria si el servidor *Openfire* només tingues quatre usuaris.

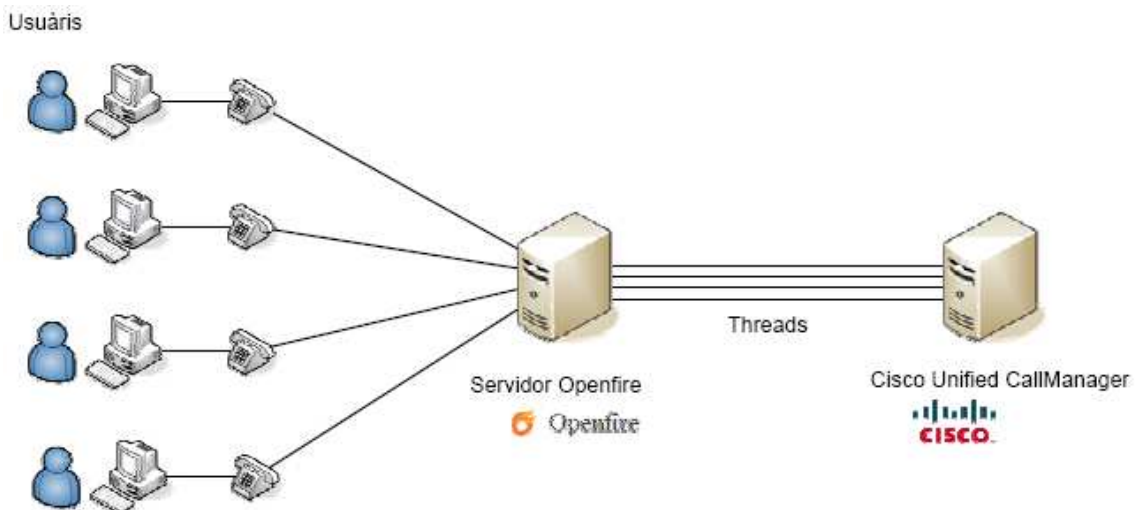


Figura 2.6

2. Donar d'alta l'usuari a una taula temporal pròpia del *plugin*. En el nostre cas aquesta taula conté: el número d'empleat, el nom, la MAC i el número del telèfon IP. Aquestes dades són les que utilitzarà l'aplicació que desenvoluparé.

L'ús d'aquesta taula permet augmentar la velocitat d'accés a l'hora d'obtenir les dades de l'usuari ja que ens estalvia el fet d'anar-les a buscar cada vegada al Directori Actiu i n'evita la seva sobrecàrrega.

Per emplenar aquesta taula ho podem fer de dues formes: o bé a través de la funcionalitat d'efectuar trucades que ens ofereix el propi *plugin*, o bé efectuant una càrrega inicial de tots els usuaris del Directori Actiu mitjançant una aplicació externa.

La següent figura mostra el contingut de la taula d'usuaris del *plugin*.

Currently Registered CCM Phone Users/Addresses



Username	Display Name	Device & Addresses
1082		
1482		
166		
2099		
300		
326		
364		
412		
436		
499		
500		

Figura 2.7

3. Monitoritzar i analitzar tots els events que li arribin a través dels diferents canals i efectuar les operacions oportunes segons el tipus d'event, ja sigui la realització d'una trucada mitjançant la funcionalitat del *plugin*, la identificació d'una trucada entrant, multi conferència, desviament de trucades, etc...

D'aquesta manera quan un usuari rep una trucada, el servidor *Openfire* se n'entera (avisat pel *Cisco Unified CallManager*) i el *plugin* s'encarrega d'anar a buscar la informació corresponent a l'usuari a la taula d'usuaris i enviar-la a través de missatgeria instantània XMPP al client. El que haurà de fer l'aplicació de l'identificador de trucades

que desenvoluparé és tractar aquesta informació i mostrar-la al client en forma de finestra emergent a la pantalla del PC.

A continuació mostro la figura 2.8 per tal d'entendre la importància de dotar el servidor *Openfire* amb el *PhonePlugin* i les seves funcionalitats.

La figura mostra el procés (a grans trets) de tot el que passa des de que un usuari A fa un clic2call a un usuari B i aquest rep la trucada.

Al pas 0 el servidor *Openfire* ja ha emplenat la taula d'usuaris del *plugin* amb la informació de l'usuari A extreta dl D.A. i l'usuari B disposa de l'aplicació client Identificador de Trucades en funcionament (connexió XMPP client-servidor establerta).

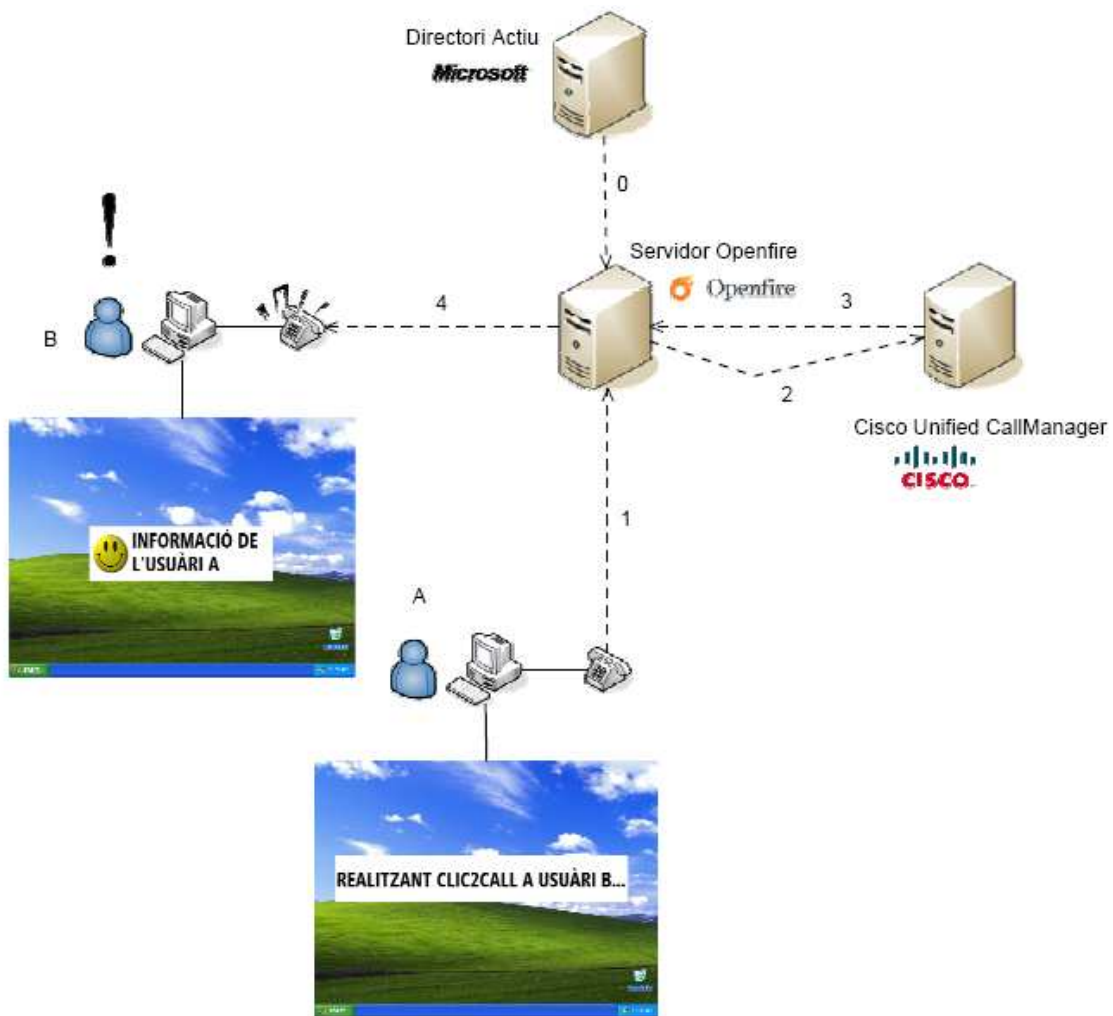


Figura 2.8

Al pas 1 l'usuari A realitza una trucada a l'usuari B (funcionalitat que ofereix el *PhonePlugin*).

Al pas 2 el servidor *Openfire* informa al *CallManager* a través del *PhonePlugin* que l'usuari A vol realitzar una trucada a l'usuari B. S'estableix el canal (*thread*) en el cas que no existeixi.

Al pas 3 el *CallManager* envia la trucada a l'usuari B i avisa a l'*Openfire* que l'usuari A està trucant a l'usuari B.

Al pas 4 l'*Openfire* mitjançant el *PhonePlugin* envia a través d'XMPP la informació emmagatzemada a la taula d'usuaris corresponent a l'usuari A.

Finalment a l'usuari B se li obre una finestra emergent mostrant aquesta informació (aquesta part és la que haure de desenvolupar jo).

2.3.2.2.El LoginUnicPlugin

L'altre *plugin* del que disposem s'anomena *loginUnicPlugin*. La funció d'aquest *plugin* consisteix en fer que qualsevol usuari que faci *login* a una sessió de Windows de qualsevol PC de l'empresa al telèfon connectat a aquest PC se li assigna automàticament el número de telèfon d'aquest empleat.

Així no importa a quin PC vagi un empleat a treballar ja que sempre s'emportarà el seu número de telèfon amb ell.

Aquest *plugin* en principi no afecta al desenvolupament de l'identificador de trucades.

2.4.La centraleta CUCM

2.4.1.Descripció

L'empresa per la que es desenvoluparà el projecte disposa d'una potent centraleta capaç de processar tot el tràfic de trucades IP. Es tracta d'una centraleta *Cisco Unified Communications Manager* anomenada també *CallManager* o CUCM.

El *CallManager* és capaç d'estendre les funcions i capacitats de la telefonia empresarial als dispositius de telefonia per paquets de la xarxa com ara telèfons IP, aplicacions multimèdia, portes d'enllaç de veu sobre IP, etc...

La centraleta brinda als usuaris amb les característiques de la telefonia convencional fusionades amb una sèrie d'avançades funcionalitats com: mobilitat, presència, preferència i conferència multimèdia.

A part de les funcionalitats el *CallManager* ofereix una sèrie de característiques que les empreses han de tenir en compte. Es tracta de que és una centraleta escalable, distribuïble i disponible.

D'aquesta manera es poden agrupar diferents centraletes *CallManager* i administrar-les com si fossin una sola centraleta dins de la xarxa IP. Això proporciona una escalabilitat de 1 a 30.000 telèfons per grup, equilibri de càrrega i redundància.

Una altra característica important que ofereixen aquest tipus de centraletes és l'anomenat control d'admissió de trucades (CAC) que s'encarrega de mantenir la qualitat del servei (QoS) de veu. Per fer-ho desvia totes les trucades cap a la xarxa PSTN (xarxa telefònica convencional) quan no es disposa de la suficient amplada de banda necessària per transmetre-les garantint d'aquesta manera una bona qualitat en les trucades que es realitzin.

A continuació podem veure una possible solució de la gestió telefònica d'una empresa mitjançant un *CallManager*.

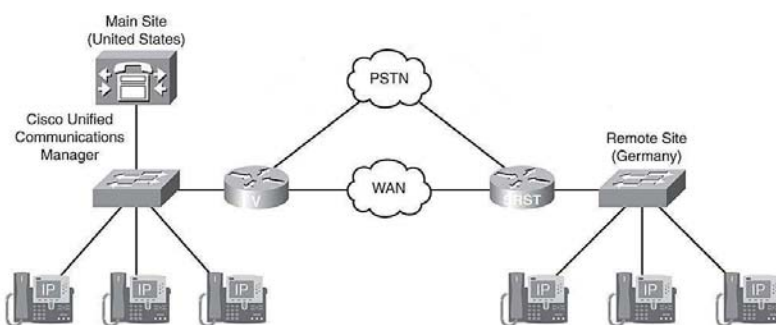


Figura 2.9

2.4.2.Com afecta el projecte

La centralita CUCM és una part important del projecte ja que s'encarrega de gestionar totes les trucades, tant les que es fan com les que es reben, tant les internes, com les externes.

En el nostre cas el CUCM s'encarrega d'avisar al *plugin PhonePlugin* de totes les trucades que s'estan realitzant en cada instant de temps a través dels canals que ha obert el *plugin*. D'aquesta manera l'*Openfire* sap en quin moment i a quin usuari li ha d'enviar els missatges instantanis via XMPP.

3. DESENVOLUPAMENT DE L'APLICACIÓ

Aquest apartat pretén explicar com ho he fet per desenvolupar l'aplicació en .NET anomenada Identificador de Trucades. S'hi pot trobar des de la metodologia que he fet servir fins a la funció de qualsevol de les classes que componen el projecte passant per les descripcions de les diferents interfícies que permeten a l'usuari interactuar amb aquesta aplicació.

3.1. Metodologia de desenvolupament utilitzada

Pel que fa a la metodologia emprada per desenvolupar l'aplicació podria dir que he seguit una metodologia amb força paral·lelismes a la Mètrica 3 però ni molt menys tant estricta. Tot i que si es dona un cop d'ull a la forma com funciona aquesta metodologia (veure apartat de bibliografia i referències web) si que puc dir que la manera com he desenvolupat l'aplicació s'hi assembla.

A continuació podem veure com seria exactament la metodologia de desenvolupament emprada.

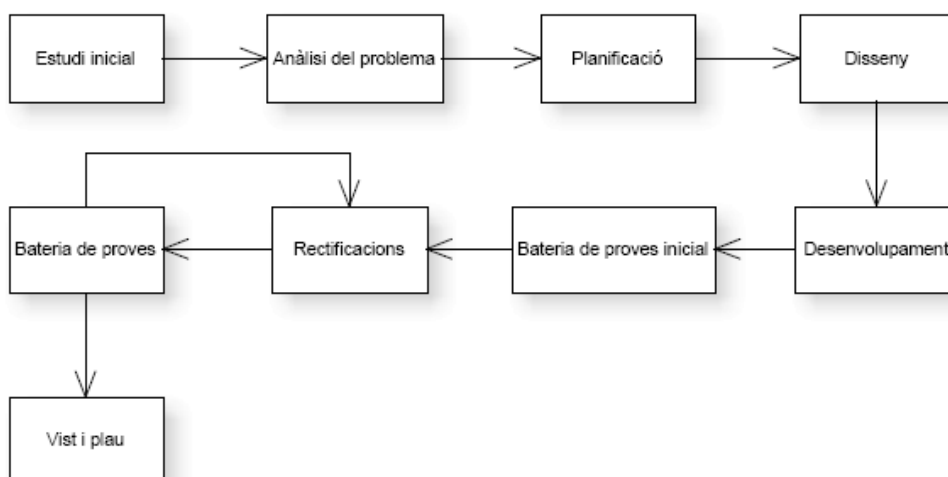


Figura 3.1

3.2.Anàlisi i disseny

3.2.1.Necessitats de l'usuari

Un empleat pot rebre al llarg del dia un volum considerable de trucades de gent tant externa com interna a l'empresa. Aquestes trucades poden tractar sobre temes ben diferents les unes de les altres. Si a un usuari que es passa el dia enganxat al telèfon li mostres per pantalla les dades de la persona que truca, aquest disposa d'un breu període de temps per tal de posar-se en el paper necessari per afrontar la trucada.

D'altra banda un usuari també necessita poder accedir ràpidament a l'historial de trucades rebudes i des d'allà disposar de diverses opcions per posar-se en contacte amb una persona que l'ha trucat fa una estona.

El fet de dotar als empleats amb les característiques descrites en aquest apartat farà que millorin el seu rendiment ja que per una banda li posarem diverses funcionalitats interessants a un clic de ratolí i per altra li facilitarem en temps real tota la informació sobre la persona que l'està trucant si es disposa d'aquesta informació en alguna banda.

3.2.2.Estudi de viabilitat

La viabilitat del projecte s'haurà de determinar un cop desenvolupada l'aplicació mitjançant una bateria de proves i ve condicionada per la capacitat de processament dels ordinadors que la utilitzaran.

L'empresa disposa de bastants models de PC que ha anat adquirint al llarg del temps i d'un ampli ventall d'aplicacions. Això provocarà gairebé amb tota seguretat que determinats empleats facin servir moltes aplicacions a la vegada i que aquestes aplicacions absorbeixin gran part dels recursos de la seva màquina per la qual cosa podria no ser viable afegir noves aplicacions.

Cal afegir que això és només una suposició i que hauré d'esperar a realitzar la bateria de proves per veure si realment es compleix o no.

La bateria de proves consistirà en distribuir l'aplicació a un PC i veure com reacciona quan la posem en funcionament. Posteriorment es distribuirà a una sèrie de PCs de persones específiques que actuaran de testers i finalment amb el vist i plau d'aquestes persones es distribuirà una aplicació a una oficina per veure com reacciona en un entorn real.

Si al final passa tots els tests es pot donar la possibilitat de que es decideixi distribuir a tots els terminals de l'empresa. En canvi si no supera alguna de les fases esmentades anteriorment hauré de mirar de corregir els possibles errors o fins hi tot optimitzar algun dels processos per fer el projecte viable.

3.2.3. Anàlisi econòmic i tècnic

Econòmicament el projecte és beneficiós per a l'empresa ja que al ser desenvolupat per un estudiant com a PFC el cost de l'aplicació es zero. En canvi si l'empresa decidís adquirir un producte del mercat molt probablement el cost es dispararia.

Actualment no he estat capaç de trobar cap aplicació de mercat que s'assembli a la que desenvoluparé jo. D'altra banda tots sabem que fer desenvolupar una aplicació per una empresa externa costa molts de diners. És per això que més endavant es pot veure un possible pressupost del que costaria desenvolupar aquesta aplicació.

Tot i això, abans de posar l'aplicació en marxa es realitzaran una sèrie de proves per garantir-ne el correcte funcionament (tal i com s'ha explicat en l'apartat anterior) i si no es supera alguna d'aquestes proves es procedirà a la modificació de l'aplicació.

3.2.4.Requisits

3.2.4.1.Requisits tècnics

Per garantir el correcte funcionament de l'aplicació cada usuari ha de disposar de:

- Un PC funcionant amb Windows XP i el *framework* 2.0 de Microsoft.
- Un telèfon IP connectat entre el PC i la xarxa de l'empresa per tal de poder realitzar i rebre trucades de veu IP.
- Una sessió de presència inicialitzada en el PC mitjançant les seves credencials per tal de que el servidor de missatgeria sàpiga a on ha d'enviar els missatges corresponents a les trucades destinades a l'usuari.

La infraestructura de l'empresa ha de disposar de:

- Una centraleta de telefonia IP capaç de gestionar totes les trucades que es realitzin ja siguin internes o externes.
- Un servidor de missatgeria instantània capaç d'enviar missatges referents a les trucades que rebin els clients.
- Un servidor web que contingui les fotografies de format carnet de tots els empleats de l'empresa.

Altres requisits tècnics importants:

Per tal de fer funcionar l'aplicació no ha de ser necessari que un empleat introdueixi les seves credencials ja que aquesta s'ha d'autenticar automàticament.

3.2.4.2.Requisits funcionals

- L'aplicació client es nodrirà d'un fitxer de configuració per tal d'ajustar els seus paràmetres (tant els de connexió com els propis de l'aplicació). Per tant qualsevol canvi que es vulgui fer a nivell de la infraestructura necessària per al funcionament de l'aplicació haurà de quedar reflectit al fitxer de configuració.
- Per tal que el servidor de missatgeria instantània pugui enviar informació corporativa sobre les persones que realitzen trucades s'haurà d'haver extret prèviament aquesta informació de la base de dades on estigui emmagatzemada

(per exemple el directori actiu AD) i s'haurà d'emmagatzemar en una base de dades del servidor. D'aquesta manera cada vegada que el servidor necessiti enviar aquesta informació a un client ja la tindrà disponible i millorarà l'eficiència.

- En el cas que no es disposi de la informació sobre la persona que realitza una trucada l'aplicació accedirà als contactes de l'*Outlook* de la persona que rep la trucada amb la intenció de poder obtenir aquesta informació.
- En el cas que no es pugui obtenir la informació de la persona que truca ni del servidor de missatgeria instantània ni de la llista de contactes de l'*Outlook* la finestra emergent es mostrarà només amb el número de telèfon des del qual s'està realitzant la trucada.
- Les fotografies dels empleats residiran en un servidor web i no en el servidor de missatgeria instantània.
- En el cas de que una fotografia no estigui disponible l'aplicació mostrarà una fotografia estàndard amb un dibuix d'una persona.

3.3. Diagrames d'activitats

Els diagrames d'activitats mostren el flux d'activitats involucrades en un procés.

3.3.1. Procés de *login*

Tot seguit podem veure el diagrama d'activitats que mostra el procés de connexió de l'identificador de trucades. Aquest procés inclou des de la posta en marxa de l'aplicació fins que l'aplicació (client) queda connectada i en espera. També inclou la possibilitat que durant el procés de connexió es produeixi algun error.

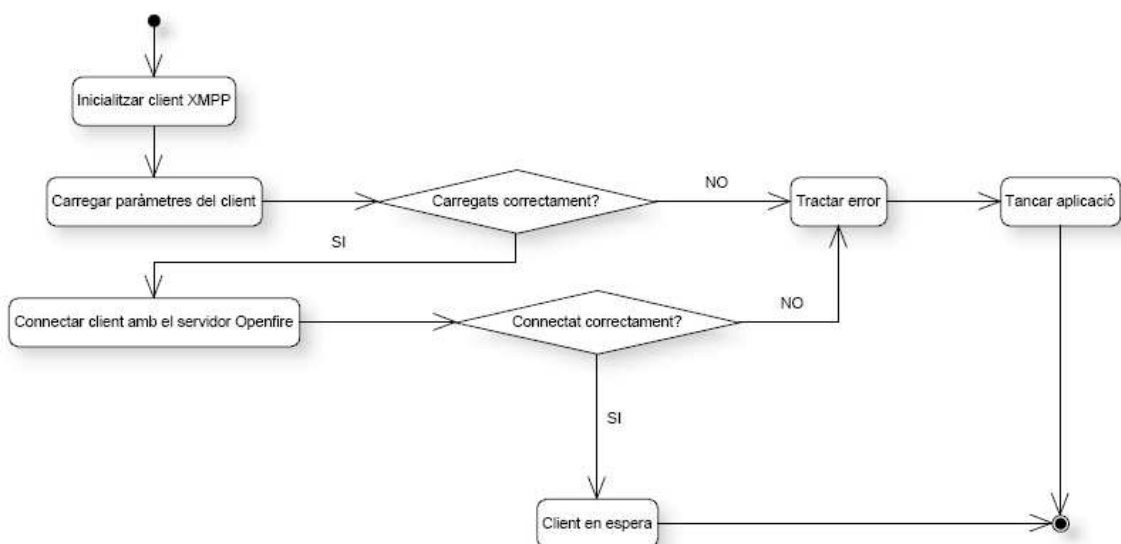


Figura 3.2

3.3.2. Procés d'arribada de trucades

A continuació podem veure els passos que segueix el client (que està en espera) quan li arriba una trucada. Les trucades poden ser externes a l'empresa o internes. Per tant el client reaccionarà de forma diferent si es dona un cas o un altre.

En el cas de que la trucada sigui interna, ja disposarem de totes les dades necessàries per mostrar en la finestra emergent (*popup*). Per tant aquesta es mostrarà al mateix temps que s'introduiran les dades a l'historial de trucades.

En canvi si la trucada és externa no es disposaran de totes les dades per mostrar a la finestra emergent (només del telèfon). En aquest cas s'hauran d'anar a buscar les dades a la llista de contactes de l'*Outlook*.

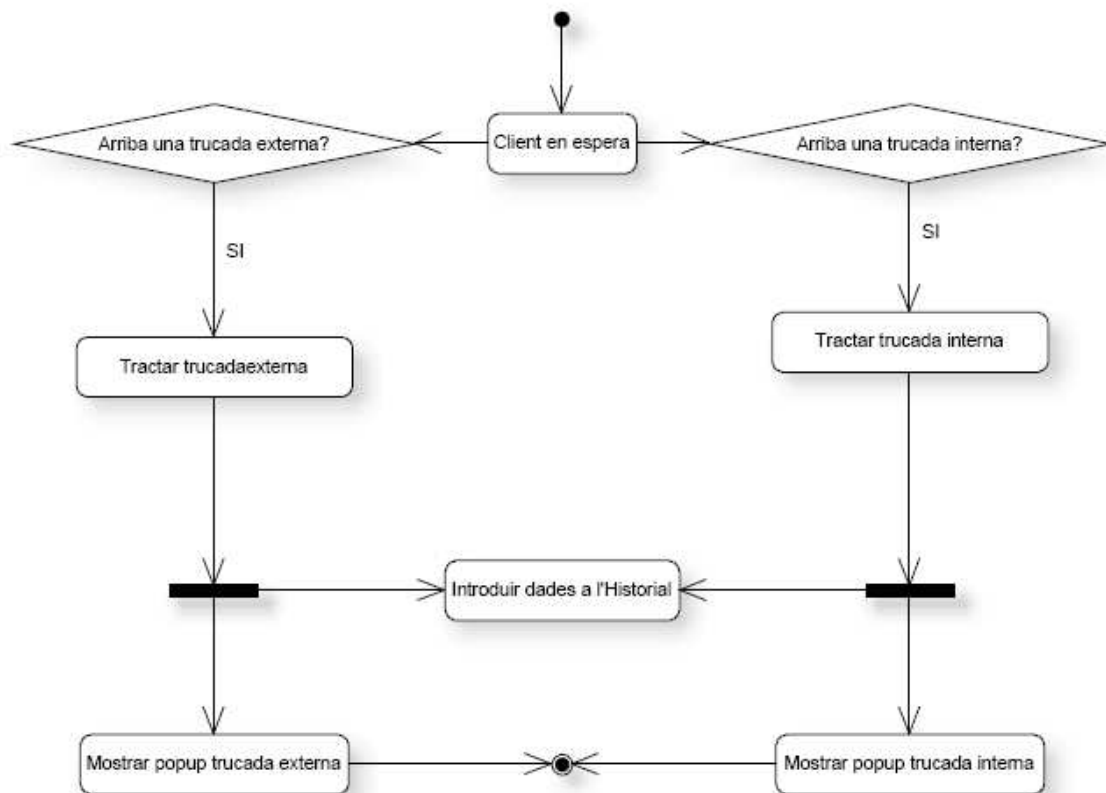


Figura 3.3

3.3.3. Procés de mostrar un *popup* corresponent a una trucada externa

El següent diagrama d'activitats mostra el flux d'activitats corresponent al cas que es mostri el *popup* d'una trucada externa.

Aquí es pot apreciar com un cop el *popup* s'està mostrant l'usuari pot realitzar diverses accions comunes als dos tipus de *popups* (trucades externes i trucades internes) com ara

tancar el *popup* immediatament, visualitzar l'història de trucades o fer clic sobre el mail de la persona que truca si se'n disposa (s'ha trobat el contacte a l'*Outlook*) per tal d'enviar-li un correu electrònic.

També es pot veure l'acció pròpia del *popup* d'una trucada externa. Es tracta d'afegir un nou contacte a l'*Outlook* (corresponent a la persona que està realitzant la trucada).

Cal destacar el fet que si l'usuari no realitza cap acció el *popup* desapareixerà automàticament al cap d'una estona. Per altra banda si l'usuari decideix realitzar qualsevol acció (excepte la de tancar el *popup*) el *popup* també desapareixerà.

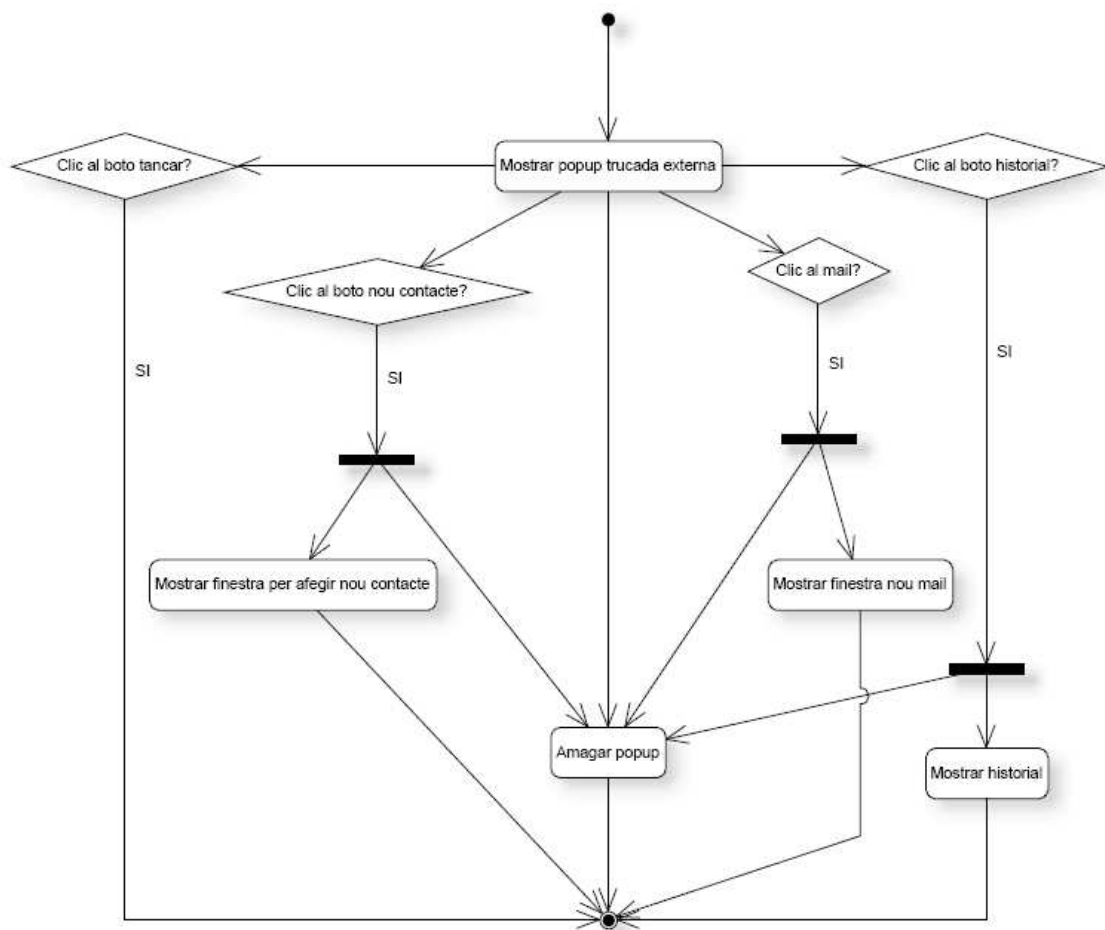


Figura 3.4

3.3.4. Procés de mostrar un *popup* corresponent a una trucada interna

El diagrama que es mostra a continuació permet veure totes les accions que pot realitzar un usuari en el cas que hagi entrat una trucada interna i s'estigui mostrant el *popup* corresponent a les trucades internes.

Cal destacar que en aquest cas com que es disposen de totes les dades de la persona que realitza la trucada sempre es podrà realitzar l'acció enviar correu electrònic i visualitzar la informació del departament al que pertany la persona que truca a la intranet de l'empresa.

L'usuari també disposa de les accions comunes com ara tancar el *popup* o veure l'historial de trucades i com en el cas anterior si l'usuari realitza qualsevol acció (excepte la de tancar el *popup*) el *popup* s'amagarà automàticament un cop transcorregut el temps durant el qual s'ha de mostrar per pantalla.

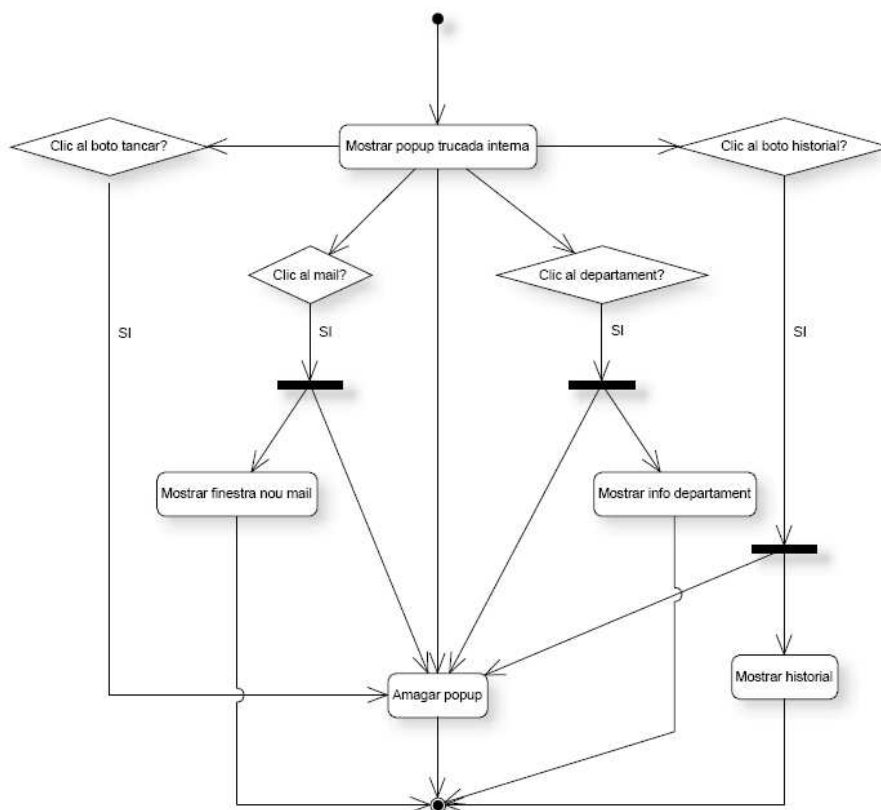


Figura 3.5

3.3.5. Procés de client en espera

Seguidament es mostra el diagrama d'activitats de les accions que pot realitzar l'usuari mentre el client està connectat i a l'espera. Aquestes accions son dues: o bé fer clic al menú sortir, la qual cosa fa que es surti de l'aplicació o bé fer clic al menú historial.

En el cas de fer clic al menú historial l'aplicació mostrarà per pantalla l'historial de trucades on l'usuari podrà realitzar una sèrie d'accions: fer clic a un número de telèfon per tal de realitzar una trucada mitjançant el clic2call al número de telèfon clicat, fer clic a un departament per tal de visualitzar la informació continguda a l'intranet de l'empresa sobre el departament clicat o bé fer clic sobre una adreça de correu electrònic per tal d'enviar un nou correu mitjançant l'*Outlook*.

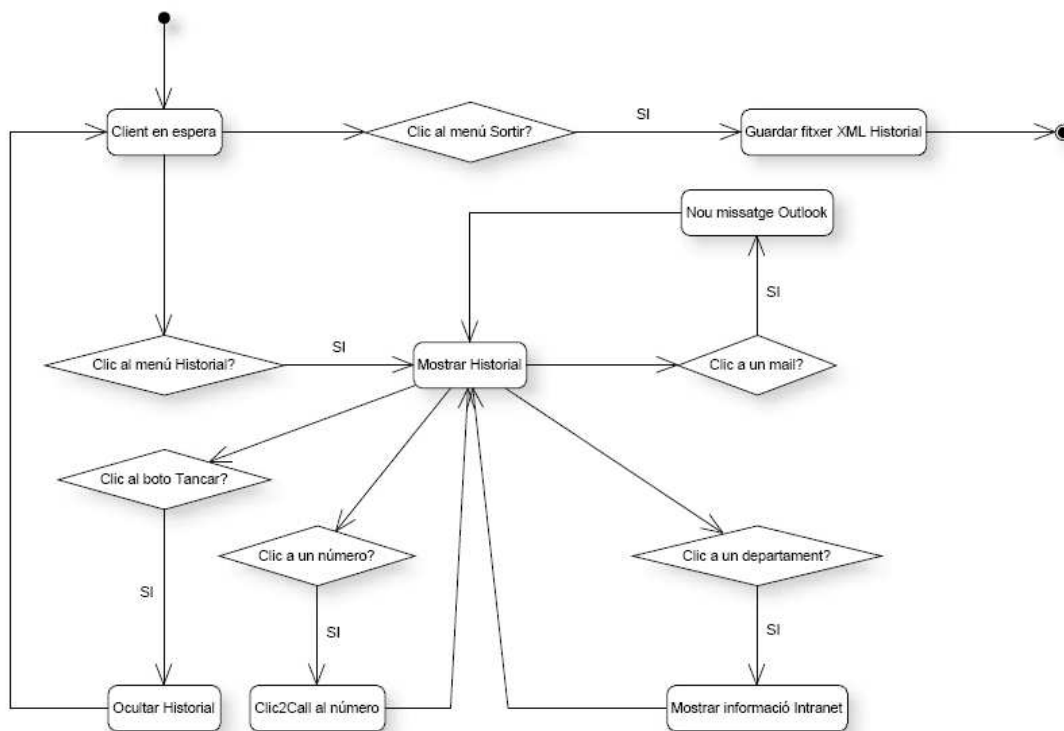


Figura 3.6

3.4. Diagrama de classes i fitxes CRC

El diagrama de classes mostra les classes del projecte i la manera com estan relacionades entre elles.

Habitualment dins de cada classe es mostren els seus atributs i les seves funcions però en el meu cas, com que les classes contenen molts atributs i moltes funcions he decidit mostrar només les relacions entre classes mitjançant un diagrama de classes i mostrar els atributs i funcions de cada classe mitjançant fitxes CRC.

Per tal que el codi quedi el màxim ordenat possible he decidit utilitzar com més classes millor. D'aquesta manera a part de quedar ordenat també queda més repartit i es molt més senzill anar a buscar possibles errors a l'hora de depurar-lo ja que crec que és millor tenir quatre classes amb cinc funcions cada una que no tenir una classe amb vint funcions.

Cal destacar també que totes les classes del projecte es divideixen en 4 grups o nivells. El primer nivell conté la classe principal anomenada Client.cs. La classe client és l'encarregada de fer la gestió de les classes dels nivells 2 i 3.

El nivell 2 conté la classe Jabber.cs. Aquesta classe és molt important ja que disposa dels mètodes necessaris per connectar el client al servidor *Openfire* i rebre els missatges corresponents a cada trucada. A més és la única classe que no he desenvolupat jo.

El nivell 3 conté totes les classes gestores com ara GestorHistorial.cs, GestorPopups.cs, GestorPopupText.cs, etc... Com el seu nom indica aquestes classes s'encarreguen de gestionar els diferents objectes que conté el projecte.

Finalment el nivell 4 conté les classes que representen els objectes com ara Registre.cs, Popup.cs, Trucada.cs, etc... Aquestes classes contenen els atributs dels objectes que representen i les funcions necessàries per llegir-los o modificar-los. També consten d'algunes accions específiques que poden realitzar certs objectes en el cas que la seva

classe gestora ho requereixi ja sigui de forma automàtica o sota petició directe de l'usuari de l'aplicació.

A continuació podem veure les classes que formen el projecte i la forma en la que estan estructurades i seguidament les corresponents fitxes CRC.

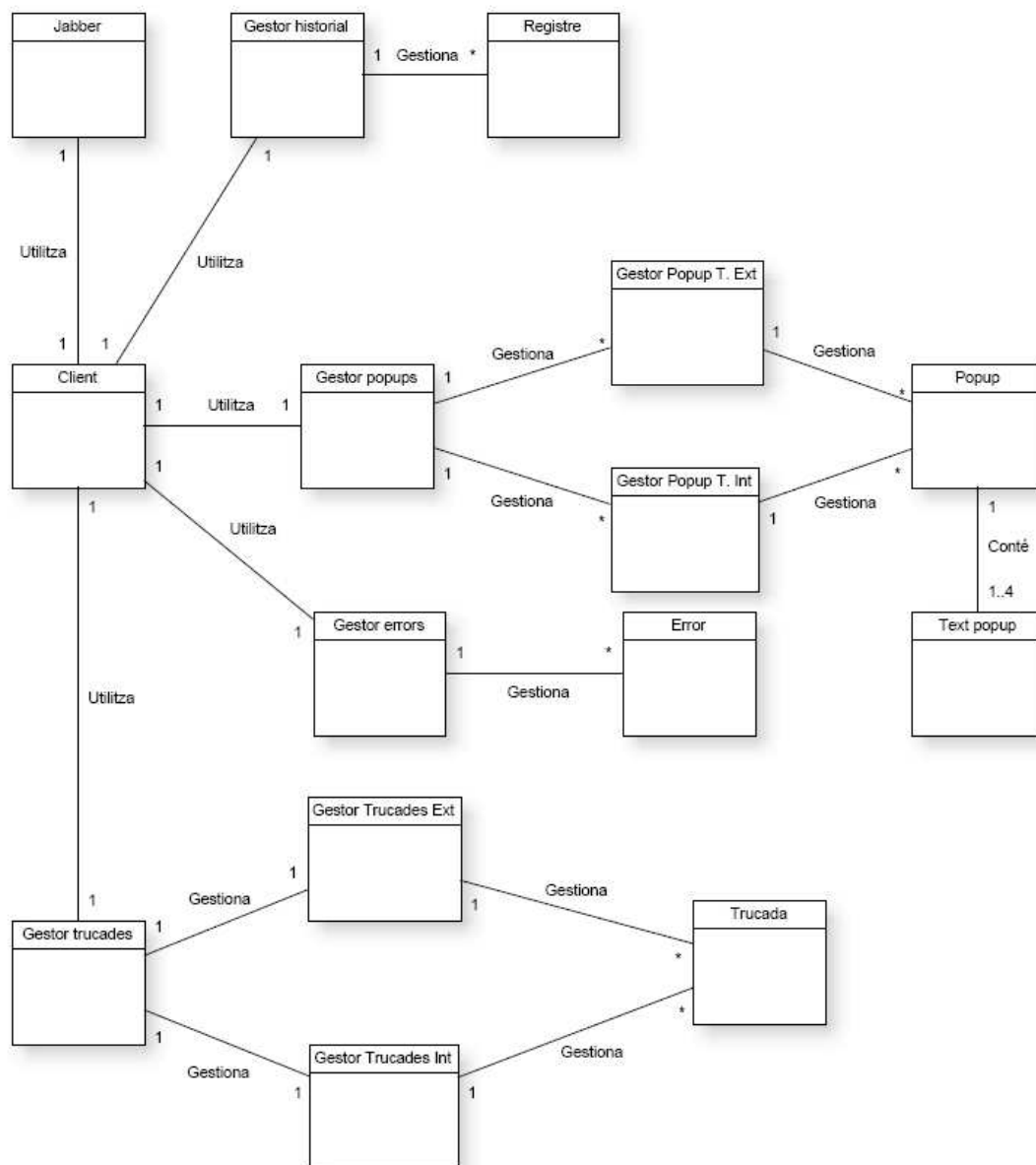


Figura 3.7

3.4.1.La classe Client.cs

Al diagrama de classes podem apreciar com tot gira al voltant d'aquesta classe Client.cs. És la classe principal del projecte.

Classe	Client
Descripció de la classe	Classe encarregada de gestionar la connexió amb el servidor i els missatges que arriben via XMPP.
Responsabilitats	Col·laboracions
-Establir connexió amb el servidor Openfire. -Tractar els missatges XMPP que arribin. -Gestionar el menú de l'aplicació	GestorHistorial GestorPopups GestorErrors GestorTrucades
Client #GH: GestorHistorial #GP: GestorPopups #GE: GestorErrors #GT: GestorTrucades //Funció per inicialitzar el client +Client() //Funció per obtenir els paràmetres de connexió i realitzar la connexió amb el servidor //Openfire #Establir_Connexió() //Funció per processar l'arribada de missatges XMPP #Si_Arriba_Missatge_XMPP() //Funció que defineix el menú de l'aplicació #Crea_Menu() //Funció del botó sortir del menú #Sortir_clic() //Funció del botó historial del menú #Historial_clic()	

Figura 3.8

3.4.2.La classe Jabber.cs

Aquesta classe no l'he desenvolupat jo. Conté totes les funcions necessàries per fer la connexió entre l'aplicació i el servidor *Openfire* per tal de poder rebre missatges via XMPP sobre les trucades que rep el client.

3.4.3. La classe GestorHistorial.cs

La classe GestorHistorial.cs s'encarrega de gestionar l'historial de trucades. Les seves funcions són: editar el fitxer xml on s'emmagatzemen totes les trucades i fer-ne el manteniment, mostrar i ocultar l'historial de trucades i nodrir l'historial amb el contingut del fitxer xml.

Classe	GestorHistorial
Descripció de la classe	Gestiona el fitxer XML que conté els registres de l'historial de trucades. També gestiona la pantalla de l'historial i li dona funcionalitats.
Responsabilitats	Col·laboracions
S'encarrega de gestionar l'historial de trucades.	Registre
GestorHistorial //Boto per tancar el Form #Tancar: Button //Columnes del DataGridView #Hora: DataGridViewTextBoxColumn #Nom: DataGridViewTextBoxColumn #Telefon: DataGridViewLinkColumn #Departament: DataGridViewLinkColumn #Correu: DataGridViewLinkColumn //estructura on s'emmagatzemarà el contingut del fitxer XML #Graella: DataGridView //Fitxer XML que conté els registres de l'historial de trucades #doc_xml: XmlDocument //Constructor del gestor +GestorHistorial() //Crea el Form de l'historial #InitializeComponent() //Afegeix una trucada al fitxer XML +Afegir_Trucada(nom:string, num:string, dept:string, mail:string) //Funció que carrega el fitxer XML i mostra l'historial +Mostrar_Historial() //Funció del boto tancar +Tancar_Historial(sender:object, e:EventArgs) //Funció per tractar els events de la Graella #ClicAContingutGraella(sender:object, e:DataGridViewCellEventArgs) //Funció per fer el manteniment de l'Historial +Mantenir_Historial(nTrucades:int)	

Figura 3.9

A continuació podem veure l'estructura del fitxer xml que conté l'històric de trucades. A la quarta trucada podem veure com tots els camps excepte el número de telèfon estan buits. Això és degut a que es tracta d'una trucada externa i no s'ha trobat la informació de la persona que ha realitzat la trucada enlloc.

```
<?xml version="1.0" encoding="utf-8" ?>
<Trucades>
  <Trucada Data="02/04/2009 12:31:56" Nom="aaa bbb ccc" Num="1234" Departament="333" Correu="abc@caixagirona.es" />
  <Trucada Data="02/04/2009 12:36:09" Nom="ddd eee fff" Num="4567" Departament="333" Correu="def@caixagirona.es" />
  <Trucada Data="02/04/2009 12:36:41" Nom="ggg hhh iii" Num="2223" Departament="444" Correu="ghi@caixagirona.es" />
  <Trucada Data="02/04/2009 12:38:31" Nom="" Num="12345678" Departament="" Correu="" />
</Trucades>
```

Figura 3.10

3.4.4. La classe Registre.cs

Aquesta classe conté l'estructura d'un registre de l'històric de trucades i les funcions necessàries per crear-lo i modificar-lo.

En un principi s'havia pensat que aquesta classe fos l'encarregada d'afegir un registre al fitxer XML però per motius d'eficiència s'ha acabat fent a la classe GestorHistorial.cs.

S'ha decidit així perquè si es feia d'inserció al fitxer des de la classe Registre.cs implicava carregar i guardar el fitxer XML cada vegada que es volia inserir una trucada. En canvi fent-ho a la classe GestorHistorial.cs només s'ha de carregar el fitxer una vegada quan s'inicialitza l'aplicació. Fent-ho així tampoc m'estalvio guardar el fitxer cada vegada que es rep una trucada ja que si no ho fes així i l'aplicació es tanqués degut a un error del sistema es perdria la informació corresponent a totes les trucades rebudes.

Classe	Registre
Descripció de la classe	Conté les funcions necessàries per inserir i modificar registres al fitxer XML. (al final no s'utilitza per temes d'eficiència)
Responsabilitats	Col·laboracions
Gestiona el fitxer XML on s'emmagatzemen les trucades rebudes	
Registre	
#data: string	
#nom: string	
#num: string	

```

#dept:string
#mail:string
/Constructor sense paràmetres
+Registre()
//Constructor amb paràmetres
+Registre(nomPT:string , numPT:string, deptPT:string, mailPT:string)
//Funcions per obtenir/modificar els valors del Registre
+ObtenirData:string
+ObtenirNom:string
+ObtenirNum:string
+ObtenirDept:string
+ObtenirMail:string

```

Figura 3.11

3.4.5. La classe GestorPopups.cs

Aquesta classe s'encarrega de gestionar els *popups* tant de trucades externes com internes. Conté les funcions necessàries per mostrar els *popups*. Aquestes funcions creen i emplenen els *popups* amb les dades de les quals es disposa corresponents a la persona que realitza la trucada. Cal destacar també que aquesta classe és la que s'encarrega d'interactuar amb el servidor web que conté les fotografies dels empleats per tal d'obtenir la foto corresponent a la persona que realitza la trucada i així mostrar-la a la finestra emergent corresponent a les trucades internes de l'empresa.

Classe	GestorPopups
Descripció de la classe	Classe per gestionar els diferents tipus de <i>popup</i> (el de trucades externes i el de trucades internes).
Responsabilitats	Col·laboracions
Crea i mostra PopupText i PopupTInt	PopupTExt PopupTInt
GestorPopups	
//Constructor sense paràmetres	
+GestorPopups()	
//Funció que crea i mostra un PopupTInt (trucada interna)	
+MostraPopupTInt(nom:string, num:string, dept:string, mail:string, TApareixer:int, TMantenir:int, TDesapareixer:int)	
//Funció que crea i mostra un PopupTExt (trucada externa)	
+MostraPopupTExt(nom:string, num:string, dept:string, mail:string, TApareixer:int, TMantenir:int, TDesapareixer:int)	

Figura 3.12

3.4.6.La classe GestorPopupTExt.cs

Aquesta classe s'encarrega de fer tota la gestió dels *popups* corresponents a les diferents trucades externes (de les que probablement no es disposi de més informació que el numero de telèfon de la persona que realitza la trucada) que puguin rebre els empleats. Per fer-ho disposa de tots els mètodes necessaris per crear, modificar, mostrar per pantalla i ocultar aquestes finestres emergents així com les funcions encarregades de gestionar els events de ratolí que es puguin realitzar sobre aquestes.

Classe	PopupTExt
Descripció de la classe	Classe que gestiona els PopupTExt (Popups dedicats a trucades externes)
Responsabilitats	Col·laboracions
Conté els mètodes per crear, mostrar, amagar i modificar un PopupTExt. Afegeix la funcionalitat d'afegir un contacte a l'Outlook.	Popup
PopupTExt	
#p: Popup	
//Rectangles del contingut dels texts del popup	
#RectangleNom: Rectangle	
#RectangleNum: Rectangle	
#RectangleDept: Rectangle	
#RectangleMail: Rectangle	
//Events del Popup	
#nEventsMostrar: Int	
#nEventsAmagar: Int	
#nEventsVisible: Int	
#nIncrementMostrar: Int	
#nIncrementAmagar: Int	
//Events de ratoli	
#bSiRatoliSobrePopup: Bool	
#bSiRatoliSobreTanca: Bool	
#bSiRatoliSobreNouCont: Bool	
#bSiRatoliSobreNom: Bool	
#bSiRatoliSobreNum: Bool	
#bSiRatoliSobreMail: Bool	
#bSiRatoliSobreDept: Bool	
#bSiRatoliSobreHistorial: Bool	
#bSiRatoliApretat: Bool	
#bMantenirSiRatoli: Bool	
#bTornarAMostrarSiRatoli: Bool	
//Timer per fer les animacions	
#timer: Timer	
//Variable per definir els diferents estats que pot presentar el PopupTInt	
#estatPopupTInt: EstatsPopupTInt	

```

//Rectangles on aniran representades les parts del PopupTInt
+RectNom:Rectangle
+RectNum:Rectangle
+RectDept:Rectangle
+RectMail:Rectangle
#RectangleAreaDeTrebali:Rectangle
//Definició de les parts clicables del PopupTInt
+NomClicable:Bool
+NumClicable:Bool
+DeptClicable:Bool
+MailClicable:Bool
+TancaClicable:Bool
+HistorialClicable:Bool
+NouContClicable:Bool
+ActivarSeleccioRectangle:Bool
//Events del PopupTInt
+NomClic:EventHandler
+NumClic:EventHandler
+DeptClic:EventHandler
+MailClic:EventHandler
+TancaClic:EventHandler
+HistorialClic:EventHandler
+NouContClic: EventHandler
//Constructor
+PopupTExt()
//Retorna l'estat del PopupTInt
+EstatPopupTExt:EstatsPopupTExt
//Indica si el PupupTInt ha de romandre visible quan te el ratolí a sobre
+MantenirSiRatoli:Bool
//Indica si el PopupTInt ha de reaparèixer si passem el ratolí per sobre quan esta
//desapareguent
+TornarAMostrarSiRatoli:Bool
//Funció per mostrar el PopupTExt
+Show(strNom:String, strNum:String, strDept:String, strMail:String,
nTempsEnApareixer:Int, nTempsMantenir:Int, nTempsEnAmagar:Int)
//Funció per amagar el PopupTExt
+Hide()
//Estableix la imatge de fons del PopupTExt a partir d'un nom de fitxer
+SetImFonsPopupTExt(strNomFitxer:String, transparencyColor:Color)
//Estableix la imatge de fons del PopupTExt a partir d'una imatge
+SetImFonsPopupTExt(imatge:Image, color:Color)
//Estableix la imatge del boto de tancar a partir d'un nom de fitxer
+SetImBotoTanca(strNomFitxer:String, color:Color, posicio:Point)
//Estableix la imatge del boto de tancar a partir d'una imatge
+SetImBotoTanca(imatge:Image, color:Color, posicio:Point)
//Estableix la imatge del boto historial a partir d'un nom de fitxer
+SetImBotoHistorial(strNomFitxer:String, color:Color, posicio:Point)
//Estableix la imatge del boto historial a partir d'una imatge
+SetImBotoHistorial(imatge:Image, color:Color, posicio:Point)
//Estableix la imatge del boto per afegir un nou contacte a partir d'un nom de fitxer

```

```

+SetImBotoNouCont(strNomFitxer:String, color:Color, posicio:Point)
//Estableix la imatge del boto per afegir un nou contacte a partir d'una imatge
+SetImBotoNouCont(imatge:Image, color:Color, posicio:Point)
//Estableix la foto de la persona que truca a partir d'un nom de fitxer
+SetImFoto(strNomFitxer:String, color:Color, posicio:Point)
//Estableix la foto de la persona que truca a partir d'una imatge
+SetImFoto(imatge:Image, color:Color, posicio:Point)
//Dibuixa el boto de tancar
#DibuixaBotoTancar(grfx:Graphics)
//Dibuixa el boto de l'històric
#DibuixaBotoHistorial(grfx:Graphics)
//Dibuixa el boto de nou contacte
#DibuixaBotoNouContacte(grfx:Graphics)
//Dibuixa el boto per afegir un nou contacte
#DibuixaBotoNouCont(grfx:Graphics)
//Dibuixa la foto de la persona que truca
#DibuixaFoto(grfx:Graphics)
//Dibuixa un text
#DibuixaText(grfx:Graphics)
//Funció per ajustar els tamanys dels rectangles al text
#CalcularRectanglesRatoli()
//Funció per posar una imatge a una regió
#ImatgeARegio(imatge:Bitmap, color:Color):Region
//Funció del Timer
#OnTimer(obj:Object, ea:EventArgs )
//Funcions dels diferents events de ratolí
#OnMouseEnter(ea:EventArgs)
#OnMouseLeave(ea:EventArgs)
#OnMouseMove(mea:MouseEventArgs)
#OnMouseDown(mea:MouseEventArgs)
#OnMouseUp(mea:MouseEventArgs)
#OnPaintBackground(pea:PaintEventArgs)

```

Figura 3.13

3.4.7. La classe GestorPopupTInt.cs

Aquesta classe s'encarrega de gestionar les trucades internes (de les quals gairebé sempre en disposarem de tota la informació) que puguin rebre els empleats. A l'igual que la classe anterior conté tots els mètodes necessaris per crear, mostrar, etc... aquest tipus de finestres emergents.

Classe	PopupTInt
Descripció de la classe	Classe que gestiona els PopupTInt (Popups dedicats a trucades internes)
Responsabilitats	Col·laboracions
Conté els mètodes per crear, mostrar, amagar i modificar un PopupTInt	Popup
PopupTInt #p: Popup //Rectangles del contingut dels texts del popup #RectangleNom: Rectangle #RectangleNum: Rectangle #RectangleDept: Rectangle #RectangleMail: Rectangle //Events del Popup #nEventsMostrar: Int #nEventsAmagar: Int #nEventsVisible: Int #nIncrementMostrar: Int #nIncrementAmagar: Int //Events de ratoli #bSiRatoliSobrePopup: Bool #bSiRatoliSobreTanca: Bool #bSiRatoliSobreNom: Bool #bSiRatoliSobreNum: Bool #bSiRatoliSobreMail: Bool #bSiRatoliSobreDept: Bool #bSiRatoliSobreHistorial: Bool #bSiRatoliApretat: Bool #bMantenirSiRatoli: Bool #bTornarAMostrarSiRatoli: Bool //Timer per fer les animacions #timer: Timer //Variable per definir els diferents estats que pot presentar el PopupTInt #estatPopupTInt: EstatsPopupTInt //Rectangles on aniran representades les parts del PopupTInt +RectNom: Rectangle +RectNum: Rectangle +RectDept: Rectangle +RectMail: Rectangle #RectangleAreaDeTreball: Rectangle //Definició de les parts clicables del PopupTInt +NomClicable: Bool +NumClicable: Bool +DeptClicable: Bool +MailClicable: Bool +TancaClicable: Bool +HistorialClicable: Bool +ActivarSeleccioRectangle: Bool //Events del PopupTInt +NomClic: EventHandler	

```

+NumClic:EventHandler
+DeptClic:EventHandler
+MailClic:EventHandler
+TancaClic:EventHandler
+HistorialClic:EventHandler
//Constructor
+PopupTInt()
//Retorna l'estat del PopupTInt
+EstatPopupTInt:EstatsPopupTInt
//Indica si el PopupTInt ha de romandre visible quan te el ratolí a sobre
+MantenirSiRatoli:Bool
//Indica si el PopupTInt ha de reaparèixer si passem el ratolí per sobre quan esta
//desapareguent
+TornarAMostrarSiRatoli:Bool
//Funció per mostrar el PopUp
+Show(strNom:String, strNum:String, strDept:String, strMail:String,
nTempsEnApareixer:Int, nTempsMantenir:Int, nTempsEnAmagar:Int)
//Funció per amagar el PopUp
+Hide()
//Estableix la imatge de fons del PopupTInt a partir d'un nom de fitxer
+SetImFonsPopupTInt(strNomFitxer:String, transparencyColor:Color)
//Estableix la imatge de fons del PopUp a partir d'una imatge
+SetImFonsPopupTInt(imatge:Image, color:Color)
//Estableix la imatge del boto de tancar a partir d'un nom de fitxer
+SetImBotoTanca(strNomFitxer:String, color:Color, posicio:Point)
//Estableix la imatge del boto de tancar a partir d'una imatge
+SetImBotoTanca(imatge:Image, color:Color, posicio:Point)
//Estableix la imatge del boto historial a partir d'un nom de fitxer
+SetImBotoHistorial(strNomFitxer:String, color:Color, posicio:Point)
//Estableix la imatge del boto historial a partir d'una imatge
+SetImBotoHistorial(imatge:Image, color:Color, posicio:Point)
//Estableix la foto de la persona que truca a partir d'un nom de fitxer
+SetImFoto(strNomFitxer:String, color:Color, posicio:Point)
//Estableix la foto de la persona que truca a partir d'una imatge
+SetImFoto(imatge:Image, color:Color, posicio:Point)
//Dibuixa el boto de tancar
#DibuixaBotoTanca(grfx:Graphics)
//Dibuixa el boto de l'historial
#DibuixaBotoHistorial(grfx:Graphics)
//Dibuixa la foto de la persona que truca
#DibuixaFoto(grfx:Graphics)
//Dibuixa un text
#DibuixaText(grfx:Graphics)
//Funció per ajustar els tamanys dels rectangles al text
#CalcularRectanglesRatoli()
//Funció per posar una imatge a una regió
#ImatgeARegio(imatge:Bitmap, color:Color):Region
//Funció del Timer
#OnTimer(obj:Object, ea:EventArgs )
//Funcions dels diferents events de ratolí

```

```

#OnMouseEnter(ea:EventArgs)
#OnMouseLeave(ea:EventArgs)
#OnMouseMove(mea:MouseEventArgs)
#OnMouseDown(mea:MouseEventArgs)
#OnMouseUp(mea:MouseEventArgs)
#OnPaintBackground(pea:PaintEventArgs)

```

Figura 3.14

3.4.8. La classe Popup.cs

Aquesta classe conté l'estructura bàsica de les finestres emergents. Elements com textos i botons estan definits en aquesta classe. També consta de les funcions necessàries per modificar aquests elements. D'aquesta manera canviar als atributs dels *popups* fa que sigui una tasca relativament senzilla i brinda la possibilitat de modificar-ne l'aspecte en qualsevol moment (per exemple quan l'aplicació passi pel departament d'Organització per tal de dotar-la d'imatge corporativa).

Classe	Popup
Descripció de la classe	Classe que conte els principals atributs d'un objecte Popup i les funcions necessàries per modificar-los.
Responsabilitats	Col·laboracions
Crea i modifica objectes Popup	TextPopup
Popup //Imatge de fons del Popup #ImFonsPopUp: Bitmap //Boto per tancar el Popup #ImBotoTanca: Bitmap #LocImBotoTanca: Point #TamImBotoTanca: Size //Boto per mostrar l'historial #ImBotoHistorial: Bitmap #LocImBotoHistorial: Point #TamImBotoHistorial: Size //Boto per afegir un contante #ImBotoContacte: Bitmap #LocImBotoContacte: Point #TamImBotoContacte: Size //Foto de la persona que truca #ImFoto: Bitmap #LocImFoto: Point #TamImFoto: Size	

```

//Nom, numero, departament i correu electrònic de la persona que truca
#NomPT:TextPopup
#NumPT:TextPopup
#DeptPT:TextPopup
#TMailPT:TextPopup
//Constructors amb i sense paràmetres
+Popup()
+Popup(iFons:Bitmap, iBT:Bitmap, lBT:Point, tBT:Size, iBH:Bitmap, lBH:Point,
tBH:Size, iBC:Bitmap, lBC:Point, tBC:Size, iPT:Bitmap, lPT:Point, tPT:Size,
rNoPT:Rectangle, noPT:String, rNuPT:Rectangle, nuPT:String, rDePT:Rectangle,
dePT:String, rMaPT:Rectangle, maPT:String)
//Funcions per obtenir o posar els diferents atributs del Popup
+ObtenirImFons:Bitmap
+ObtenirImBotoTancar:Bitmap
+ObtenirLocBotoTancar:Point
+ObtenirTamBotoTancar:Size
+ObtenirImBotoHistorial:Bitmap
+ObtenirLocBotoHistorial:Point
+ObtenirTamBotoHistorial:Size
+ObtenirImBotoContacte:Bitmap
+ObtenirLocBotoContacte:Point
+ObtenirTamBotoContacte:Size
+ObtenirImFoto:Bitmap
+ObtenirLocFoto:Point
+ObtenirTamFoto:Size
//Atributs del nom de la persona que truca
+ObtenirTPNomPT:TextPopup
+ObtenirTextNom:String
+ObtenirColNormalNom:Color
+ObtenirColEventNom:Color
+ObtenirFontNormalNom:Font
+ObtenirFontEventNom:Font
//Atributs del numero de telèfon de la persona que truca
+ObtenirTPNumPT:TextPopup
+ObtenirTextNum:String
+ObtenirColNormalNum:Color
+ObtenirColEventNum:Color
+ObtenirFontNormalNum:Font
+ObtenirFontEventNum:Font
//Atributs del departament de la persona que truca
+ObtenirTPDeptPT:TextPopup
+ObtenirTextDept:String
+ObtenirColNormalDept:Color
+ObtenirColEventDept:Color
+ObtenirFontNormalDept:Font
+ObtenirFontEventDept:Font
//Atributs del correu electrònic de la persona que truca
+ObtenirTPMailPT:TextPopup
+ObtenirTextMail:String
+ObtenirColNormalMail:Color

```

```

+ObtenirColEventMail:Color
+ObtenirFontNormalMail:Font
+ObtenirFontEventMail:Font

```

Figura 3.15

3.4.9. La classe TextPopup.cs

Classe encarregada de definir els texts dels *popups*. És una classe que complementa l'anterior. He decidit posar totes les funcions corresponents als texts de les finestres emergents en una classe a part per tal de facilitar la gestió de la classe Popup.cs.

Classe	TextPopup
Descripció de la classe	Classe que conté els atributs propis de l'objecte text així com els constructors, les funcions per consultar-los o modificar-los i les funcions que descriuen els diferents events que es produiran al fer clic a sobre d'un objecte text.
Responsabilitats	Col·laboracions
Crear un text, consultar-lo, modificar-lo i dotar-lo de diverses funcionalitats. TextPopup #text:String #colTextNormal:Color #colTextEvent:Color #fontTextNormal:Font #fontTextEvent:Font //Constructor sense paràmetres +TextPopup() //Constructor amb paràmetres +TextPopup(txt:string, CNTxt:Color, CETxt:Color, FNText:Font, FEText:Font) //Funció per obtenir o posar el contingut del text de TextPopup +ObtenirText:string //Funció per obtenir o posar el color normal del text de TextPopup +ObtenirColorNormal:Color //Funció per obtenir o posar el color event del text de TextPopup +ObtenirColorEvent:Color //Funció per obtenir o posar la font normal del text de TextPopup +ObtenirFontNormal:Font //Funció per obtenir o posar la font event del text de TextPopup +ObtenirFontEvent:Font //Funció per definir l'event de fer clic a un TextPopup del tipus telèfon +ClicTelf()	


```

//Funció per definir l'event de fer clic a un TextPopup del tipus departament
+ClicDept()
//Funció per definir l'event de fer clic a un TextPopup del tipus correu electrònic
+ClicMail()

```

Figura 3.16

3.4.10. La classe GestorErrors.cs

Aquesta classe s'encarrega de crear i inserir al fitxer de log tots els errors que es puguin produir durant el procés de connexió (*login*) de l'aplicació amb el servidor *Openfire*. En un futur proper esta previst que gestioni el màxim nombre d'errors que es puguin produir durant l'execució de l'aplicació per tal de facilitar-ne la depuració. Per això és capaç d'inserir errors en un fitxer per tal que aquests quedin emmagatzemats físicament i puguin ser consultats si fos necessari.

Classe	GestorErrors
Descripció de la classe	Classe encarregada de fer la gestió dels possibles errors que es puguin produir durant l'execució de l'aplicació.
Responsabilitats	Col·laboracions
-Generar errors en format XML -Introduir els errors al fitxer XML d'errors	Error
GestorErrors +doc_xml: XmlDocument // Constructor + GestorErrors() // Funcions + Afegir_Error (tr: Trucada)	

Figura 3.17

A la Figura 3.18 es mostra el contingut del fitxer xml on s'emmagatzemen els errors que puguin sorgir durant l'execució de l'aplicació.

```
<?xml version="1.0" encoding="utf-8"?>
<Errors>
  <Error Data="02/04/2009 12:31:56" Text="No s'ha pogut establir la connexio amb el servidor Openfire" />
  <Error Data="02/04/2009 12:36:09" Text="No s'ha pogut establir la connexio amb el servidor Openfire" />
</Errors>
```

Figura 3.18

3.4.11.La classe Error.cs

Conté l'estructura bàsica d'un error i les funcions necessàries per crear-los i modificar-los. Els errors que es vagin produint durant l'execució de l'aplicació s'aniran inserint a un fitxer de log per tal de controlar-los i depurar-los.

Classe	Error
Descripció de la classe	Classe que defineix un Error.
Responsabilitats	Col·laboracions
Conte funcions per crear i modificar Errors.	
Error	
#data: string	
#textError: string	
//Constructors	
+ Error()	
+ Error(dt: string, txtErr: string)	
//Funcions	
+ ObtenirData: string	
+ ObtenirTextError: string	

Figura 3.19

3.4.12.La classe GestorTrucades.cs

Aquesta classe s'encarrega de gestionar els dos tipus de trucades que es poden produir (internes o externes). Disposa d'una funció per cadascuna d'elles que el que fan és transferir el missatge xml (enviat pel servidor *Openfire*) corresponent a cada trucada que

arriba al client per tal que pugui ser desglossat i emplenar així els atributs d'una Trucada amb aquesta informació perquè posteriorment es pugui generar la finestra emergent.

Classe	GestorTrucades
Descripció de la classe	Classe per fer el tractament de les trucades.
Responsabilitats	Col·laboracions
S'encarrega de gestionar les trucades tant si son externes com internes.	GestorTInt GestorTExt
GestorTrucades	
//Constructor de la classe	
+GestorTrucades()	
//Funció per fer el tractament de les trucades externes	
+TractarTrucadaExterna(trucada:string):Trucada	
//Funció per fer el tractament de les trucades internes	
+TractarTrucadaInterna(trucada:string):Trucada	

Figura 3.20

3.4.13.La classe GestorTExt.cs

S'encarrega de fer la gestió de les trucades externes que pugui rebre l'usuari. És la classe que conte una funció per tal de desglossar la informació enviada pel servidor *Openfire* i encarregada d'anar a buscar informació corresponent a la persona que realitza la trucada a la llista de contactes de l'*Outlook*. Un cop ha obtingut el màxim d'informació genera una Trucada (veure classe Trucada.cs) i n'emplena els atributs.

Classe	GestorTExt
Descripció de la classe	Classe encarregada de gestionar trucades externes.
Responsabilitats	Col·laboracions
Aquesta classe emplena els camps d'una Trucada amb els texts corresponents. Com que es tracta d'una trucada externa va a buscar les dades del contacte a l'Outlook.	Trucada
GestorTExt	
#missatge:string	
#trucada:Trucada	
//Constructors	
+TrucadaExt()	

```

+TrucadaExt(ms:string)
//Funció encarregada de tractar el missatge que es rep des del servidor Openfire
+TractarMissatge():Trucada

```

Figura 3.21

3.4.14. La classe GestorTInt.cs

Classe encarregada de desglossar la informació que arriba al client procedent del servidor *Openfire*. Un cop desglossada crea una *Trucada* (veure classe *Trucada.cs*) i n'emplena els atributs amb la informació de la persona que realitza la trucada.

Classe	GestorTInt
Descripció de la classe	Classe encarregada de gestionar trucades internes.
Responsabilitats	Col·laboracions
Aquesta classe s'encarrega d'emplenar els camps d'una <i>Trucada</i> amb la informació que li arriba a través del servidor <i>Openfire</i> . Com que les trucades són internes ja disposa de tota la informació.	<i>Trucada</i>
GestorTInt #missatge: string #trucada: Trucada //Constructors + TrucadaInt() + TrucadaInt(ms:string) //Funció encarregada de tractar el missatge que es rep des del servidor <i>Openfire</i> + TractarMissatge():Trucada	

Figura 3.22

3.4.15. La classe Trucada.cs

Conté les funcions necessàries per crear *Trucades* i modificar-ne els atributs (nom, numero de telèfon, departament i correu electrònic).

Classe	Trucada
Descripció de la classe	Defineix una trucada.
Responsabilitats	Col·laboracions
Aquesta classe s'encarrega de crear	

trucades. També conté les funcions necessàries per assignar o obtenir els atributs d'una Trucada.

Trucada

#nom:**string**

#num:**string**

#dept:**string**

#mail:**string**

//Constructors

Trucada()

Trucada(nomPT:**string**, numPT:**string**, deptPT:**string**, mailPT:**string**)

//Funcions

+ObtenirNom:**string**

+ObtenirNum:**string**

+ObtenirDept:**string**

+ObtenirMail:**string**

Figura 3.23

3.5.Pantalles de l'aplicació

En aquest apartat pretenc explicar com han estat dissenyades les diferents pantalles que podrà veure l'usuari de l'aplicació mentre aquesta es trobi en funcionament.

En primer lloc cal destacar que un cop es posi en funcionament l'aplicació l'usuari no veurà cap finestra nova a la seva pantalla. L'únic que veurà és una icona a la part inferior dreta de la pantalla de la qual es podrà desplegar un petit menú on es podran veure les opcions Historial i Sortir.

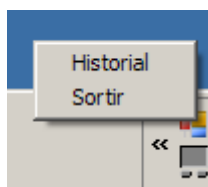


Figura 3.24

A continuació explicaré detalladament les diferents interfícies que veurà l'usuari en el cas que arribi una trucada externa, interna o vulgui veure l'historial de trucades. Cal destacar que aquestes interfícies no són definitives ja que abans de posar l'aplicació en explotació el projecte haurà de passar pel departament d'Organització de l'empresa per tal de dotar-lo d'imatge corporativa. Per això podran ser canviades les mides, colors, posicions, etc...

3.5.1.Els botons

Per fer els botons dels *popups* he hagut de fer servir una imatge dividida en tres parts. L'aplicació mostra només una de les tres parts de la imatge segons els events de ratolí que es produeixen. Si el ratolí no està a sobre del botó es mostra una part de la imatge, si el ratolí passa per sobre es mostra una altra de les parts i si es fa clic al botó es mostra la tercera part de la imatge.

A la figura 3.25 podem veure un exemple de com ha de ser una imatge d'un botó.



Figura 3.25

He de dir que l'exemple no són els botons que he fet servir a l'hora de desenvolupar l'aplicació. Com que segurament a l'hora de dotar d'imatge corporativa el programa s'hauran de canviar els botons m'he creat uns botons estàndard com els que es mostren a continuació.



Figura 3.26

Com es pot apreciar aquest tipus de botó que he creat no canvia segons els events de ratolí ja que les tres parts són la mateixa imatge. He fet servir aquest botó per fer tots els botons dels diferents tipus de *popup*.

3.5.2. Pantalla del *popup* d'una trucada externa

Tant la finestra emergent de les trucades internes com la de les externes estan fetes d'una superposició d'imatges i texts. És a dir, el fons de la finestra és una imatge, els diferents botons són imatges i la foto que apareix també és una imatge. D'altra banda la informació de la persona que truca (nom, numero de telèfon, etc...) són texts.

Un cop obtingudes totes les parts de la finestra es munta l'estructura i es mostra mitjançant una funció que realitza una animació.

Com que es tracta d'una trucada externa, la majoria de les vegades no disposarem de la informació de la persona que esta realitzant la trucada (a no ser que trobem el contacte a la llista de contactes de l'*Outlook*). Per això els diferents texts de la finestra emergent no es mostraran i a on hauria d'aparèixer la foto de la persona que truca hi apareixerà una foto per defecte.

També podem veure els tres botons que formen aquesta finestra emergent. El de més a l'esquerra serveix per afegir un contacte nou a l'*Outlook*. És molt útil si volem guardar la informació de la persona que ens està trucant per tal de tenir-la disponible per trucades posteriors. El botó del mig serveix per visualitzar l'historial de trucades. I finalment el botó de més a la dreta serveix per ocultar immediatament el *popup*.

En aquest cas si s disposa del correu electrònic de la persona que realitza la trucada l'usuari disposarà de l'opció de fer clic al mail per tal d'enviar-li un correu electrònic a traves de l'*Outlook*.

Això és el que veurà l'usuari si li entra una trucada externa.

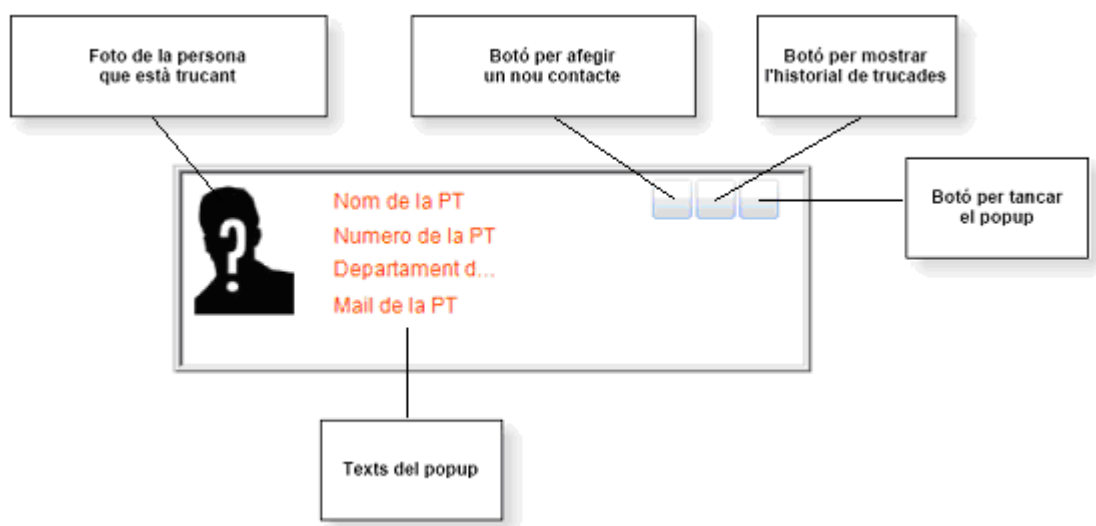


Figura 3.27

3.5.3.Pantalla del *popup* d'una trucada interna

En el cas de que es rebí una trucada interna apareixerà la fotografia de la persona que realitza la trucada així com la possibilitat de fer clic sobre el seu correu electrònic per tal d'enviar-li un missatge o bé sobre el seu departament per accedir a la informació d'aquest departament continguda a la intranet de l'empresa. En aquest cas cap dels camps de text del *popup* apareixeran en blanc perquè es suposa que sempre disposarem de la informació de la persona que truca. Per això aquí desapareix l'opció d'afegir el contacte a la llista de contactes de l'*Outlook*.

També es mantenen l'opció de visualitzar l'historial i d'amagar el *popup*.

Aquesta és la finestra emergent corresponent a una trucada interna.

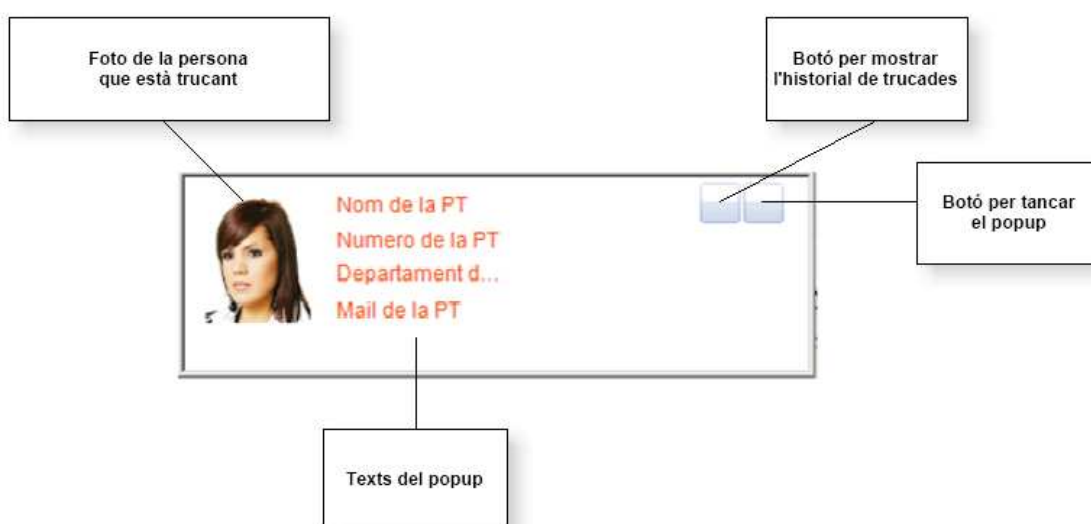


Figura 3.28

3.5.4.Pantalla de l'historial de trucades

Aquesta és la única pantalla dissenyada íntegrament amb el *Visual Studio* ja que no he trobat la manera de mostrar una graella de dades a sobre d'una imatge com si es tractés d'un dels *popups* (en principi aquesta era la idea).

Així doncs l'historial té com a base un *Form* i com a components un botó (per tancar-lo) i un *DataGridView* format per cinc columnes: la data, el nom, el número de telèfon, el departament i el correu electrònic. D'aquestes columnes, a les tres últimes s'hi pot fer clic.

Si es fa clic a un número de telèfon es realitzarà una trucada mitjançant *clic2call* al número de telèfon.

Si es fa clic a un departament es visualitzarà per pantalla (a través del navegador web) la informació corresponent al departament continguda a la intranet de l'empresa.

Si es fa clic a un mail s'obrirà un nou correu electrònic de l'*Outlook* que tindrà com a remitent la persona a la que correspongui l'adreça a la que s'hagi fet clic.

Aquí podem veure com és aquest historial de trucades

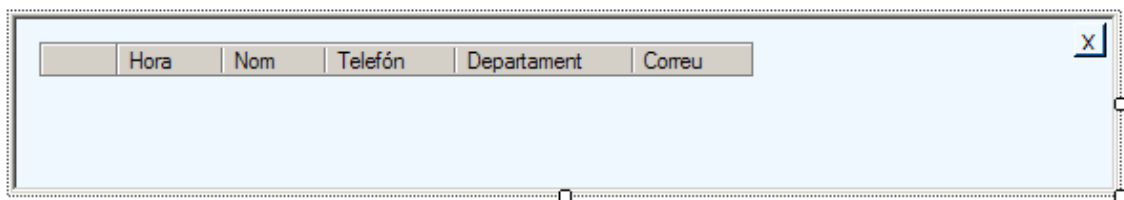


Figura 3.29

A l'igual que en les pantalles anteriors aquesta pantalla de l'historial de trucades és provisional ja que segurament més endavant em veure obligat a modificar-la per tal de dotar a l'aplicació de la corresponent imatge corporativa.

3.6.Com fer el *popup* i els seus elements

3.6.1.La finestra emergent senzilla

A continuació passo a descriure el que seria una primera versió de les diferents finestres emergents (anomenades també *popups*) que formen la part visual i interactiva de l'aplicació.

Principalment podríem dir que una finestra emergent o *popup* es un element que es mostra a la pantalla que conté informació interessant per a l'usuari.

A la següent figura mostro la imatge que he fet servir per crear aquesta primera versió de finestra emergent.



Figura 3.30

Un cop tinc la base he de fer que aquesta es mostri per pantalla. Per fer-ho, ho podria fer de moltes maneres ja sigui fent que aparegui de cop al centre de la pantalla o en una cantonada o bé mitjançant algun tipus d'animació.

Després de donar-hi unes quantes voltes he optat per mostrar la finestra emergent a la cantonada inferior esquerra de la pantalla ja que si la mostro al mig podria molestar als usuaris (a tots ens molesta la publicitat que es mostra en forma de *popup* quan naveguem per la xarxa i més si estem treballant).

D'altra banda he decidit que la pantalla no es mostri de cop sinó que ho faci mitjançant l'animació típica de la majoria d'aplicacions de missatgeria instantània que coneixem. És a dir, que es comenci a mostrar de baix cap a dalt, es mantingui durant un cert temps i si l'usuari no hi interactua desaparegui de la mateixa forma que ha aparegut però a l'inversa (de dalt cap a baix).

Per fer-ho m'he de definir els diferents estats en que es pot trobar el *popup* per tal de poder-lo controlar. Aquests estats són els següents: amagat, apareixent, visible i desapareixent.

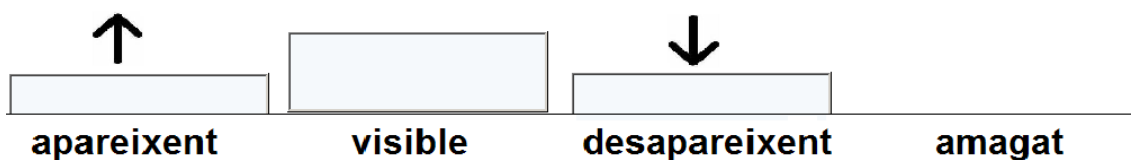


Figura 3.31

Un cop definits els estats de la finestra emergent necessitaré unes variables que em permetin controlar la temporització dels diferents estats. Es a dir que defineixin el temps que tarda a mostrar-se per pantalla, el temps durant el qual es manté el *popup* visible i el temps que tarda a desaparèixer.

A aquestes variables les anomenaré: “nTempsEnApareixer”, “nTempsMantenir” i “nTempsEnAmagar” i seran de tipus enter.

Definides les variables necessàries per fer l'animació de la finestra emergent ara em dispo a definir les variables que necessitaré per controlar les diferents interaccions de l'usuari amb el *popup*.

En aquesta primera versió com que l'únic que podrà fer l'usuari és passar el ratolí per sobre només em caldrà una variable de control que anomenaré “bSiRatoliSobrePopup”. Serà una variable de tipus booleà i em permetrà detectar (treballant juntament amb unes funcions pròpies del Visual C#) si el cursor del ratolí es troba a sobre de la finestra emergent.

D'aquesta manera podré fer certes coses com ara incloure un cas dintre de la funció encarregada de fer passar l'estat del *popup* de visible a amagat que faci que si el ratolí es troba a sobre de la finestra emergent (que vol dir que l'usuari hi esta interactuant) aquesta es mantingui en estat visible i no desaparegui.

...

SI bSiRatoliSobrePopup == **CERT I** estat == visible **LLAVORS**

mantenir

ALTRAMENT

amagar

FSI

...

I d'una forma semblant em permetrà fer que si la finestra emergent està desapareixent i l'usuari hi passa el ratolí per sobre aquesta torni a aparèixer sigui quin sigui el punt en el que es troba.

A continuació podem veure un esquema del que seria un gràfic d'estats que mostra el funcionament d'aquest *popup* simple.

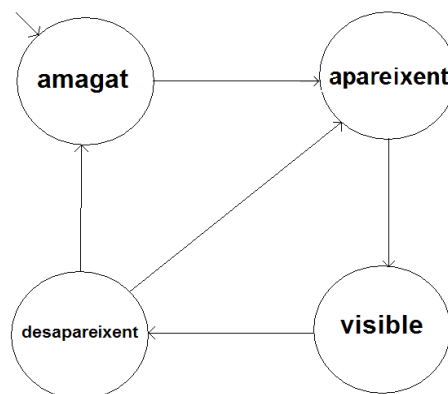


Figura 3.32

El *popup* començarà en l'estat amagat. Si s'activa la funció encarregada de mostrar-lo primer de tot passarà de l'estat amagat a l'estat apareixent i posteriorment a l'estat visible. Per fer aquesta transició tardarà "nTempsEnApareixer".

Un cop el *popup* es trobi en l'estat mostrar si ningú hi interactua es mantindrà en aquest estat durant "nTempsMantenir". Un cop exhaurit aquest temps passarà a l'estat desapareixent. D'altra banda si algú interactua amb la finestra aquesta es mantindrà a l'estat actual (mentre s'interactua amb el *popup* la variable "nTempsMantenir" es seguirà decrementant per tal que tant bon punt l'usuari deixi d'interactuar aquesta s'amagui sense que aquest s'hagi d'esperar).

Així doncs un cop exhaurit el temps "nTempsMantenir" i si ningú està interactuant amb la finestra aquesta passarà automàticament a l'estat desapareixent i seguidament a l'estat amagat. Per fer aquesta transició tardarà "nTempsEnAmagar".

Si mentre el *popup* es troba a l'estat desapareixent l'usuari hi interactua aquest tornarà a l'estat apareixent sense reiniciar la variable "nTempsMantenir".

3.6.2. Afegint botons i imatges

Primer de tot explicaré com funcionen els botons de la finestra emergent ja que de fet els botons i les imatges funcionen de la mateixa manera. L'única diferència és que els botons són interactius i les imatges no. Per això un cop entès com funcionen els botons s'entendrà com funcionen les imatges.

Com ja he explicat anteriorment un botó no es res més que una imatge rectangular dividida en tres parts. Una per cada estat del botó (quan està en repòs, quan l'usuari hi passa el ratolí per sobre i quan l'usuari el prem).

Per tant, per afegir un botó a la finestra emergent necessitaré tres paràmetres. El primer correspondria a la imatge (.bmp) del botó, la qual anomenaré "ImBotoTanca" ja que en el cas del *popup* senzill el que vull fer és afegir-hi el típic botó de tancar del qual

disposen la majoria d'aplicacions per tal que permeti a l'usuari fer desaparèixer la finestra a l'instant quan ell ho desitgi. Una segona variable (de tipus *point*) es correspondrà al punt on vull que aparegui aquest botó i es dirà "LocImBotoTanca" i finalment una tercera variable (de tipus *size*) que em definirà la mida que vull mostrar de la imatge del botó i que anomenaré "TamImBotoTanca".

Aquesta ultima variable em servirà per decidir quina de les tres parts del botó vull mostrar en cada moment segons m'indiquin les variables que detecten la interactivitat de l'usuari amb aquest botó.

Per tal de controlar la interacció de l'usuari amb el botó em caldran dues variable. La primera, que anomenaré "bSiRatoliSobreTanca" (de tipus booleà) em servirà com hem vist anteriorment per controlar si l'usuari passa el ratolí per sobre del botó.

En aquest cas, com que l'usuari pot fer clic a sobre del botó (recordem que amb la base del *popup* només hi pot passar el ratolí per sobre) necessitaré una variable de tipus event (del Visual C#) que em permeti saber en quin moment l'usuari prem el botó del ratolí i que anomenaré "TancaClic".

Combinant les variables que defineixen el botó amb les variables que controlen la interactivitat de l'usuari amb aquest tindrè el següent:

Suposem que disposem del següent botó.



Figura 3.33

La variable "ImBotoTancar" es correspondria a la imatge en si mentre que la variable "LocImBotoTanca" es correspondria al punt de la pantalla on vull mostrar el botó (a l'hora de fer el codi s'haurà de comprovar sempre que aquest punt es correspongui a un punt dels que formen l'àrea de la finestra emergent sinó s'haurà de fer que no es mostri)

i la variable “TamImBotoTanca” seria l'alçada i la base d'una de les tres parts de la imatge en si (alçada de la imatge , llargada de la imatge / 3).

Així doncs a l'hora de mostrar el botó es poden donar tres casos. Són els següents:

- Cas 1. L'usuari no interactua amb el botó per tant les variables “bSiRatoliSobreTanca” i “TancaClic” es troben totes dues a “0”. El que he de mostrar es el rectangle format per la part central de la imatge.



Figura 3.34

Aquest rectangle esta format per dues variables. La primera es correspon al punt inicial de la imatge on comença el rectangle que en el nostre cas seria $(\text{llargada de la imatge}/3)*1$ mentre que la segona defineix la quantitat d'imatge que hem d'agafar i es correspon amb la variable “TamImBotoTanca”.

- Cas 2. Es tracta del cas en que l'usuari passa el ratolí per sobre del botó cosa que provoca que la variable “bSiRatoliSobreTanca” es posi a “1”. Per tant la part de la imatge que volem mostrar es la següent:



Figura 3.35

Les variables que formen el rectangle en aquest cas són en primer lloc $(\text{llargada de la imatge}/3)*2$ i “TamImBotoTanca”.

- Cas 3. Finalment si l'usuari fa clic al botó la variable "TancaClic" es posa a "1" i per tant vull mostrar el següent rectangle.



Figura 3.36

En aquest cas les variables que formen el rectangle a mostrar són 0 i "TamImBotoTanca".

Cal destacar el fet que quan es detecti que l'usuari ha premut el botó (la variable "TancaClic" es posa a "1") s'ha d'activar el mètode o funció corresponent al botó. En aquest cas com que el que volem és que la finestra emergent es tanqui, quan es detecti que s'ha premut el botó el *popup* haurà de passar de l'estat visible a l'estat amagat passant per l'estat desapareixent.

Pel que fa a les imatges com ja he comentat al principi d'aquest apartat és exactament igual que a l'hora d'afegir un botó amb la única diferència que (de moment) l'usuari no pot interactuar amb elles i per tant m'estalvio tota la part de control d'interactivitat.

3.6.3. Afegint texts

Els texts de la finestra emergent funcionen de manera semblant als botons però en comptes de tres imatges que defineixen els estats en que es troba un botó i que permeten fer una petita animació quan l'usuari de l'aplicació hi interactua els texts només disposen de dos estats i aquests es defineixen canviant el tipus de lletra entre l'un i l'altre.

Per exemple si vull mostrar un text quan l'usuari no hi interactuï ho faré de la següent manera.

Text de mostra

En canvi quan l'usuari hi passi el ratolí per sobre el que faré és mostrar el mateix text amb el mateix tipus de lletra però aquesta vegada canviaré el text a negreta.

Text de mostra

D'aquesta manera tant senzilla s'aconsegueix crear un efecte d'interactivitat força interessant.

A diferència dels botons, quan l'usuari fa clic al un text no es necessari crear una tercera animació. Tot i que seria possible trobo força difícil poder crear un efecte de profunditat en un text.

Així doncs per definir un text necessitaré quatre variables que m'indiquin el tipus de lletra que faré servir i el color (aquestes variables les ajustaré per tal de donar imatge corporativa a l'aplicació). S'anomenaran "colTextNormal" per definir el color del text quan l'usuari no hi interactuï, "colTextEvent" per definir el color del text quan l'usuari hi interactuï, "fontTextNormal" per definir la font del text quan l'usuari no hi interactuï i "fontTextEvent" per definir la font del text quan l'usuari hi interactuï.

Un cop definides aquestes variables el text s'emmarcarà a dins d'una variable de tipus rectangle (com en el cas de les imatges i els botons) amb la seva corresponent ubicació dins de la finestra emergent i el seu corresponent tamany.

3.6.4.Resultat

Un cop definides les diferents variables ja només falta una funció que em permeti ajuntar les diferents parts a la imatge de fons del *popup* i aquest en seria el resultat.

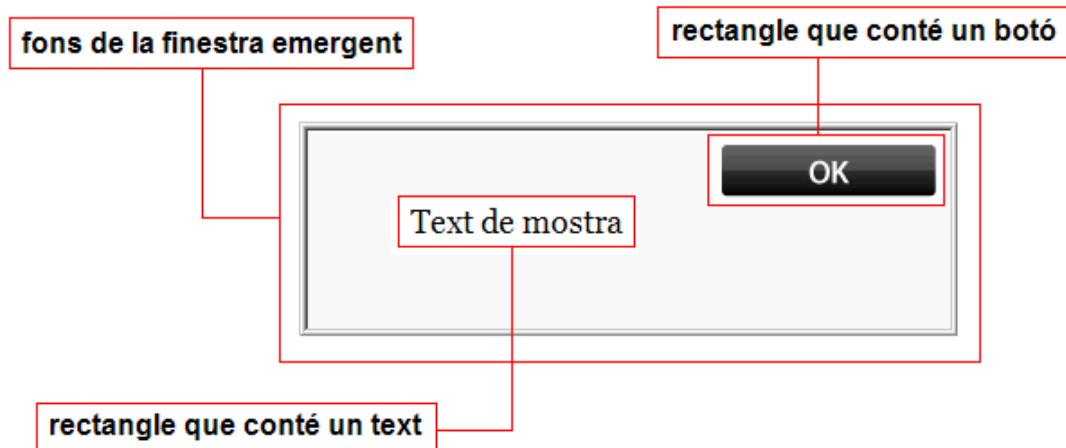


Figura 3.37

Un cop muntada la finestra emergent s'haurà de disposar d'una funció per mostrar-la i d'una altra per amagar-la. Aquestes funcions utilitzaran un "timer" per dur a terme els tempos de les corresponents animacions.

A part d'això també s'hauran d'implementar les funcions corresponents al text i al botó per tal que s'activin quan l'usuari hi faci clic.

4.PROVES, VERSIÓ FINAL I DISTRIBUCIÓ

En aquest apartat s'explica com es farà tota la part corresponent al testeig de l'aplicació i les diferents transformacions que patirà aquesta fins que arribi a l'usuari final així com la forma com es distribuirà als usuaris i la bateria de proves que es realitzarà.

4.1.Primeres proves

Un cop disposem d'una primera versió de l'aplicació, aquesta serà distribuïda manualment a unes quantes persones de l'empresa que faran de "beta testers". Aquestes persones seran algun dels meus companys de departament i algun superior.

Durant un temps aquestes persones provaran l'aplicació i possiblement em suggeriran fer algunes modificacions per tal de millorar-ne algun aspecte.

4.2.Dotant l'aplicació d'imatge corporativa

Un cop l'aplicació obtingui el vist-i-plau de les persones encarregades de fer les proves serà enviada al departament d'Organització de l'empresa per tal de dotar-la d'imatge corporativa.

Aquest procés consistirà en ajustar els colors, les formes, els botons, etc... de totes les interfícies de l'aplicació per tal que aquesta no desentoni a l'hora d'executar-se juntament amb les altres aplicacions que ha desenvolupat l'empresa.

4.3. Paquetització de l'aplicació

Un cop s'obtingui la versió definitiva aquesta s'haurà d'empaquetar per poder-ne fer la distribució als PCs dels empleats de l'empresa.

Per distribuir software l'empresa disposa d'un servidor SMS (*Systems Management Server*) de Microsoft. Aquest servidor és capaç, entre altres coses, d'enviar un programa a tots els PCs de l'empresa i executar-ne la instal·lació.

En el meu cas, ja que l'aplicació no disposa d'un procés d'instal·lació gaire complicat. Els únics fitxers que cal distribuir per instal·lar l'aplicació són: l'executable `IdentificadorDeTrucades.exe`, el fitxer que conté l'històric de trucades `Hist.xml`, un fitxer de paràmetres `Param.xml` i un fitxer d'errors `Err.xml`. Per tant la instal·lació consistirà en generar un directori anomenat `C:\IdentificadorDeTrucades` i copiar a dins d'aquest directori els quatre fitxers de l'aplicació.

Cal tenir en compte que la instal·lació de l'aplicació ha de ser completament desatesa, és a dir, automàtica i invisible. L'usuari final no s'ha d'adonar que s'estan fent coses en el seu PC. D'aquesta manera a l'hora de distribuir-la no s'interromprà a l'empleat de les tasques que estigui duent a terme en aquell precís moment.

Per fer-ho he generat el següent *Visual Basic Script*.

```
Dim objshell
Set objshell = CreateObject("WScript.Shell")

'Creo el directòri C:\IdentificadorDeTrucades
objshell.Run "cmd.exe /c mkdir C:\IdentificadorDeTrucades",0,true

'Copio els fitxers a la carpeta C:\IdentificadorDeTrucades
objshell.Run "cmd.exe /c xcopy /y IdentificadorDeTrucades.exe C:\IdentificadorDeTrucades",0,true
objshell.Run "cmd.exe /c xcopy /y Hist.xml C:\IdentificadorDeTrucades",0,true
objshell.Run "cmd.exe /c xcopy /y Err.xml C:\IdentificadorDeTrucades",0,true
objshell.Run "cmd.exe /c xcopy /y Param.xml C:\IdentificadorDeTrucades",0,true
```

Figura 4.1

Podem veure com es crea el directori i hi copio els diferents fitxers. El paràmetre '0' que podem veure és per fer que les còpies siguin invisibles per l'usuari i el 'true' serveix per fer que no s'executi una instrucció fins que no s'hagi acabat l'execució de la instrucció anterior.

Un cop tenim el fitxer .vbs ja podem generar un paquet al servidor SMS. Aquest paquet executarà a les màquines que nosaltres decidim el fitxer .vbs instal·lant d'aquesta manera l'aplicació.

4.4. Estudi del rendiment

Un cop paquetitzada l'aplicació ha arribat l'hora de distribuir-la (mitjançant l'SMS) a les oficines per tal que els empleats hi comencin a treballar i veure així com afecta realment l'execució del programa al rendiment dels PCs dels treballadors.

Per fer-ho, primer es distribuirà a una sola oficina i si en surt una valoració positiva es provarà en un conjunt més ampli d'oficines (quatre o cinc).

Si durant aquest període de proves es detectes que per culpa d'aquesta aplicació el rendiment de les màquines baixa o al contrari, que el propi rendiment de l'aplicació baixa per culpa de les màquines es procedirà directament a la modificació (optimització) del codi de l'aplicació per tal de solventar el problema.

4.5. Distribució de l'aplicació

En el cas que l'estudi del rendiment explicat a l'apartat anterior surti favorable es procedirà a la distribució massiva de l'aplicació a tots els PCs de l'entitat, tant d'oficines com de Serveis Centrals.

He d'afegir que aquesta distribució tot i ser massiva no es realitzarà de cop sinó que s'anirà fent gradualment. Actualment tota la xarxa d'oficines està geogràficament subdividida en vuit zones. D'aquesta manera, per seguretat, la distribució es realitzarà zona per zona i finalment als Serveis Centrals. Així si es detectes qualsevol error durant el procés de distribució aquest es podria aturar minimitzant-ne les conseqüències sempre i quant es detectes l'error a temps.

4.6. Posta en marxa i aturada de l'aplicació

Aquesta aplicació s'haurà de posar en marxa automàticament cada vegada que un empleat es posi a treballar en un PC. Per fer-ho es modificarà un *script* que s'executa cada vegada que un empleat inicia la seva sessió de Windows.

D'altra banda pel que fa a l'apagada de l'aplicació es pot realitzar de dues maneres. O bé tancant la sessió de Windows de l'usuari o bé fent clic al menú sortir de la pròpia aplicació situat a la part inferior dreta de la pantalla.

5.RESULTATS

5.1.Objectius assolits

Com s'ha pogut veure durant la realització del projecte s'ha estudiat un sistema i s'ha desenvolupat una aplicació en funció de les especificacions que el sistema estudiat demanava i amb l'objectiu de complir una funció determinada.

Així doncs a la conclusió del projecte és que disposa d'una solució (actualment en fase de proves) que ofereix a l'usuari un sistema d'identificació de trucades amb un conjunt de funcionalitats molt interessants com ara la gestió de l'històric de trucades, la possibilitat de realitzar trucades mitjançant la funcionalitat clic2call, enviar correus electrònics, veure la informació departamental d'una persona, etc...

5.2. Temporització real

A continuació es mostra la figura 5.1 que mostra la temporització real del projecte.

Number	Task	Start	End	Duration	2008												
					December	January	February	March	April	May	June	July	August	September			
1	PRIMERA FASE	21/12/2008	13/3/2009	59													
1.1	ESTUDI DE LA INFRAESTRUCTURA	21/12/2008	13/3/2009	59													
1.2	ESTUDI DE L'ENTORN .NET	5/1/2009	12/1/2009	5													
1.3	ESTUDI DE LA PROGRAMACIÓ EN C#	5/1/2009	2/2/2009	20													
2	SEGONA FASE	9/2/2009	10/6/2009	87													
2.1	ESTUDI DELS REQUISITS	9/2/2009	1/6/2009	15													
2.2	DESENVOLUPAMENT DE L'APLICACIÓ	9/2/2009	10/6/2009	87													
3	TERCERA FASE	10/6/2009	31/8/2009	58													
3.1	ESTUDI BENDIMENT	10/6/2009	31/8/2009	58													
3.2	DISTRIBUCIÓ	31/7/2009	31/8/2009	21													

Figura 5.1

Es pot veure com aquesta temporització dista de la temporització prevista. Això és degut a diversos factors com ara el fet d'integrar-me en un nou equip de treball, conèixer l'entorn, o haver tingut de retrocedir en diferents punts durant el desenvolupament de l'aplicació. Tot això ha fet que el projecte s'hagi retardat una mica respecte la previsió inicial.

Tot i això cal destacar que en breu començarà la tercera fase del projecte i que després d'haver parlat del tema amb els responsables del projecte per part de l'empresa les expectatives són molt favorables.

5.3.Conclusions

El fet d'haver tingut l'oportunitat de desenvolupar un projecte per a una empresa m'ha motivat molt i ja tinc ganes de veure com tots els empleats la utilitzen i veure també quines són les seves reaccions.

D'altra banda he pogut veure en primera persona la complexitat que suposa desenvolupar un software per a una empresa. Reunions, idees, dificultats, investigació, programar, errors, etc... han estat presents cada dia dels que he dedicat a desenvolupar aquesta aplicació. Tot això m'ha servit per conèixer gent nova i aprendre'n moltes coses.

Ha estat, en definitiva, una experiència molt positiva i que espero que no sigui l'última.

5.4. Treball futur

Durant la realització d'aquest PFC s'ha assolit el desenvolupament d'una primera versió de l'aplicació de l'Identificador de Trucades. No obstant el projecte continuarà durant encara força temps.

Actualment s'han començat a fer les proves de l'aplicació però un cop l'aplicació estigui a ple rendiment s'haurà de començar a pensar com fer-ho per afegir-hi més funcionalitats com podrien ser: la realització de videoconferències, la inclusió d'un teclat numèric per tal de poder marcar números de telèfon des de la pantalla del PC, la possibilitat de posar una trucada en espera o desviar-la cap a una altra persona (aquí potser s'hauria de dissenyar un altre tipus de *popup* per saber que la trucada que esta arribant a l'usuari ve desviada per un altre usuari), etc...

Una altra característica que podria ser interessant de cares al futur seria la opció de que aquesta aplicació passes a formar part d'una aplicació més gran que actualment està desenvolupant l'empresa i que es caracteritza per oferir als empleats tots els programes que utilitzen en un de sol.

4.5. Coneixements adquirits

Durant el desenvolupament del projecte he après moltes coses com:

- Millora de la capacitat de treballar en equip. Segurament la millor experiència adquirida durant la realització de l'aplicació.
- Experiència de programació basada en events (tecnologia .NET) de la qual en tenia un coneixement gairebé nul.
- Experiència en programació utilitzant el llenguatge C# que no havia utilitzat mai.
- Experiència en l'eina de programació *Visual Studio*.
- El funcionament del servidor *Openfire*, els seus *plugins*, la centraleta CUCM i el Directori Actiu de Microsoft.
- La dificultat de desenvolupar coses en una empresa on cada persona s'encarrega d'alguna cosa i tot ha de ser provat moltes vegades abans de ser donat per bo.

6.BIBIOGRAFIA I REFERÈNCIES

BIBLIOGRAFIA

- [1] Schuller, Joseph. (2001) *Aprendiendo UML en 24 horas*. PRENTICE HALL
- [2] Charre Ojeda, Francisco. (2006) *Visual Studio 2005 (Guia práctica para usuarios)*. ANAYA
- [3] Charre Ojeda, Francisco. (2002) *Visual C# .NET*. ANAYA MULTIMEDIA
- [4] Sharma, Mayank. (2008) *Openfire Administration*. PACKT PUBLISHING

REFERENCIES

- [5] cisco.com, Pagina web de l'empresa CISCO amb informació sobre els seus productes, .COM
www.cisco.com
- [6] xmpp.org, Pagina web dels creadors del protocol XMPP amb informació sobre aquest protocol, .ORG
xmpp.org
- [7] igniterealtime.com, Pagina web dels desenvolupadors del servidor *Openfire*, .COM
www.igniterealtime.com
- [8] csi.map.es, Pagina web del govern amb informació sobre els estàndards que s'han de seguir a l'hora de fer un projecte de programació, .ES
<http://www.csi.map.es/csi/metrica3/index.html>
- [9] msdn.microsoft.com, Pagina web amb tota la informació necessària per programar en *Visual Studio* i tecnologia .NET, .COM
<http://msdn.microsoft.com>

7.GLOSSARI

- [1] **.NET** – Plataforma (*framework*) de desenvolupament d'aplicacions creada per Microsoft que proveeix a l'usuari una gran quantitat de llibreries per tal de poder treballar amb interfícies, bases de dades, aplicacions web, etc...
- [2] **Veü sobre IP** – Anomenada també VoIP (*Voice over Internet Protocol*) és un protocol de transferència de veü a través de la xarxa IP.
- [3] **Sistema de presència** – Sistema que ofereix mobilitat als seus usuaris. Per exemple en una empresa on cada usuari treballa sota una configuració específica gracies a aquest sistema no importa amb quin PC de l'empresa treballi l'usuari ja que sempre mantindrà la seva configuració
- [4] **Directorí Actiu AD** – Tecnologia desenvolupada per Microsoft per tal d'oferir una sèrie de serveis de xarxa com ara LDAP, Kerberos, DNS, etc...
- [5] **Escalable** – Característica d'un sistema que indica la seva capacitat d'expandir-se sense perdre la qualitat del seu servei.
- [6] **QoS** – Qualitat del servei (*Quality of Service*). Tecnologies que garanteixen la transmissió de certa quantitat de dades en un temps determinat.
- [7] **PSTN** – Xarxa telefònica commutada (*Public Switched Telephone Network*).
- [8] **GPL** – Llicència pública general (*General Public License*). Llicència dedicada a la protecció de la lliure distribució, modificació i ús del software.
- [9] **Jabber** – Anomenat també XMPP. És un protocol de missatgeria instantània.
- [10] **Plugin** - Aplicació que dota una aplicació principal d'alguna funcionalitat específica sense necessitat de modificar-la.
- [11] **API** – Interfície de comunicació entre components de software.

8.AGRAÏMENTS

En primer lloc vull agrair a en Lluís Cassú del departament de Producció i Sistemes de Caixa Girona tot el temps i paciència que m'ha dedicat.

A en Robert Miralles, cap del departament de Producció i Sistemes de Caixa Girona per haver-me donat l'oportunitat de fer aquest projecte.

Al meu tutor Pere Vila per l'ajuda i el temps que m'ha dedicat a l'hora de fer aquesta memòria i les orientacions que m'ha anat donant durant la realització del projecte.

A tots ells, moltes gracies!

Agost del 2009, Girona