Faculdade de Engenharia da Universidade do Porto

Ferramenta de Comunicação para Colaboração usando Quadros Interactivos no contexto da educação

Ricardo António Lourenço Amado Preto

Relatório de Dissertação
Mestrado Integrado em Engenharia Informática e Computação

Orientador: Prof. Dr. Pedro Alexandre Guimarães Lobo Ferreira do Souto
Co-orientador: Prof. Dr. Rui Pedro Amaral Rodrigues

28 de Janeiro de 2011

A Communication Tool using Interactive Screens for Education

Ricardo António Lourenço Amado Preto

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Luís Paulo Reis

Vogal Externo: Filipe Pacheco

Orientador: Pedro Alexandre Guimarães Lobo Ferreira do Souto

_____

Resumo

A constante evolução da tecnologia assim como a cada vez maior facilidade de acesso à internet vieram permitir o desenvolvimento de aplicações de colaboração e comunicação em tempo real e entre utilizadores dispersos pelo globo. O facto de o governo ter efectuado recentemente um investimento no sentido de equipar as escolas com meios tecnológicos (quadros interactivos, computadores portáteis, ligações 3G À internet) abriu portas a novas formas de interacção entre professores e alunos.

A presente tese pretende responder a estas recentes necessidades, fornecendo uma ferramenta inovadora que permitirá aumentar o nível de interacção na sala de aula. O presente trabalho propõe-se desenvolver uma aplicação que permite editar uma tela de desenho de um quadro interactivo de forma colaborativa, assim como fornecer ferramentas de comunicação entre os utilizadores, nomeadamente trocas de mensagens de texto e realização de conferências áudio/vídeo.

Para a realização deste projecto foi primeiramente desenvolvida uma *Framework*, utilizando a f*ramework* WCF disponibilizada pela Microsoft. A *Framework* desenvolvida permite implementar funcionalidades de comunicação e colaboração numa qualquer aplicação que possua tecnologias compatíveis. Posteriormente foi elaborado uma pequena aplicação colaborativa de forma a verificar se a *Framework* respondia às necessidades que lhe foram atribuídas. Por último foi realizado o protótipo final que consiste na adição de funcionalidades colaborativas e de comunicação a uma aplicação já existente nomeadamente o UbiStudio. Este produto foi mesmo apresentado na feira internacional BETT, permitindo recolher algum *feedback* da interface e reponsividade do mesmo. O facto de este produto ter sido já dado a conhecer ao público, aliado aos testes efectuados permitem concluir que o resultado da solução obtida cumpre com os objectivos inicialmente traçados.

Abstract

The constant evolution of technology and the increasing ease of internet access enabled the development of communicative and real time collaborative applications. The fact that the government has recently made an investment in order to equip schools with technology (interactive whiteboards, laptops and 3G internet connections) opened the door to new forms of interaction between teachers and students.

This thesis aims to answer to these recent requirements, providing an innovative tool that will increase the level of interaction in the classroom. This study proposes to develop an application that lets the edition of an interactive whiteboard in a collaborative way, while providing communication tools between the users such as text messages and audio/video conferencing.

In order to develop such application, a Framework was initially developed using the framework WCF provided by Microsoft. This Framework allows the implementation of communication and collaboration features into any application with compatible technology. After the development of this Framework a small and simple collaborative application was created using the developed Framework. This application allowed to see if the goals of the Framework were met. Finally the final prototype was developed. This prototype consisted on adding collaborative and communicative features to an existing application named UbiStudio. The product was presented at the international showcase BETT allowing to collect some feedback on the interface and response of the product. The fact that this product has already been made known to the public, joined with the tests performed, allowed to conclude that the product obtained meets all the goals initially set.

Agradecimentos

O presente espaço serve para agradecer a todas as pessoas sem as quais não teria sido possível desenvolver este trabalho. Desta forma permitam-me nesta secção, ao contrário do restante documento, utilizar um tom mais pessoal.

É normal na presente secção agradecer-se a todas as pessoas que acompanharam o percurso deste trabalho mas no entanto não o vou fazer. Não vou agradecer a todos aqueles que já estão comigo desde que me conheço. Não agradecerei a todos aqueles, que apesar de saberem que teria que madrugar no dia seguinte me obrigaram a não me deitar a horas decentes. Não agradecerei embora devesse pois apesar das poucas horas de sono que me proporcionaram, fizeram com que a minha cabeça deixasse por horas de pensar no que tinha que fazer e como o tinha que fazer, o que muito sinceramente foi de uma grande ajuda porque havia dias em que já estava cheio de a ouvir. Não agradecerei embora devesse pois tiveram o dom de me manter sempre animado e pronto para mais um dia de trabalho. Também não vou agradecer a quem me aturou e a quem me deu a possibilidade de agora estar a redigir este documento. No fundo não vou agradecer porque sei que todos sabem o quão grato estou por me terem dado tudo o que me deram e como é lógico me vão continuar a dar. A presente secção será destinada a um outro leque de pessoas. Dedico-a a todos aqueles que não me conheciam, a todos aqueles que apesar de nunca ter tido o prazer de conhecer pessoalmente, me deram muito mais do que poderia pedir e esperar.

Agradeço em primeiro lugar ao professor Pedro Souto meu orientador, e ao professor Rui Rodrigues meu co-orientador. Agradeço pela irrepreensível disponibilidade que ambos sempre demonstraram e pelas importantes sugestões e reparos que me providenciaram.

Agradeço em segundo lugar aos responsáveis pela empresa UbiWhere onde realizei a tese. Obrigado não só por me darem todos os recursos que necessitei para elaborar este trabalho, como também a liberdade e peso que sempre deram às minhas opiniões no decorrer do presente trabalho.

Agradeço em terceiro lugar a todos os colaboradores da UbiWhere pela disponibilidade e pelo ambiente agradável que sempre me providenciaram. Destes, permitam-me apenas destacar dois colaboradores da empresa que me deram um pouco mais do que todos os outros. Permitam-me agradecer de uma forma um pouco mais efusiva aos colaboradores Ricardo Machado e Bruno Silva. Obrigado pelo constante suporte que me deram e pela preciosa ajuda na integração final da aplicação com o UbiStudio. Obrigado por me terem dado o privilégio não só de conhecer esta formidável aplicação como de a melhorar transformando num produto ainda mais inovador e completo.

Agradeço em quarto lugar a uma das pessoas mais importantes para o sucesso do presente trabalho. Um dos maiores medos que tinha no início dos trabalhos, eram as duas horas diárias de comboio que iria realizar. Convínhamos que 1 hora de comboio às 8 da manhã onde o a vontade de abrir os olhos é mínima e mais 1 hora às 18h30 depois de olhar para o computador durante tantas horas (e ter acordado tão cedo) são condições temíveis para qualquer ser humano. Daí a minha total gratidão para a rapariga que não só eliminou o medo que tinha destas duas horas como as transformou num par de horas sinceramente agradáveis. É mesmo esquisito dizer

que se a viagem demorasse o dobro do tempo, temo que não me custasse minimamente fazê-la. Obrigado Diana Almeida não só por me aturares todos os dias (salvo quando a menina se atrasou) como por me teres dado um bocadinho de ti, bocadinho esse que eu guardo com muito carinho.

Os meus sinceros agradecimentos a todos,
Ricardo Preto

Índice

Lista de figuras

Lista de Tabelas

Abreviaturas e Símbolos

API	Application Programming Interface
COM+	Component Object Model
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
J2EE	Java 2 Enterprise Edition
MSMQ	Message Queuing
RAM	Random Access Memory
REST	Representational State Transfer
RTP	Real-Time Protocol
RTMFP	Real-Time Media Flow Protocol
RTMP	Real-Time Messaging Protocol
SOAP	Simple Object Access Protocol
TIC	Tecnologias da Infomação e Comunicação
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VOIP	Voice over Internet Protocol
WCF	Windows Communication Foundation
WPF	Windows Presentation Foundation
WS	Windows Security
WSE	Web Service Enhancements
XML	Extensible Marked Language

Capítulo 1

Introdução

A proposta da presente tese consiste na implementação de uma ferramenta colaborativa utilizando todas as potencialidades que a evolução da tecnologia e a Internet vieram disponibilizar, inserindo-as directamente na sala de aula.

No decorrer do último século a sociedade portuguesa sofreu uma grande alteração na sua filosofia e qualidade de vida. A principal responsável por esta alteração foi a evolução sofrida pela tecnologia no decorrer dos últimos anos. Hoje em dia a tecnologia encontra-se presente em quase todas as áreas da sociedade, contribuindo de forma importante para a sua evolução e qualidade de serviços. A tecnologia aliada ao aparecimento e divulgação da Internet, permitiu estabelecer pontes de comunicação acessíveis entre utilizadores dispersos pelo globo. Este facto permitiu não só uma maior facilidade na troca de conhecimentos e informação, assim como a partilha de recursos anteriormente disponíveis apenas a um reduzido leque de utilizadores. O facto de a Internet ser um recurso de fácil acesso, contribuiu para a popularização do mesmo e para o acentuado crescimento do seu número de utilizadores, tal como pode ser verificado na Fígura 1.

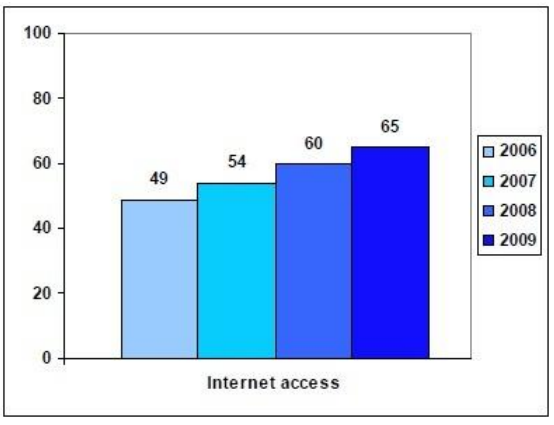


Figura 1: Percentagem de utilizadores da EU que detinham uma ligação à Internet no ano de 2006 até 2009[IntEU]

O consumo de Internet em Portugal seguiu o mesmo comportamento do consumo de Internet na União Europeia, verificando-se em 2009 que 61,1% dos domicílios de Portugal detinham uma ligação à Internet [IntPT]. O aparecimento e consequente utilização deste recurso, veio disponibilizar uma maior possibilidade de realização de trabalho de forma colaborativa, assim como permitir uma maior troca de informação entre entidades separadas fisicamente. A presente tese tem como principal objectivo aproveitar este novo paradigma de comunicação, inserindo-o na sala de aula. Apesar da importância que as TIC (Tecnologias da Informação e Comunicação) podem representar no sucesso académico dos estudantes, através principalmente da troca de informação e de recursos com outras instituições académicas e/ou empresariais [FDNT], estas não se encontram ainda em perfeita harmonia com o sistema educativo português. Na maioria dos casos a utilização das TIC é feita por iniciativa dos estudantes, ficando estes penalizados no caso de não terem conhecimento da existência ou da forma correcta de utilização deste recurso para obtenção de informação. O sistema educativo português ao contrário de áreas como a medicina que abraçaram desde cedo a evolução das TIC, não soube tirar partido do grande desenvolvimento da tecnologia, não tendo realizado uma revolução/evolução dos seus métodos aproveitando as potencialidades que as TIC disponibilizam. Esse facto, aliado ao facto de os jovens tomarem cada vez mais cedo contacto com as novas tecnologias tornou a sala de aula tradicional num ambiente desmotivante, contribuindo este último facto para a diminuição do aproveitamento escolar, "*Quem lida normalmente com jovens ou está atento aos seus interesses, sabe o quanto os telemóveis, computadores, iPod, consolas, etc, são do seu agrado. A facilidade com que aderem e lidam com estes objectos é um campo de possibilidades tão rico e cheio de potencialidades para aprendizagens várias que é totalmente absurdo que a escola não o utilize para atingir os seus objectivos pedagógicos*" [INTEDU]. O aproveitamento escolar de um determinado aluno, é directamente dependente do grau de atenção e de motivação que este dedica à aula [INTEDU,TEDPT], logo se o ambiente da sala de aula não for interactivo e dinâmico, este perderá mais facilmente a motivação e consequentemente a concentração. "*Quando se abre a sala de informática (vulgo sala TIC - Tecnologias de Informação e da Comunicação) o entusiasmo é geral. Os jovens dirigem-se imediatamente para o computador e esperam impacientemente a autorização do professor para o ligar e experimentar.*"[INTEDU]. A utilização de material audiovisual na sala de aula assume um papel importante no que diz respeito à motivação dos alunos. Nesse sentido, o governo português tem ao longo dos últimos anos, apostado em equipar as escolas com melhores meios tecnológicos, com o intuito de trazer a sala de aula para o século XXI. Para esse efeito o governo equipou as escolas com quadros interactivos e facilitou o acesso a computadores pessoais e internet móvel aos alunos do ensino público [MEDU]. Para além disso o governo formou ainda os seus docentes em competências de tecnologias de informação, de forma a estes poderem utilizar todas as potencialidades providenciadas pela inserção de meios tecnológicos na sala de aula [MEDU]. A inclusão de quadros interactivos assim como a disponibilização de ligações à Internet, abriu portas à possibilidade de aparecimento de aplicações desenvolvidas especificamente para a sala de aula com o intuito de criar um ambiente dinâmico e apelativo aos seus intervenientes.

1.1 - Motivação

No ambiente de ensino tradicional, o professor é o único responsável pela preparação e apresentação do conteúdo que será leccionado na sala de aula. O aluno assume uma postura passiva frente aos conteúdos, limitando-se a retê-los da melhor forma possível inquirindo o professor por iniciativa própria sempre que alguma dúvida surja. Por outro lado, a utilização de metodologias de trabalho colaborativo no decorrer de uma aula, apelam a uma maior participação dos alunos, impulsionando assim uma maior troca de informação entre alunos e professores. Em ambientes colaborativos, é dada a possibilidade de os alunos interagirem e contribuírem eles próprios na construção de conhecimento, aumentando assim a participação dos mesmos e fazendo-os sentir-se

parte integrante da sala de aula. Foi neste sentido que a presente tese surgiu, propondo-se disponibilizar um *software* pensado para o ambiente de uma sala de aula e contendo ferramentas de comunicação e colaboração. Esta aplicação tomará partido não só da constante evolução sofrida pela tecnologia, como também da facilidade de acesso e recente inserção da mesma nas instituições de ensino públicas. Pretende-se com o presente trabalho virtualizar a sala de aula, de modo a que todo o conteúdo preparado pelo professor possa ser visualizado e editado por qualquer um dos alunos, independentemente de estes estarem ou não presentes fisicamente na sala de aula. Desta forma, para além de um maior dinamismo e colaboração dos alunos na construção do conhecimento a ser transmitido, estudantes com necessidades especiais poderão interagir como qualquer outro aluno e participar de forma mais activa na sala de aula. Estes estudantes deverão poder através do conforto das suas casas, visualizar e interagir em tempo real com o conteúdo disponível no quadro interactivo, assim como comunicar com qualquer um dos alunos/professores que estiverem presentes na sessão/sala de aula.

1.2 - Descrição do problema

O objectivo principal do presente projecto é o desenvolvimento de uma aplicação colaborativa e de comunicação orientada à partilha de um quadro interactivo. O projecto prevê a adição de um módulo de comunicação/colaboração a uma aplicação que permite a edição de uma tela de um quadro interactivo. Esta aplicação disponibiliza já diversas ferramentas de colaboração, sendo que após a conclusão do presente trabalho, parte destas funcionalidades deverão ser possíveis de utilizar de forma colaborativa.

A aplicação ao qual serão estendidas funcionalidades colaborativas será o *UbiStudio* que se encontra actualmente a ser desenvolvido pela empresa *UbiWhere*.

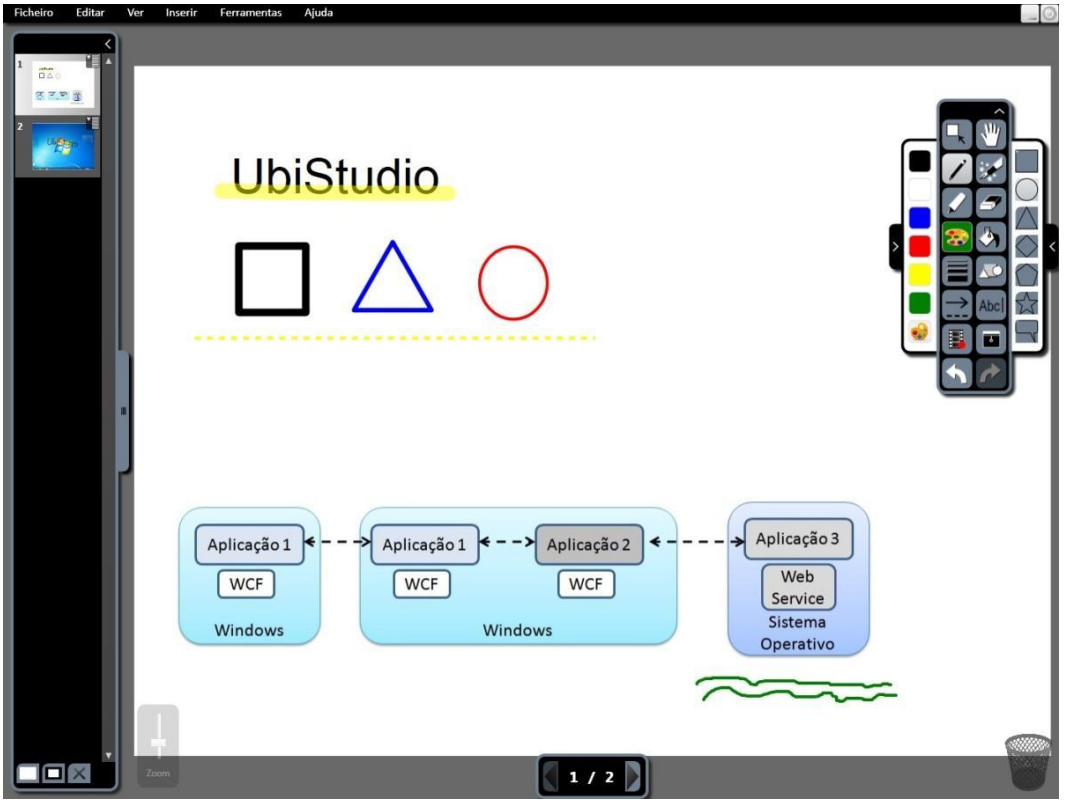


Figura 2: Ambiente de trabalho da aplicação *UbiStudio*

Como pode ser visualizado pela imagem Figura 2, o *UbiStudio* disponibiliza diversas ferramentas de edição de uma tela de desenho através da manipulação de uma interface intuitiva e apelativa.

Pretende-se com o presente trabalho que a edição da tela de desenho da aplicação anterior passe a ser colaborativa, permitindo que utilizadores separados fisicamente possam visualizar as alterações efectuadas entre si.

Apresenta-se de seguida um possível cenário de utilização:

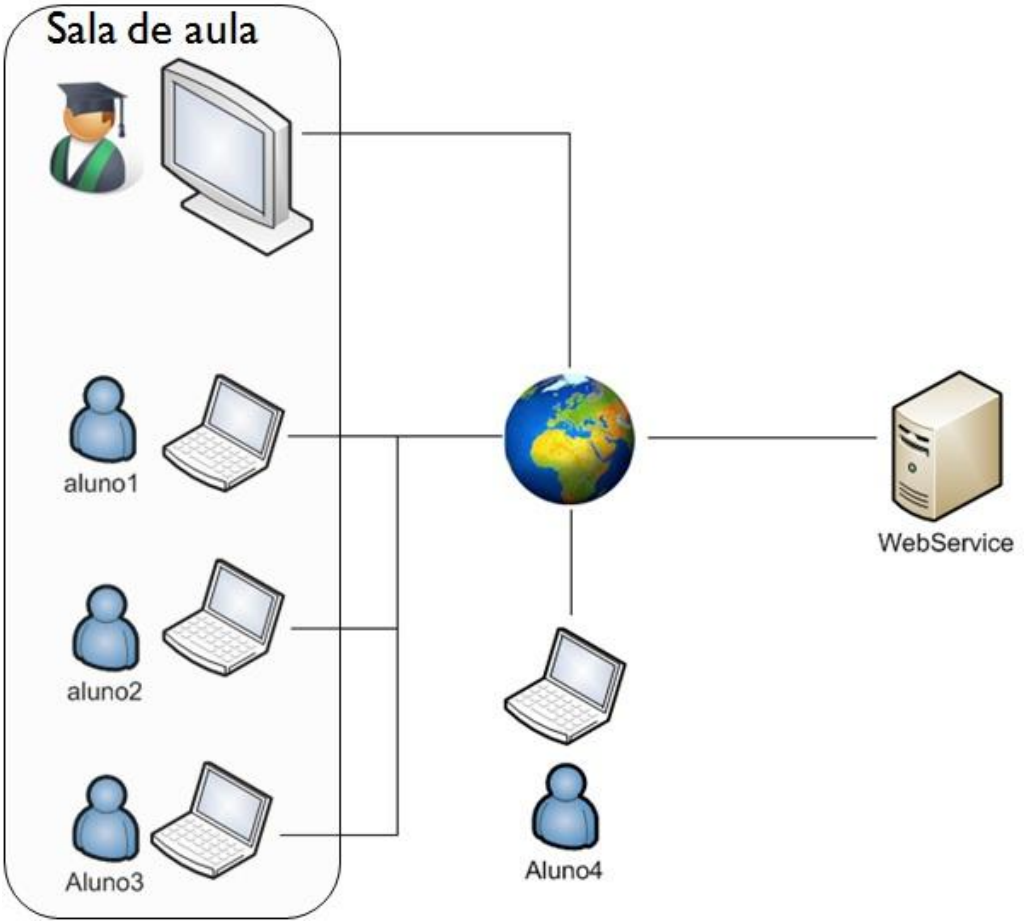


Figura 3: Cenário de utilização da aplicação desenvolvida

Como pode ser inferido da Figura 3, deverá ser possível para qualquer utilizador registado numa determinada sessão visualizar as alterações que os outros utilizadores realizarem na tela. Por outro lado esse mesmo utilizador deverá poder editar a tela de desenho, sendo que as suas alterações serão também visíveis a todos os outros utilizadores. O único requisito para o cenário anterior é que todos os utilizadores possuam uma conexão à Internet. Através da implementação do cenário anterior será possível, não só aumentar a interacção dos alunos numa sala de aula como possibilitar que alunos que não estejam presentes fisicamente na sala de aula possam acompanhar o decorrer da mesma com um mínimo de restrições no que diz respeito à sua participação.

A aplicação final deverá disponibilizar as seguintes funcionalidades:

- Implementar um sistema de chat
- Implementar um sistema de partilha de vídeo/áudio
- Implementar um sistema de partilha de ficheiros
- Implementar um sistema de edição de uma tela de desenho num quadro interactivo e de forma olaborativa
- Implementar sistema hierárquico de controlo de permissões
- Disponibilizar todos estes serviços recorrendo a uma interface apelativa

1.3 – Estrutura da dissertação

Para além da introdução, a presente dissertação contém mais cinco capítulos. No próximo capítulo é apresentado o estado de arte relativo a aplicações inseridas no âmbito do presente trabalho, assim como analisadas ferramentas passíveis de ser utilizadas para a implementação da solução final. No capítulo 3 é realizada uma breve análise ao trabalho a desenvolver apresentando as tecnologias a utilizar bem como o cenário de utilização da solução proposta. No capítulo 4 é apresentada a metodologia de desenvolvimento e descrita a solução proposta para resolver o problema levantado pelo presente trabalho. São apresentadas as soluções para os três componentes a desenvolver no decorrer dos trabalhos, enumerando para cada um destes as suas restrições, requisitos, e arquitectura física. É também neste capítulo apresentada a interface esperada para cada um dos protótipos desenvolvidos. No capítulo 5 são apresentados os detalhes de implementação assim como os resultados obtidos. Para cada um dos produtos são descritos os seus principais módulos/componentes e apresentada a sua interface final. No capítulo 6 são apresentadas as conclusões retiradas, assim como identificado o trabalho futuro após a conclusão da presente tese.

Capítulo 2

Estado da Arte

No presente capítulo será apresentado o estado da arte, relativo a tecnologias, aplicações e recursos que actualmente se encontram disponíveis para a realização/implementação de metodologias colaborativas e/ou de comunicação.

2.1 - Aplicações colaborativas

Existem actualmente, um número considerável de aplicações que disponibilizam algumas ferramentas de comunicação e colaboração. Apesar do número destas aplicações ser consideravelmente maior das aplicações contempladas no presente capítulo, foram apenas referenciadas no mesmo, aquelas que possuem ou possuíram um maior grau de utilização. No presente capítulo são analisadas as seguintes aplicações:

- Skype
- Google Wave
- Camões
- Edoboard
- WiZiQ

2.1.1 - Skype [Skype,Skype2]

A empresa Skype Limited foi fundada em Agosto de 2003 por Niklas Zennstroem e Janus Friis. O seu principal produto Skype foi criado por Ahti Heinla, Priit Kasesalu e Jaan Tallinn, e permite estabelecer um canal de comunicação entre dois utilizadores, independentemente do local onde estes se encontrem no globo, desde que possuam uma ligação à Internet. A aplicação foi construída baseada em tecnologia VOIP (*Voice Over Internet Protocol*) que possibilita, utilizando uma ligação à Internet, que dois utilizadores comuniquem por voz de forma eficiente. Devido ao sucesso atingido, o Skype tem sofrido algumas alterações e adição de funcionalidades ao longo dos tempos, contando de momento com os seguintes recursos:

- Realização de chamadas de voz entre computadores

- Realização de chamadas de vídeo/voz entre computadores
- Realização de chamadas de voz entre computador e telefone/telemóvel
- Realização de chamadas em conferências (chamada de vídeo/áudio partilhada para "n" outras pessoas)
- Envio/recepção de ficheiros
- Sistema de troca de mensagens de texto entre utilizadores do Skype
- Envio de mensagens de texto para telemóveis
- Possibilidade de utilização de um sistema de *voicemail*

Como é possível verificar o Skype disponibiliza um elevado leque de funcionalidades no que diz à comunicação. Devido ao seu elevado sucesso e universalização, têm vindo a ser desenvolvidos *plugins* com o intuito de tornar o Skype numa ferramenta colaborativa. Neste âmbito destacam-se os seguintes *plugins*:

- TalkAndWrite
- IDroo

Ambas as aplicações foram construídas com a intenção de fornecer ferramentas colaborativas aproveitando todas as ferramentas de comunicação implementadas pelo Skype. As ferramentas disponibilizam os seguintes recursos [T&W,IDroo]:

- Disponibilização de uma tela de desenho, com algumas ferramentas de edição da mesma
- Possibilidade de partilha da tela de forma colaborativa e em tempo real entre n utilizadores

No que diz respeito a limitações ambas as aplicações não permitem a inserção de vídeos na conferência nem interagir com o ambiente de trabalho sem sair ou minimizar a aplicação. A aplicação TalkAndWrite limita ainda o número máximo de utilizadores numa sessão a dez utilizadores.

2.1.2 - Google Wave [GWAVE,GWHigh,WGW]

O Google Wave é um serviço disponibilizado pela Google e desenvolvido para ser acessível a qualquer utilizador que possua uma ligação à Internet. O Google Wave foi desenvolvido pela Google com o intuito de fornecer um ambiente ágil, intuitivo e apelativo, orientado à colaboração e comunicação entre diversos utilizadores. Este produto foi apresentado no ano de 2009 no decorrer da conferência *Google I/O* e funciona embebido num browser. O Google Wave providencia os seguintes serviços:

- Correio electrónico
- Mensagens instantâneas
- Web chat
- Redes sociais
- Gestor de projectos
- *Wiki*

O protocolo e plataforma deste produto foram disponibilizados publicamente, pelo que se assiste a uma constante disponibilização de novos serviços/funcionalidades para o Google Wave por parte de diversas instâncias. É então possível desenvolver extensões e adicionar recursos/funcionalidades ao Google Wave adaptando-o a vários cenários de utilização. Existem actualmente várias extensões que implementam a partilha de forma colaborativa de uma tela de desenho entre vários utilizadores. No entanto estas extensões não foram disponibilizadas por uma

entidade certificada, verificando-se mesmo alguns problemas de implementação e performance.

Apesar das inúmeras funcionalidades colaborativas e comunicativas disponibilizadas por este serviço, o mesmo foi descontinuado pela Google no mês de Agosto de 2010. Embora o serviço ainda se encontre disponível e funcional, a empresa anunciou que já abandonou o projecto, pelo que este não seria alvo de melhoramentos ou adição de funcionalidades. A disponibilização do próprio serviço não é também certa, podendo o mesmo deixar de ser disponibilizado num futuro próximo.

2.1.3 - Camões [Camoes]

A plataforma Camões foi desenvolvida pela empresa Microfil com o intuito de disponibilizar uma aplicação de características colaborativas e comunicativas, a ser utilizada nas salas de aulas tanto por professores como alunos. A plataforma prevê a utilização de um quadro interactivo e de computadores pessoais para a edição de uma tela de desenho de forma colaborativa. O software disponibiliza algumas ferramentas de edição da tela, que podem ser utilizadas pelos vários utilizadores tanto interagindo directamente no quadro como utilizando o seu computador pessoal. A colaboração do quadro é realizada em tempo real, podendo mesmo os utilizadores não se encontrarem no mesmo espaço físico mas participar na edição da tela de desenho.

2.1.4 - Edoboard [Edoboard]

O Edoboard é uma plataforma acessível a partir de um browser com ligação à Internet. O Edoboard foi criado com o intuito de servir o mundo académico, comprometendo-se a servir de ferramenta de comunicação entre professores e alunos. O Edoboard permite que um professor/tutor, crie uma sala de aula virtual e permita que um ou mais alunos se liguem a ela. Este ambiente virtual possui os seguintes recursos:

- Partilha de áudio/vídeo possibilitando assim a realização de conferências entre professores e alunos
- Partilha em tempo real e de forma colaborativa de uma tela de desenho
- Ferramentas de matemática (calculadora, desenho de funções, elaboração de equações)
- Partilha de ficheiros
- Gestão de aulas e respectivos alunos no calendário do utilizador
- Partilha, marcação e comentário de exercícios realizados entre a comunidade de utilizadores do Edoboard

O Edoboard é uma plataforma acessível a partir de um browser com ligação à Internet. O Edoboard foi criado com o intuito de servir o mundo académico, comprometendo-se a servir de ferramenta.

Tal como foi referido esta aplicação está restringida à utilização de um browser. Devido a isto não pode utilizar todas as funcionalidades que uma aplicação de *desktop* poderá usufruir. Para além disso, se o utilizador não possuir uma ligação à internet, não poderá utilizar a aplicação mesmo que não pretenda entrar em sessão com outros colaboradores.

2.1.5 - WiZiQ [WiZiQ]

WiZiQ é um serviço/plataforma disponível online que permite que vários utilizadores (professores e alunos) criem uma sala virtual, e de forma dinâmica e colaborativa troquem informação entre eles. O serviço disponibiliza os seguintes recursos:

- Partilha de áudio/vídeo permitindo assim a realização de conferências entre utilizadores

- Partilha de forma colaborativa de uma tela de desenho
- Possibilidade de integrar um ficheiro Power Point na sessão
- Partilha do ecrã
- Suporte para ficheiros do tipo Microsoft Word, Flash, PDF.

2.2 - Tecnologias

No presente capítulo, serão apresentadas tecnologias, nomeadamente plataformas que permitem a implementação de funcionalidades de comunicação e colaboração. Para cada uma das plataformas será feita uma análise dos módulos que disponibiliza assim como tecnologias que utiliza.

2.2.1 - Windows Communication Foundation [IWCF,WWCF]

O WCF (Windows Communication Foundation) foi criado pela Microsoft e lançado com a *framework* .Net3.0. Esta API foi disponibilizada com o intuito de fornecer um modelo unificado de programação para a construção de aplicações distribuídas orientadas a serviços, assim como para a implementação de comunicação entre processos e/ou aplicações. O WCF disponibiliza um conjunto de ferramentas que permitem separar a lógica de negócio da componente de comunicação, encapsulando-a num único módulo construído utilizando WCF. Utilizando o WCF é ainda possível comunicar entre aplicações desenvolvidas em diferentes linguagens para diferentes sistemas operativos, desde que estas sigam os mesmos *standards* do serviço que utilizam.

A plataforma .Net, antes da disponibilização da API WCF, disponibilizava vários módulos que ofereciam a capacidade de implementar diversos paradigmas de comunicação. Através da introdução do WCF, e tal como é ilustrado na Figura 4, todos estes paradigmas foram disponibilizados na mesma plataforma, facilitando o acesso e utilização das mesmas às equipas de programação.

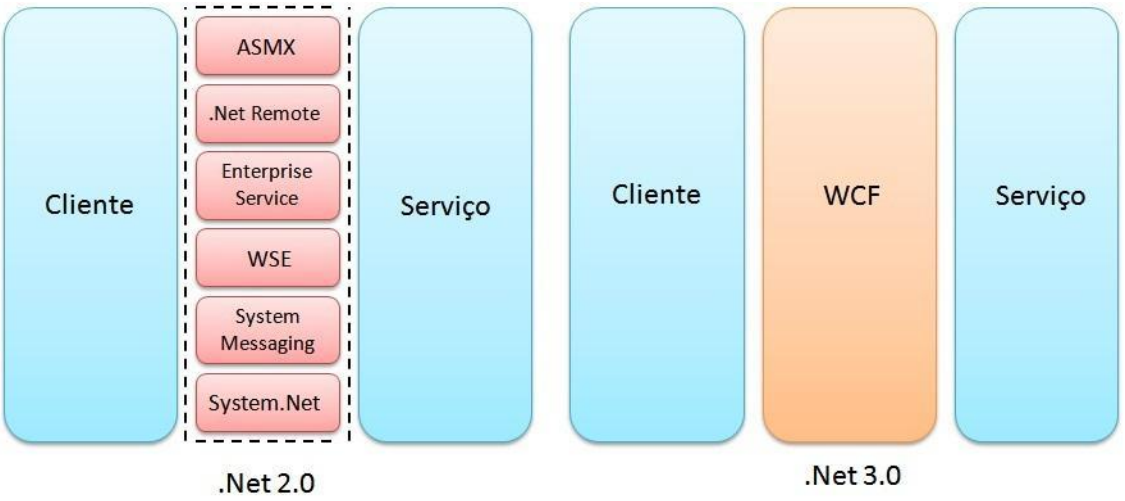


Figura 4: Módulos unificados no WCF

Apresentam-se de seguida cada uma das APIs que a *framework* WCF veio unificar:

- **ASMX:** Também conhecido por ASP.NET Web Services, é uma API de comunicação que tem como principal característica a capacidade de comunicar com aplicações baseadas na plataforma J2EE.

- **Net Remoting:** É uma API da Microsoft que tem como principal objectivo a implementação de comunicação entre aplicações desenvolvidas na .Net framework, apresentando a melhor *performance* nesta situação, quando comparada com outas APIs
- **Enterprise Services:** API que permite o acesso a recursos do monitor transacional do Windows, denominado COM+. [3]
- **WSE:** Também conhecido por *Web Services Enhancement*, possibilita a comunicação com aplicações baseadas na plataforma J2EE tal como ASMX usando mensagens utilizando o protocolo SOAP ou REST. O WSE implementa ainda as especificações WS-* nas suas mensagens baseadas no protocolo SOAP, adicionando um maior grau de segurança no que diz respeito à protecção das suas mensagens.
- **System.Messaging:** API que disponibiliza uma interface para MSMQ (*Message Queuing*). Tecnologia frequentemente utilizada para comunicar com aplicações que nem sempre estão disponíveis.
- **System.Net:** API que permite a comunicação entre diferentes aplicações, implementando uma comunicação baseada no protocolo HTTP

Um dos aspectos centrais a ter em conta aquando da implementação de um canal de comunicação entre processos/aplicações é a segurança. Esta preocupação encontra-se também presente no WCF através do suporte do mesmo a um subconjunto significativo das especificações WS-*, tais como os protocolos *WS-Security* e *WS-Trust* entre outros. A fiabilidade das mensagens também não foi descurada no WCF permitindo este o uso do protocolo *WS-ReliableMessaging* para esse efeito. O *Windows Communication Foundation* utiliza mensagens SOAP na comunicação entre processos, garantindo desta forma a interoperabilidade com outras plataformas *Web Service* ou qualquer outro processo que utilize este protocolo na formação das suas mensagens. O WCF permite à equipa de programação definir o tipo de codificação pretendido para as mensagens, podendo escolher entre XML e binário. De notar que a utilização do formato binário para a codificação de mensagens só deverá ser utilizado em cenários onde todas as extremidades de comunicação envolvidas utilizam WCF.

O WCF distingue dois módulos *WCF Service* e *WCF Client*. O primeiro implementa todas as funcionalidades do serviço e espera que o módulo Client estabeleça uma ligação, respondendo aos pedidos provenientes deste. Este módulo disponibiliza publicamente e através da internet um contrato contendo uma interface que define métodos que disponibiliza assim como uma interface com os métodos que deverão ser implementados pelas aplicações que se ligarem a este. O módulo *WCF Client* deverá conter o contrato disponibilizado pelo serviço a que se ligar e deverá ser o responsável por inicializar a ligação com o serviço.

WCF Service

Para a implementação do canal de comunicação em WCF e consequente disponibilização de um serviço a um ou mais clientes, é necessária a definição de um WCF Service que disponibilize e implemente o serviço pretendido. O WCF Service, e tal como pode ser visualizado na Figura 5, é composto por três componentes principais:

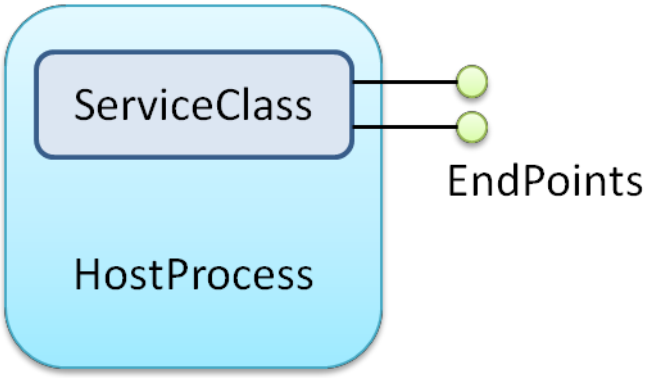


Figura 5: Ilustração dos componentes principais de um *WCF Service*

- ***Service Class***: Define e implementa os métodos a serem disponibilizados aos clientes. Esta informação define um *Service Contract* e deverá ser obrigatoriamente igual entre todas as aplicações que estão presentes em qualquer lado da comunicação. Caso algum dos métodos referidos receba ou retorne alguma variável cujo tipo não seja nativo do sistema, é ainda necessário proceder-se a uma definição do tipo de formato de dados a enviar/receber que serão suportados em adição aos já nativos no sistema. Esta definição assume a forma de um *DataContract* e tal como um *ServiceContract* também tem de ser coincidente em todos os processos que estabeleçam uma comunicação entre si.
- ***Host process***: Processo onde o serviço se encontra hospedado, podendo ser por exemplo no IIS, *Windows Service* ou *Console Applications*.
- ***Endpoint***: Conjunto de informações que caracterizam um canal de comunicação.

Para a eficiente e correcta definição de um *Endpoint*, é necessário definir as seguintes propriedades visíveis na Figura 6:

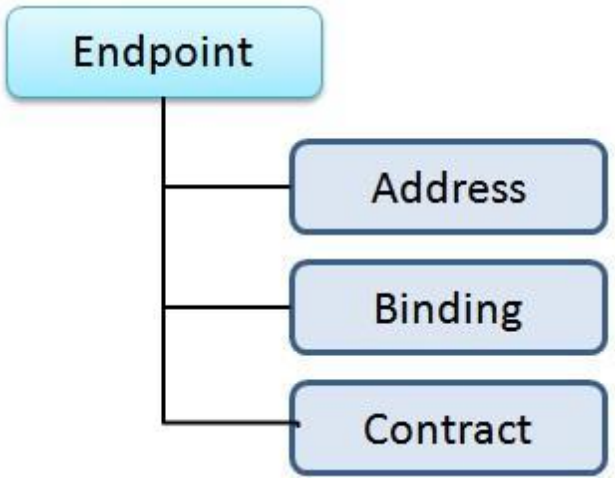


Figura 6: Ilustração da estrutura de um *Endpoint*

- ***Address***: Indica o endereço onde o serviço se encontra disponível. É composto por duas propriedades, sendo elas um URI e uma *Identity*. A primeira serve para definir a localização

do serviço através da Internet, sendo que a última é utilizada para efeitos de segurança.

- ***Binding***: Especifica o modo como o cliente comunica com o serviço. A escolha do tipo de *binding* define o protocolo de transporte a utilizar (TCP, http, …), o modo de codificação a utilizar nas mensagens (binário ou texto) e a especificação de propriedades relativas à segurança. Apesar do WCF fornecer uma lista considerável de tipos de *binding* a aplicar (BasicHttpBinding, NetTcpBinding, …), este também permite a especificação de um tipo de *binding* que não se encontre implementado por omissão no WCF.
- ***Contract***: Identifica as funcionalidades que o serviço disponibiliza ao cliente. Na sua especificação podem ser encontradas as operações passíveis de serem invocadas pelo cliente, o formato das mensagem, o tipo de dados de entrada a utilizar para invocar a operação e o tipo de resposta que o cliente deverá receber após a realização de um determinado pedido.

WCF Client

Para se poder estabelecer uma ligação entre aplicações utilizando o WCF para além da criação do serviço, é necessário também proceder à implementação de um cliente partilhe todos os termos (serviços, protocolos de comunicação) definidos no serviço. Um *WCF Client* é um objecto local que representa um dado WCF Service, denominando-se habitualmente por proxy. O proxy será o responsável pela comunicação com o serviço, trocando mensagens com este e disponibilizando, de forma transparente, todas as funcionalidades do serviço ao utilizador, tal como é possível verificar na Figura 7:

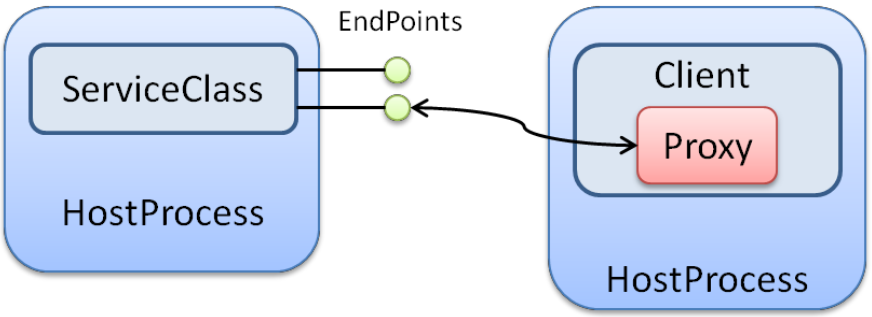


Figura 7: Iustração da forma de ligação entre um *WCF Service* e um *WCF Client*

Para se poder implementar correctamente o *proxy* é necessário saber previamente o contrato que o serviço com o qual se pretende comunicar disponibiliza. Para a obtenção desta informação é possível utilizar uma funcionalidade disponível no Visual Studio ou através da execução do programa "svcutil" na linha de comandos. Caso o serviço só seja acessível através da Internet, ambos os serviços, são capazes também de gerar toda a informação necessária se lhe for dado o URL onde o serviço se encontra hospedado, e este disponibilizar essa informação através da Internet.

2.2.2 - Google Wave [GWAPI,GWARQ]

Tal como foi referido anteriormente, a plataforma e API do *Google Wave* foram na sua maioria disponibilizadas publicamente, de forma a possibilitar o desenvolvimento de extensões para o mesmo, por parte de qualquer entidade. Esta API encontra-se orientada e optimizada para o desenvolvimento de ferramentas colaborativas, disponibilizando por omissão diversas funcionalidades para esse efeito.

O *Google Wave* apresenta os seguintes componentes visíveis na Figura 8:

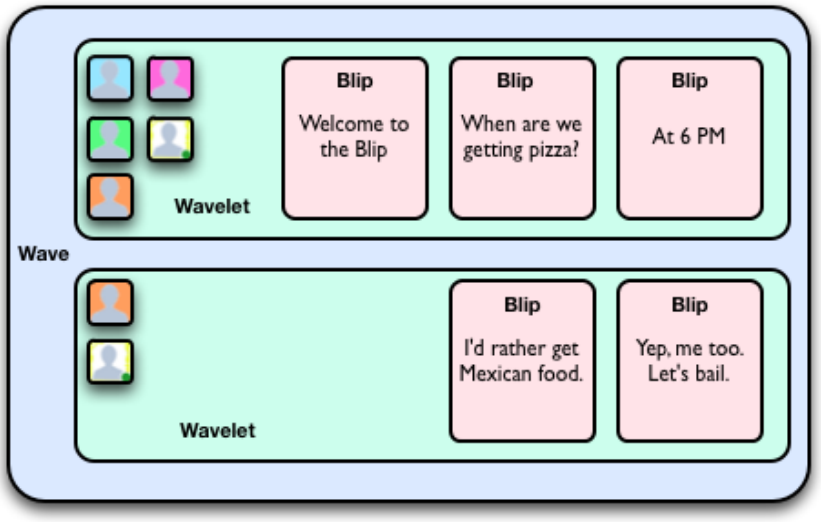


Figura 8: Ilustração da estrutura de algumas entidades que fazem parte do protocolo do Google Wave[GWAPI]

- ***Wave***: Consiste num conjunto de Wavelets e possuí um *id* único sob a forma de nome@dominio.
- ***Wavelet****:* Uma *Wavelet* possui um *id* único no universo de uma determinada Wave. Uma *Wavelet* contém um ou mais *documents* assim como um ou mais *Participant*s. Todos os participantes que constam numa determinada *Wavelet* possuem permissão para ler e escrever em qualquer conteúdo pertencente à mesma *Wavelet.*
- ***Document****:* Possui um id único dentro de um determinado *Wavelet* e é neste que reside o conteúdo introduzido e partilhado pelos utilizadores.
- ***Blip****:* Conteúdo inserido pelos utilizadores num determinado documento
- ***Data document***: Contém informação que embora não seja visível aos utilizadores, ajuda a construir/verificar o conteúdo inserido pelos mesmos (dicionário, histórico de alterações, …)
- ***Participant****:* Guarda a informação relativa a um utilizador que tem permissão para visualizar/editar todo o conteúdo residente numa *Wavelet*
- ***Extension****:* Mini-aplicação que corre numa determinada *Wave.* Uma extensão pode ser um *Gadget* ou um *Robot*
- ***Gadget***: Aplicação que adiciona funcionalidades/recursos numa determinada *Wave*
- ***Robot****:* Participantes autónomos desenvolvidos com inteligência artificial. Um *Robot* pode interagir com utilizadores ou retribuir informação que é interna ou externa a uma *Wave.*

O protocolo utilizado para a comunicação entre instâncias pelo *Google Wave* consiste numa extensão ao protocolo XMPP [20]. O *Google Wave* utiliza as funcionalidades de resolução de IP e portas assim como autentificação TLS (Transport Layer Security) e encriptação de conexão definidas no referido protocolo.

2.2.3 - LiveCycle Collaboration Service [LCCS]

A LiveCycle Collaboration Service é uma plataforma desenvolvida pela Adobe, que permite a implementação de funcionalidades comunicativas e colaborativas em aplicações desenvolvidas em Flash ou Flex. Por exemplo a aplicação Edoboard referida anteriormente no presente documento, foi desenvolvido com o auxílio desta tecnologia [ASC].

A LCCS implementa uma arquitectura do tipo cliente/servidor, onde as aplicações residem do

lado do cliente e os serviços são disponibilizados no servidor. A plataforma apresenta-se os componentes ilustrados na Figura 9:

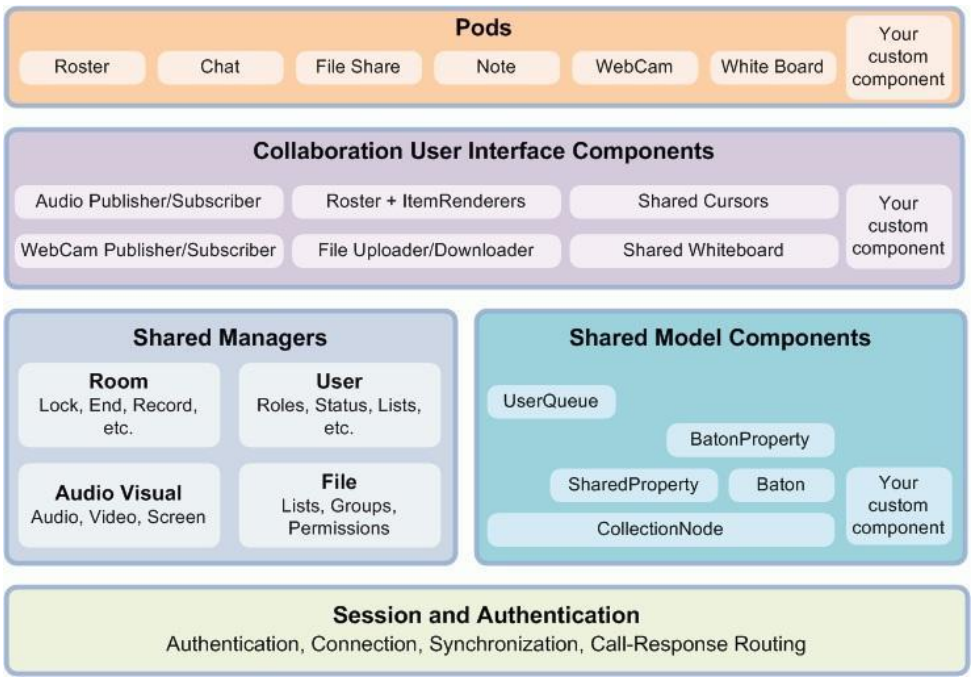


Figura 9: Módulos disponibilizados pelo LCCS

- ***Session and authentication***: Este módulo contém as funcionalidades referentes à conexão e manutenção de uma ligação a um determinado servidor. A autenticação dos utilizadores assim como a gestão das ligações de diferentes utilizadores a um serviço é realizada com recurso ao presente módulo.
- ***Shared Model Components****:* Componente de baixo nível que possibilita a definição da forma como a colaboração é implementada num determinado recurso. Caso o utilizador pretenda, pode definir uma nova forma de colaboração de um determinado recurso no que diz respeito à segurança e resolução de problemas de concorrência do mesmo.
- ***Shared Managers***: Este módulo disponibiliza as quatro APIs fundamentais no desenvolvimento de uma aplicação utilizando LCCS. Cada uma das APIs disponibiliza ferramentas de criação, gestão e customização de recursos como sessões, utilizadores, áudio visual e ficheiros.
- ***Collaboration User Interface Components***: Este componente disponibiliza interfaces gráficas customizáveis para os recursos mais utilizados na adição de ferramentas colaborativas a uma aplicação.
- ***Pods****:* Neste módulo estão presentes mini-aplicações que disponibilizam as funcionalidades mais utilizadas na implementação de ferramentas colaborativas numa aplicação.

No que diz respeito à implementação de comunicação entre os diversos processos, a plataforma disponibiliza os protocolos RTMP e RTMFP e permite a criação de canais de comunicação baseados nos protocolos TCP, UDP. Para além da implementação de canais de comunicação cliente/servidor, a plataforma oferece ainda a possibilidade de estabelecer um canal de comunicação cliente/cliente utilizando uma solução *peer to peer* própria.

2.3 - Conclusões

No presente capítulo foram apresentadas várias aplicações e tecnologias passíveis de ser inseridas no âmbito do presente trabalho. No que diz respeito às aplicações, apesar de algumas delas implementarem algumas das funcionalidades pretendidas para a aplicação, algumas destas apresentam também algumas falhas no que diz respeito a funcionalidades de edição de uma tela. Destas falhas é de referir o descontinuação do Google Wave produto que foi a grande aposta da empresa Google no ano 2009.

No que diz respeito a tecnologias foram analisadas três *framework* que poderiam ser utilizadas para desenvolver uma aplicação colaborativa e com funcionalidades de comunicação. Estas tecnologias foram analisadas pois são as ferramentas disponibilizadas pelas principais empresas de software que disponibilizam ferramentas orientadas à temática aqui estudada. Das três analisadas a que aparenta ter um maior nível de compatibilidade com a aplicação UbiStudio é a *framework* da Microsoft WCF. Este facto prende-se essencialmente por utilizar tecnologias compatíveis com a aplicação o que levará a uma melhor integração e à não inserção de novas restrições à aplicação final.

Capítulo 3

Concepção de uma Ferramenta de Comunicação

No presente capítulo é analisado o problema proposto, e feito um primeiro levantamento dos passos a tomar para o resolver.

A proposta do presente trabalho, propunha a implementação de uma ferramenta de comunicação e colaboração que permitisse a partilha de uma tela de desenho de um quadro interactivo em tempo real e com recurso a uma ligação à internet. Para o desenvolvimento de uma ferramenta com estas características, deve-se desde logo analisar os seguintes aspectos:

- Aplicação onde a mesma será utilizada
- Tecnologias a utilizar
- Cenário de utilização da ferramenta

3.1 – Aplicação alvo

O produto final do presente trabalho deverá consistir numa nova versão da aplicação UbiStudio sendo que esta deverá disponibilizar ferramentas colaborativas de edição da tela de desenho. Desta forma, a plataforma foi desenvolvida tendo em conta as características e restrições desta aplicação. Apresenta-se de seguida na Figura 10 a interface actual desta aplicação, assim como as suas funcionalidades[Ubi].

Figura 10: Aplicação UbiStudio

O UbiStudio disponibiliza as seguintes funcionalidades:

- **Tela Principal** - Área onde serão desenhados e inseridos todos os elementos de uma apresentação.
- ***Tool* Box** - Paleta de ferramentas disponíveis para criação de conteúdos.
- **Selecção** - Ferramenta para seleccionar objectos da tela.
- **Caneta** – Permite escrever ou desenhar sobre a tela.
- **Caneta Mágica** – Criação através do desenho livre na tela de formas geométricas, que serão reconhecidas pelos sistema. (Nota: o sistema apenas reconhece polígonos convexos).
- **Marcador -** Permite realçar uma determinada zona da tela
- **Borracha** - Ferramenta para apagar objectos da tela.
- **Selecção de cores** - Altera a cor de uma forma ou de uma linha, tendo em conta o que está seleccionado na altura.
- **Ferramenta de preenchimento** - Altera a cor de preenchimento das formas para a cor que estiver seleccionada no momento da utilização da ferramenta.
- **Espessura** - Controla a espessura do contorno de uma forma ou de uma linha.
- **Formas** - Desenha formas pré-definidas.
- **Setas** - Disponibiliza vários tipos de setas para utilizar na apresentação.
- **Caixa de Texto** - Insere blocos de texto, que podem ser formatados, copiados e editados naturalmente.
- **Gravação** - Grava a apresentação ou uma parte desta. Pode também capturar o ecrã ou uma área para uma imagem.
- **Cortina** - Esta ferramenta cobre todos os objectos da tela actual e é completamente redimensionável, permitindo revelar os conteúdos conforme o interesse.
- **Vista Desktop** – Permite tirar anotações sobre o Ambiente de Trabalho ou utilizar aplicações do sistema sem ter que fechar a aplicação.
- **Adicionar conteúdos Multimédia** – Possibilidade de criar apresentações utilizando vídeos, áudio, imagens e componentes Flash.
- ***Undo*** - Volta atrás, desfazendo as últimas acções, sequencialmente.
- ***Redo*** - Recupera as acções desfeitas com a funcionalidade anterior, também de forma sequencial.
- **Painel de Pré-Visualização** – possibilidade de visualizar e controlar os slides que fazem parte da apresentação. Permite a adição, remoção, reordenamento ou copia de elementos de uns slides para os outros.
- **Navegador de Páginas** – Permite a navegação sequencialmente entre os slides da apresentação.
- **Controlador de Zoom -** Controla o zoom actual da tela.
- **Menu da aplicação** – Permite a definação das preferências da aplicação, guardar a apresentação, importar ficheiros de outras aplicações, exportar para .PDF, imagens e outros formatos.
- **Reciclagem** – Permite a recuperação de objectos apagados através de um simples "drag and drop" da reciclagem para o slide actual.

Embora o UbiStudio tenha sido desenvolvido com o objectivo de fornecer ferramentas de edição para um quadro interactivo, este não limita o seu uso a este tipo de *hardware*. A aplicação poderá ser utilizada em qualquer computador que possua os seguintes requisitos [UbiInfo]:

- Computador compatível com Windows com Pentium IV ou equivalente com 512 Mb de RAM
- Sistema Operativo Windows 7, Vista ou XP

- 40 MB de espaço disponível no disco rígido, até 900 MB caso seja necessário instalar o .Net framework 4.0 e os codecs para reprodução de vídeo.
- Placa de vídeo a cores SVGA de 65K ou outra compatível com o mesmo desempenho

A possibilidade da utilização do UbiStudio num computador de secretária ou num portátil torna possível a implementação da partilha de uma tela de desenho entre vários utilizadores, sem a obrigatoriedade de estes, deterem um quadro interactivo. Para além disso, este software pode ser utilizado em vários tipos de quadro interactivo independentemente do seu fabricante e características, desde que os mesmos possuam as características a nível de *hardware* acima mencionadas.

No que diz respeito a restrições/características adicionais desta aplicação, vale a pena referir que a mesma foi desenvolvida utilizando as tecnologias *WPF* (*Windows Presentation Foundation*) e C#, ambas disponibilizadas pela *Microsoft*, pelo que o desenvolvimento da plataforma de colaboração deverá ser compatível com estas tecnologias.

3.2 – Sistema Operativo e tecnologias a utilizar

Sistema Operativo

Tendo em conta que a ferramenta desenvolvida deverá ser utilizada pelo UbiStudio, o sistema operativo escolhido para desenvolver a ferramenta e com o qual a mesma será compatível será o Windows. Apesar de esta escolha possuir a desvantagem da limitação do uso da ferramenta a este sistema operativo, a aplicação alvo (UbiStudio) também possui a mesma restrição pelo que não estão a ser adicionadas restrições ao produto final.

Tecnologias

No que diz respeito às tecnologias utilizadas, e tendo em conta as tecnologias do *UbiStudio*, foi seleccionada a linguagem C# para o desenvolvimento da ferramenta de forma a facilitar a sua integração. Após esta escolha, e tendo em conta o capítulo "Estado de Arte" presente no presente documento, a *framework* de comunicação escolhida foi a WCF (*Windows Communication Foundation)*. Para a realização desta escolha foi tido em conta o Sistema Operativo com a qual a ferramenta teria de ser compatível e a linguagem que seria utilizada para o desenvolvimento da ferramenta. De todas as estudadas, foi a WCF que apresentou a maior compatibilidade visto ela ser um produto também da *Microsoft* e uma das linguagens de desenvolvimento da ferramenta ser o C#.

A ferramenta utilizada para o desenvolvimento da plataforma de comunicação foi o Microsoft Visual Studio 2010, devido à sua compatibilidade com todas as tecnologias acima mencionadas, e ao facto de o próprio UbiStudio ter sido desenvolvido com recurso a esta ferramenta. Através da utilização desta ferramenta, prevê-se mais facilitada a integração final entre os módulos desenvolvidos e a aplicação UbiStudio.

3.3 – Cenário de Utilização

Como já foi referido anteriormente, a ferramenta a produzir será utilizada posteriormente para a adição de funcionalidades colaborativas e de comunicação ao software UbiStudio. No entanto, este software ainda se encontrava em fase de desenvolvimento, aquando da proposta e início de trabalhos referentes ao presente trabalho. Devido a este facto, foi então decidido desenvolver-se uma *framework* independente do UbiStudio que permitisse a adição de funcionalidades de

comunicação/colaboração a uma aplicação. Esta *framework* deveria implementar toda a lógica de comunicação, disponibilizando então um cliente que possuísse uma API que permitisse utilizar todas as funcionalidades da *framework* desenvolvida. Desta forma, seria possível desenvolver-se uma plataforma que pudesse ser utilizada no *UbiStudio* ou em qualquer outra aplicação que utilizasse tecnologias compatíveis. Para além disso, o próprio UbiStudio poderia sofrer alterações (adição/remoção de funcionalidades) sem ser necessário alterar a lógica de comunicação/colaboração da *framework*.

3.4 – Conclusões

Apesar de a *framework* a desenvolver pretender ser genérica e utilizável por diversas aplicações, esta não deve abdicar de nenhuma das características acima mencionadas. A *framework* deve ser desenvolvida em primeira instância para a aplicação alvo, nomeadamente o UbiStudio, sendo que todas as decisões tomadas no que diz respeito ao seu desenho e tecnologias deverão sempre ter em conta as restrições do mesmo.

Capítulo 4

Proposta de Solução

Neste capítulo é apresentada a solução proposta para o problema levantado pelo presente trabalho. É apresentada primeiramente a solução da *framework* de comunicação e colaboração, assim como dois protótipos funcionais que deverão utilizar a *framework* desenvolvida para realizar trabalho colaborativo numa tela de uma sessão.

4.1 – Metodologia de Desenvolvimento

Para o desenvolvimento de uma *framework* com as características acima mencionadas, foi primeiramente realizado um levantamento dos requisitos que a *framework* deveria obedecer. Posteriormente foram identificadas as funcionalidades a disponibilizar pela *framework* de forma a tornar esta o mais geral e eficiente possível. Uma vez desenvolvidas as funcionalidades básicas da *framework*, foi então desenvolvido um pequeno protótipo que implementava algumas funcionalidades de trabalho colaborativo utilizando a *framework* desenvolvida por forma a analisar o desempenho da mesma. Por último foi desenvolvido o protótipo final que consistiu na adição de funcionalidades colaborativas e de comunicação à aplicação Ubistudio.

A implementação da *framework*, assim como de ambos os protótipos seguiu a metodologia de desenvolvimento *Sahimi Waterfall* que consiste numa variante do modelo *Waterfall* [Sashi]. Esta metodologia foi utilizada pois permite avançar ou recuar entre as várias fases de desenvolvimento permitindo assim alterar e adaptar a *framework* e/ou o protótipo utilizando o *feedback* adquirido para completar cada uma das fases. No que diz respeito aos testes realizados, estes foram maioritariamente efectuados pelo autor da presente tese assim como alguns colaboradores da empresa onde a mesma foi desenvolvida. Esta metodologia permitiu utilizar o *feedback* de cada um dos testes e reavaliar o *design* e funcionalidades da solução obtida, alterando-a e corrigindo-a se necessário. Esta metodologia foi baseada no modelo em Espiral [Spiral], e permitiu o melhoramento da solução obtida a cada uma das iterações de testes realizadas. O facto de ambos os modelos anteriormente referidos serem metodologias ágeis de desenvolvimento de Software, permitiu resolver os problemas assim que estes foram detectados mesmo que estes, tenham sido detectados numa fase de desenvolvimento diferente quando comparada com a fase que deu origem ao problema.

4.2 – *Framework*

Apresentam-se de seguida a solução proposta para a *framework* de comunicação/colaboração a desenvolver.

4.2.1 - Visão geral

A *framework* de comunicação pretende oferecer funcionalidades básicas a qualquer aplicação que pretenda implementar ferramentas de comunicação e colaboração.

4.2.2 - Funcionalidades

A *framework* deverá disponibilizar as seguintes funcionalidades:

- Criar/Terminar/Ligar /Desligar de sessão
- Troca de ficheiros entre utilizadores em tempo real
- Troca de mensagens de texto em tempo real
- Sistema de vídeo-conferência em tempo real entre utilizadores
- Partilha de um ou mais recursos entre utilizadores em tempo real
- Sistema hierárquico de permissões de utilizadores

4.2.3 - Pressupostos e dependências

A *framework* aqui proposta foi desenhada e desenvolvida para aplicações que correm numa máquina com o sistema operativo Windows. A *framework* foi desenvolvida para aplicações desenvolvidas com recurso à *framework* .Net 4.0 desenvolvida pela Microsoft e que utilizem a linguagem C#, pelo que a utilização da *framework* desenvolvida noutro tipo de tecnologia não é garantida.

4.2.4 - Utilizadores

A *framework* disponibiliza por omissão um sistema hierárquico de utilizadores com o intuito de gerir as permissões que os mesmos possuem numa determinada sessão colaborativa. A *framework* distingue quatro tipos de utilizador, visíveis na Figura 11:

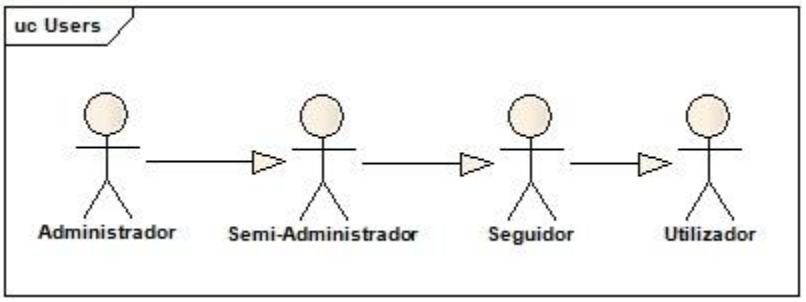


Figura 11: Utilizadores

- **Administrador:** Utilizador responsável pela criação de uma determinada sessão no servidor. Este utilizador tem acesso a todas as funcionalidades da plataforma, sendo o responsável por gerir todas as permissões inerentes aos utilizadores registados na sua sessão.

Para além disso é também responsável pela definição dos recursos e funcionalidades passíveis de serem utilizadas em modo de colaboração em tempo real.

- **Semi-Administrador:** Utilizador nomeado pelo administrador da sessão e que possui permissão para aceder a todas as funcionalidades de uma sessão.
- **Seguidor:** Utilizador que se encontra registado numa determinada sessão. Este utilizador terá acesso a todas as funcionalidades aprovadas pelo administrador da sessão onde ambos se encontram.
- **Utilizador:** Utilizador que não se encontra associado a nenhuma sessão. Este utilizador terá apenas acesso à visualização das sessões disponíveis num determinado momento, assim como a possibilidade de requisitar o acesso a uma delas.

4.2.5 - Modelo de Casos de Utilização

Apresentam-se de seguida os requisitos funcionais da *framework* desenvolvida. Estes requisitos não incluem nenhuma funcionalidade inerente a um quadro interactivo pois a *framework* foi desenvolvida de uma forma genérica e para uma qualquer aplicação que tencione implementar partilha de um determinado recurso.

4.2.5.1 - Visão Geral

Apresenta-se de seguida na Figura 12 a visão geral do modelos de casos de utilização da *framework* desenvolvida.

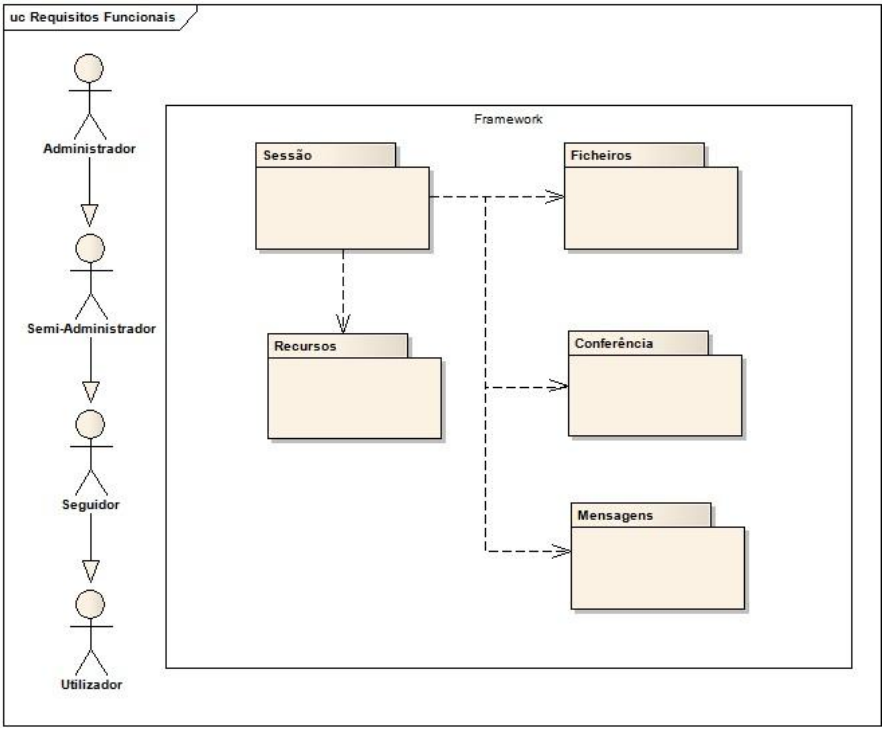


Figura 12: Visão geral dos requisitos funcionais da *Framework*

Como é possível verificar, a *framework* encontra-se estruturada em cinco módulos que serão analisados com maior pormenor do decorrer do presente documento

4.2.5.2 - Módulo Sessão

O módulo sessão é responsável pela disponibilização das funcionalidades básicas e que dizem respeito à criação e gestão de uma sessão.

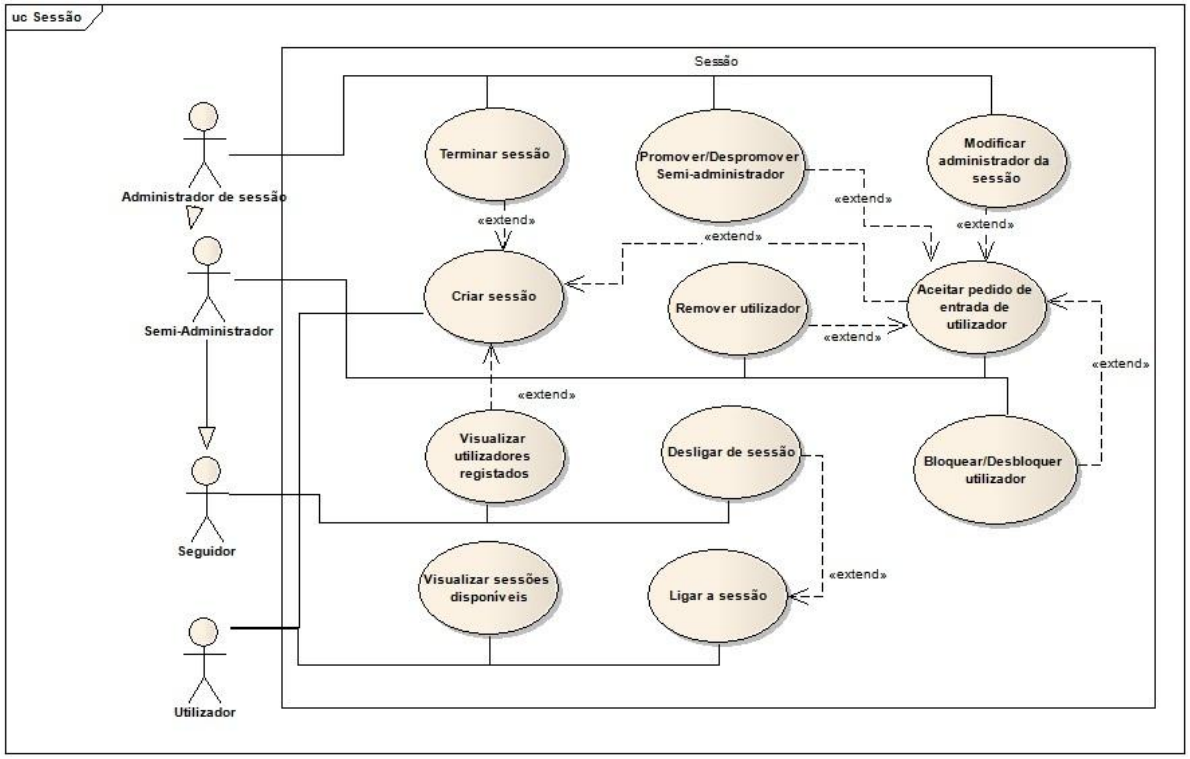


Figura 13: Módulo Sessão

Como é possível visualizar na Figura 13, o módulo sessão é responsável pela disponibilização das funcionalidades de criação, associação e término de sessões. Para além disso disponibiliza ainda a possibilidade de um administrador nomear um utilizador registado na sessão de Semi-Administrador. A funcionalidade Bloquear/Desbloquer utilizador, permite que o Administrador/Semi-Administrador retire todas as permissões de utilização de recursos a um determinado utilizador, ficando este possibilitado apenas a receber as alterações provenientes da sessão sem poder intervir na mesma. É ainda de referir que o utilizador com *ranking* Utilizador é o único que possui permissão para Criar uma determinada sessão. Isto deve-se ao facto de um determinado utilizador só poder estar associado a uma determinada sessão. Este utilizador após criar uma determinada sessão, automaticamente passa a possuir o *ranking* de Administrador da sessão que criou.

4.2.5.3 – Módulo Recursos

O módulo Recursos é o módulo responsável pela colaboração de um determinado recurso. Este módulo permite que um utilizador, após criar uma determinada sessão, adicione um recurso, assim como as respectivas funcionalidades que serão utilizadas de forma colaborativa entre todos os utilizadores da sessão. Para esse efeito, o módulo, após receber a notificação da utilização de uma determinada funcionalidade por parte de um utilizador, e após verificar se este tem permissão para a utilização da mesma, notifica todos os utilizadores registados na sessão que um determinado

utilizador utilizou uma funcionalidade de um certo recurso. Desta forma, a alteração efectuada no recurso através de uma determinada funcionalidade, pôde ser propagada por todos os utilizadores da da sessão. As funcionalidades a implementar pelo módulo, encontram-se ilustradas na Figura 14.

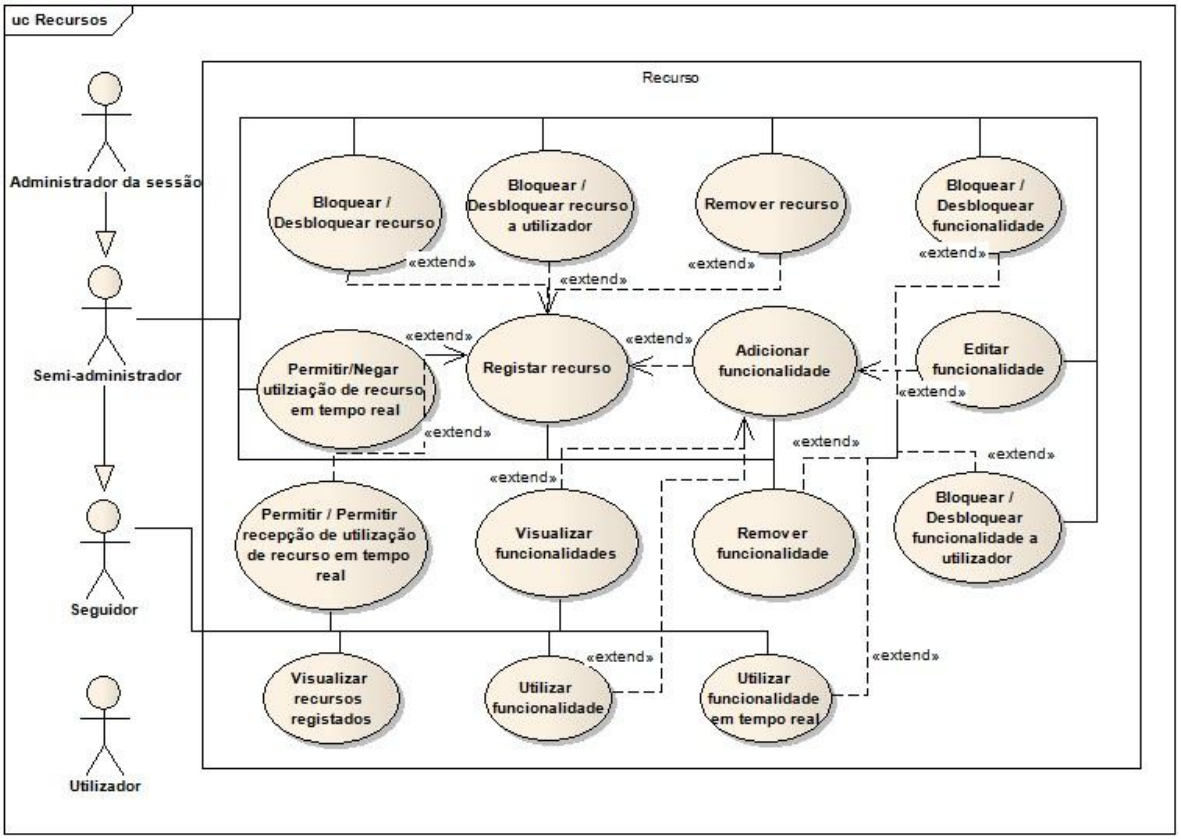


Figura 14: Módulo Recursos

Tal como é possível verificar pela figura anterior, o módulo disponibiliza um elevado leque de funcionalidades que permitem gerir os recursos de uma determinada sessão. Assim, para além de disponibilizar ferramentas que permitem adicionar/remover/editar recursos/funcionalidades, permite também que um utilizador com um determinado nível hierárquico, gira as permissões de utilização dos recursos/funcionalidades da sessão. Para esse efeito é disponibilizado o método Bloquear/Desbloquear recursos ou funcionalidades. Esta acção pode ter como alvo um determinado utilizador, ou a sessão em geral e impede que mais utilizadores, possuam permissão para utilizar um determinado recurso de forma colaborativa. É apenas de notar que este bloqueio só se aplica a utilizadores que se encontrem no *ranking* de "Seguidor", visto os "Semi-Administadores" e "Administrador" não possuírem restrições na utilização de recursos/funcionalidades da sessão.

A figura anterior apresenta ainda as funcionalidades "Utilizar funcionalidade" e "Utilizar funcionalidade em tempo real". Para melhor se perceber a diferença e o porquê da existência destas duas funcionalidades, tomemos como exemplo a implementação de trabalho colaborativo numa tela de desenho de um quadro interactivo utilizando a *framework* em análise. Neste caso, existiria então apenas um recurso registado, nomeadamente a tela de desenho, que apresentaria várias funcionalidades, sendo que uma delas seria desenhar uma linha. Para efeitos de colaboração, o acto de desenhar uma linha poderia ser dividido em três actos distintos:

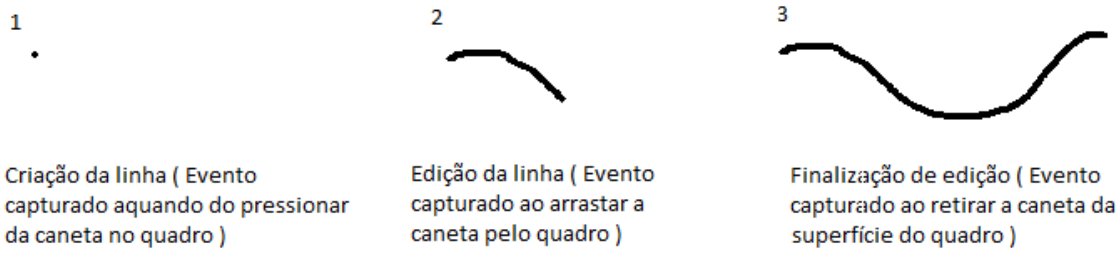


Figura 15: Exemplo de utilização de funcionalidade

As fases 1 e 2 da criação da linha presentes na Figura 15, e embora correspondam a uma acção do utilizador, poderiam não ser imediatamente partilhadas por todos os utilizadores da sessão, uma vez que o objecto ainda não foi totalmente criado. Se estas duas fases fossem descuradas, e apenas se notificasse os utilizadores da acção 3 presente na figura anterior, estes continuariam a manter-se numa sessão partilhada e igual em todas as instâncias dos utilizadores, embora o impacto visual ficasse reduzido. No entanto, este cenário seria especialmente útil em cenários em que a ligação à Internet fosse especialmente lenta. Nestes casos, seria de maior utilidade reservar as comunicações para acções definitivas e não a edição de objectos em tempo real, que estaria a "roubar" largura de banda a acções definitivas e de maior relevância para os utilizadores. Para esse efeito, a *framework* diferencia estas duas acções e permite ao utilizador definir a que melhor lhe convém num determinado instante. As acções 1 e 2 da figura anterior utilizariam então a funcionalidade "Utilizar funcionalidade em tempo real", sendo que a acção 3 utilizaria a funcionalidade "Utilizar funcionalidade". Desta forma, e se o utilizador assim o quisesse (e possuísse permissões para tal), poderia impedir todos os utilizadores ou apenas alguns (aqueles que possuíssem ligações mais lentas) de utilizar uma determinada funcionalidade em tempo real. Da mesma forma, um utilizador, após se aperceber que a sua ligação à Internet não possuí qualidade suficiente para receber todas as acções, pode impedir a recepção da utilização de funcionalidades da sessão em tempo real por parte de outros utilizadores, recebendo assim só as acções definitivas que os mesmos tomaram.

4.2.5.4 – Módulo Mensagens

O módulo Mensagens pretende disponibilizar aos utilizadores um sistema de *chat* interno de troca de mensagens de texto. As suas funcionalidades são apresentadas na figura seguinte:

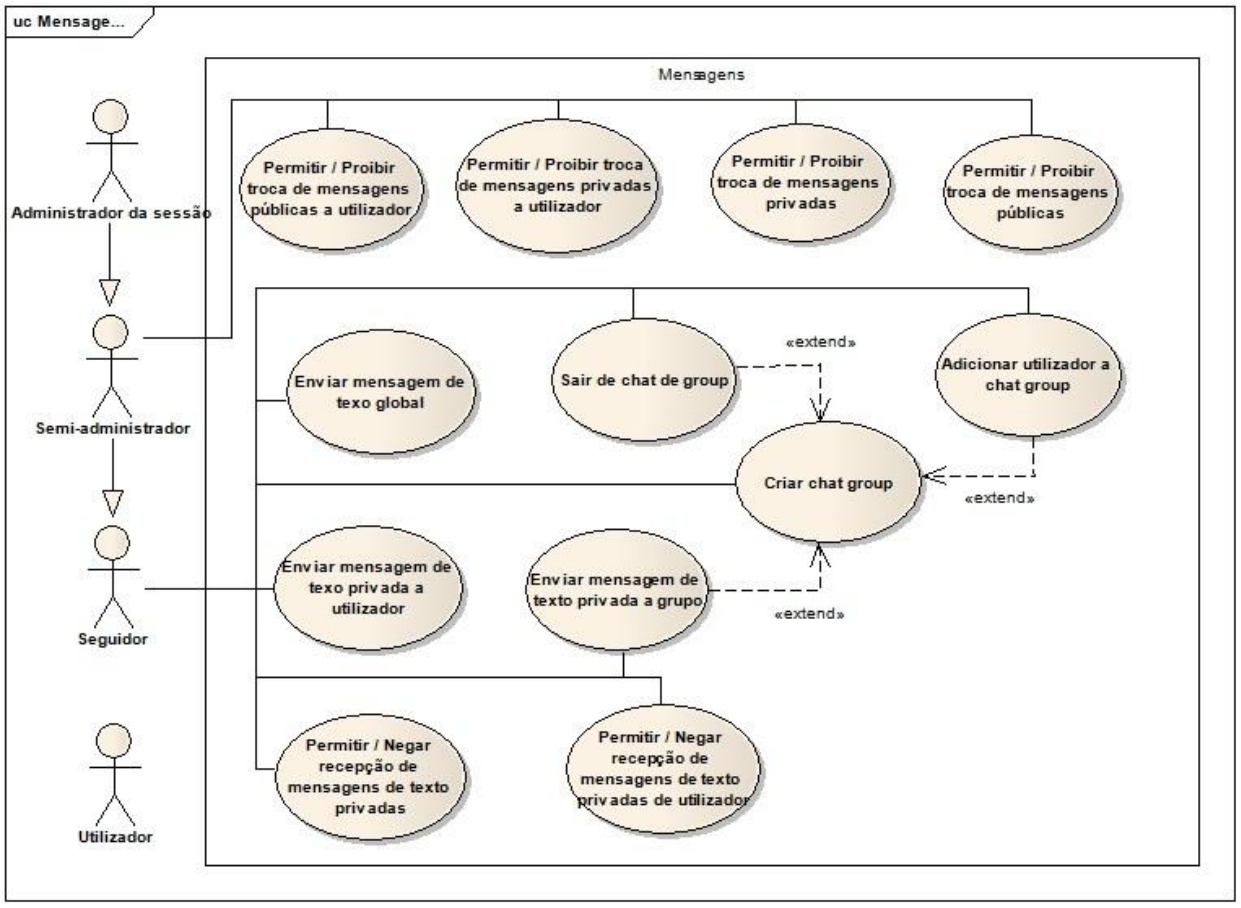


Figura 16: Módulo Mensagens

Tal como é possível visualizar na Figura 16, a *framework* disponibiliza três tipos de *chat*:

Chat Global: As mensagens trocadas neste canal são visíveis entre todos os utilizadores registados numa determinada sessão.

Chat privado entre dois utilizadores: As mensagens trocadas neste canal são apenas visíveis aos dois utilizadores intervenientes, sendo o estabelecimento e manutenção da comunicação invisível para qualquer outro utilizador não importando o seu ranking.

Chat privado entre um grupo de utilizadores: As mensagens trocadas neste tipo de *chat* são apenas visíveis para os utilizadores registados no mesmo, independentemente do seu *ranking*. A única forma de um utilizador se registar neste chat é se um dos intervenientes no mesmo o convidar. Este chat não permite que um utilizador seja removido do chat por um outro utilizador independentemente do seu ranking. A única forma de um utilizador sair de um determinado grupo de mensagens é se o mesmo assim o desejar.

Tal como nos módulos anteriores, a *framework* disponibiliza ferramentas de gestão do presente módulo. Para o efeito, um utilizador com as devidas permissões, pode proibir a troca de mensagens no chat global ("Proibir Mensagens Públicas") de forma generalizada (ficando este apenas editável pelos Semi-Administradores e Administrador da sessão) ou apenas a alguns utilizadores. É apenas de notar que os utilizadores englobados no ranking "Seguidor" e que não possuam permissões para contribuir no chat global, podem visualizar as mensagens trocadas no

mesmo. A *framework* disponibiliza também a funcionalidade "Proibir troca de mensagens privadas" que impede que utilizadores (com o ranking "Seguidor") troquem mensagens privadas entre si ou em grupos privados. Tal como na opção anterior, estes utilizadores apesar de não poderem enviar mensagens privadas, continuam a poder recebê-las. A *framework* permite ainda que um utilizador impeça a recepção de mensagens privadas de forma generalizada ou de um determinado utilizador. No caso de o utilizador impedir um outro de lhe enviar mensagens privadas apenas receberá mensagens do utilizador bloqueado se o mesmo possuir um ranking superior ao seu, ou se se encontrarem no mesmo grupo privado.

4.2.5.5 – Módulo Conferência

O módulo Conferência disponibiliza aos utilizadores um sistema de vídeo-conferência interno. As suas funcionalidades são apresentadas na Figura 17:

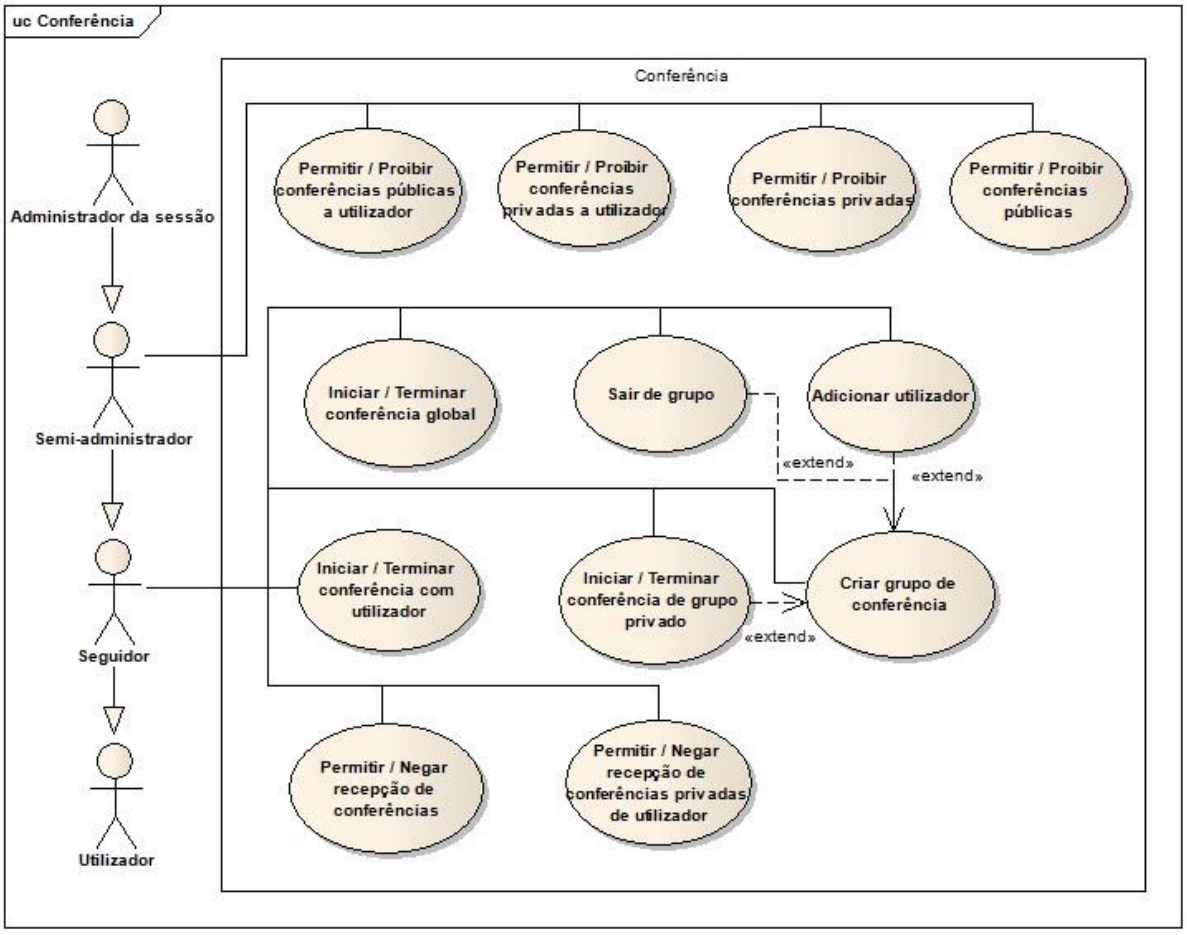


Figura 17: Módulo Conferência

Tal como é possível visualizar na figura anterior e seguindo a mesma linha do módulo "Mensagens" apresentado anteriormente a *framework* disponibiliza três tipos de conferência:

- **Conferência Global**: Uma conferência inicializada neste modo por um utilizador ficará disponível a qualquer utilizador.

- **Conferência privada entre dois utilizadores:** Neste modo, uma conferência é apenas partilhada entre os dois utilizadores intervenientes, ficando inacessível a todos os outros utilizadores. Caso o utilizador inicie mais do que uma conferência neste modo com mais do que um utilizador, apenas os seus dados áudio/vídeo serão partilhados com mais do que um utilizador sendo invisível aos utilizadores que se encontram a conferenciar consigo o facto de este utilizador estar a conferenciar com mais do que um utilizador.

- **Conferência privada entre um grupo de utilizadores:** Neste modo, todos os dados áudio/vídeo de cada um dos utilizadores, estão disponíveis a todos os utilizadores registados no grupo de momento.

Tal como em todos módulos apresentados previamente, a *framework* também neste módulo disponibiliza ferramentas de gestão. Para o efeito, um utilizador com as devidas permissões, pode proibir conferências inicializadas em modo público ("Proibir Conferências Públicas") de forma generalizada (ficando esta opção apenas disponível para Semi-Administradores e Administrador da sessão) ou apenas a alguns utilizadores. É de notar que como em casos semelhantes já explanados anteriormente, embora alguns utilizadores não possuam permissão para iniciar conferências, continuam a poder recebê-las de utilizadores que tenham essa permissão. A *framework* disponibiliza também a funcionalidade "Proibir Conferências privadas" que impede que utilizadores (com o ranking "Seguidor") iniciem conferências privadas entre si ou em grupos privados. Tal como na opção anterior, estes utilizadores apesar de não poderem enviar mensagens privadas, continuam a poder recebê-las. A *framework* permite ainda que um utilizador impeça a recepção de conferências privadas de forma generalizada ou de um determinado utilizador. Neste caso, o utilizador não receberá mais os dados áudio/vídeo do utilizador bloqueado independentemente do seu ranking.

4.2.5.6 – Módulo Ficheiros

O presente módulo permite a troca de ficheiros entre os utilizadores registados na sessão. Apresentam-se de seguida as suas funcionalidades:

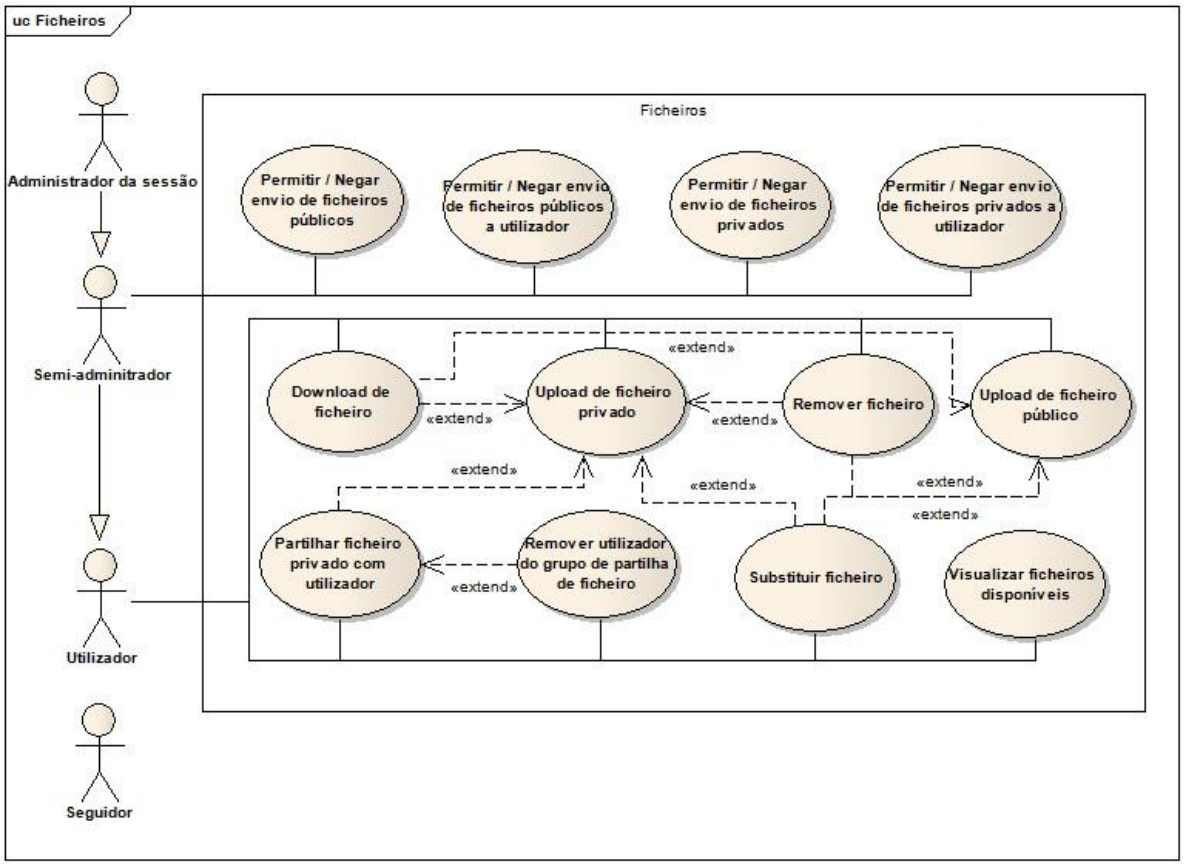


Figura18: Módulo Ficheiros

Tal como é possível visualizar na Figura 18 a *framework* disponibiliza funcionalidades básicas que dizem respeito à troca de ficheiros entre os utilizadores. Para esse efeito um utilizador, se possuir permissões para tal, poderá realizar o upload de ficheiros de uma forma pública ou privada. No caso de o ficheiro ser público, todos os utilizadores serão notificados da existência do mesmo no servidor, caso contrário, apenas os utilizadores com os quais o utilizador partilhou o ficheiro, ou utilizadores com ranking superior ao mesmo, serão notificados. Este módulo, e tal como todos os módulos anteriores, também permite que utilizadores com ranking elevado, executem uma gestão das permissões dos utilizadores com ranking "Seguidor". Para esse efeito a *framework* disponibiliza várias ferramentas (ilustradas na figura anterior) que permitem Permitir/Negar o upload de ficheiros públicos/privados.

4.2.6– Requisitos Não Funcionais

Apresentam-se de seguida os requisitos não funcionais atribuídos à *framework* e que deverão ser alvo de análise aquando do desenvolvimento da mesma.

Eficiência: Pretende-se que a *framework* apresente um tempo de resposta reduzido. Este factor é crucial pois o objectivo da *framework* é possibilitar a implementação de trabalho colaborativo. Se a *framework* apresentar uma resposta lenta, este facto será perceptível ao utilizador final e consequentemente levará a uma desmotivação e desinteresse na aplicação por parte do utilizador.

Segurança: Os dados de cada utilizador, deverão estar apenas disponíveis a utilizadores registados na mesma sessão e com autorização para tal. Assim sendo, a partilha do vídeo/áudio, assim como

das mensagens de texto e ficheiros pelos utilizadores deverão ser invisíveis a todos os utilizadores alheios à conversação/partilha.

Fiabilidade: A *framework* deverá apresentar um comportamento fiável ao longo da sua utilização, e qualquer erro deverá ser reportado à aplicação através do retorno de uma mensagem identificativa do mesmo. A *framework* deverá ainda ser capaz de recuperar de qualquer erro de forma autónoma devido ao facto de esta ser utilizada por inúmeros utilizadores ao mesmo tempo.

Usabilidade: A usabilidade da aplicação deverá ser também tida em conta no desenvolvimento da presente *framework*. Deverá ser disponibilizada uma API que permita um acesso intuitivo a todas as funcionalidades da *framework*.

Escalabilidade: Visto a *framework* ter sido desenhada para ser utilizada por inúmeros utilizadores ao mesmo tempo, a mesma deverá ser desenvolvida sempre com atenção à sua escalabilidade por forma a manter uma resposta o mais rápida e eficiente possível.

4.2.7– Arquitectura Física

A Figura 19 ilustra a arquitectura física proposta para a implementação da *framework:*

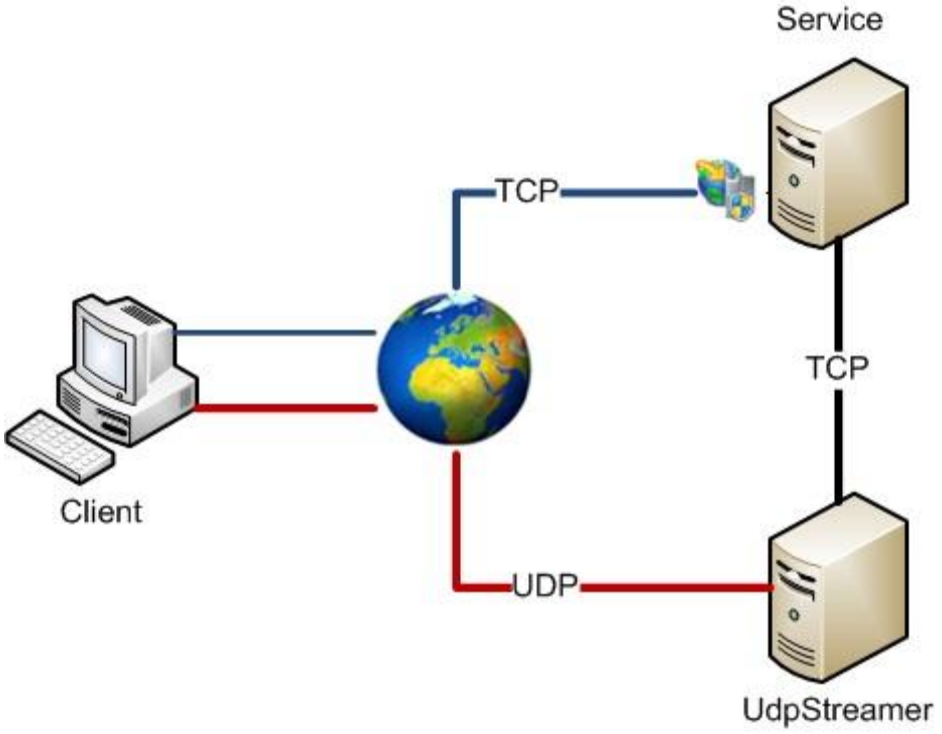


Figura 19: Arquitectura física da *Framework*

Como é possível verificar pela imagem anterior a *framework* encontra-se dividida em três componentes distintas. Cada um destes três componentes executa uma função específica e são os responsáveis por garantir uma solução eficiente e generalizada. As componentes Service e UdpStreamer são as responsáveis pela implementação de todas as funcionalidades anteriormente descritas. A componente Service é a principal componente desenvolvida, implementando todas as

funcionalidades de todos os módulos excepto do módulo conferência. A implementação deste módulo é partilhada entre a componente Service e a componente UdpStreamer, onde a componente Service implementa todas as funcionalidades de gestão de conferências e a componente UdpStreamer se limita a receber e enviar os pacotes de áudio/vídeo aos utilizadores respectivos. Desta forma é possível separar o *streaming* de vídeo do resto da lógica de colaboração, podendo mesmo estas duas componentes funcionar em máquinas separadas aumentando assim a *performance* do serviço em geral. É de notar que estas duas componentes comunicam através de uma ligação que utiliza o protocolo TCP. Este tipo de protocolo de ligação é utilizado pois garante o envio e recepção de uma mensagem entre ambas as partes, notificando uma falha caso seja impossível contactar o alvo. A componente Client é a componente que deverá ser integrada em qualquer aplicação que tencione utilizar a *framework*. Esta componente disponibiliza uma API simples e intuitiva que esconde toda a lógica da *framework* e permite estabelecer uma ligação com a componente Service e a componente UdpStreamer.

No que diz respeito aos ambientes onde as componentes irão correr, estes não diferem muito entre si. Todas as componentes deverão ser utilizadas num sistema operativo Windows, sendo que a componente Service deverá ser disponibilizada online através do software IIS 7. No que diz respeito às portas de comunicação, a máquina que aloja a componente Service deverá disponibilizar uma porta com o protocolo TCP a ligações externas. A máquina onde será alojada a componente UdpStreamer deverá libertar uma porta com o protocolo TCP e uma porta com o protocolo UDP. A porta TCP irá ser utilizada pela componente Service, sendo que a porta UDP será utilizada pela componente Client.

4.3– Protótipo 1

A seguinte secção serve para apresentar o primeiro protótipo a desenvolver, e que servirá como prova de conceito da *framework* desenvolvida.

4.3.1 - Visão geral

Tendo em conta que a *framework* deverá ser integrada numa aplicação para quadros interactivos, o presente protótipo deverá implementar algumas funcionalidades básicas que dizem respeito à implementação de uma aplicação colaborativa para quadros interactivos.

4.3.2 - Funcionalidades

O primeiro protótipo deverá implementar as seguintes funcionalidades:

- Criar/Terminar/Ligar a/Desligar de sessão
- Troca de mensagens de texto em tempo real
- Sistema hierárquico de permissões de utilizadores
- Desenho de uma linha de forma colaborativa
- Edição da linha (utilização da borracha) de forma colaborativa

4.3.3 - Pressupostos e dependências

O protótipo desenvolvido apresenta os seguintes pressupostos e dependências:

- A *framework* implementa as funcionalidades a utilizar pelo protótipo e encontra-se disponível pela Internet
- Existe uma ligação à Internet
- A aplicação corre num sistema operativo Windows

4.3.4 – Utilizadores

A aplicação deverá utilizar o sistema hierárquico disponível na *framework*, distinguindo quatro tipos de utilizadores (Administrador, Semi-Administrador, Seguidor, Utilizador) possuindo estes as permissões previamente identificadas.

4.3.5 - Interface com o utilizador

O design da interface do presente protótipo não deverá ser uma preocupação. O protótipo deverá disponibilizar apenas uma interface simples que permita testar algumas funcionalidades implementadas pela *framework* de modo a ser perceptível o sucesso/insucesso da *framework* desenvolvida. Apresenta-se de seguida a interface esperada para o primeiro protótipo desenvolvido:

Janela Inicial:

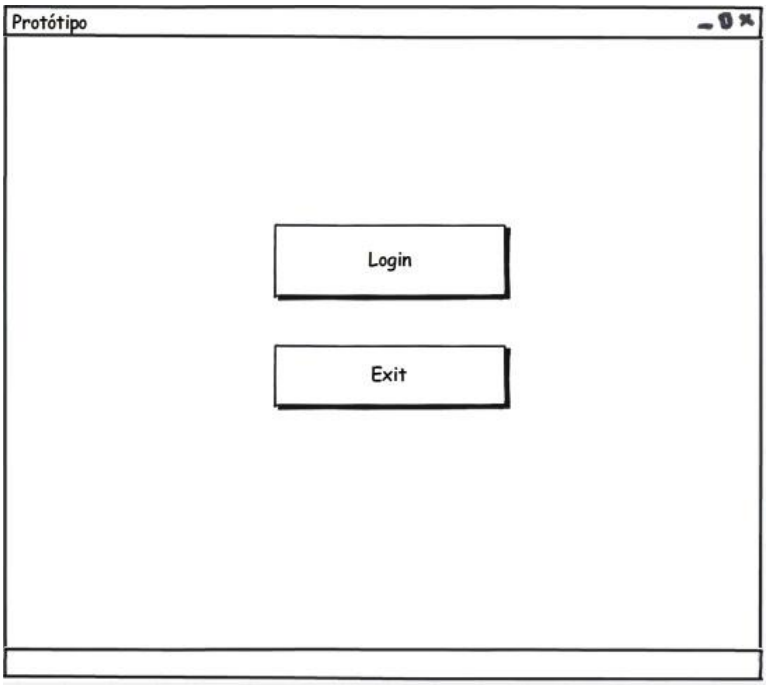


Figura 20: Interface inicial do protótipo 1

Devido ao facto de este protótipo se inserir apenas no âmbito de teste/prova de conceito à *framework* desenvolvida, o utilizador só terá acesso à tela de desenho após realizar o login no Serviço. Devido a isso a primeira janela da aplicação apenas disponibiliza a opção Login para continuar a avançar na aplicação (tal como é visível na Figura 20), redireccionando caso o utilizador assim o deseje para a janela a seguir apresentada.

Definição do nome:

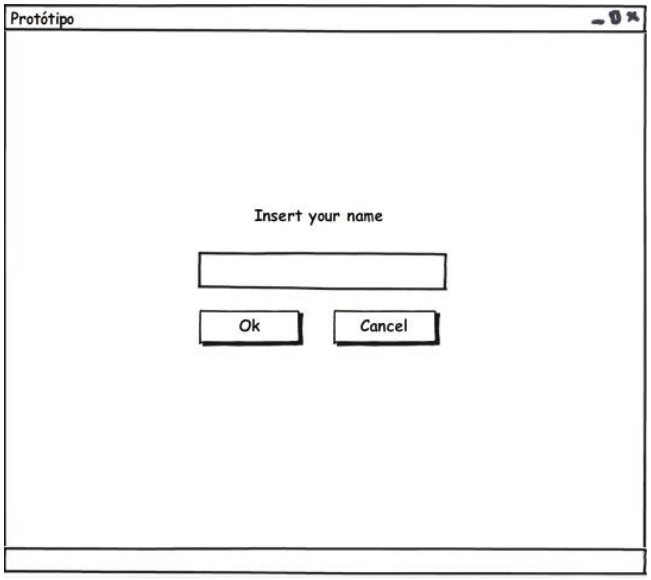


Figura 21: Interface de login do protótipo 1

Na janela ilustrada na Figura 21 o utilizador poderá definir o nome com que os outros utilizadores o associarão numa determinada sessão.

Escolha da sessão

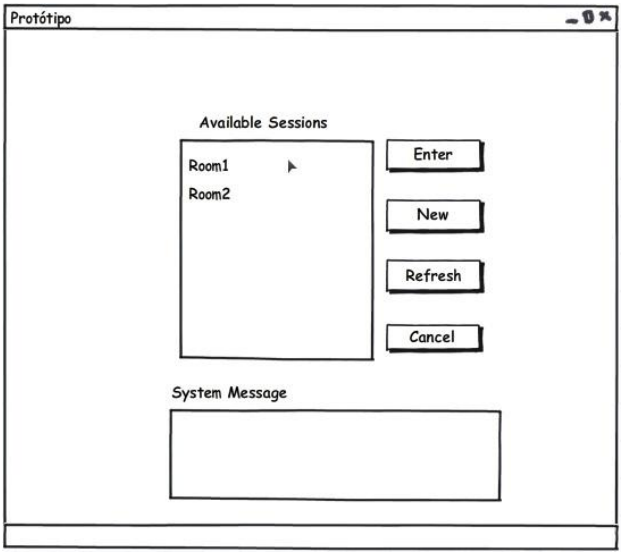


Figura 22: Interface de escolha de sessão do protótipo 1

Na janela ilustrada na Figura 22 o utilizador deverá optar por juntar-se a uma determinada sessão ou criar uma sessão nova. É de notar que embora o utilizador possua uma opção "Refresh", as sessões deverão ser actualizadas automaticamente assim que são criadas. É também de referir que a

caixa de texto associada à *tag* "System Message" não é editável, estando esta reservada a mensagens de erro provenientes da *framework*.

Criação de uma Nova Sessão:

Caso o utilizador opte por criar uma nova sessão, deverá ser apresentada interface ilustrada pela figura 23:

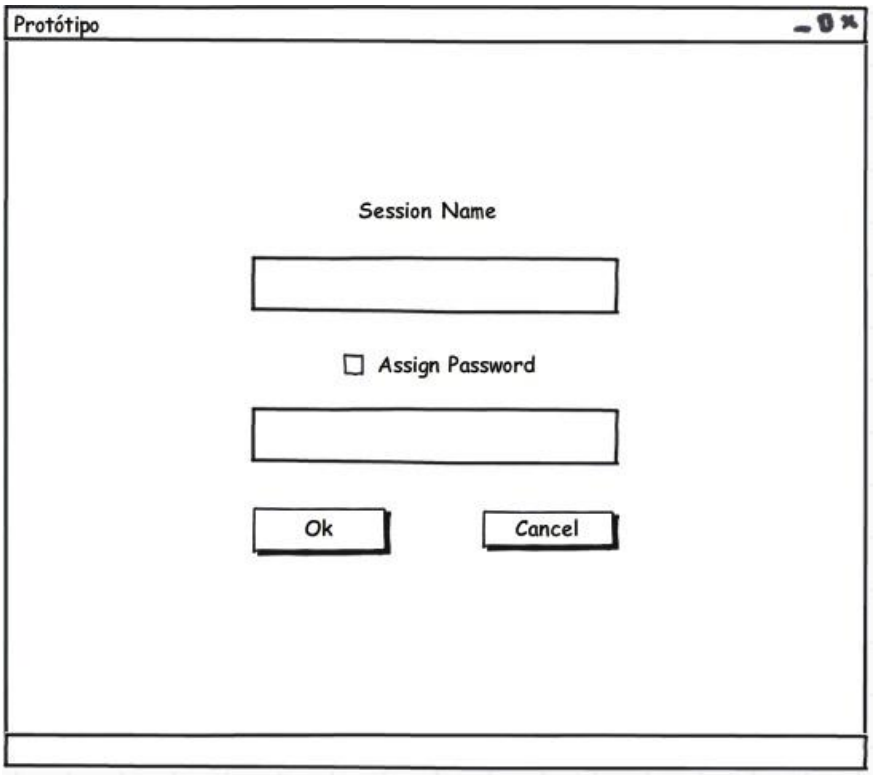


Figura 23: Interface de criação de uma nova sessão no protótipo 1

Na presente janela o utilizador deverá definir o nome da sessão, assim como determinar se a mesma deverá ser protegida e em caso afirmativo definir a *password*.

GUI principal

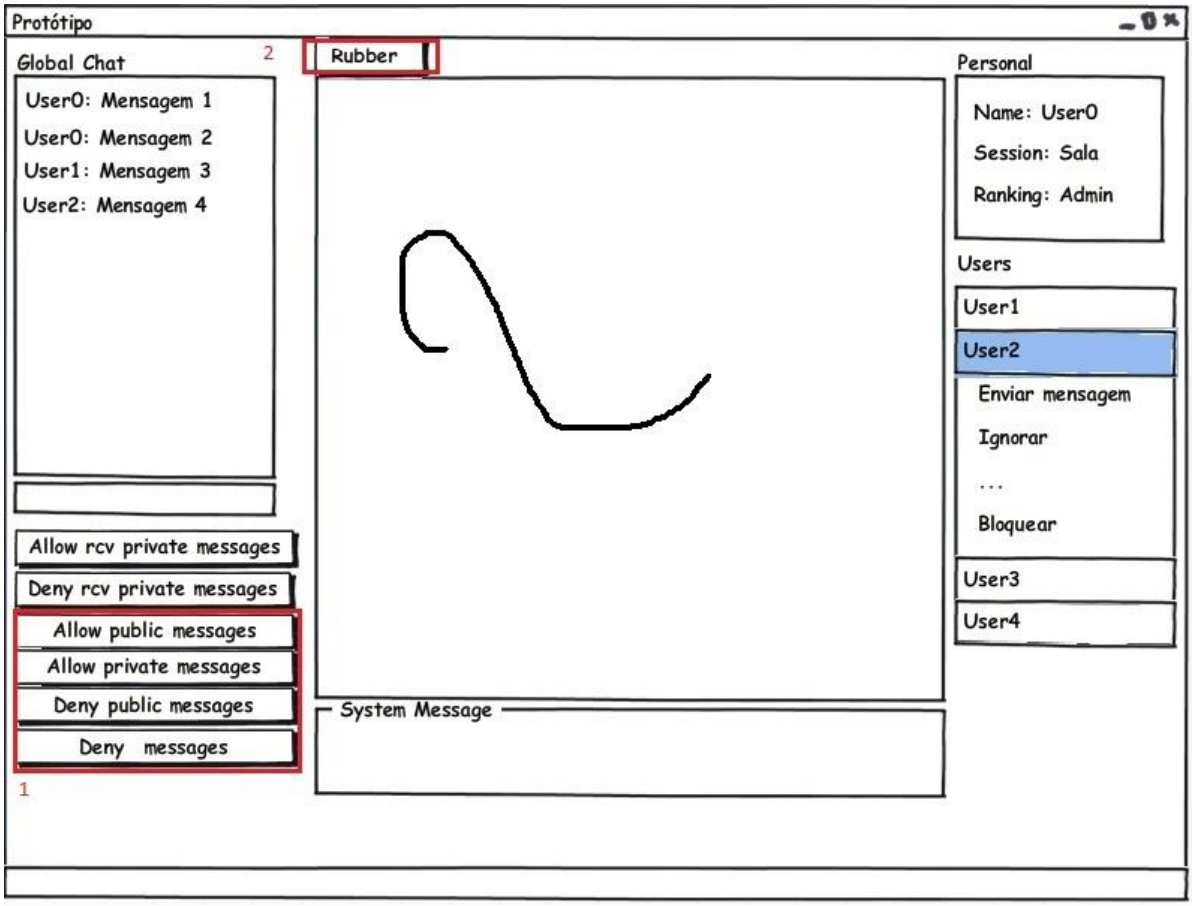


Figura 24: Interface principal do protótipo 1

A Figura 24 representa a interface principal da aplicação. Como é possível verificar esta contém uma tela de desenho que permite desenhar e apagar linhas de forma livre e colaborativa. Para mudar o tipo de interacção com a tela entre desenhar e apagar, o utilizador poderá seleccionar a opção número 2 assinalada na figura anterior. Caso este se encontre com a ferramenta desenhar linha activa, então a opção assinalada por 2 será "Rubber", caso contrário a opção será "Pen", disponibilizando assim ambas.

Como é possível visualizar na figura anterior, o utilizador poderá verificar os seus dados, assim como os dados da sessão no separador "Personal". Poderá também visualizar os utilizadores registados na sessão, assim como tomar uma acção sobre estes no separador "Users". Do lado esquerdo, encontra-se situado o sistema de troca de mensagens públicas no separador "Global Chat". Neste separador, o utilizador poderá visualizar todas as mensagens públicas de todos os utilizadores, assim como contribuir para o chat através da utilização da caixa de texto situada imediatamente abaixo do "Global Chat". O utilizador possuí ainda as opções "Allow rcv private messages" e "Deny rcv private messages", que lhe permitem negar ou permitir a recepção de mensagens privadas por parte de outros utilizadores. É de notar que o comportamento destas opções deverá estar conforme o comportamento especificado pela *framework*. As opções assinaladas pela secção 1, deverão estar só visíveis a utilizadores que possuam ranking de "Administrador" ou "Semi-Administrador, e o seu funcionamento, tal como o das funcionalidades anteriormente descritas deverá estar de acordo com a *framework*. Tal como foi expresso anteriormente, a interface em análise permite tomar uma acção sobre um utilizador, sendo que uma das acções será enviar uma mensagem privada. Aquando da selecção desta opção, deverá ser visível uma nova janela que irá ser descrita seguidamente.

Janela de mensagens privadas

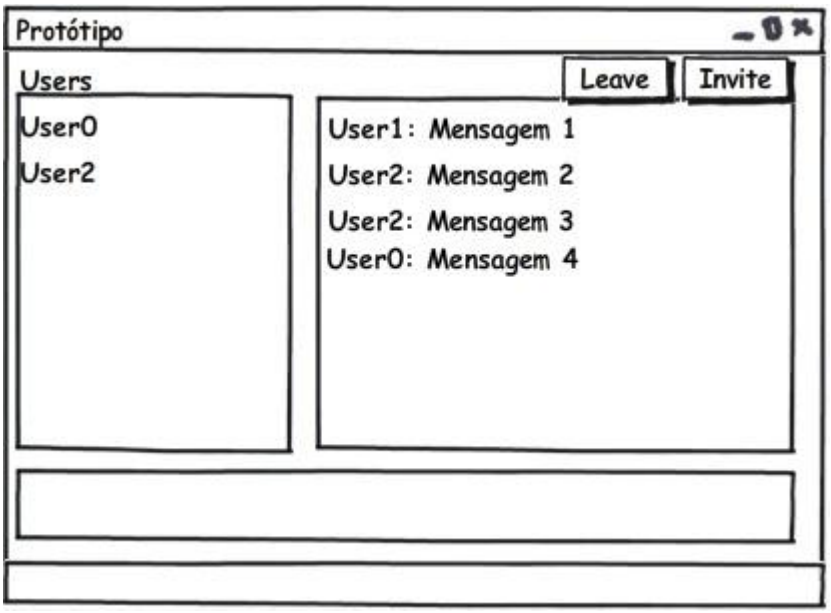


Figura 25: Interface de mensagens privadas do protótipo 1

Tal como é possível verificar na Figura 25, a aplicação disponibilizará uma janela para a troca de mensagens privadas entre um ou mais utilizadores. O utilizador poderá, utilizando esta interface, visualizar os utilizadores residentes no grupo privado, visualizar as mensagens de grupo desde que o mesmo entrou, convidar um utilizador para se juntar ao grupo, sair do grupo ou enviar uma mensagem de texto para todos os elementos do grupo.

4.3.6 - Arquitectura Física

A Figura26 ilustra a arquitectura física proposta para a implementação do protótipo em análise:

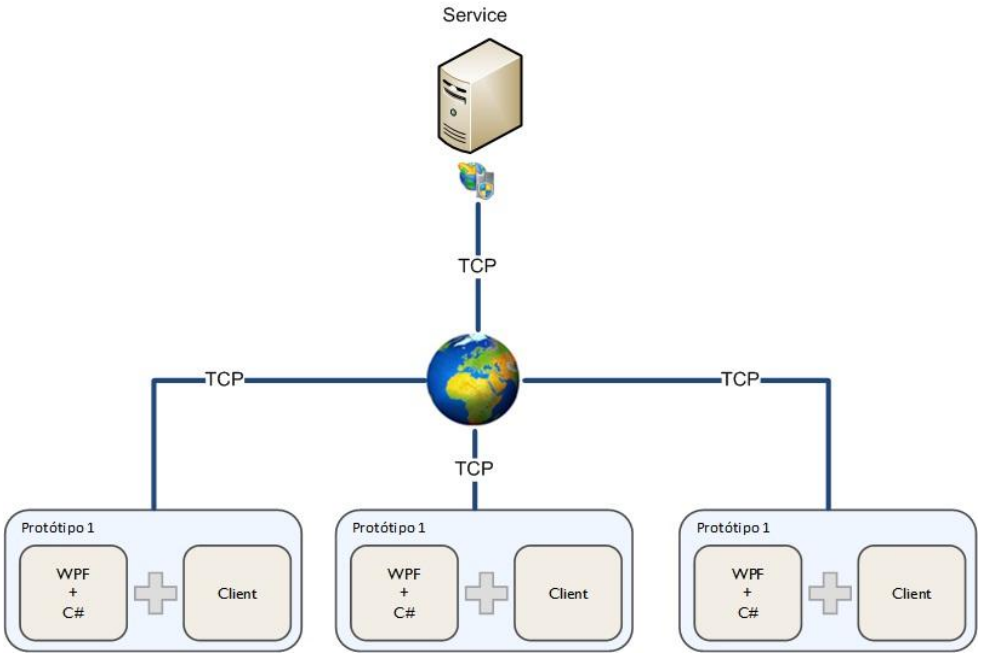


Figura 26: Arquitectura física do protótipo 1

É possível verificar pela Figura 26 que a aplicação deverá ser capaz (utilizando a componente Client) de comunicar com a componente Service através de uma ligação TCP. Após o estabelecimento desta ligação, o protótipo deverá ser capaz de criar uma sessão e de partilhar uma tela de desenho com outras instâncias do protótipo a correr em máquinas diferentes. O protótipo deverá ter como base um sistema operativo Windows e ter acesso a uma ligação à Internet.

4.4 – Integração com o UbiStudio (Protótipo 2)

A seguinte secção serve para apresentar a solução proposta para o protótipo final a desenvolver.

4.4.1 – Visão geral

Tal como já foi referido anteriormente, o protótipo final deverá consistir numa expansão de funcionalidades da aplicação Ubistudio.

4.4.2 – Funcionalidades

O protótipo final deverá implementar as seguintes funcionalidades, todas de forma colaborativa:

- Criar/Terminar/Ligar a/Desligar de sessão
- Troca de mensagens de texto em tempo real
- Sistema de vídeo-conferência em tempo real entre utilizadores
- Desenho livre sobre a tela
- Utilização da borracha
- Desenho de formas
- Desenho e reconhecimento de formas utilizando a caneta
- Edição de objectos (Cor, tamanho, Espessura)
- Ferramenta de preenchimento
- Desenho de setas
- Caixa de texto e ferramentas de edição de texto
- Adicionar imagem
- Adicionar/Remover slides (Normais e tela infinita)
- Copiar/Cortar/Eliminar objectos
- Reconhecimento de texto utilizando desenho livre
- Alterar fundo de slide
- Selecção de objectos

4.4.3 – Pressupostos e dependências

A aplicação final apresenta os seguintes pressupostos e dependências:

- A *framework* implementa as funcionalidades a utilizar pelo protótipo e encontra-se disponível pela Internet
- Existe uma ligação à Internet
- A aplicação corre num sistema operativo Windows

4.4.4 – Utilizadores

A aplicação deverá utilizar o sistema hierárquico disponível na *framework*, distinguindo quatro tipos de utilizadores (Administrador, Semi-Administrador, Seguidor, Utilizador).

4.4.5 – Interface com o utilizador

Tal como foi dito anteriormente, o protótipo desenvolvido é uma extensão de uma aplicação já existente (UbiStudio) pelo que a interface deverá ser o mais adaptada possível à interface existente na aplicação. A Figura 27 ilustra a interface actual da aplicação Ubistudio:

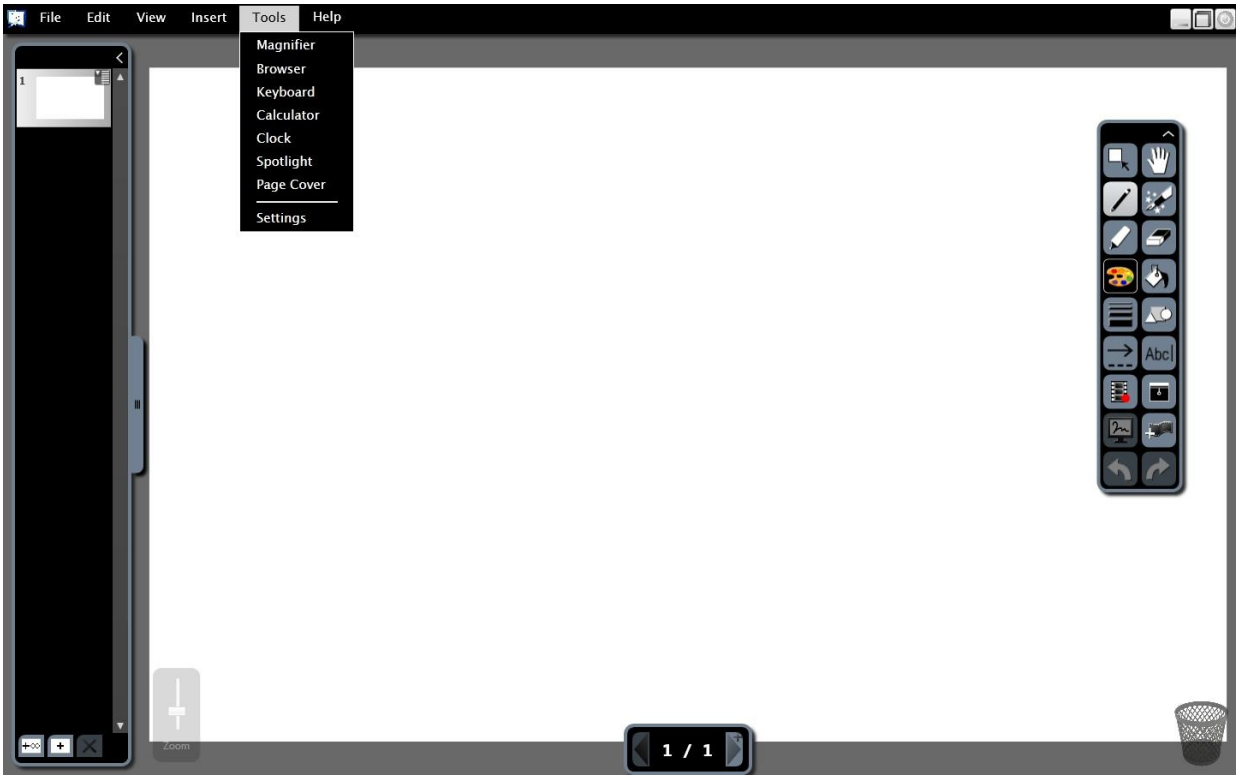


Figura 27: Interface do UbiStudio

Tendo a interface anterior em conta, apresentam-se de seguida as interfaces propostas para o protótipo:

Inicialização

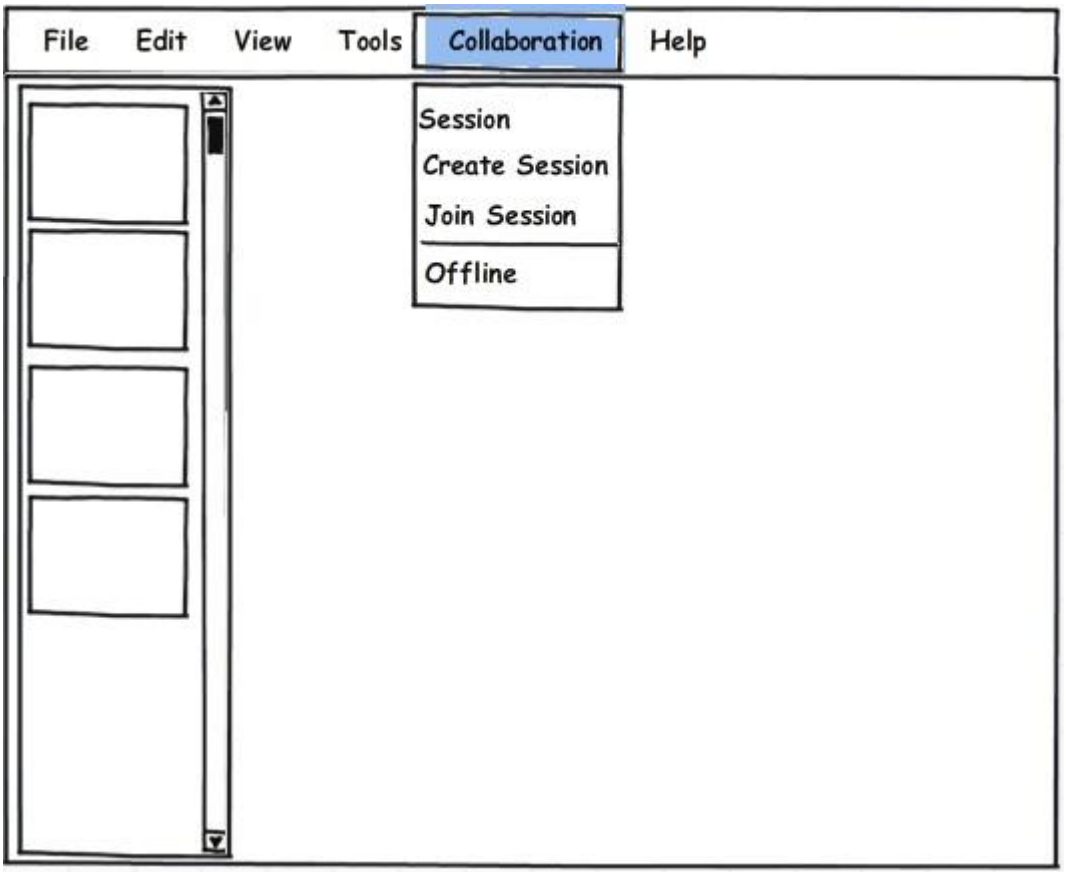


Figura 28: Interface inicial do protótipo 2

Tal é possível verificar na Figura 28, foi adicionada mais uma opção à lista de menus existente no cabeçalho da aplicação. Desta forma, é possível fornecer ao utilizador as funcionalidades de inicialização de sessão colaborativa, sem alterar radicalmente a interface da solução. O utilizador poderá então optar por criar uma sessão, ou juntar-se a uma sessão previamente criada.

Criar Sessão

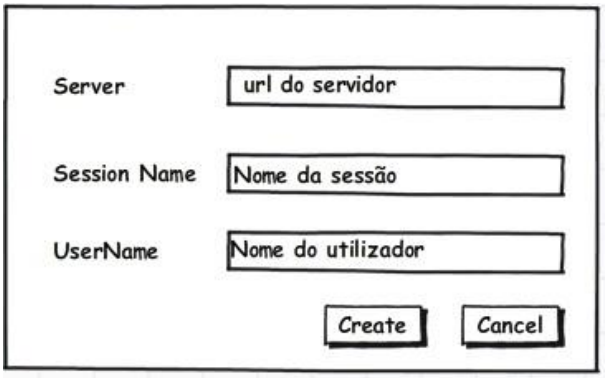


Figura 29: Interface de criação de sessão do Protótipo 2

Como é possível verificar pela Figura 29, o utilizador poderá definir o servidor ao qual deseja ligar-se, o nome da sessão criada e o nome com que deseja ser visualizado pelos outros utilizadores.

Ligar a Sessão

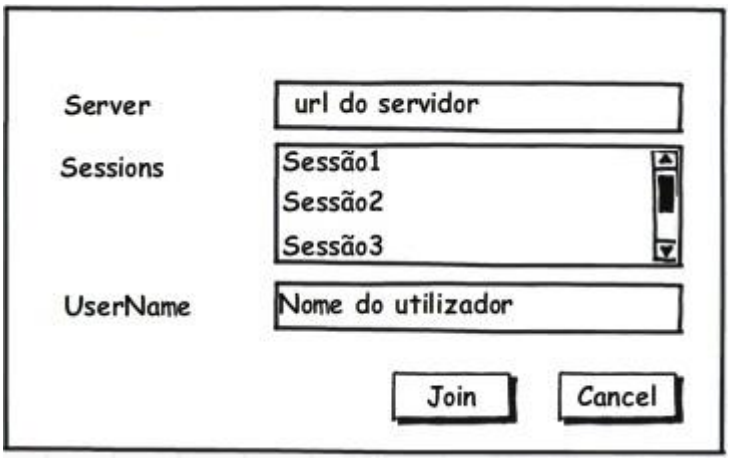


Figura 30: Interface de ligação a sessão do Protótipo 2

É possível visualizar pela Figura 30, que o utilizador poderá personalizar o servidor, assim como o seu nome na sessão alvo, podendo visualizar também todas as sessões criadas ate ao momento.

Sessão Colaborativa

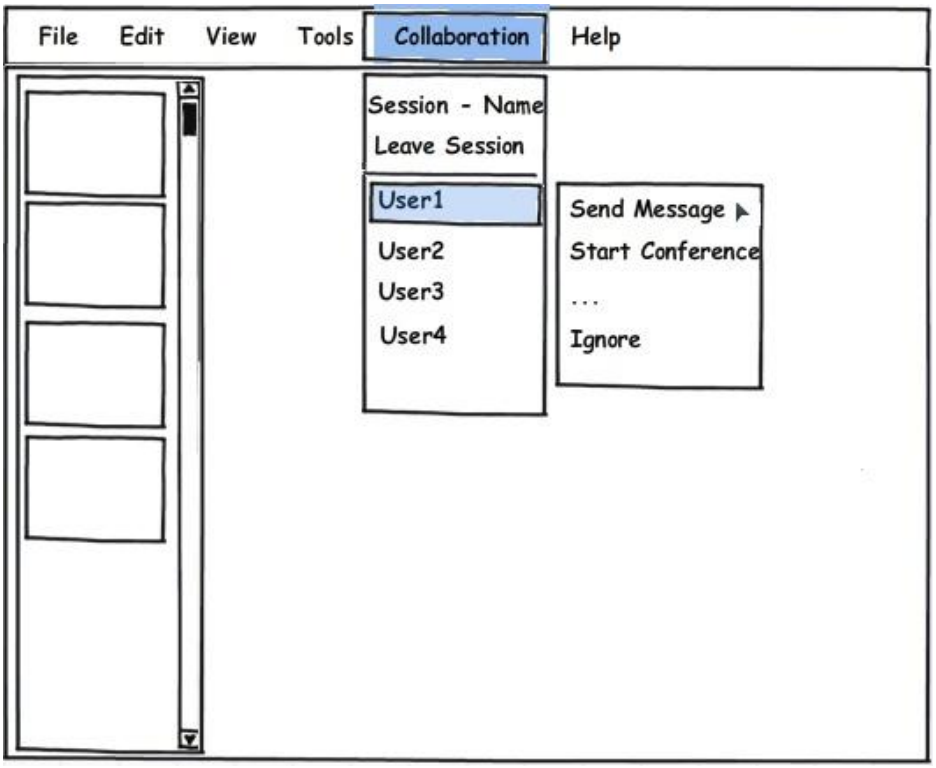


Figura 31: Interface principal do protótipo 2

A Figura 31 consiste na interface do UbiStudio após o utilizador se registar numa determinada sessão. O utilizador poderá visualizar os utilizadores registados na sessão num determinado momento, e tomar acções sobre os mesmos. A interface proposta para além de alterar pouco a interface já existente, permite disponibilizar todas as funcionalidades de uma forma intuitiva e eficaz.

Janela de Conferência

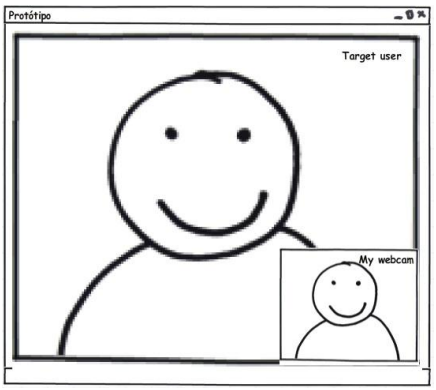


Figura 32: Interface de conferência do protótipo 2

O utilizador poderá visualizar os dados de vídeo de um outro utilizador, através da janela ilustrada pela Figura 32. Devido às limitações da interface, assim como à limitação temporal do presente trabalho, a aplicação apenas permitirá numa primeira fase, modo de conferência entre duas pessoas. O utilizador poderá abrir mais do que uma janela de conferência mas não poderá criar grupos de conferência permitidos pela *framework*.

Janela de Chat

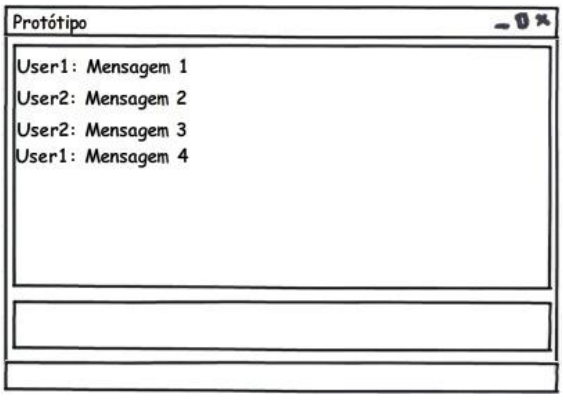


Figura 33: Interface de conversação de chat no protótipo 2

Devido às limitações impostas pela janela de conferência, o utilizador não poderá criar grupos de conversação numa primeira fase do protótipo. O utilizador poderá no entanto abrir várias janelas de conversação com vários utilizadores ilustradas pela Figura 33.

4.4.6 - Arquitectura Física

A Figura 34 ilustra a arquitectura física proposta para a implementação do protótipo final:

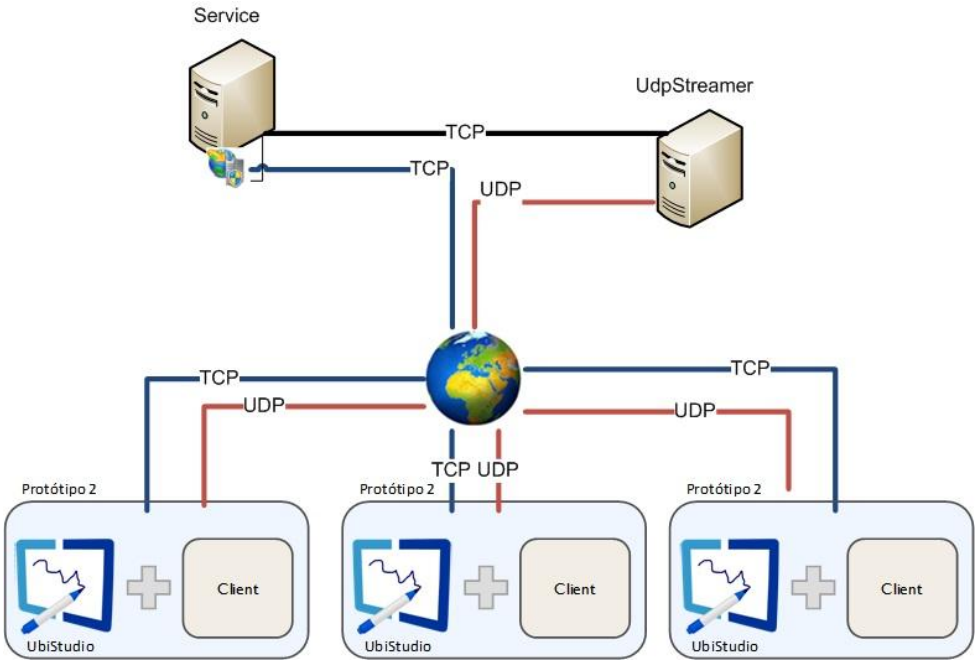


Figura 34: Arquitectura física do protótipo 2

Como é possível verificar na figura anterior, o protótipo deverá ser capaz de se ligar à componente Service e à componente UdpStreamer através de recurso a uma ligação á Internet. A componente Client deverá estar embebida na aplicação UbiStudio de forma a esta junção ser completamente transparente ao utilizador final. A aplicação deverá correr sob o sistema operativo Windows, requisito esse que já existia na aplicação UbiStudio antes da inserção da componente colaborativa.

4.5 – Conclusões

Tal como foi apresentado no presente capítulo, deverão ser desenvolvidos três produtos. Deverá primeiramente ser desenvolvida uma *framework* que permita a implementação de funcionalidades de comunicação e colaboração a uma qualquer aplicação. Posteriormente deverá ser desenvolvida uma pequena aplicação que utilizando a *framework* implementa algumas funcionalidades comunicativas e de colaboração de uma tela de desenho. Posteriormente, e depois de verificada a usabilidade da *framework* esta deverá ser utilizada para desenvolver funcionalidades colaborativas, assim como disponibilizar ferramentas de comunicação na aplicação UbiStudio.

Capítulo 5

Implementação

Tendo em conta todas as considerações apresentadas anteriormente, foram desenvolvidas uma *framework* e duas aplicações colaborativas. Os detalhes de implementação de cada uma delas são apresentados no presente capítulo.

5.1 – *Framework*

A *framework* desenvolvida é passível de ser dividida em três módulos. Estes módulos são apresentados nas próximas secções.

5.1.1 – Componente Service

A componente Service é a componente principal da *framework* desenvolvida. Esta componente consiste num serviço autónomo desenvolvido utilizando a *framework* WCF e alojado num computador recorrendo ao *software* IIS para o disponibilizar publicamente através da internet. Este serviço é responsável por definir e implementar a maioria das funcionalidades da *framework* desenvolvida. Esta componente, e tal como qualquer serviço desenvolvido utilizando a framework WCF, disponibiliza publicamente as suas funcionalidades (através de metadados) num contrato que contém as operações disponibilizadas pelo serviço e que será utilizado pela componente Client da *framework* desenvolvida. Esta definição encontra-se nos ficheiros IServer.cs e Server.cs criados pelo Microsoft Visual Studio 2010 aquando da criação de um projecto que utilize a *framework* WCF. Para além disso, o Visual Studio cria também um ficheiro intitulado Web.config, que contém a definição das características do canal de comunicação a disponibilizar, assim como a identificação dos ficheiros que contêm a interface e implementação das operações a permitir. Apresenta-se de seguida na Figura 35 um diagrama ilustrativo das componentes que constituem a componente Service desenvolvida.

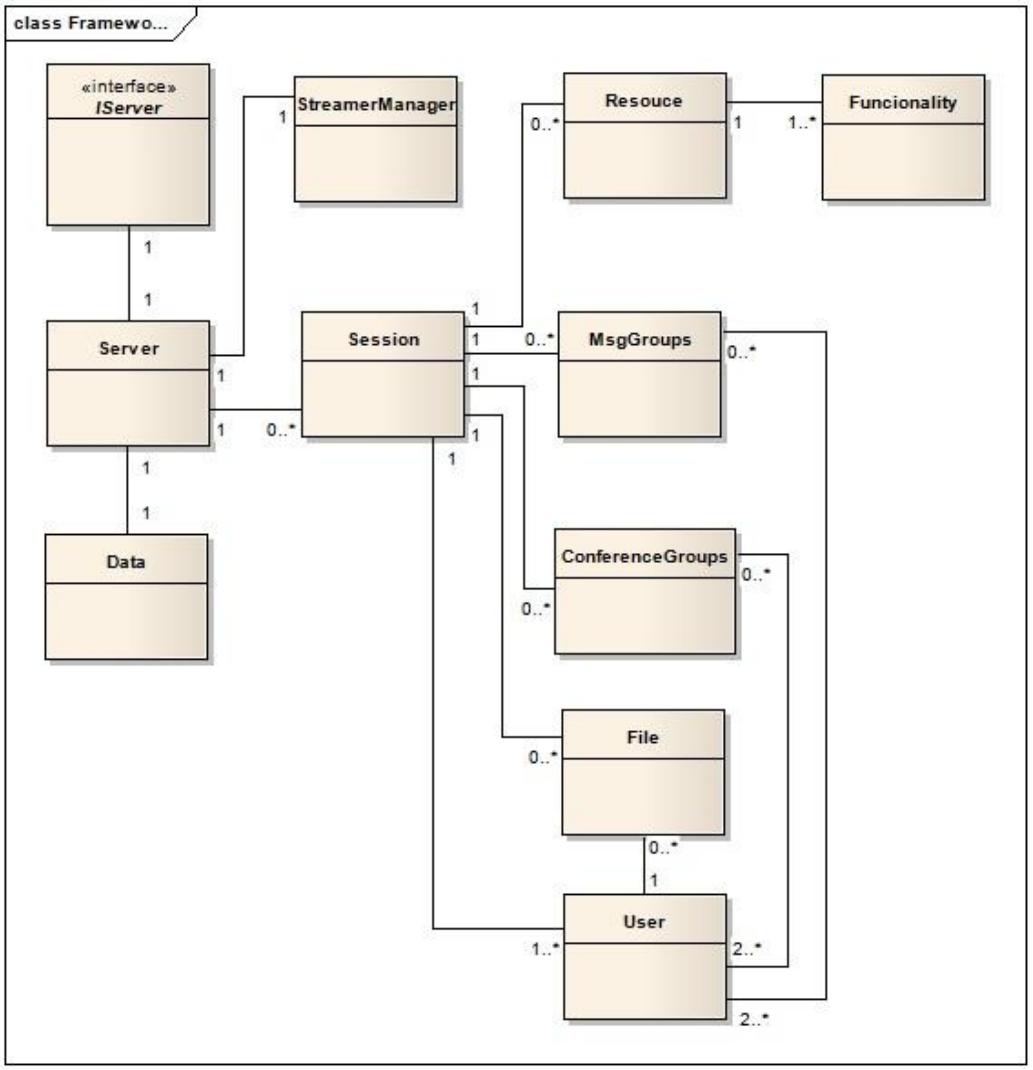


Figura 35: Diagrama representativo dos principais componentes da componente Service

IServer (Conjunto de Interfaces)

Esta componente contém uma interface denominada por IServer que contém a definição de todos os métodos disponibilizados ao exterior e que são implementados na classe Server. Todos os métodos aqui registados têm a si atribuídos os tipos de objectos serializáveis que suportam, assim como se a aplicação que invocou o método espera ou não uma resposta (a instância que iniciou a ligação e invocou determinado método poderá ou não esperar um valor de retorno). Para além disso, é também nesta componente que estão definidos os métodos que o serviço poderá invocar na instância que iniciou a comunicação com este, residindo os mesmos numa só interface denominada ICallback. Os métodos residentes nesta última interface deverão ser implementados na instância que iniciou a comunicação pois o serviço poderá invocá-los de forma autónoma desde que o canal de comunicação ainda esteja disponível. É através destes métodos que os utilizadores são notificados de acções que não foram inicializadas por si. Este ficheiro está disponível em anexo de forma a mostrar os métodos inerentes ao serviço criado. É apenas de notar na figura anterior que as ligações entre a classes MsgGroups e ConferenceGroups com a classe User possuem uma multiplicidade 2..*. Isto deve-se ao facto de ambas as classes precisarem de ver a si associados no mínimo dois utilizadores. Um grupo de conferência ou de troca de mensagens de texto não poderá ser criado com apenas um utilizador, nem continuar a existir no sistema se apenas um utilizador

restar registado nesse grupo.

Server (Classe)

A presente classe é responsável por implementar todos os métodos definidos na interface IServer. Esta componente poderá criar uma ou mais sessões, sendo esta a responsável por reencaminhar todos os métodos referentes a uma determinada sessão para a sessão correspondente. Para além disto esta classe possui ainda a capacidade de comunicar com outra componente da *framework* (nomeadamente a componente UdpStreamer) através da utilização dos métodos existentes na classe StreamerManager.

Data

A componente Data contém um conjunto de classes que definem todos os tipos de objectos serializáveis que não são nativos no sistema e que deverão ser suportados pela *framework*. De entre os vários objectos não nativos suportados destacam-se os seguintes:

- **SessionToken**: Contém toda a informação inerente a uma determinada sessão como por exemplo: nome da sessão, utilizadores registados, ficheiros disponíveis, …
- **UserData:** Contém toda a informação referente a um determinado utilizador
- **FileData:** Contém a informação referente a um determinado ficheiro disponível numa determinada sessão.
- **ResourceToken:** Contém a informação relativa à utilização de um determinado recurso por um determinado utilizador.
- **MsgToken:** Contém a informação relativa a uma determinada mensagem de texto enviado por um certo utilizador numa sessão a decorrer no momento.

StreamerManager (Classe)

A classe StreamerManager é responsável pela comunicação com a componente UdpStreamer da *framework* gerindo um canal baseado em TCP para a comunicação com a mesma. Este canal é responsável por registar a permissão de criação de conferências numa determinada sessão por determinados utilizadores na componente UdpStreamer, assim como definir os utilizadores que deverão receber os pacotes áudio/vídeo enviados por um determinado utilizador.

Session (Classe)

Esta classe pretende emular uma sala de aula/reunião permitindo que vários utilizadores se liguem à mesma e utilizem recursos registados na mesma de forma colaborativa. Para além disso, esta classe permite que utilizadores registados na mesma, troquem mensagens de texto entre si ou realizem conferências áudio/vídeo entre si.

Resource (Classe)

Esta classe contém a identificação assim como as funcionalidades associadas a um determinado recurso. As permissões de acesso a um determinado recurso também se encontram implementadas na classe em análise

Functionality (Classe)

A presente classe representa uma funcionalidade de um determinado recurso. Esta funcionalidade tem a si associada uma identificação assim como as permissões de acesso à mesma.

MsgsGroup (Classe)

A classe MsgsGroup contém os utilizadores registados num determinado grupo de troca de mensagens de texto. As funcionalidades de adição/remoção de utilizadores de um determinado grupo de mensagens também se encontram implementadas na presente classe.

ConferenceGroup (Classe)
A presente classe segue a mesma lógica da classe anterior mas aplicada a grupos de conferência áudio/vídeo.

File (Classe)
A classe File contém a informação relativa a um determinado ficheiro. Esta classe permite ainda a alteração da *source* de um ficheiro previamente enviado ao serviço.

User (Classe)
A classe User contém a informação relativa a um determinado utilizador registado numa sessão. É nesta classe que residem os grupos de chat/conferência em que o mesmo se encontra, assim como a informação pessoal associada ao mesmo (nome, canal de comunicação activo, …). Esta classe permite para além de aceder, alterar toda esta informação.

Encontra-se disponibilizado no anexo A, o ficheiro Web.config, assim como as interfaces ICallback e Iserver. Estes ficheiros permitirão verificar os métodos actualmente disponibilizados pela *framework*, permitindo assim uma melhor percepção das ferramentas da mesma. Destaca-se apenas no presente capítulo a utilização das funcionalidades de utilização de recurso devido ao facto de estas terem de permitir a sua utilização por diferentes aplicações.

A implementação das ferramentas de colaboração, ao contrário das ferramentas de comunicação, foi alvo de especial cuidado devido à imprevisibilidade do tipo de objectos a serem partilhados. Devido a esse facto, surgiram então os conceitos de Recurso e Funcionalidade previamente apresentados. Para se melhor perceber a utilização das funcionalidades de colaboração, exemplifica-se de seguida um cenário de utilização.

Tomemos como exemplo a operação de iniciar o desenho de uma linha. Para notificar esta operação a todos os elementos, é necessário enviar toda a informação necessária para replicar esta acção. Essa informação poderá ser a seguinte:

- Identificador da linha criada
- Coordenadas [x,y] do ponto inicial
- Espessura
- Côr

A *framework* para este efeito disponibiliza o seguinte método:

```
sessionToken useResource(string userId, string session, string resourceId, string
functionalityId, object[] args);
```

Identifica-se de seguida cada um dos argumentos do método previamente apresentado:

- userId: Identificador único do utilizador a utilizar a funcionalidade.
- session: Identificador da sessão onde a funcionalidade do recurso estará a ser utilizada
- resourceId: Identificador do recurso. No presente cenário poderia ser “BOARD”, pois o mesmo poderia ser efectuado numa tela de desenho. É ainda de referir que os recursos e respectivas funcionalidades são definidos aquando da criação de uma determinada sessão pelo utilizador responsável pela mesma.
- funcionalityId: Identificador da funcionalidade. Neste cenário poderia ser “CREATELINE”.
- args: Conjunto de objectos que permitem recriar a acção. No presente cenário, este array conteria o identificador da linha criada, as coordenadas, a espessura e a cor da linha criada.

A componente Service, após receber este pedido irá invocar o método sendResourceCommand para cada utilizador definido na interface ICallback e implementado por cada uma das instâncias dos utilizadores. O argumento deste método contém a informação do id do recurso e funcionalidade, assim como o *array* previamente descrito. Cada instância do utilizador é responsável posteriormente por identificar a funcionalidade e a partir do *array* recebido recriar a acção do utilizador responsável pela utilização da funcionalidade. Tal como é possível inferir pela informação anterior, a *framework* não define o tipo nem a ordem dos objectos referentes a um determinado recurso. Esta abordagem permitiu assim abstrair a framework e mantê-la geral o suficiente para ser utilizada por diferentes aplicações.

5.1.2 – UdpStreamer

A componente UdpStreamer foi criada com o intuito de realizar *streaming* de pacotes áudio/vídeo entre os vários utilizadores. Para esse efeito, a componente disponibiliza uma porta UDP e espera que uma outra aplicação envie os pacotes áudio/vídeo que deseja partilhar com outras instâncias. A componente disponibiliza ainda uma outra porta baseada no protocolo TCP com o intuito de receber mensagens da componente Service da *framework*. Estas mensagens servem para definir os utilizadores que possuem permissão para partilhar os seus pacotes áudio/vídeo, assim como os utilizadores que receberão estes dados.

A componente UdpStreamer consiste em dois módulos. O módulo UDPMessage e o módulo Server. O módulo UDPMessage contem a definição do objecto serializável que a componente Server aceitará na recepção e envio de mensagem por parte de outras instâncias tanto pela comunicação TCP como pela comunicação UDP. Este módulo é disponibilizado tanto à componente Service como à componente Client no formato de uma dll. O módulo Server é responsável pela recepção e envio de pacotes de áudio/vídeo, assim como a recepção dos comandos da componente Service da *framework* que definirão os utilizadores que retêm direito de enviar/receber os dados de vídeo/áudio. Este módulo encontra-se estruturado numa única *class* que lança duas *Threads* (uma para o protocolo TCP e outra para o protocolo UDP) que estão constantemente à espera de receber mensagens numa determinada porta. Após a recepção de uma mensagem, a aplicação deverá processá-la reenviando os dados áudio/vídeo para os respectivos utilizadores (caso a mensagem tenha sido recebida pela porta UDP) ou alterando as permissões dos utilizadores (caso a mensagem tenha sido recebida na porta TCP). É de notar que a escolha de ambos os protocolos que definem o canal de comunicação de cada uma das portas não foi seleccionada ao acaso. A comunicação para envio/recepção de pacotes de áudio faz uso do protocolo UDP pois um utilizador não precisa de confirmação do sucesso de envio de dados, assim como o envio dos mesmos em caso de falha deve ser descartado processando-se o envio do próximo pacote. Este reenvio é realizado utilizando uma comunicação ponto-a-ponto que terá sido inicializada por cada um dos clientes que irá receber os pacotes. É ainda de referir, que aquando da quebra de um dos canais de comunicação UDP será o cliente o responsável por restabelecer a ligação. Já a ligação com a componente *Service* segue necessariamente o protocolo TCP pois é fulcral que a componente Service detenha em todo o instante o controlo do número de conferências activas assim como das permissões de cada um dos utilizadores, desta forma, não pode ser descartada nenhuma mensagem entre as componentes e em caso de falha a componente Service deverá ser notificada.

5.1.3– Client

A componente Client disponibiliza uma API que permite o acesso às funcionalidades da componente Service. Esta componente é composta por três módulos distintos apresentados de seguida:

UdpClient

Esta componente consite apenas na dll criada pela componente UdpStreamer nomeadamente no módulo UdpMessage. Esta .dll define o objecto que depois de serializado deverá ser enviado para a componente UdpStreamer contendo os pacotes de áudio/vídeo.

TcpClient

A componente TcpClient deverá ser responsável por interagir com a componente Service da framework. Para esse efeito, a mesma contém um ficheiro que contém o contrato obtido através dos metadados disponibilizados pela componente Service e que se intitula Service.cs. Este contrato contém uma interface dos métodos disponibilizados pelo serviço residente na componente Service e que poderão ser utilizados por qualquer utilizador. Este ficheiro foi gerado automaticamente recorrendo à ferramenta *Service Model Metada Utility Tool* (Svcutil.exe) disponibilizada pela Microsoft. Esta ferramenta extrai os metadados disponibilizados pela componente Service e gera de forma autónoma uma interface que contém os métodos, assim como os objectos esperados como argumento ou retorno de cada uma das operações. Para além disso, é gerada também uma interface que contém os métodos passíveis de serem invocados pela componente Service na instância que contém a componente CLient e que inicializou a conexão. Este contrato é utilizado para gerar uma API que permite aceder a todas as operações definidas e implementadas pela componente Service. O ficheiro que contém esta implementação intitula-se Client.cs e contém uma classe que depois de correctamente inicializada, é responsável por iniciar a comunicação com a componente Service e invocar cada uma das operações que a mesma disponibiliza. Para inicializar esta classe, o utilizador deverá definir apenas a URI contendo o endereço da máquina onde se encontra hospedado o serviço, a classe onde é implementada a interface dos métodos invocáveis pela componente Service na instância que iniciou a comunicação, e o seu *id* a ser utilizado para a sua identificação perante a componente Service. A componente TcpCLient contém ainda mais um ficheiro que se intitula Personal.cs. Este ficheiro contém uma classe que contém toda a informação relativa ao utilizador numa determinada sessão.

5.2 – Protótipo 1

O primeiro protótipo foi desenvolvido recorrendo à framework desenvolvida anteriormente e à framework WPF (Windows Presentation Foundation) disponibilizada pela Miscorsoft. No que diz respeito à implementação, destacam-se três arquivos de maior relevância para a aplicação desenvolvida:

SessionGUI

Este arquivo consiste em dois ficheiros nomeadamente SessionGUi.xaml e SessionGUI.cs. A componente SessionGUI.xaml define o desenho da interface gráfica principal que contém a tela de desenho, assim como várias opções de gestão de permissões de uma determinada sessão. A componente SessionGUI.cs inclui a implementação dos métodos disponibilizados pela interface gráfica, assim como dos métodos responsáveis por actualizar a tela de desenho desencadeados pela edição da tela colaborativa realizada por outros utilizadores.

Proxy (Classe)

Esta classe é responsável por inicializar a componente Client da *framework*, assim como invocar os métodos definidos na sua API e que permitem utilizar as funcionalidades da componente Service.

Callback (Classe)

Esta classe implementa os métodos passíveis de ser invocados pela componente Service da *Framewok* após o protótipo estabelecer comunicação com esta.

A *framework* requer que cada instância que inicie uma comunicação forneça um identificador. Isto deve-se ao facto de a framework guardar toda a informação de um determinado utilizador (nome, sessão, apontador para o canal de comunicação, ...) num dicionário, sendo que o mesmo necessitará de uma chave única para cada novo utilizador adicionado. Este identificador deverá ser único para cada utilizador independentemente da sessão em que o mesmo se encontre. Para esse efeito, o presente protótipo utiliza uma *GUID* (Globally Unique Identifier) para se registar perante o serviço. Este identificador é um inteiro de 128 bits que é gerado aleatoriamente pelo sistema operativo. É de notar que é impossível os sistemas operativos de dois computadores diferentes gerarem uma chave igual, e ainda que o mesmo sistema operativo não gerará a mesma chave enquanto pelo menos o próprio não for reinicializado.

No que diz respeito à utilização de recursos e funcionalidades de edição de uma tela de desenho de forma colaborativa numa determinada sessão, o protótipo regista apenas um recurso no serviço, identificado pela palavra-chave "BOARD" e associando-lhe as seguintes funcionalidades:

- CREATESTROKE: Inicializa a criação de uma linha, enviado o ponto de criação e o identificador da linha. Este identificador consiste numa *Guid* (previamente apresentada na presente secção) e que garante que os diferentes objectos (linhas) criados, contêm sempre um identificador diferente.
- RTUPDATESTROKE: Adiciona um ponto a uma determinada linha contendo o identificador da linha, assim como as coordenadas [x,y] do ponto.
- UPDATESTROKE: Envia todos os pontos de uma determinada linha, assim como o identificador da mesma após a mesma ter sido finalizada.
- CREATECOMPLETESTROKE: Inicializa a criação de uma linha, enviando também todos os pontos que a mesma contém.
- RTDELETESTROKE: Elimina um conjunto de linhas, enviando os seus identificadores

É de notar que as funcionalidades "CREATESTROKE", "RTUPDATESTROKE", "CREATECOMPLETESTROKE" apesar de serem utilizadas com o mesmo objectivo (criar uma linha), são utilizadas de forma a diminuir a quantidade de informação enviada aos utilizadores numa determinada sessão, utilizando mesmo dois métodos distintos disponibilizados pela *framework*. As funcionalidades "CREATESTROKE" e "RTUPDATESTROKE" utilizam o método "useRealTimeResource" e a funcionalidade "CREATECOMPLETESTROKE" utiliza o método "useResource". Se um utilizador definir que não pretende receber a informação de criação de objectos em tempo real, o mesmo não será notificado das acções representadas por "CREATESTROKE" e "RTUPDATESTROKE" recebendo apenas o comando de finalização de criação de uma linha "COMPLETESTROKE". A utilização de cada uma das funcionalidades anteriormente descritas segue sempre a mesma sequência de funcionamento. Exemplifica-se de seguida a utilização da funcionalidade "CREATESTROKE".

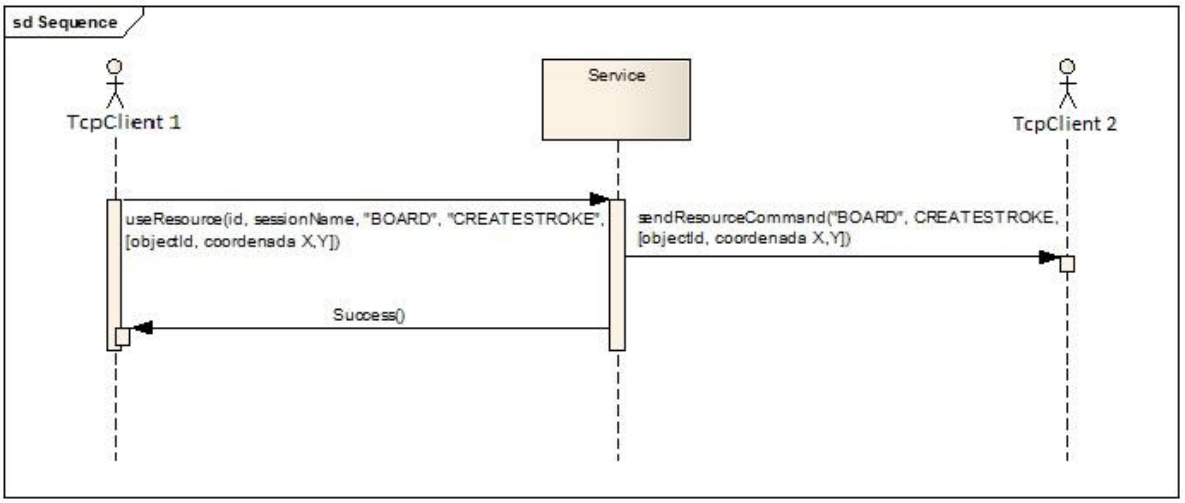


Figura 36: Utilização de recurso

Como é possível verificar pela Figura 36, o utilizador responsável pela criação da linha, envia a notificação da utilização da funcionalidade "CREATESTROKE" no recurso "BOARD", enviando ainda os dados da linha criada. A componente Service, após ter recebido a notificação da utilização da funcionalidade, notifica os utilizadores da sessão, invocando o método sendResourceCommand, implementado pelos utilizadores e definido no contrato do Serviço.

Apresentam-se de seguida as interfaces do protótipo 1 após o término do seu desenvolvimento:

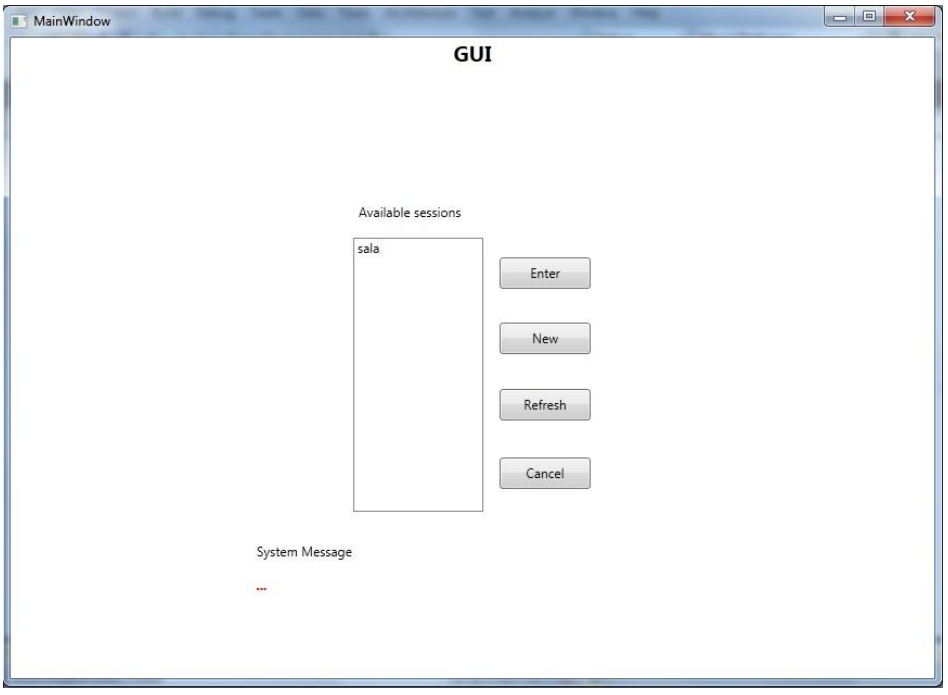


Figura 37: Interface de escolha de sessão do Protótipo 1

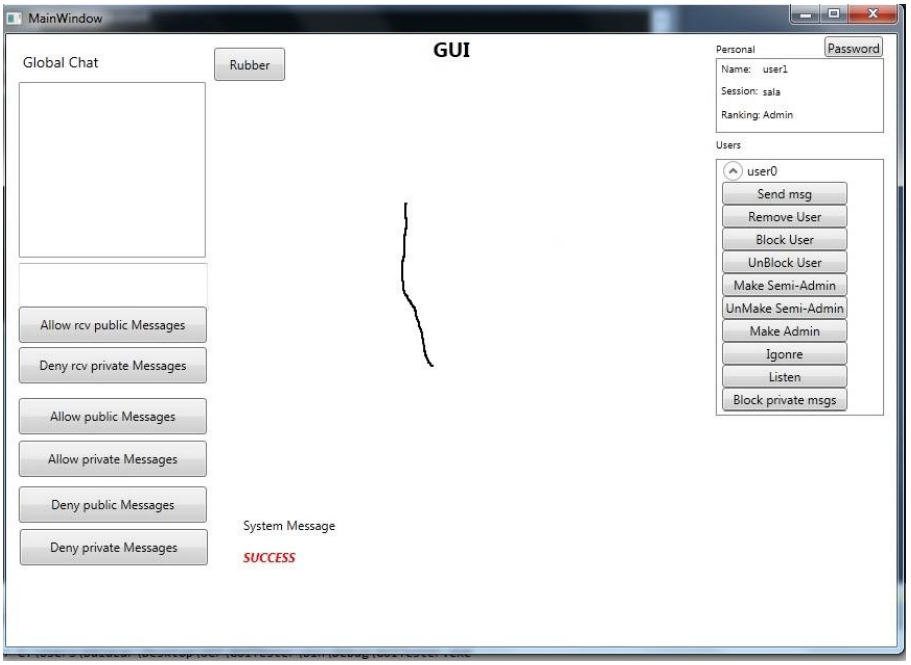


Figura 38: Interface principal do Protótipo 1

Como é possível verificar pelas Figuras 37 e 38, a interface desenhada e apresentada no capítulo anterior para o presente protótipo foi implementada com sucesso (disponibilizam-se no anexo B as restantes interfaces deste protótipo). A interface apresenta ainda mais uma característica que permite identificar objectos a serem editados num determinado momento e que é ilustrada na figura 39:

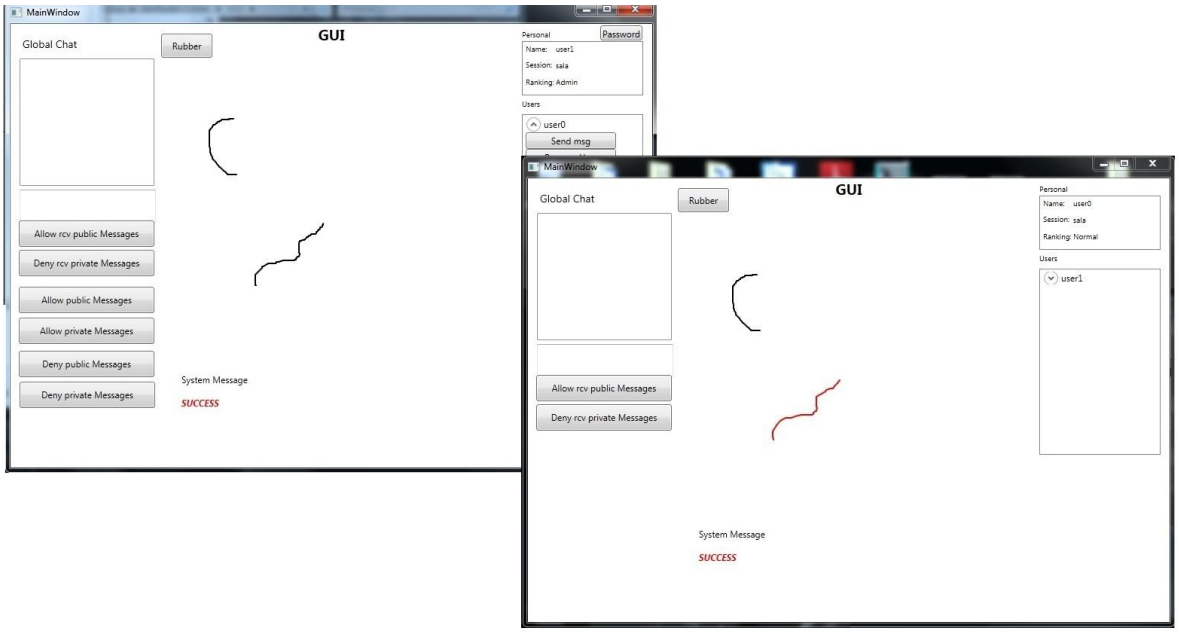


Figura 39: Criação de linha em tempo real

Como é possível verificar pela figura anterior, e apesar de o protótipo não disponibilizar uma funcionalidade para a mudança de cor existem duas linhas na tela, que embora possuam os mesmos pontos, apresentam uma cor diferente. Esta diferença pretende mostrar ao utilizador que a linha vermelha se encontra a ser editada por outro utilizador e ainda não está finalizada a sua criação. Esta diferenciação na interface entre linhas terminadas e linhas em criação foi implementada devido ao facto de se ter decidido que uma linha só poderia ser editada por outros utilizadores quando for terminada a sua criação. Desta forma os utilizadores têm a percepção visual dos objectos que podem ou não editar. Um cenário da situação previamente apresentado é ilustrado na Figura 40:

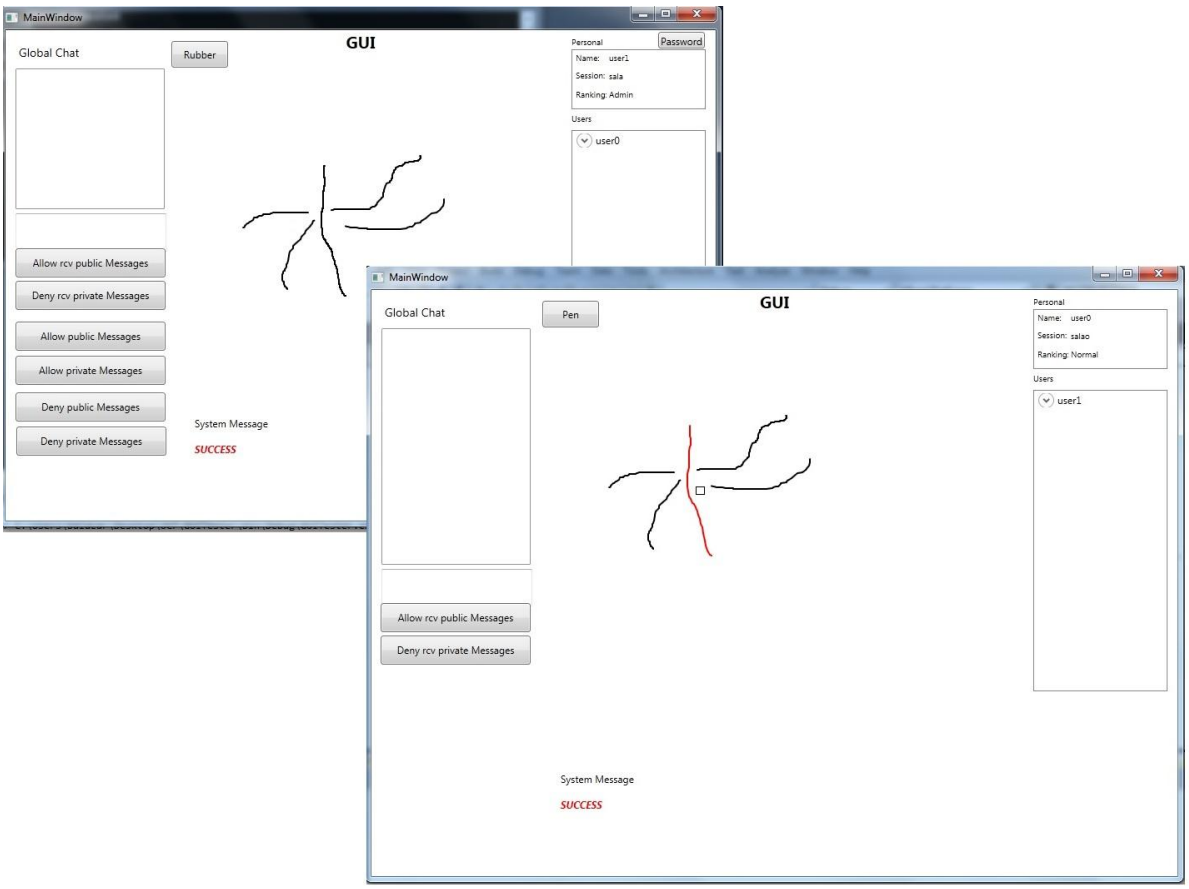


Figura 40: Utilização da borracha em tempo real (Protótipo 1)

É passível de verificar que o utilizador da aplicação que se encontra do lado esquerdo da imagem anterior se encontra a criar a linha vertical na tela colaborativa. Por outro lado, o utilizador da aplicação do lado direito utilizou a borracha apagando os pontos em falta das restantes duas linhas. Como é possível inferir, a borracha terá a certa altura interceptado a linha em criação pelo outro utilizador mas como é possível verificar na imagem anterior, esta não foi alterada, atingindo-se assim o objectivo anteriormente apresentado.

5.3 – Integração com o UbiStudio (Protótipo 2*)*

O presente protótipo consiste na aplicação final do presente trabalho. Este protótipo consiste na aplicação UbiStudio mas agora com ferramentas de comunicação entre utilizadores assim como

funcionalidades colaborativas para a edição de uma tela de desenho. O UbiStudio, tal como o protótipo anterior, utiliza a *framework* WPF pelo que a implementação das interfaces dos módulos colaborativos também utiliza esta *framework*. Destacam-se de seguida os principais módulos desenvolvidos para a aplicação UbiStudio:

Proxy: Classe responsável pela inicialização da componente Client da *framework* de colaboração e comunicação desenvolvida. Esta classe utiliza ainda a API disponibilizada pela componente Client para comunicar com a componente Service.

Callback: Classe que implementa a Interface definida na componente Service para comunicações inicializadas por esta. Esta interface encontra-se também definida no contrato disponibilizado pela componente Client que foi gerado através dos metadados da componente Service.

WebCamCapturer: Módulo que capta a imagem actual disponível na webcam actual do utilizador.

MicroCapturer: Classe que capta os dados áudio actuais provenientes do microfone do utilizador.

WebCamService: Classe que gere as janelas actuais de vídeo/áudio conferência do utilizador

WebCamStreamer: Classe que trata os dados áudio/vídeo de outros utilizadores e os insere na interface actual do utilizador.

UdpClient: Classe que utiliza a biblioteca UdpMessage disponibilizada pela componente Client da *framework* e inicializa e gere um canal de comunicação UDP com a componente UdpStreamer da *framework*. Esta classe é ainda responsável por enviar e receber os pacotes áudio/vídeo

CollaborativeService: Componente que inicializa a classe Proxy anteriormente apresentada. Esta componente é responsável por realizar a ligação entre os métodos invocados na interface do UbiStudio e o proxy responsável por enviar e receber mensagens da componente Service assim como guardar o estado da sessão e permissões actuais do utilizador.

CollaborationGuiController: Componente responsável por implementar todos os métodos invocados pelas funcionalidades utilizadas pelos outros utilizadores na sessão. Após o método de utilização de funcionalidade de um determinado recurso ser invocado no módulo Callback, este verifica qual é a funcionalidade utilizada e invoca o método correspondente no módulo CollaborationGuiControler que irá actualizar a tela do utilizador emulando a acção tomada pelo utilizador responsável pelo uso de uma determinada funcionalidade. É também nesta classe que é preparada a informação, assim como identificada a funcionalidade que o utilizador actual realizou na tela de desenho. Posteriormente esta informação é passada ao módulo proxy que se encarregará de a enviar à componente Service da *framework*.

Tal como no protótipo anterior a presente aplicação precisa também de definir o seu identificador para poder comunicar com a componente Service da *framework*. Este identificador deve ser único e deverá permitir verificar se, aquando de uma nova ligação por parte de um utilizador este é um utilizador não registado no serviço ou por outro lado este se está a ligar novamente após ter saído ou a ligação se ter perdido. Devido a esta necessidade (a aplicação poderá ser fechada e re-aberta instantes depois, a ligação à internet poderá ter sido quebrada ou o utilizador expulso de uma determinada sessão) a utilização de uma chave gerada em *runtime* não permitiria identificar se o utilizador se tinha previamente ligado a uma determinada sessão (possuindo já um perfil que continha as suas permissões e identificação) pois perder-se-ia aquando do término da aplicação. Para esse efeito definiu-se que o identificador de um utilizador deveria ser sempre que possível, o *mac address* da placa de rede que estiver a ser utilizada para o mesmo se ligar ao serviço. Esta abordagem no entanto não contempla o cenário em que utilizador se encontra a utilizar uma placa 3G para aceder à internet. Neste caso a placa em utilização não tem associado um *mac adress*, não sendo possível atribuir assim um id único baseado nesta variável. No entanto este cenário é fácil de ser identificado pois ocorre sempre que o *mac address* da placa activa não é possível de ser determinado e no entanto existe uma ligação à internet. Nestes casos, é atribuído ao

utilizador um identificador que equivale ao *mac address* da primeira placa de rede registada no sistema e que possui um *mac address*. É ainda de referir que este identificador é utilizado para comunicar com ambas as componentes Service e UdpStreamer.

Tal como no protótipo anterior, também no presente protótipo foi decidido que apenas um utilizador poderá editar um determinado objecto num determinado instante. Desta forma, adaptou-se a *framework* para atribuir a cada utilizador de uma determinada sessão uma certa cor. Desta forma, é possível identificar visualmente o utilizador que se encontra a editar um determinado objecto. Os objectos criados por um utilizador possuirão então inicialmente e enquanto o mesmo os está a criar a cor do utilizador responsável por essa acção. Para além disso, o *adorner* utilizado para seleccionar objectos, deverá também possuir a cor, assim como o *username* do utilizador responsável pela selecção em todas as instâncias que se encontram na sessão e que não são as utilizadas pelo utilizador a editar/criar/seleccionar o objecto. Para além disso, não deverão ser disponibilizadas opções de edição nos objectos já seleccionados por outro utilizador.

No que diz respeito aos recursos e funcionalidades de edição da tela de desenho, a aplicação regista apenas um recurso, nomeadamente "BOARD". Este recurso disponibiliza diversas funcionalidades (a listagem completa encontra-se no Anexo C), entre as quais:

- COMPLETESTROKE: Completa a criação de uma determinada linha contendo todos os pontos desta.
- COMPLETESHAPE: Finaliza a criação de uma determinada forma geométrica enviando a altura e largura finais da mesma
- SELECTITEMS: Selecciona um ou mais items
- MOVEITEMS: Actualiza a posição de um ou mais items após ter sido terminada a edição da posição dos mesmos
- REMOVEITEM: Remove um ou mais items previamente seleccionais
- DELETEITEMS: Elimina um ou mais items após o utilizador ter terminado a sua acção
- RESIZE: Actualiza o tamanho de um determinado objecto após um utilizador ter terminado de realizar o redimensionamento.
- ADDPAGE: Adiciona um slide à sessão actual
- CNHGBH: Altera o fundo de um determinado slide
- CNHGPAGEPOS: Altera a posição de um determinado slide na lista de slides da sessão
- REMOVEPAGE: Elimina um determinado slide da sessão
- FADDED: Adiciona um determinado ficheiro à sessão
- ROTATE: Completa a rotação de um determinado objecto
- COPYDI: Copia um ou mais items
- CUTDI: Corta um ou mais items
- PASTEDI: Cola um ou mais items
- DUPLICATEDI: Duplica um ou mais items
- ADDPAGECOVER: Adiciona uma cortina a um determinado slide
- CREATEERTXT: Insere uma caixa de texto

É apenas de referir que para além da implementação de todas as funcionalidades inicialmente definidas como prioritárias para o sucesso da aplicação, foi ainda adicionada a funcionalidade de adicionar/remover/editar cortina. Esta funcionalidade permite ocultar o conteúdo de uma parte de um determinado slide a todos os utilizadores.

A utilização das funcionalidades previamente referidas na tela de desenho, segue as seguintes regras definidas de forma a facilitar a interacção entre os utilizadores:

- Um objecto, apenas pode ser editado por um utilizador num determinado instante.

- Após a selecção de um determinado objecto, todos os outros utilizadores deixam de ter permissão de edição do objecto seleccionado
- Um slide não poderá ser removido se um utilizador estiver a editar/criar um objecto no mesmo.
- Uma cortina apenas pode ser editada por um utilizador num determinado instante.

Apresenta-se de seguida a interface da aplicação desenvolvida ilustrada pelas Figuras 41, 42 e 43:

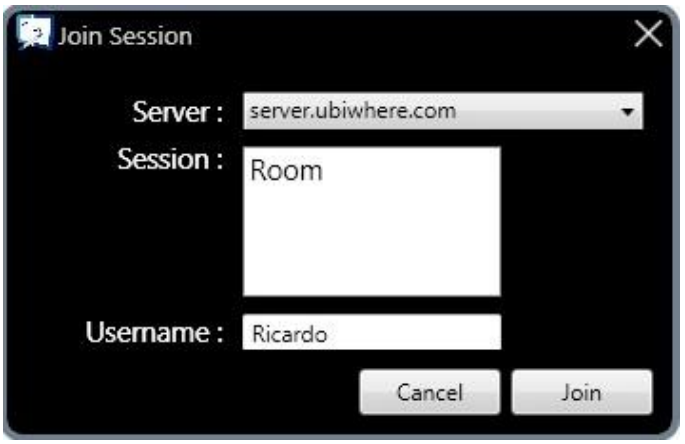


Figura 41: Escolha da sessão e definição do UserName assim como do uri do servidor (Protótipo 2)

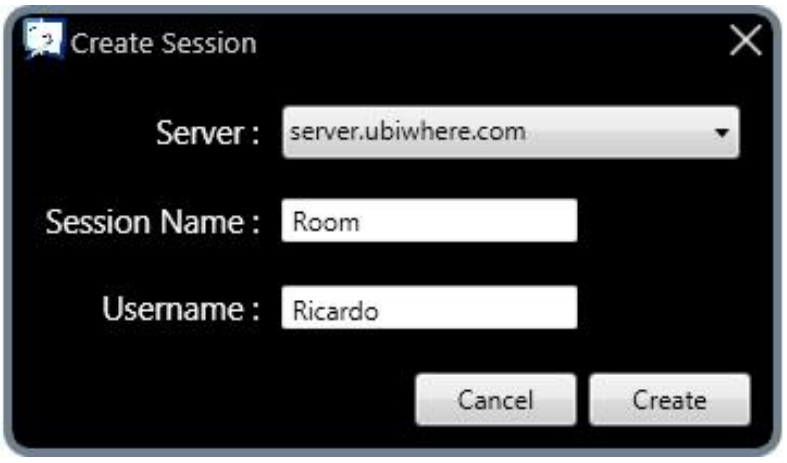


Figura 42: Criação de uma determinada sessão (Protótipo 2)

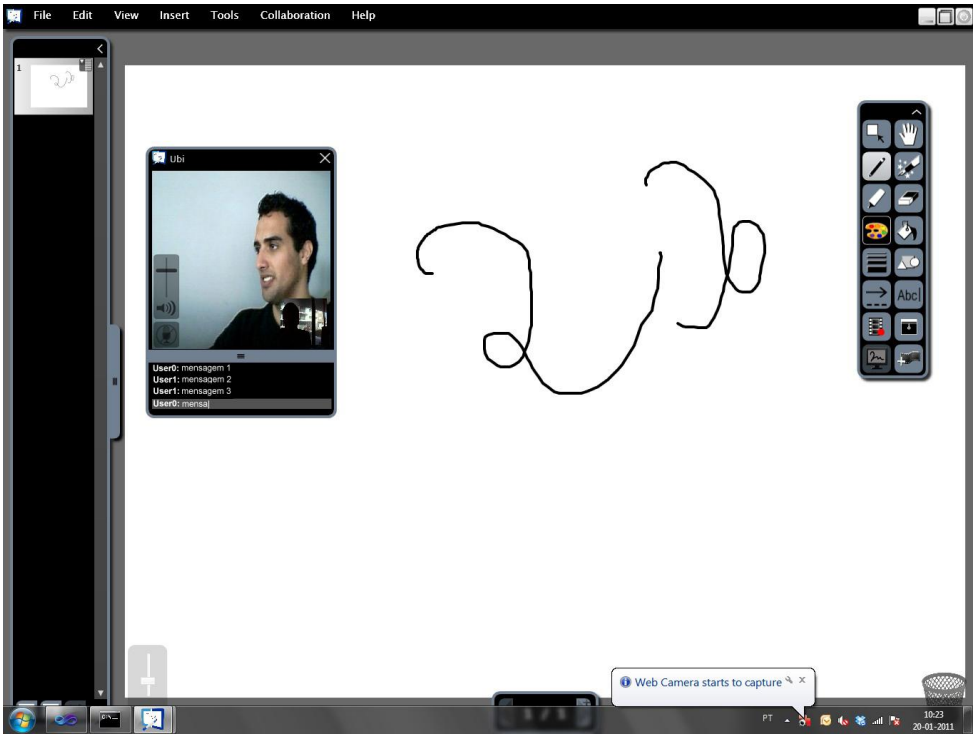


Figura 43: Interface principal da aplicação (Protótipo 2)

Tal como é possível verificar pelas imagens anteriores, todos os templates apresentados no capítulo anterior foram respeitados, excepto o que diz respeito à janela de conversação por conferência e por texto. Estas duas janelas foram fundidas numa única, de forma a diminuir a quantidade de objectos na interface, dando assim maior destaque à tela. Apresenta-se de seguida na Figura 44 uma ilustração em maior destaque desta componente da interface.

Figura 44: Janela de conferência Figura (Protótipo 2)

Tal como é possível verificar o utilizador poderá ajustar a quantidade da janela que deseja atribuir à vídeo-conferência e às mensagens de texto. Para além disso o utilizador poderá regular o volume áudio da informação recebida, assim como deixar de disponibilizar o seu microfone a este utilizador (o microfone continuará a ser disponibilizado a outros utilizadores noutras janelas).

Anteriormente foram referidas as restrições no que diz respeito à edição colaborativa de objectos. Estas restrições foram contempladas na interface desenvolvida de forma a permitir que os utilizadores possuam uma experiência mais intuitiva aquando da utilização da aplicação. A aplicação contempla os seguintes cenários:

Criação de um objecto

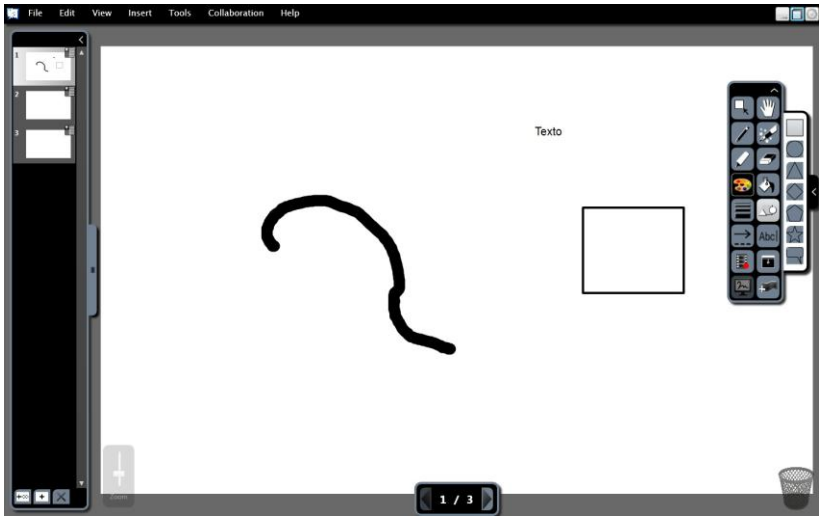


Figura 45: Criação de uma linha (Protótipo 2)

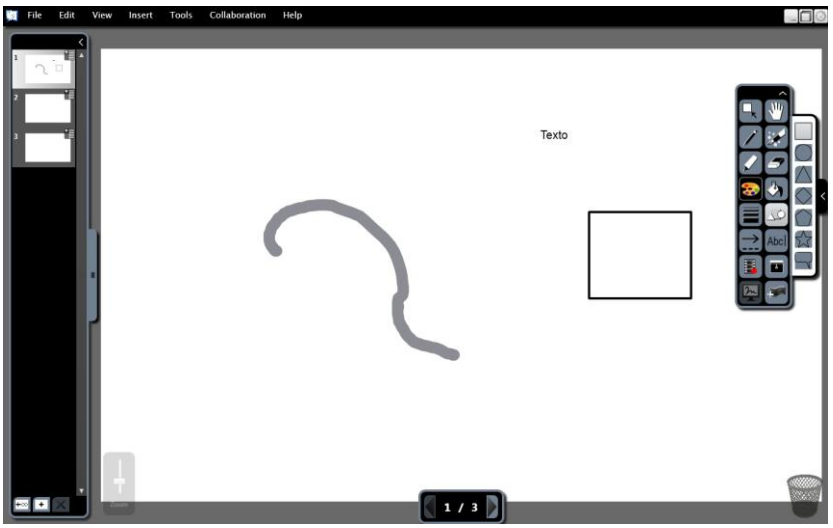


Figura 46: Visualização da criação de uma linha por parte de outro utilizador (Protótipo 2)

As Figuras 45 e 46 correspondem à utilização da aplicação num determinado instante, onde ambos os utilizadores estão registados na mesma sessão. É no entanto possível verificar que a linha

desenhada na tela apresenta uma cor diferente em ambas as aplicações. De fato, a verdadeira cor da linha desenhada corresponde à cor da Figura 45 (preta). O facto de o utilizador da figura 46 possuir uma cor diferente, deve-se ao facto de o utilizador da primeira imagem ter visto a si atribuída esta cor. Devido a isto, todos os objectos que o mesmo criar, vão possuir esta cor até ao momento em que o mesmo completar a sua criação. De facto após o utilizador acabar de desenhar a linha (levantar a caneta do quadro/largar a tecla esquerda do rato), a tela do utilizador figura 46 foi actualizada ficando com o aspecto da Figura 47 de seguida apresentada:

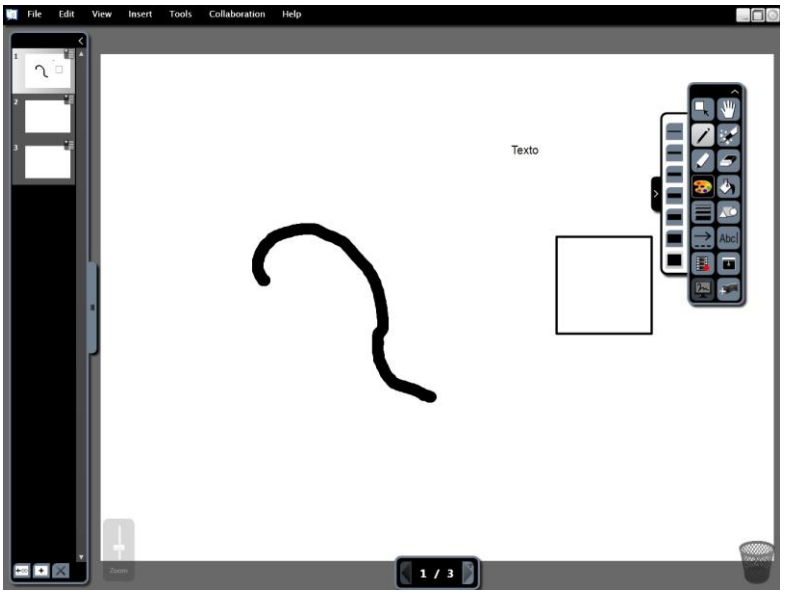


Figura 47: Finalização da criação da linha (Protótipo 2)

Selecção de um objecto

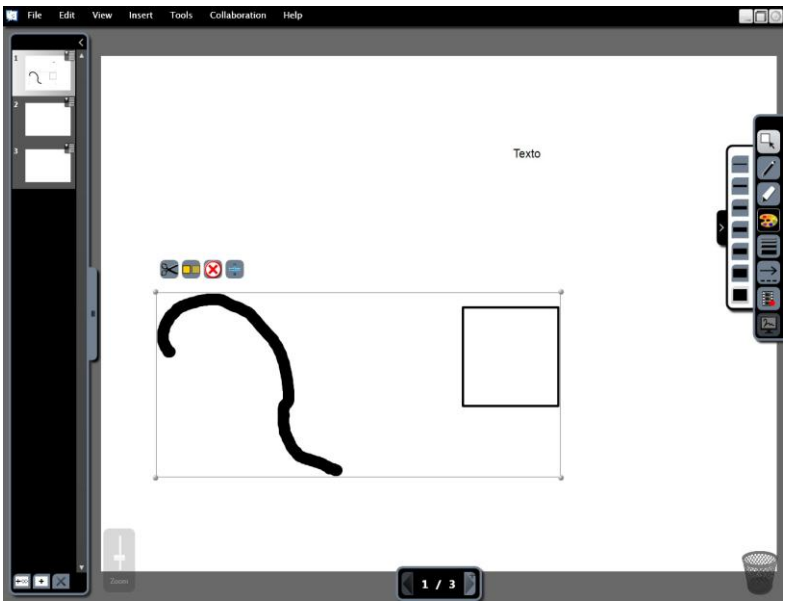


Figura 48: Selecção de um objecto (Protótipo 2)

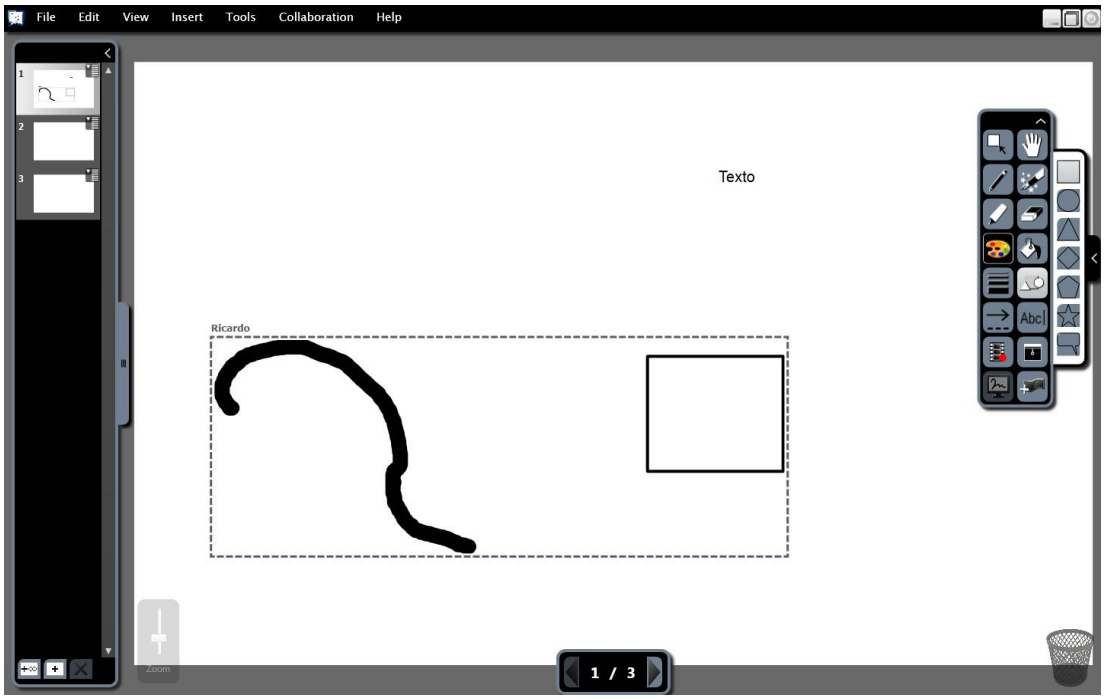


Figura 49: Selecção de um objecto por parte de outro utilizador (Protótipo 2)

Como é possível verificar pelas Figuras 48 e 49, o *adorner* dos objectos seleccionados é diferente entre as duas instâncias embora estas se encontrem a correr a mesma versão da aplicação e os utilizadores se encontrem na mesma sessão colaborativa. Este cenário ocorre porque o utilizador da figura 48 foi o responsável pela selecção dos objectos, sendo que todos os outros utilizadores obterão a interface da Figura 49 onde o *adorner* dos objectos seleccionados detém a cor atribuída ao utilizador responsável pela selecção, assim como o seu userName (no caso anterior Ricardo). Para além disso é ainda visível que os utilizadores que não são responsáveis pela selecção dos objectos não possuem a interface de edição de objectos visível na imagem 46. A inclusão destes elementos na interface permite então ao utilizador aperceber-se imediatamente de todos os objectos passíveis de ser editados, assim como os utilizadores que se encontram a realizar uma acção sob os objectos da tela.

No que diz respeito à implementação das funcionalidades referidas no capítulo anterior e referentes a este protótipo, foram todas implementadas, tendo-se ainda adicionado a implementação da funcionalidade adicionar cortina. Esta funcionalidade permite adicionar de forma colaborativa uma cortina que ocultará parte de um determinado slide e encontra-se ilustrada na figura seguinte:

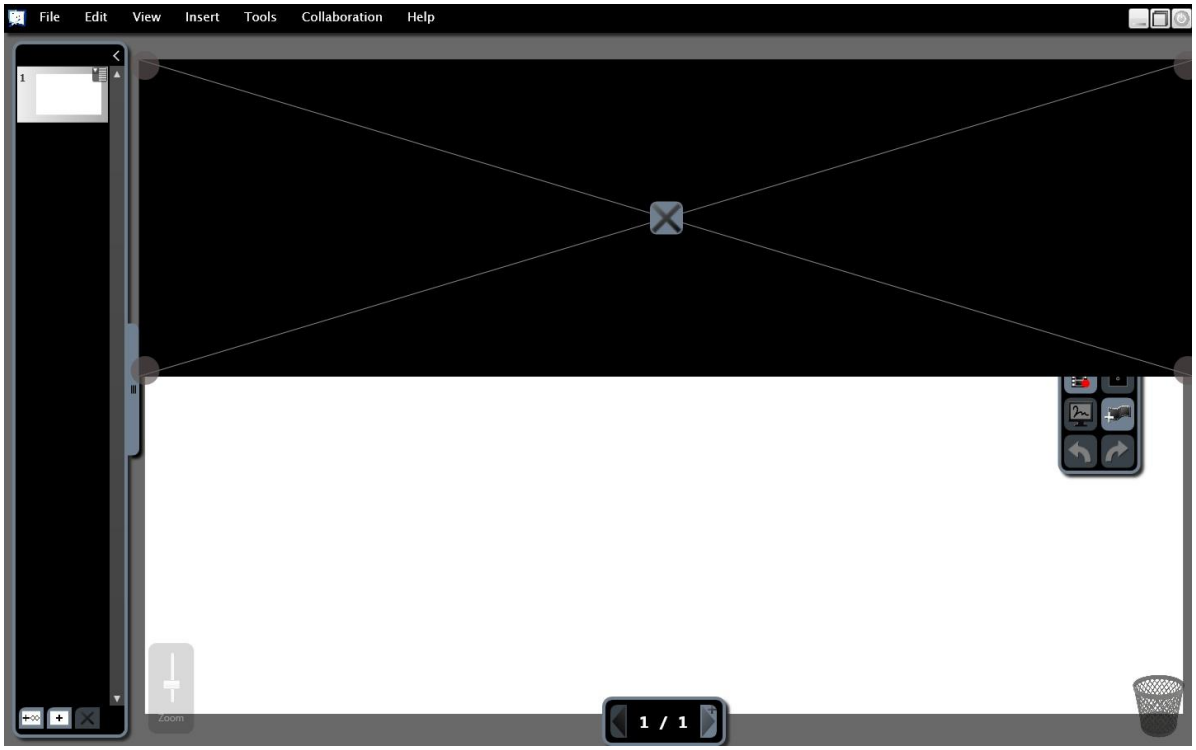


Figura 50: Exemplificação da cortina

5.4 – Testes e Resultados

No decorrer da elaboração de ambos os protótipos foram estabelecidas sessões até um máximo de seis utilizadores, sendo que em todos os casos, o tempo de resposta obtido foi sempre inferior a um segundo mantendo-se a impressão de que as alterações realizadas na tela colaborativa numa determinada instância, eram actualizadas em tempo real em todas as outras instâncias.

Para se perceber melhor a *performance* da *framework*, foi medido o tempo necessário para a utilização de um determinado recurso ser propagada para todos os utilizadores de uma determinada sessão. Para esse efeito foram utilizadas várias máquinas pelo que primeiramente teve que ser levado a cabo um processo de "sincronização". Para a realização deste processo, foram levados a cabo os seguintes passos:

- Definição da máquina que conterá o tempo de referência.
- Registo de dez leituras do tempo actual com uma precisão de milissegundos em todas as máquinas e no mesmo instante que a máquina referência.
- Subtracção de cada um dos tempos obtidos para cada instante, e obtenção da média dos valores calculados anteriormente, obtendo-se assim a média da diferença do tempo actual entre cada uma das máquinas e a máquina referência.
- Registo do desfasamento do tempo actual em relação à máquina referência para cada uma das máquinas.

O processo anterior permitiu sincronizar de forma razoável o tempo actual nos diversos computadores a utilizar para a realização dos testes à *framework*. Todos os testes apresentados de seguida foram precedidos do processo anteriormente descrito, sendo que os tempos obtidos contemplam já a diferença do tempo actual obtido em máquinas diferentes no mesmo instante.

Para se analisar com maior detalhe a *performance* da *framework* sem contemplar ainda o atraso

imprevisível de uma ligação à internet, foi estabelecida uma sessão entre dezasseis utilizadores, sendo que, todos se encontravam a utilizar a mesma rede onde se encontrava alojada a componente Service. A máquina responsável pelo alojamento desta componente apresenta as seguintes características:

Processador: IntelCore 2Quad CPU Q855 @2.66Gz
Ram: 1.00 GB
Sistema Operativo: Windows Server Enterprise (32 bits), Service Pack 2
Velocidade de upload da ligação à Internet: 321 Kbps
Velocidade de download da ligação à Internet: 1555 Kbps

Para os presentes testes, foram utilizados quatro computadores, sendo que cada um foi responsável por inicializar quatro instâncias do protótipo 2 (a forma de cálculo do identificador do utilizador perante a *framework* foi alterada por forma a permitir iniciar mais do que um utilizador na mesma máquina. O id do utilizador passou a ser calculado através da geração de uma GUID, sendo que será diferente a cada instância lançada). Todas as máquinas se encontravam a utilizar o sistema operativo Windows 7 (32 bits) e todas possuíam a mesma ligação de rede local.

Foi realizado primeiramente um teste com o intuito de verificar o tempo necessário para a utilização de um determinado recurso ser propagada a todos os utilizadores. Foram então inicializadas dezasseis instâncias do protótipo 2, juntos todos os utilizadores na mesma sessão, e enviado um comando de utilização de um determinado recurso por parte de um determinado utilizador. Antes de o comando ser enviado à componente Service, foi imprimido o tempo actual na instância do utilizador a utilizar o recurso (O utilizador responsável pela utilização do recurso encontrava-se a utilizar a máquina que contém o tempo referência). Cada um dos utilizadores, imprimiu também o tempo actual assim que a componente Service, invocou o método de utilização de uma funcionalidade em cada um deles. Para cada um dos utilizadores foi calculada a diferença entre o tempo no instante de recepção da utilização de um recurso com o tempo no instante em que o mesmo foi utilizado. Posteriormente foi calculado o tempo médio de recepção entre os quinze utilizadores (o utilizador que utilizou a funcionalidade não recebeu nenhuma notificação do servidor), assim como obtido o tempo máximo de recepção entre os utilizadores.

Este processo foi repetido três vezes, alterando-se apenas o tamanho do array de informação inerente à utilização de uma funcionalidade de um determinado recurso. Foram utilizados arrays com 50, 100 e 250 bytes respectivamente em cada um dos testes realizados Para cada um dos diferentes tamanhos foram realizadas 20 leituras, sendo que as tabelas que se seguem são a média dos resultados obtidos em milissegundos.

Tabela 1: Média de *delay* Teste 1

Bytes	*Delay (ms)*
50	56
100	83
250	204

Tabela 2: Máximo de *delay* Teste 1

Bytes	*Delay (ms)*
50	65
100	94
250	215

O tamanho dos pacotes realmente enviado pela rede é superior à quantidade de *bytes* apresentada nas tabelas anteriores, devido não só ao facto de ser enviada mais informação para utilização de uma funcionalidade de um recurso (id do utilizador, id da sessão, id do recurso e id da funcionalidade) mas também como consequência do *overhead* inserido para a codificação desta informação de acordo com o protocolo *SOAP*. No entanto como esta codificação é necessária e efectuada de forma autónoma pelo sistema operativo e *frameworks* de comunicação utilizadas, este não foi medido nem contemplado nos testes. De facto, visto este *overhead* ser previsivelmente constante e necessário para o funcionamento da *framework* desenvolvida, a *performance* obtida pela *framework* deverá já contemplar este factor de atraso. Devido a isto apenas se contempla nos casos de teste e para efeitos de tamanho da mensagem enviada, a única variável que previsivelmente alterará o *delay* entre o momento de utilização de um recurso e o momento de recepção da utilização por parte de outro utilizador. Quanto à quantidade de informação enviada, de momento o protótipo 2 na maioria dos casos, associa um array de 10 bytes à utilização de um determinado recurso, pelo que a quantidade de informação associada a cada um dos testes permite desde já tirar ilações acerca do desempenho da *framework*.

No teste anterior, assim como na maioria dos testes realizados, as aplicações que partilhavam uma determinada sessão estabeleciam uma ligação de rede local com o servidor responsável por alojar a componente *Service*, não sendo assim tido em conta o previsível acréscimo de *delay* devido a uma conexão utilizando a internet. Para analisar o comportamento verificado neste cenário, o teste anterior foi novamente realizado, mas desta vez, todos os utilizadores utilizaram uma ligação à internet para se ligarem à componente Service. Foram então utilizadas dois tipos de ligação à Internet nomeadamente uma ligação permanente por cabo utilizando um serviço disponibilizado por um ISP (Internet Service Provider) e uma ligação 3G utilizando uma placa móvel *wireless* que permite o acesso à Internet. É de referir que o número de utilizadores que utilizaram cada uma das ligações é igual, sendo que oito utilizadores utilizaram uma ligação por cabo, e oito utilizadores utilizaram a rede 3G. No que diz respeito ao presente caso de teste, convém também referir que a componente Service se encontrava num servidor alojado nas instalações da empresa em Aveiro, e todas as aplicações que se ligariam a esta componente se encontravam na cidade do Porto. É ainda de referir que os resultados a seguir apresentados são a média de 20 leituras efectuadas:

Tabela 3: Média de *delay* Teste 2

Bytes	*Cabo (ms)*	*3G (ms)*
50	56	52
100	593	500
250	779	801

Tabela 4: Máximo de *delay* Teste 2

Bytes	*Cabo (ms)*	*3G (ms)*
50	129	88
100	562	554
250	856	833

Após a realização dos testes anteriores, pretendeu-se simular um ambiente onde todos os utilizadores se encontrassem a utilizar um determinado recurso, cenário esse mais plausível com a realidade da utilização da aplicação desenvolvida. Neste cenário todos os utilizadores estariam a enviar/receber notificações à/provenientes da componente Service. Tendo em conta que a utilização de um recurso deve ter associado um número mínimo de bytes para manter a *performance* da *framework*, e tendo ainda em conta que a utilização da maior parte das funcionalidades por parte do UbiStudio (Protótipo 2) utiliza menos de 10 bytes, no cenário do presente teste todos os utilizadores utilizam com um intervalo de cerca de 75 milissegundos uma funcionalidade que vê a si associada

uma array de 50 bytes. Mais uma vez no presente teste foi medido o tempo que demorou a ser propagada a utilização de uma determinada funcionalidade por parte de um determinado utilizador. Neste teste no entanto a *framework* para além da notificação da utilização do recurso deste utilizador era também responsável por enviar todas as outras notificações de utilização de recursos de todos os outros utilizadores a todos os utilizadores registados na sessão. Esta situação leva a que a cada 75 mili-segundos a *framework* receba 16 notificações de utilização de um recurso, sendo que para cada uma delas, terá de notificar 15 utilizadores do seu uso. Realizou-se primeiramente o cenário descrito utilizando uma rede local e realizadas 20 leituras, apresentando-se de seguida uma tabela com a média de resultados em milissegundos:

Tabela 5: Tabela de Delay Teste 3

Delay	*Delay (ms)*
Médio	98
Máximo	115

Tal como no cenário anterior, a experiência foi repetida mas desta vez recorrendo a ligações utilizando a internet. Os utilizadores foram igualmente distribuídos pela rede 3G e pela rede por cabo tendo-se obtido os seguintes resultados

Tabela 6: Tabela de Delay Teste 4

Ligação	*Cabo (ms)*	*3G (ms)*
Média	1325	1152
Máximo	2230	2115

Como é possível verificar pelos tempos anteriores, embora estes apresentem tempos relativamente altos, também correspondem a um cenário raro onde todos os utilizadores se encontrem a utilizar uma funcionalidade que vê a si associada uma considerável quantidade de informação (array de 50 bytes).

A componente UdpStreamer foi também alvo de análise, sendo que foram estabelecidas várias conferências entre utilizadores. Para melhor se poder analisar a performance da componente UdpStreamer foi medido o *delay* associado a uma determinada conferência. Para esse efeito, de 100 em 100 milissegundos para além de enviar o *frame* actual da webcam, enviou-se também uma *label* que continha o identificador da *frame*. Registou-se previamente o tempo actual antes de se proceder ao envio da *frame*, registando-se posteriormente o tempo actual na máquina que recebeu a *frame* correspondente. Subtraindo-se os tempos obtem-se o *delay* associado a uma determinada *frame*. Numa primeira fase, foi inicializada apenas uma conferência entre dois utilizadores, ambos utilizando a mesma rede onde se encontra a componente Service, feitas 20 leituras dos tempos e obtendo-se a seguinte média de resultados:

Tabela 7: Tabela de Delay Teste 5

Delay	*Delay (ms)*
Médio	115
Máximo	354

Posteriormente foram adicionados dois utilizadores e estes dois iniciaram uma conferência entre si, existindo na sessão duas conferências que correspondem a quatro utilizadores a enviar os seus dados áudio/vídeo, sendo que cada pacote de áudio/vídeo seria reencaminhado apenas para um

utilizador e todos os utilizadores estavam na mesma rede local. A conferência inicializada no teste anterior apresentou os seguintes resultados:

Tabela 8: Tabela de Delay Teste 6

Delay	*Delay (ms)*
Médio	112
Máximo	341

Foi posteriormente adicionado mais um utilizador, e inicializadas conferências entre todos os utilizadores da sessão. Isto traduziu-se num cenário onde existiam cinco utilizadores a enviar os seus dados áudio/vídeo sendo que cada pacote de dados era distribuído por cinco utilizadores. Obtiveram-se neste cenário os seguintes resultados:

Tabela 9: Tabela de Delay Teste 7

Delay	*Delay (ms)*
Médio	360
Máximo	456

Por último, foi inicializada a mesma conferência entre os mesmos utilizadores só que desta vez, um dos utilizadores utilizou uma rede 3G para se ligar à sessão. Apresentam-se de seguida os tempos obtidos.

Tabela 10: Tabela de Delay Teste 8

Ligação	*Cabo (ms)*	*3G (ms)*
Média	455	1152
Máximo	540	1560

No que diz respeito à interface do protótipo final desenvolvido, não foram feitos testes de usabilidade pois a interface do mesmo, segue o modelo do produto anterior. Para além disso, a disponibilização das funcionalidades de desenho não sofreu qualquer alteração no que diz respeito à sua interface pelo que o utilizador não notará grandes diferenças quando comparada à usabilidade do produto anterior.

5.5 – Demonstração

É de mencionar que o produto desenvolvido foi ainda apresentado na feira internacional BETT que decorreu em Londres nos dias 12 a 15 de Janeiro. Esta feira serve de montra a produtos e tecnologias orientados ao âmbito da educação e desenvolvidos por empresas de todo o mundo. O facto do protótipo desenvolvido ter marcada presença nesta feira pela empresa UbiWhere, deixa transparecer que o mesmo poderá vir a ser encarado no futuro como uma aplicação a comercializar. Nesta feira foi estabelecida uma sessão entre dois computadores presentes na feira, assim como dois computadores presentes na sede da empresa em Aveiro. O *delay* obtido entre os dois computadores de Londres foi cerca de dois segundos, *delay* esse aceitável tendo em conta que a componente *Service* se encontrava na sede da empresa em Aveiro. No que diz respeito à experiência de utilização dos utilizadores, a maioria assinalou como importante e agradável a percepção visual dos objectos seleccionados por outros utilizadores. O impacto visual da colaboração em tempo real de todas as funcionalidades de desenho da tela também foi alvo de referência por parte da maioria dos utilizadores, revelando alguns destes que a ferramenta seria muito útil em alguns cenários inerentes a uma sala de aula.

5.6 – Conclusões

O presente capítulo apresentou a implementação realizada para cada um dos produtos propostos na solução. A *framework* desenvolvida foi utilizada com sucesso na implementação de duas aplicações colaborativas, sendo que uma delas era a aplicação alvo, nomeadamente o UbiStudio. Ambas as interfaces dos protótipos foram implementadas com sucesso, assim como as funcionalidades propostas para cada um deles.

A secção de resultados permite desde já tirar algumas ilações sobre o comportamento da *framework*. O desempenho da componente Service denota alguma diferença no tempo de resposta quando comparados os testes efectuados em rede local com os testes efectuados utilizando ligações à Internet, sendo que no entanto o *delay* associado à Internet é muito difícil de prever e controlar. No entanto, os testes realizados em rede local, permitem verificar que a *performance* da *framework* atinge níveis razoáveis mesmo quando o número de ligações ao servidor é relativamente alto. Há ainda que ter em conta que a máquina onde esta está alojada não possui características óptimas principalmente a nível de memória RAM e a nível de ligação à Internet, pelo que o alojamento noutro local deverá ser ponderado. A componente UdpStreamer, tal como a componente Service, também demonstrou um acréscimo de tempo quando foi utilizada uma ligação 3G. Este acréscimo é consideravelmente maior quando comparado ao acréscimo da componente Service, devido ao facto de neste caso a quantidade de informação a circular ser cerca de 300 vezes maior. No entanto a tabela 10 permite verificar que a principal responsável por este decréscimo de *performance* não é a componente UdpStreamer mas sim a taxa de upload da placa 3G utilizada. De facto, esta tabela permite verificar que os pacotes enviados pelo utilizador na rede local para o utilizador na rede 3G demoraram menos de metade do tempo quando comparados com os pacotes enviados pelo utilizador na rede 3G para o utilizador na rede local.

Por último é de realçar que a aplicação foi apresentada na feira internacional BETT. Apesar de a aplicação ainda estar em fase de desenvolvimento, esta experiência efectuada em Londres permitiu não só verificar o *delay* associado a uma ligação tão longa como também verificar a reacção das pessoas ao produto desenvolvido, nomeadamente a questões de interface e de interesse na aplicação.

Capítulo 6

Conclusões e Trabalho Futuro

O principal objectivo do presente trabalho era o desenvolvimento de uma aplicação colaborativa para quadros interactivos. Estes objectivos foram não só totalmente cumpridos, como foi apresentada uma solução genérica para a implementação de funcionalidades de comunicação e colaboração, e que permite a sua utilização em outras aplicações.

A *framework* desenvolvida mostrou no decorrer do presente trabalho, que foi capaz de responder a todos os objectivos inicialmente traçados. Esta *framework* permitiu o desenvolvimento de duas aplicações completamente independentes sem que a mesma sofresse qualquer tipo de alteração. A implementação de funcionalidades colaborativas na aplicação UbiStudio revelou-se relativamente simples através do uso da *framework* desenvolvida. De facto, para cada nova funcionalidade, é apenas necessário definir os objectos necessários para a sua criação e incorporá-los num *array* que deverá ser enviado com o identificador da funcionalidade. É também necessário implementar o método responsável por criar o mesmo objecto na tela de desenho apenas com a informação disponível no *array* recebido. Toda a lógica responsável por disseminar esta informação pelos utilizadores é completamente transparente facilitando assim a implementação. O facto de a *framework* disponibilizar também a gestão de sessões, assim como um sistema hierárquico de utilizadores faz com que estes factores não possam ser abstraídos pela aplicação. As ferramentas de comunicação fornecidas pela framework permitem também desenvolver rapidamente um sistema de chat interno para uma sessão sem grande esforço.

No que diz respeito aos desempenho da *framework* desenvolvida, esta apresentou resultados razoáveis que poderão ainda ser melhorados, principalmente na componente UdpStreamer. Esta componente foi a que apresentou os piores tempos, o que seria previsível visto ser a que mais tráfego albergará.

A aplicação final desenvolvida não só implementa todas as funcionalidades previstas no início dos trabalhos, como foi ainda adicionada a possibilidade de serem iseridas cortinas nos slides por forma a ocultar partes do mesmo. A aplicação atingiu ainda um grau de desenvolvimento suficiente para ser apresentada numa feira internacional tão conceituada como a BETT o que mostra as potencialidades que a mesma de momento detém.

No que diz respeito a trabalho futuro este prende-se principalmente pela adição de funcionalidades à aplicação UbiStudio, assim como a optimização das ferramentas de comunicação/colaboração utilizadas por este. Tal como foi apresentado anteriormente, a componente Service, decresce consideravelmente o seu desempenho com o aumento da quantidade de objectos

presentes no *array* associado à utilização de funcionalidades de um determinado recurso. De momento, as funcionalidades associadas à borracha apresentam o pior comportamento, devido ao facto de estarem a utilizar mais de 250 bytes no *array* de objectos. Estas funcionalidades deverão ser alvo de análise por forma a diminuir a quantidade de informação que será necessário enviar para se poder recriar uma acção em todos os utilizadores. Para além disso, os *codecs* utilizados para comprimir/descomprimir os pacotes de áudio/vídeo não são suficientemente optimizados, sendo que a *performance* da aplicação UbiStudio decresce quando o utilizador inicia inúmeras conferências áudio/vídeo, situação essa que poderá ser optimizada. No que diz respeito a funcionalidades da aplicação, não foram implementadas a adição de vídeos de forma colaborativa, assim como a utilização do *undo/redo*. Estas funcionalidades não foram implementadas por não terem sido identificadas como prioritárias assim como devido à sua complexidade e limitação temporal do presente trabalho. No entanto, ambas as funcionalidades são passíveis de ser implementadas, com as ferramentas que foram disponibilizadas. É ainda de referir que não estão ainda de momento a ser utilizadas as ferramentas de comunicação que permitem a criação de grupos privados, nem a incorporação do sistema de *chat* global. Esta situação deveu-se não à falta de implementação destas funcionalidades pela *framework* (o primeiro protótipo permitia a criação de grupos de conversação utilizando a *framework* para gerir os mesmos, assim como um *chat* de conversação global) mas devido à falta de um desenho consensual na interface. Estas funcionalidades poderiam no entanto atrair os utilizadores pois permitiriam uma maior troca de ideias entre certos indivíduos numa determinada sessão, podendo assim contribuir em última instância para um melhor desenvolver dos trabalhos.

Também não deverá ser descurado o melhoramento das condições de alojamento da *framework* desenvolvida. O alojamento numa máquina de melhor rendimento e com uma melhor ligação à Internet ou mesmo numa *cloud* utilizando por exemplo o serviço Microsoft Azure deverá ser analisado.

Referências

[ASC] Adobe, “Showcase”, Disponível em http://www.adobe.com/devnet/flashplatform/services/collaboration/showcase/. Último acesso a Junho de 2010

[Camoes] Microfil, “Plataforma Camões”, Disponível em http://www.microfil.pt/pt/homepage/plataforma-camoes/. Último acesso a Junho de 2010.

[EdoBoard] “Edoboard”, Disponível em http://www.edoboard.com/en/. Último acesso a Junho de 2010

[FDNT] Luís Paulo Leopoldo Mercado, ”FORMAÇÃO DOCENTE E NOVAS TECNOLOGIAS”, IV Congresso RIBIE, Brasilia 1998

[GWAPI] Google, “Google Wave API Overview”, Disponível em http://code.google.com/intl/pt- PT/apis/wave/guide.html. Último acesso a Julho de 2010

[GWARQ] Lassen ,Sam Thorogood, “Google Wave Federation Architecture”, 2009

[GWave] Google wave, “About”, Disponível em http://wave.google.com/about.html. Último acesso a Junho de 2010

[GWHigh] Google, “Highlights from Google I/O 2009”, Disponível em http://code.google.com/intl/pt-PT/events/io/2009/. Último acesso a Junho de 2010

[IDroo] “IDroo Features”, Disponível em http://www.idroo.com/features. Último acesso a Junho de 2010

[INTEDU] José Alberto Lencastre et Maria José Araújo, “IMPACTO DAS TECNOLOGIAS EM CONTEXTO EDUCATIVO FORMAL”, disponível em http://www.fpce.up.pt/ciie/publs/mjosearaujo/Impacto_Tecnologias_Contexto_Educativo_Formal.pdf

[IntEU] Anna LÖÖF, Heidi SEYBERT, “Internet usage in 2009 - Households and Individuals”. Disponível em http://epp.eurostat.ec.europa.eu/cache/ITY_OFFPUB/KS-QA-09-046/EN/KS-QA-09-046-EN.PDF. Último acesso a Junho de 2010

[IntPT] OberCom, “A Internet em Portugal 2009”. Disponível em http://www.obercom.pt/client/?newsId=428&fileName=rel_internet_portugal_2009.pdf. Último acesso a Junho de 2010

[IWCF] msdn, “Introducing Windows Communication Foundation in .NET framework 4”, Disponível em http://msdn.microsoft.com/en-us/library/ee958158.aspx. Último acesso a Julho de 2010

[LCCS] Adobe, “Adobe Flash Collaboration Service. Developer Guide”, 2009

[MEDU] Ministério da Educação, “Plano Tecnológico da Educação: um meio para a melhoria do desempenho escolar dos alunos”. Disponível em http://www.min-edu.pt/np3/2237.html. Último acesso a Junho de 2010

[Sashi]. Rising, Jim (2009) “Sashimi Waterfall Software Development Proccess”, Managed Mayhem, http://www.managedmayhem.com/2009/05/06/sashimi-waterfallsoftware-development-process/ . Último acesso a Setembro de 2010.

[Skype] “About Skype”, Disponível em http://about.skype.com/. Último acesso a Junho de 2010

[Skype2] “Free Calls. It´s what Skype is made for”, Disponível em http://www.skype.com/intl/en-us/features/allfeatures/skype-to-skype-calls/. Último acesso a Junho de 2010

[Spiral] Barry W. Boehm (1988) “A Spiral Modelo f Stoftware Development and Enhacement”, Disponível em http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/spiral.pdf. Último acesso a Janeiro de 2011

[TEDPT] Elias Blanco e Bento Silva, “TECNOLOGIA EDUCATIVA EM PORTUGAL: CONCEITO, ORIGENS, EVOLUÇÃO, ÁREAS DE INTERVENÇÃO E INVESTIGAÇÃO”, Revista Portuguesa de Educação 1993

[T&W] “TalkAndWrite”,Disponível em http://www.talkandwrite.com/pt/solutions/education/ Último acesso a Junho de 2010

[Ubi] UbiStudio, Disponível em http://www.ubistudio.eu/index.htm, acedido a Dezembro de 2010

[UbiInfo] Folheto UbiStudio, Disponível em http://www.ubistudio.eu/files/ubistudio_folheto.pdf. Último acesso a Dezembro de 2010

[WGW] Google, “What is Google Wave”, Disponível em http://code.google.com/intl/pt-PT/apis/wave/. Último acesso a Junho de 2010

[WiZiQ] WiZiQ, “WiZiQ helps you Learn and Teach Online”, Disponível em http://www.wiziq.com/. Último acesso a Junho de 2010

[WWCF] mdsn, “What is Windows Communication Fountation”, Disponível em http://msdn.microsoft.com/en-us/library/ms731082.aspx.Último acesso a Julho de 2010

Anexo A: Componente Service da *framework*

A.1 – Ficheiro Web.config

```
<?xml version="1.0"?>
<configuration>
  <system.serviceModel>
    <services>
      <service behaviorConfiguration="ucfBehaviour" name="UCFServer.Service">
        <endpoint address="" binding="netTcpBinding"
bindingConfiguration="ucfBinding"
          contract="UCFServer.IServer"/>
        <endpoint address="mextcp" binding="mexTcpBinding"
contract="IMetadataExchange"/>
      </service>
    </services>
    <behaviors>
      <serviceBehaviors>
        <behavior name="ucfBehaviour">
          <serviceMetadata httpGetEnabled="true"/>
          <serviceDebug includeExceptionDetailInFaults="true"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>

    <bindings>
      <netTcpBinding>
        <binding name="ucfBinding" portSharingEnabled="true"
receiveTimeout="01:00:00" maxReceivedMessageSize="2000000000"
                 maxBufferSize="2000000000" maxBufferPoolSize="2000000000">
          <readerQuotas maxDepth="32" maxArrayLength="2000000000"
maxStringContentLength="2000000000" maxBytesPerRead="2000000000"/>
          <reliableSession enabled="false" ordered="true"
inactivityTimeout="01:00:00"/>
          <security mode="None"/>
        </binding>
      </netTcpBinding>
    </bindings>

    <serviceHostingEnvironment multipleSiteBindingsEnabled="true"/>
  </system.serviceModel>
</configuration>
```

A.2 - Ficheiro IServer.class

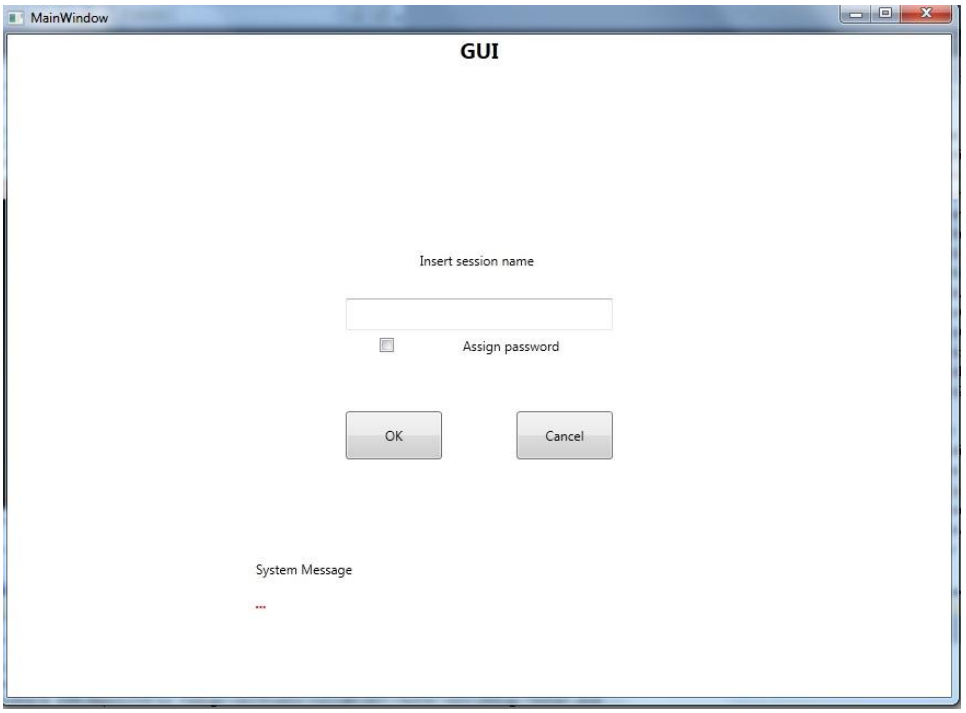
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace UCFServer
{
    /*
     * Interface do canal de comunicação servidor/cliente
    */
    public interface ICallback
    {
        [OperationContract(IsOneWay = true)]
        [ServiceKnownType(typeof(string[]))]
        [ServiceKnownType(typeof(double[]))]
        [ServiceKnownType(typeof(object[]))]
        [ServiceKnownType(typeof(float[]))]
        [ServiceKnownType(typeof(bool[]))]
        void sendSessionCommand(sessionToken msg);

        [OperationContract(IsOneWay = true)]
        void sendMessageCommand(msgToken msg);

        [OperationContract(IsOneWay = true)]
        [ServiceKnownType(typeof(string[]))]
        [ServiceKnownType(typeof(double[]))]
        [ServiceKnownType(typeof(object[]))]
        [ServiceKnownType(typeof(float[]))]
        [ServiceKnownType(typeof(bool[]))]
        [ServiceKnownType(typeof(byte[]))]
        void sendResourceCommand(resourceToken msg);

        [OperationContract(IsOneWay = true)]
        [ServiceKnownType(typeof(byte[]))]
        void sendFileCommand(FileData msg);

    }

    /*
     *  Interface do canal de comunicação cliente/servidor
     */
    [ServiceContract(CallbackContract = typeof(ICallback), SessionMode =
SessionMode.Required)]
    [ServiceKnownType(typeof(object[]))]
    public interface IServer
    {

        [OperationContract(IsOneWay = false)]
        [ServiceKnownType(typeof(resourceToken[]))]
        [ServiceKnownType(typeof(string[]))]
        [ServiceKnownType(typeof(double[]))]
        [ServiceKnownType(typeof(object[]))]
        [ServiceKnownType(typeof(float[]))]
        [ServiceKnownType(typeof(bool[]))]
```

```
        sessionToken sessionCommand(string sCommand, string arg1, string arg2, string
arg3, string arg4, string arg5, string arg6, string arg7, string arg8);


        [OperationContract(IsOneWay = false)]
        sessionToken resourceEditorCommand(string command, string userId, string
session, string resourceId, string[] functionalities);

        [OperationContract(IsOneWay = false)]
        sessionToken resourceCommand(string command, string arg1, string arg2, string
arg3, string arg4, string arg5);

        [OperationContract(IsOneWay = false)]
        [ServiceKnownType(typeof(string[]))]
        [ServiceKnownType(typeof(double[]))]
        [ServiceKnownType(typeof(object[]))]
        [ServiceKnownType(typeof(float[]))]
        [ServiceKnownType(typeof(bool[]))]
        [ServiceKnownType(typeof(byte[]))]
        sessionToken useResource(string userId, string session, string resourceId,
string functionalityId, object[] args);

        [OperationContract(IsOneWay = true)]
        [ServiceKnownType(typeof(string[]))]
        [ServiceKnownType(typeof(double[]))]
        [ServiceKnownType(typeof(object[]))]
        [ServiceKnownType(typeof(float[]))]
        [ServiceKnownType(typeof(bool[]))]
        [ServiceKnownType(typeof(byte[]))]
        void useRealTimeResource(string userId, string session, string resourceId,
string functionalityId, object[] args);

        [OperationContract(IsOneWay = false)]
        [ServiceKnownType(typeof(byte[]))]
        sessionToken conferenceCommand(string command, string arg1, string arg2,
string arg3,byte[] udpMessage);

        [OperationContract(IsOneWay = false)]
        sessionToken messageCommand(string command, string arg1, string arg2, string
arg3, byte[] udpMessage);

        [OperationContract(IsOneWay = false)]
        [ServiceKnownType(typeof(byte[]))]
        [ServiceKnownType(typeof(string[]))]
        [ServiceKnownType(typeof(double[]))]
        [ServiceKnownType(typeof(object[]))]
        [ServiceKnownType(typeof(float[]))]
        FileData fileCommand(string command, string arg1, string arg2, string arg3,
string arg4, byte[] source);

    }
}
```

Anexo B: Interface do protótipo 1

Interface de Login

Interfade de definição de nome



Interface de criação de sessão

Anexo C: Listagem de funcionalidades do recurso “Board” do protótipo 2

No que diz respeito aos recursos e funcionalidades de edição da tela de desenho, a aplicação regista apenas um recurso, nomeadamente “BOARD” com as seguintes funcionalidades:

- CREATESTROKE: Inicializa a criação de uma linha de desenho livre
- RTUPDATESTROKE: Actualiza uma determinada linha, enviando-lhe mais uma coordenada
- COMPLETESTROKE: Completa a criação de uma determinada linha contendo todos os pontos desta.
- CREATESHAPE: Inicializa a criação de uma determinada forma geométrica, identificando o tipo de forma assim como as coordenadas de criação da mesma
- RTUPDATESHAPE: Actualiza o tamanho de uma determinada forma geométrica
- COMPLETESHAPE: Finaliza a criação de uma determinada forma geométrica enviando a altura e largura finais da mesma
- SELECTITEMS: Selecciona um ou mais items
- RTMOVEITEMS: Actualiza em tempo real a posição de um ou mais items previamente seleccionados
- MOVEITEMS: Actualiza a posição de um ou mais items após ter sido terminada a edição da posição dos mesmos
- REMOVEITEM: Remove um ou mais items previamente seleccionados
- UNSELECTITEM: Desselecciona um ou mais items
- RTDELEITEMS: Apaga em tempo real um ou mais items enquanto o utilizador ainda não terminou a acção de apagar (Utilização da borracha)
- DELETEITEMS: Elimina um ou mais items após o utilizador ter terminado a sua acção
- INITRESIZE: Inicializa o processo de redimensionamento de um determinado objecto
- RTRESIZE: Actualiza em tempo real o tamanho de um determinado objecto
- RESIZE: Actualiza o tamanho de um determinado objecto após um utilizador ter terminado de realizar o redimensionamento.
- ADDPAGE: Adiciona um slide à sessão actual
- CNHGB: Altera o fundo de um determinado slide

- CNHGPAGEPOS: Altera a posição de um determinado slide na lista de slides da sessão
- REMOVEPAGE: Elimina um determinado slide da sessão
- FADDED: Adiciona um determinado ficheiro à sessão
- CREATELINE: Adiciona um determinado tipo de linha
- RTUPDATELINE: Actualiza uma determinada linha
- COMPLETELINE: Completa a criação de uma determinada linha
- RTROTATE: Aplica uma rotação a um determinado objecto
- ROTATE: Completa a rotação de um determinado objecto
- COPYDI: Copia um ou mais items
- CUTDI: Corta um ou mais items
- PASTEDI: Cola um ou mais items
- DUPLICATEDI: Duplica um ou mais items
- CHNGATTRDI: Altera a cor e espessura de um determinado objecto
- CHNDIBG: Altera o fundo de um determinado objecto
- ADDPAGECOVER: Adiciona uma cortina a um determinado slide
- RTEDITPAGECOVER: Altera a posição e tamanho de uma determinada cortina em tempo real
- EDITPAGECOVER: Termina a alteração da posição e tamanho de uma determinada linha
- REMOVEPAGECOVER: Remove uma determinada cortina de um determinado slide
- CREATEERTXT: Insere uma caixa de texto
- RTUPDATERCTXT: Actualiza o tamanho de uma determinada caixa de texto em tempo real
- COMPLETERCTXT: Termina a alteração do tamanho de uma determinada caixa de texto
- ADDTXTRCTXT: Adiciona texto a uma determinada caixa de texto
- SCROLLRCTXT: Altera a posição do scroll numa determinada caixa de texto
- COMPLETEEDITRCTXT: Complete a edição de uma determinada caixa de texto
- CREATEWIDTHSPACE: Actualiza o comprimento de uma tela infinita
- CREATEHEIGHTSPACE: Actualiza a altura de uma tela infinica
- GROUPITEMS: Agrupo dois ou mais items
- UNGROUPITEMS: Desagrupa os elementos de um determinado item
- LOCK: Bloqueia um determinado item
- UNLOCK: Desbloqueia um determinado item