

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

Unidad Zacatenco

Departamento de Computación

**Soporte multi-plataforma de espacios colaborativos
ejecutables en arreglos de dispositivos móviles**

T E S I S

Que presenta

Erik Alejandro Reyes Lozano

Para obtener el grado de

Maestro en Ciencias

en Computación

Directora de la Tesis:

Dra. Sonia Guadalupe Mendoza Chapa

Ciudad de México

Diciembre, 2016

Resumen

El incremento y la disponibilidad de servicios de comunicación ha visto un cambio en la naturaleza de los dispositivos móviles y sus aplicaciones. Antes de la aparición generalizada de servicios de comunicaciones móviles se ha tendido a centrarse en el desarrollo de dispositivos de menor tamaño y las formas de interacción que ofrecen. En particular, se ha visto un considerable enfoque en el desarrollo de técnicas y enfoques para superar las limitaciones de visualización y la ergonomía de los nuevos dispositivos (teléfonos inteligentes y tabletas). Sin embargo, el desarrollo de una nueva clase de dispositivos móviles que se enlazan con los servicios de telecomunicaciones y que ofrecen conexiones a otros sistemas (heterogeneidad), significa que tenemos que ampliar nuestra consideración para el diseño de marcos de desarrollo.

Una forma para compartir información entre usuarios en un mismo espacio colaborativo es mediante el despliegue de información con dos o múltiples dispositivos móviles (tabletas y teléfonos inteligentes) que pueden estar localizadas y conectadas en una misma red local.

En la actualidad existe una extensa gama de nuevas plataformas y dispositivos móviles que salen al mercado sin embargo, estos incorporan características particulares, propiciando que no exista una uniformidad entre ellos. En consecuencia, dichos dispositivos móviles se presentan en diversos tamaños y con diferentes características.

Esta diversidad de dispositivos y plataformas móviles no solo pone en evidencia la necesidad de poder compartir la información que se requiere en un grupo de trabajo, sino también impone nuevos requerimientos en el soporte y desarrollo de aplicaciones colaborativas, tales como la adaptación de las aplicaciones a los contextos de uso del usuario, una mayor interactividad con el usuario, el trabajo colaborativo, entre otros.

En esta tesis se toma, como caso de estudio, a la adaptación de interfaces de usuario colaborativas en los dispositivos móviles y como técnica de adaptación, a los espacios colaborativos, que se caracterizan con la interacción cara a cara de los usuarios presentes y sus dispositivos móviles que incluyen funciones de colaboración como: acoplamiento de dispositivos y servicios de comunicación entre los dispositivos. Mediante el diseño de un marco de desarrollo para la creación de aplicaciones colaborativas, que facilite la forma flexible de adaptación de un espacio de trabajo individual a un espacio de trabajo colaborativo. El marco de desarrollo de aplicaciones colaborativas surge como una respuesta a la gran variedad de dispositivos móviles en el mercado y como una opción para homogeneizar el proceso de desarrollo de aplicaciones colaborativas, al implementar y combinar un desarrollo multi-plataforma con métodos de los espacios virtuales. Esta propuesta permite un ciclo de desarrollo rápido y eficiente.

Abstract

The increase and availability of communication services has seen a change in the nature of mobile devices and their applications. Before the emergence of mobile communications services has tended to focus on the development of smaller devices and the forms of interaction they offer. In particular, there has been a considerable focus on developing techniques and approaches to overcome the display limitations and ergonomics of new devices (smartphones and tablets). However, the development of a new class of devices mobile which offer connections to other systems (heterogeneity) means that we have to expand our consideration for the design of development frameworks.

One way to share information between users in the collaborative spaces is by deploying information with two or multiple mobile devices (tablets and smartphones) that can be located and connected on the same local network.

With the constant emergence of new platforms and mobile devices which incorporate particular characteristics ensuring that there is no uniformity between them. Accordingly, said mobile devices come in various sizes and with different characteristics.

This diversity of mobile devices and platforms not only highlights the need to share the information required in a working group, but also imposes new requirements in the support and development of collaborative applications, such as the adaptation of applications to the contexts of user use, greater interactivity with the user and the collaborative work.

In this thesis is taken, as a case study, the adaptation of collaborative user interfaces in mobile devices and as an adaptation technique, to the collaborative spaces, which are characterized by the face-to-face interaction of the present users and their mobile devices that include collaboration functions such as: device coupling and communication services between devices. By designing a development framework for the creation of collaborative applications, it facilitates the flexible way of adapting an individual workspace to a collaborative workspace. The collaborative application development framework emerges as a response to the wide variety of mobile devices in the market and as an option to homogenize the collaborative application development process by implementing and combining multi-platform development with virtual space methods. This proposal allows for a fast and efficient development cycle.

Agradecimientos

Al Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, particularmente al Departamento de Computación.

Al Consejo Nacional de Ciencia y Tecnología por el apoyo económico durante estos dos años de maestría.

A la Dra. Sonia Guadalupe Mendoza Chapa, por creer en mi y darme la oportunidad de realizar esta tesis bajo su supervisión, pero sobre todo, por su guía, apoyo y extraordinarias revisiones de las cuales aprendí mucho. Su ejemplo como investigadora es una fuente de inspiración para mi vida profesional.

A los Dres. Dominique Decouchant y Amilcar Meneses Viveros, por aceptar ser revisores del presente trabajo y por su valioso tiempo invertido.

A mis padres y hermanos, por su apoyo incondicional para ayudarme a salir adelante y permitirme seguir mis sueños.

También quiero agradecer a mis amigos de la maestría Luis, Guillermo y Manuel, por las risas y por hacer de estos dos años una experiencia divertida e inolvidable. Gracias amigos.

Índice general

1. Introducción	1
1.1. Contexto de investigación	1
1.2. Antecedentes	4
1.3. Motivación	7
1.4. Planteamiento del problema	7
1.5. Objetivos	8
1.6. Hipótesis de investigación	8
1.7. Organización de la tesis	8
2. Marco teórico y trabajos relacionados	11
2.1. Definición y modelos de colaboración	11
2.2. Espacios colaborativos	13
2.2.1. Sistemas basados en espacios virtuales	14
2.2.2. Características del espacio virtual	15
2.3. Granularidad de distribución	15
2.4. Contexto de uso	16
2.4.1. Dimensiones del contexto	17
2.4.2. Localización y espacio en el contexto	19
2.5. Arquitecturas de sistemas	20
2.6. Trabajos relacionados	21
2.6.1. Stitching	21
2.6.2. Mobicast	21
2.6.3. MobiSurf	22
2.6.4. ConneCTable	22
2.6.5. Interactive Phone Call	24
2.6.6. Mindmap	25
2.6.7. MOY	25
2.7. Análisis comparativo	26
3. Análisis y diseño del marco de desarrollo	29
3.1. Requerimientos	29
3.2. Arquitectura	30
3.2.1. Descripción general	31
3.2.2. Servicio de descubrimiento	34
3.2.3. Servicio de comunicación	36

3.2.4.	Servicio de actualización	38
3.2.5.	Servicio de presencia	39
3.3.	Casos de uso	40
3.3.1.	Caso de uso: Usuario	40
3.3.2.	Caso de uso: Espacio de trabajo	44
3.3.3.	Caso de uso: Servicio de descubrimiento	47
3.3.4.	Caso de uso: Servicio de comunicación	50
3.3.5.	Caso de uso: Servicio de presencia	53
3.4.	Diagrama de despliegue y de paquetes	55
3.4.1.	Diagrama de despliegue	55
3.4.2.	Diagrama de paquetes	56
3.5.	Diagramas de clases	57
3.5.1.	Clases del paquete <i>workSpace</i>	57
3.5.2.	Clases del paquete <i>setup</i>	57
3.5.3.	Clases del paquete <i>gestures</i>	57
3.5.4.	Clases del paquete <i>global</i>	59
3.5.5.	Clases del paquete <i>services</i>	60
3.6.	Diagramas de secuencia	62
3.6.1.	Servicio de descubrimiento	63
3.6.2.	Servicio de comunicación	64
3.6.3.	Servicio de presencia	65
4.	Implementación del marco de desarrollo	67
4.1.	Metodología de desarrollo	67
4.2.	Herramientas de desarrollo empleadas	69
4.2.1.	Plataformas móviles	69
4.2.2.	Paradigma de programación	71
4.3.	Implementación del marco de desarrollo	74
4.3.1.	Servicio de descubrimiento	74
4.3.2.	Servicio de comunicación	77
4.3.3.	Servicio de actualización	87
4.3.4.	Servicio de presencia	90
4.4.	Configuración del soporte	93
5.	Desarrollo de aplicaciones mediante el marco de desarrollo	95
5.1.	Descripción de la API proporcionada por el soporte	95
5.2.	Aplicación de demostración	97
5.3.	Espacios de trabajo de la aplicación de ejemplo	99
5.3.1.	Espacio de trabajo individual	99
5.3.2.	Espacio de trabajo colaborativo	102
5.3.3.	Eventos producidos por los servicios	105
5.4.	Aspectos técnicos de la aplicación de ejemplo	106
5.4.1.	Interfaz gráfica de usuario	106
5.4.2.	Clases de apoyo	107
5.5.	Mecanismo de recuperación de objetos	108

5.6. Problemas en el desarrollo	110
6. Pruebas y resultados	115
6.1. Pruebas finales	115
6.1.1. Tipos de dispositivos usados en las pruebas	115
6.1.2. Evaluación de la sesión individual	116
6.1.3. Evaluación de la sesión colaborativa	117
7. Conclusiones y trabajo a futuro	123
7.1. Resumen de la proplemática	123
7.2. Conclusiones	124
7.3. Trabajo a futuro	124

Índice de figuras

1.1.	Clasificación <i>Computer Supported Cooperative Work</i> de acuerdo a la ACM	1
1.2.	Clasificación de <i>Frameworks</i> de acuerdo a ACM	2
1.3.	Acoplamiento de dispositivos Castro (2014)	5
1.4.	Servicios del soporte Castro (2014)	5
1.5.	Aplicación de demostración Castro (2014)	6
a.	Gesto de arrastre	6
b.	Gesto combinado	6
c.	Gesto de inclinación	6
1.6.	Gestos de acoplamiento Castro (2014)	6
1.7.	Organización del documento	10
2.1.	Modelo basado en el conjunto de la colaboración [Ellen Taylor-Powell, 2001]	12
2.2.	Diferentes niveles de comunicación, contribución, coordinación, cooperación y colaboración utilizando distintas variables [Ellen Taylor-Powell, 2001]	14
2.3.	Ejemplos de granularidad	17
2.4.	Interacción centralizada	20
2.5.	Tipos de arquitecturas [Baran, 1964]	21
2.6.	Ejemplo de interacción usando Stitching [Hinckley et al., 2014]	22
2.7.	Ejemplo de interacción usando Mobicast con tres diferentes dispositivos [Kaheel et al., 2009]	23
2.8.	Vista general de un prototipo utilizando MobiSurf [Seifert et al., 2012]	23
2.9.	Ejemplo de algunas configuraciones posibles con ConnecTables [Tandler et al., 2001]	24
2.10.	Ejemplo de interacción usando Interactive Phone Call para compartir recursos [Winkler et al., 2011]	24
2.11.	Vista general de Mindmap [Lucero et al., 2010]	25
2.12.	Niños colaborando con sus dispositivos móviles para resolver un problema en la superficie de la mesa de juego [Goh et al., 2014]	26
3.1.	Arquitectura del marco de desarrollo propuesto	32
3.2.	Flujo de la comunicación entre los servicios del marco de desarrollo	35
3.3.	Direcciones de acoplamiento entre dispositivos	36
3.4.	Gesto basado en el <i>touch</i> del dispositivo	37
3.5.	Gesto de acoplamiento mediante acelerómetro	37
3.6.	Diagrama de casos de uso de Usuario	40
3.7.	Diagrama de casos de uso del espacio de trabajo	44

3.8.	Diagrama de casos de uso del servicio de descubrimiento	47
3.9.	Diagrama de casos de uso del servicio de comunicación	50
3.10.	Diagrama de casos de uso del servicio de presencia	53
3.11.	Diagrama de despliegue del soporte	55
3.12.	Diagrama de paquetes del soporte	56
3.13.	Diagrama de clases del paquete <i>WorkSpace</i>	58
3.14.	Diagramas de clases del paquete <i>setup</i>	58
3.15.	Diagrama de clases del paquete <i>gestures</i>	59
3.16.	Diagrama de clases del paquete <i>global</i>	59
3.17.	Diagrama de clases del paquete <i>discovery</i>	60
3.18.	Diagrama de clases del paquete <i>coupling</i>	62
3.19.	Diagrama de clases del paquete <i>updating</i>	62
3.20.	Diagrama de clases del paquete <i>presence</i>	63
3.21.	Diagrama de secuencia del servicio de descubrimiento	64
3.22.	Diagrama de secuencia del servicio de comunicación	65
3.23.	Diagrama de secuencia del servicio de presencia	66
4.1.	Modelo en V	69
4.2.	Gráfica de sistemas operativos por plataforma [Statista, 2016b]	70
4.3.	Gráfica del sistema operativo Android por versión [Statista, 2016a]	71
4.4.	Gráfica del sistema operativo iOS por versión [Statista, 2016c]	72
4.5.	Gráfica de lenguajes de programación por plataforma [Developer Economics, 2014]	73
4.6.	Click largo	78
4.7.	Gestos de arrastre	79
4.8.	Ejes X y Y del acelerómetro de un dispositivo móvil	80
5.1.	Espacio de trabajo individual y botones de configuración	99
5.2.	Menú del botón de configuración	101
5.3.	Menú de la sección <i>Coupling settings</i> del botón de configuración	101
5.4.	Menú de la sección <i>Framework configuration</i> y <i>Mobile info</i> del botón de configuración	102
5.5.	Menú del botón de agregar objeto	102
5.6.	Elementos del espacio de trabajo colaborativo	103
5.7.	Posibles lados de acoplamiento para una tableta	103
5.8.	Posibles lados de acoplamiento para un teléfono inteligente	104
5.9.	Objetos dentro del área colaborativa	104
5.10.	Mensaje de registro	106
5.11.	Mensaje de conexión	106
5.12.	Mensaje de desconexión	107
5.13.	Estado inicial de una sesión colaborativa	110
5.14.	Interacción en el espacio de trabajo colaborativo	111
5.15.	Servicio de presencia verificando la sesión colaborativa	112
5.16.	Mecanismo de recuperación de objetos en una sesión colaborativa	113
5.17.	Recuperación de los objetos en una sesión colaborativa	114
6.1.	Filtrado de los mensajes de identificación	116

6.2. Porcentaje de mensajes de identificación recibidos	117
6.3. Filtrado de los mensajes de abandono	118
6.4. Porcentaje de mensajes de identificación abandono	118
6.5. Mensajes de actualización, color de fondo, presencia y actualización de listas . . .	119
6.6. Configuración de dispositivos	120
6.7. Intentos de acoplamiento usando el gesto de arrastre	120
6.8. Intentos de acoplamiento usando el gesto de inclinación	121
6.9. Intentos de acoplamiento usando ambos gestos (arrastre e inclinación)	121

Índice de tablas

2.1. Análisis comparativo (parte 1)	26
2.2. Análisis comparativo (parte 2)	27
3.1. Caso de uso 1: Realizar un gesto de acoplamiento	41
3.2. Caso de uso 2: Realizar un gesto de inclinación.	41
3.3. Caso de uso 3: Realizar un gesto de arrastre.	41
3.4. Caso de uso 4: Iniciar el espacio de trabajo.	42
3.5. Caso de uso 5: Interactuar con el espacio de trabajo.	42
3.6. Caso de uso 6: Crear y modificar objetos.	42
3.7. Caso de uso: Configurar el espacio de trabajo.	43
3.8. Caso de uso 8: Interactuar con el espacio de trabajo.	44
3.9. Caso de uso 9: Crear y eliminar objetos.	45
3.10. Caso de uso 10: Configurar el espacio de trabajo.	45
3.11. Caso de uso 11: Activar gesto de inclinación.	45
3.12. Caso de uso 12: Activar gesto de arrastre.	46
3.13. Caso de uso 13: Visualizar notificaciones.	46
3.14. Caso de uso 14: Recolectar información de dispositivos.	48
3.15. Caso de uso 15: Comprobar disponibilidad del dispositivo.	48
3.16. Caso de uso: Difundir información de identificación.	49
3.17. Caso de uso 17: Difundir la disponibilidad del dispositivo.	49
3.18. Caso de uso 18: Procesar gestos de acoplamiento.	50
3.19. Caso de uso 19: Procesar un gesto de inclinación.	51
3.20. Caso de uso 20: Procesar un gesto de arrastre.	51
3.21. Caso de uso 21: Enviar y recibir intentos de acoplamiento.	52
3.22. Caso de uso 22: Iniciar sesiones colaborativas.	52
3.23. Caso de uso 23: Difundir la presencia del dispositivo.	53
3.24. Caso de uso 24: Recolectar mensajes de presencia.	54
3.25. Caso de uso 25: Verificación de los mensajes de presencia.	54
4.1. Uso de sistemas operativos [IDC, 2016]	70
4.2. Lenguajes de programación móvil	72
6.1. Dispositivos usados en las pruebas	115

Índice de Algoritmos

1.	Propagación de los datos de identificación del dispositivo para Android e iOS	75
2.	Recolección de los datos de identificación del dispositivo para Android e iOS	76
3.	Detección de clicks largos para Android e iOS	79
4.	Detección de un gesto de arrastre para Android e iOS	81
5.	Gesto de inclinación para Android e iOS	84
6.	Configuración de gestos de acoplamiento para Android e iOS	85
7.	Propagación de un intento de acoplamiento para Android e iOS	86
8.	Captura de un intento de acoplamiento para Android e iOS	87
9.	Envío de objetos a un solo dispositivo para Android o iOS	88
10.	Envío de objetos a múltiples dispositivos Android e iOS	88
11.	Hilo en segundo plano encargado de enviar un objeto en formato JSON para Android e iOS	89
12.	Servidor local encargado de recibir y procesar objetos para Android e iOS	90
13.	Difusión de la presencia del dispositivo para Android e iOS	91
14.	Recolección de mensajes de presencia para Android e iOS	92
15.	Verificación del estado de presencia para Android e iOS	93
16.	Configuración básica del soporte para Android e iOS	94
17.	Configuración de la aplicación de ejemplo para Android e iOS	97
18.	Métodos de ejemplo de la clase <code>ExampleWorkTogether</code> para Android e iOS	97
19.	Creación de un objeto en el espacio de trabajo para Android e iOS	98
20.	Creación de un objeto externo en el espacio de trabajo para Android e iOS	98
21.	Lógica de la clase <code>WorkSpace</code> para Android e iOS	109

Capítulo 1

Introducción

1.1. Contexto de investigación

En 1964, ACM acrónimo de *Association for Computing Machinery* crea un sistema de clasificación, para organizar el campo de las ciencias de la computación. En la revisión del año 2012 [ACM, 2016] ACM clasifica 13 categorías, en dos de las cuales se ubica el contexto de esta tesis que son: ***Computer Supported Cooperative Work*** (ver figura 1.1) y ***Frameworks*** (ver Figura 1.2). A demás *Computer Supported Cooperative Work* esta estrechamente relacionado con ***HCI*** y ***Ubiquitous and mobile computing***.

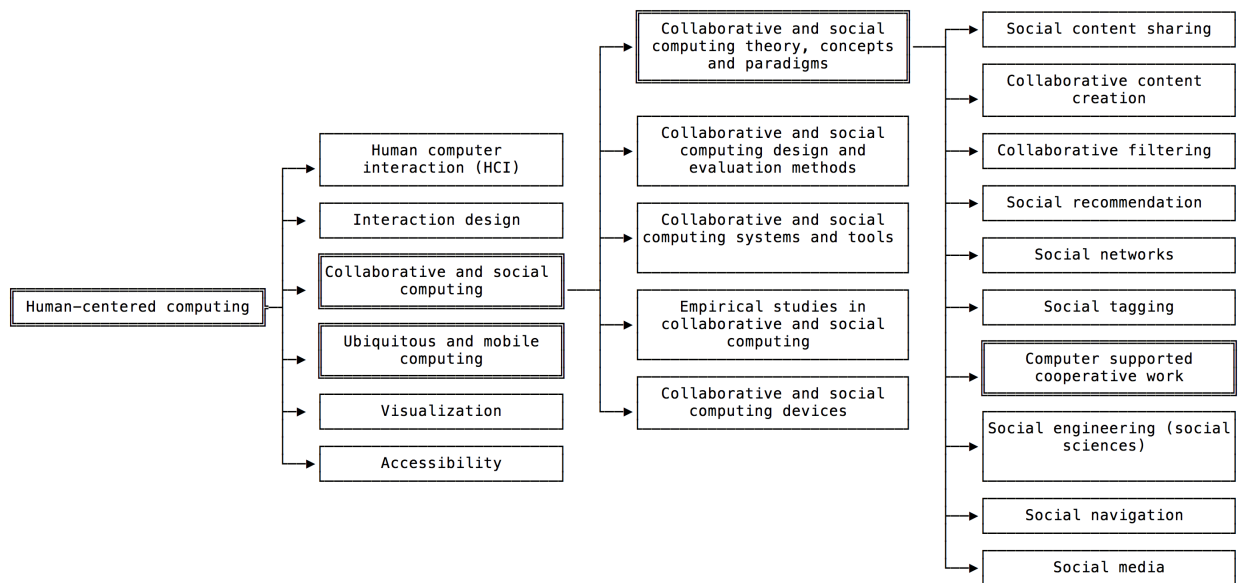


Figura 1.1: Clasificación *Computer Supported Cooperative Work* de acuerdo a la ACM

En el área de investigación de las Ciencias de Computación existe un área específica que es conocida como HCI (*Human-Computer Interaction*) [Card et al., 1983], la cual tiene como objetivo el estudio de la forma en que las tecnologías de la información influyen en el trabajo y en las actividades humanas cotidianas. El término “tecnología de información” incluye hoy en

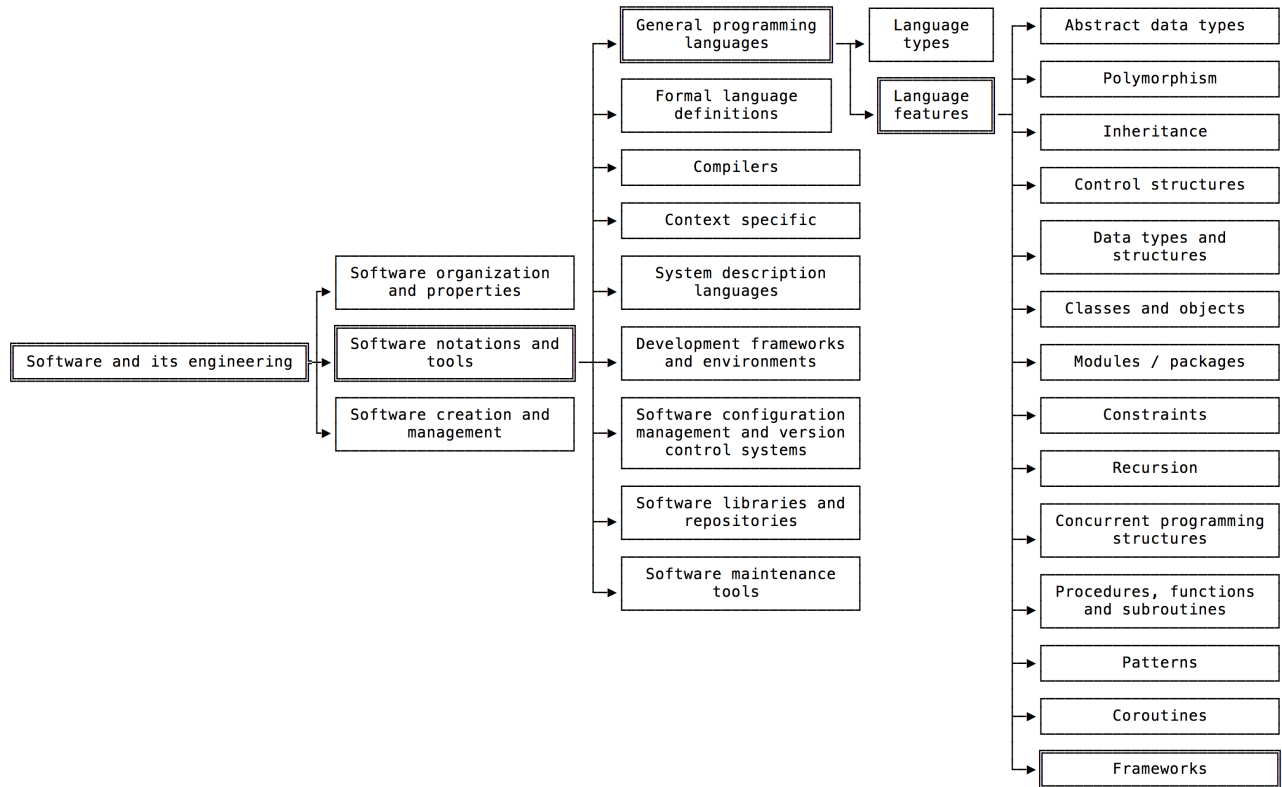


Figura 1.2: Clasificación de *Frameworks* de acuerdo a ACM

día a la mayoría de las computadoras, teclados de los teléfonos inteligentes, electrodomésticos, sistemas de navegación para automóvil y sensores, entre otros.

Actualmente, las interfaces de usuario (GUI por sus siglas en inglés) facilitan la interacción entre los usuarios y los dispositivos móviles (teléfonos inteligentes y tabletas), incluyendo la distribución de la interfaz de usuario a través de diferentes dispositivos. Esto implica que la GUI puede ser dividida, movida, copiada sobre diferentes dispositivos móviles ejecutándose en el mismo o en diferentes sistemas operativos. Estas nuevas formas de manipular la GUI se le conoce como Interfaces de Usuario Distribuidas (DUI por sus siglas en inglés) [Elmqvist, 2011].

Un *Framework* ofrece un conjunto de soluciones para un tipo de problema determinado en un contexto. Un *framework* puede ofrecer un conjunto componentes, como bibliotecas, que pueden contener plantillas que definen el funcionamiento de las aplicaciones. Según Nelson, un *framework* ayuda a los desarrolladores a proveer soluciones para un dominio específico, y proporciona diseños estructurados para una infraestructura. Cuando se crean nuevas piezas que pueden ser incluidas en el *framework*, deben tener un impacto mínimo sin afectar al resto de los componentes [Nelson, 1994].

El término CSCW (*Computer Supported Cooperative Work*) [Grudin, 1994] fue introducido por Grief y Cashman en 1984, como “una vía para describir cómo la tecnología de cómputo puede ayudar a los usuarios a trabajar juntos en grupos”. CSCW se refiere a las teorías y metodologías relativas a sistemas de soporte para equipos de trabajo, pero no incluye la parte técnica, i.e., no incluyen las aplicaciones en sí. CSCW constituye pues, según Grudin, una disciplina científica que describirá las funcionalidades de las aplicaciones de Groupware (conjunto de programas

informáticos colaborativos), partiendo del estudio teórico y práctico y de cómo las personas trabajan en grupo e interaccionan entre si. CSCW propone, a partir de estas observaciones, las pautas para el desarrollo tecnológico de esta colaboración en el logro de un fin común. CSCW constituye un campo de investigación multidisciplinario en el que intervienen áreas como la Educación, la Informática, la Psicología, la Sociología y la Economía.

La actual investigación en CSCW se centra en desarrollar nuevas tecnologías para la coordinación de grupos de personas que trabajan conjuntamente. Entre las definiciones que encontramos para CSCW son:

- Uso de las Tecnologías de la Información y Comunicación para ayudar a que los grupos de personas trabajen de forma efectiva [Malone and Crowston, 1990].
- Es la disciplina científica que trata de conocer la forma en que las personas trabajan en grupo: cómo interactúan, cómo se comunican, cómo colaboran, con el fin de proponer metodologías para el desarrollo o creación de aplicaciones o herramientas informáticas que apoyen el proceso de trabajo del grupo [Schmidt, 1991].
- Es un término que combina la comprensión del modo en que las personas trabajan en grupo con las facilidades tecnológicas de las redes de computadoras, el hardware y el software asociados [Paul, 1991].

El término *Cómputo Ubicuo* fue descrito por Weiser [Weiser, 1993]. La esencia de su visión era la creación de entornos repletos de computación y capacidad de comunicación e integrados de forma inapreciable por las personas. Uno de los objetivos más importantes del *Cómputo Ubicuo* es integrar los dispositivos computacionales, tanto como sea posible, para hacer que se incorporen en la vida cotidiana.

El *Cómputo Ubicuo* debe permitir a los usuarios que se centren en las tareas que deben hacer, no en las herramientas que deben usar, pudiendo suponer una revolución que cambie el modo de vida. El hecho de posicionar la computación en un “segundo plano” tiene dos significados:

1. Se debe integrar en los objetos, cosas, tareas y entornos cotidianos.
2. La integración se debe realizar de forma que no debe interferir en las actividades cotidianas, proporcionando un uso más cómodo, sencillo y útil.

Finalmente, el desarrollo de *frameworks* para sistemas móviles es otro de los campos de investigación en el que se inscribe el soporte de la presente tesis, el cual se puede definir básicamente como un entorno de computación en movimiento. Su objetivo consiste en permitir a un usuario acceder a datos, información u otro tipo de objetos lógicos desde cualquier dispositivo, mientras el usuario se encuentra en movimiento. El principio que persigue es “información en donde sea y cuando sea” [Talukder et al., 2010]. Debido a que el soporte propuesto está enfocado a los dispositivos móviles, es indispensable tomar en cuenta los beneficios y las desventajas que éstos acarrearán (e.g., duración de la batería, tamaño de la pantalla, la emisión y recepción de datos que se realiza a través del WiFi), los cuales ya han sido ampliamente estudiados.

1.2. Antecedentes

Como antecedente del tema de investigación se puede mencionar una tesis que propone el desarrollo de un *framework* para el trabajo individual y colaborativo compatible con el sistema operativo Android [Castro, 2014].

En la tesis de Castro (2014) se propuso un *framework* que da soporte multi-usuario para dispositivos móviles (teléfonos inteligentes y tabletas) que facilita, de forma flexible, transiciones del trabajo individual al trabajo colaborativo y viceversa, mediante técnicas de remodelación y redistribución plásticas. Este trabajo consiste en el diseño e implementación de un soporte para la plataforma Android.

Este soporte cuenta con tres servicios principales: descubrimiento de dispositivos, control de acoplamiento y distribución de actualizaciones:

- **Descubrimiento de dispositivos:** este servicio se encarga de recolectar los datos de identificación de otros dispositivos presentes en la misma red, así como de distribuir los propios. Así mismo, se encarga de verificar periódicamente que se encuentren disponibles aquellos dispositivos que ya han sido encontrados e identificados.
- **Control de acoplamiento:** este servicio tiene como propósito la creación de sesiones colaborativas al combinar dos o más espacios de trabajo individuales en un único espacio de trabajo compartido, a través de gestos de acoplamiento (ver Figura 1.4). Además este servicio envía y recibe las características de las pantallas de los dispositivos, así como de los sensores, adaptadores de red y funcionalidades del sistema operativo de cada dispositivo.
- **Distribución de actualizaciones:** este servicio se encarga de enviar y recibir aquellos mensajes que no estén relacionados con las tareas de los servicios descubrimiento de dispositivos y control de acoplamiento.

En la figura 1.4 se muestran los servicios descritos anteriormente, los cuales conforman el soporte propuesto por Castro (2014). Uno de los aspectos más importantes de este soporte es el reconocimiento y procesamiento de los gestos de acoplamiento.

Los gestos de acoplamiento pueden ser tan complejos y robustos como lo permita el hardware y software de un dispositivo móvil (ver Figura 1.6). En este *framework*, los gestos propuestos son los siguientes: arrastre, inclinación y combinados.

1. **Gesto de arrastre concurrente sobre las pantallas de ambos dispositivos:** este gesto consiste básicamente en arrastrar un dedo sobre cada una de las pantallas. Los gestos de arrastre deben ser en direcciones opuestas, de tal forma que se denoten los lados por los cuales se desea unir los espacios de trabajo (ver Figura 1.5a).
2. **Gesto de inclinación de los dispositivos en direcciones opuestas:** este gesto requiere inclinar los dispositivos móviles al menos en un cierto ángulo durante un tiempo determinado, de tal forma que los lados por los cuales se acoplarán los dispositivos permanezcan sobre la superficie (ver Figura 1.5c).
3. **Gestos combinados:** es la combinación de los dos gestos anteriores (ver Figura 1.5b).

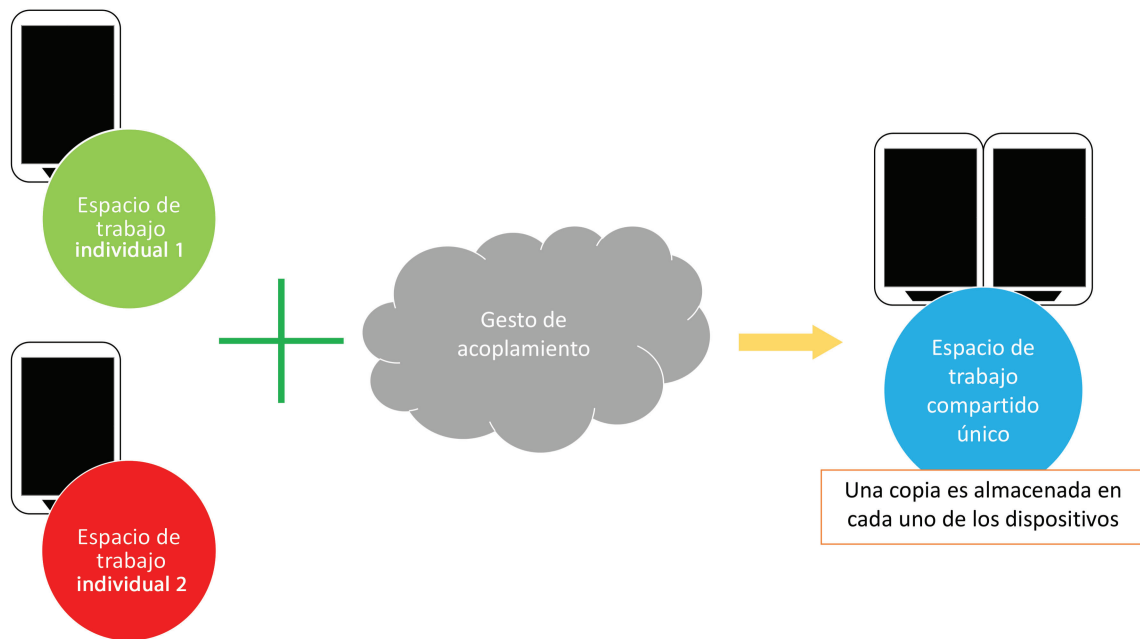


Figura 1.3: Acoplamiento de dispositivos Castro (2014)

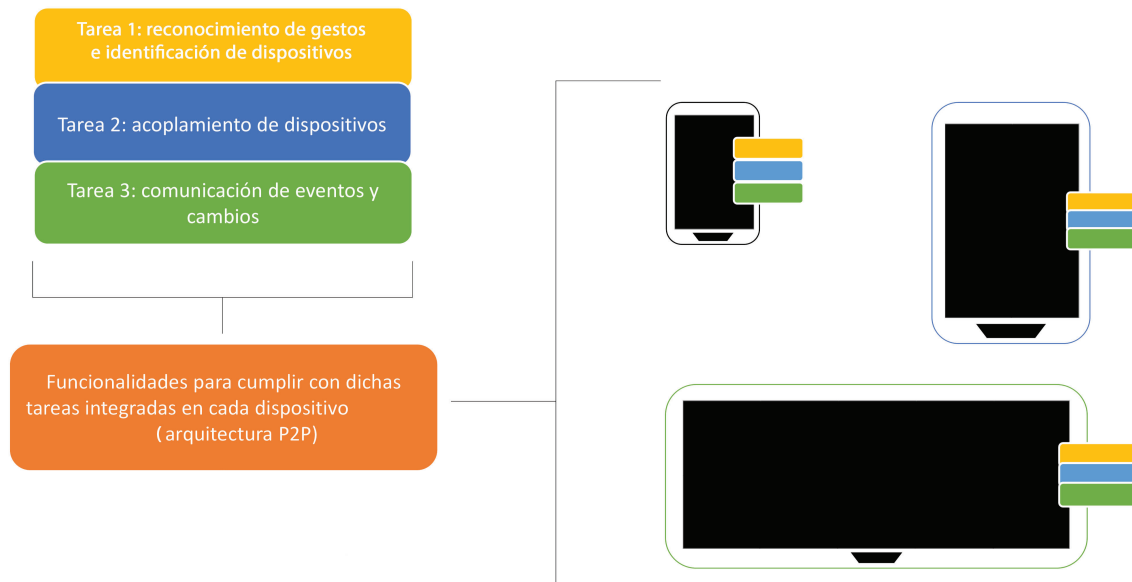


Figura 1.4: Servicios del soporte Castro (2014)



Figura 1.5: Aplicación de demostración Castro (2014)

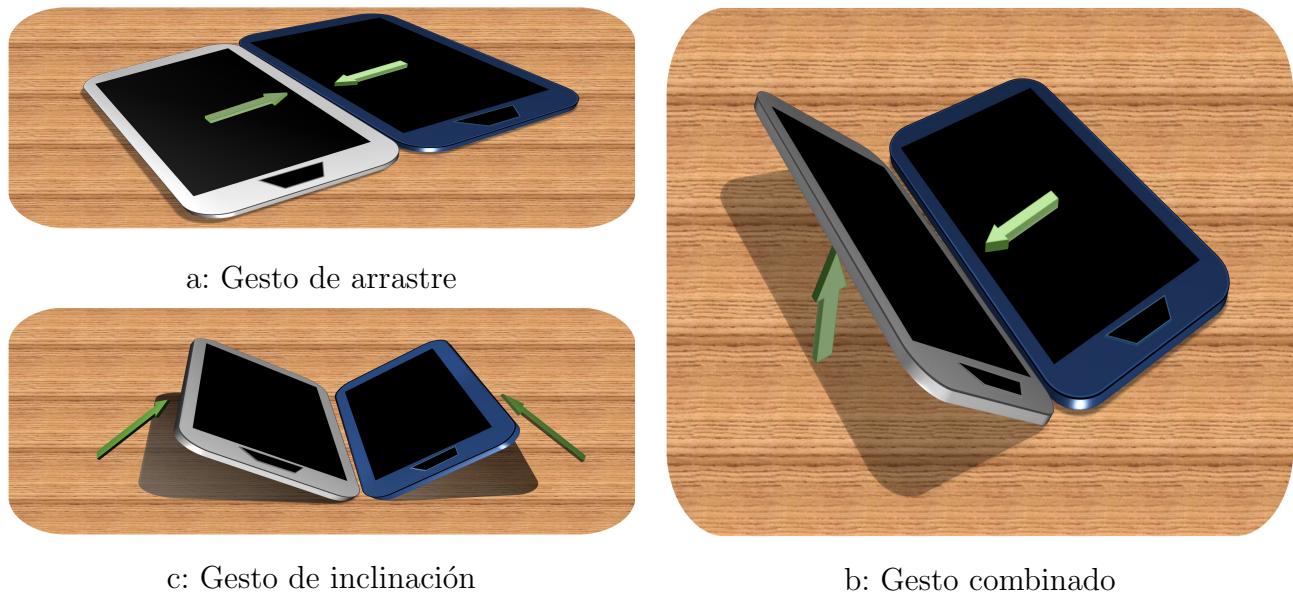


Figura 1.6: Gestos de acoplamiento Castro (2014)

El trabajo de Castro (2014) proporciona los servicios básicos mencionados anteriormente y los gestos para crear una sesión colaborativa (ver Figura 1.5). Sin embargo, el arreglo de dispositivos esta limitado a cuatro dispositivos y no es homogéneo, ya que solo soporta dispositivos móviles con sistema operativo Android. Además no cuenta con una configuración básica para el soporte (e.g., selección de gestos).

1.3. Motivación

Al inicio de la creación de los dispositivos móviles, su propósito era el uso exclusivo para la comunicación entre personas, pero con la evolución de la tecnología (*hardware y software*) estos se modificaron en tamaño y capacidad. Es por esta razón que originalmente fueron pensados para el uso personal por lo que la interacción entre dispositivos era limitada. Sin embargo, actualmente con los avances en las tecnologías de comunicación y de interacción los dispositivos móviles permiten nuevas formas de interactuar e.g., colaborativa.

El gran espacio y tamaño de las pantallas [Campos et al., 2013], que poseen los nuevos teléfonos inteligentes y tabletas, proporcionan espacios de trabajo con mayores dimensiones de tamaño, presentándose nuevas formas interactuar y colaborar. En paralelo a esta revolución tecnológica, las superficies interactivas de los dispositivos móviles tienen diferentes tamaños. De hecho, en los últimos años, hay un incremento en el uso de teléfonos inteligentes y tabletas. Existen oportunidades de innovación en el desarrollo de soportes, pero los espacios físicos y virtuales no han abordado todavía a fondo, el problema de llevar las actividades de colaboración eficaces, utilizando múltiples superficies interactivas, especialmente en ámbitos de trabajo complejos.

1.4. Planteamiento del problema

Hoy en día existe una gran diversidad de marcos de desarrollo para aplicaciones móviles (e.g., de uso individual o colaborativas) que establecen una conexión por medio de una red inalámbrica. Las arquitecturas que pueden usar estos marcos son: centralizadas, descentralizadas o distribuidas. La mayoría de los marcos de desarrollo hacen uso de arquitecturas centralizadas, pero presentan inconvenientes debido a su funcionamiento, ya que la información debe pasar por un nodo principal (servidor). En situaciones en que el nodo principal falla se pueden producir cuellos de botella e indisponibilidad del servidor.

En particular, existen algunos prototipos que permiten la interacción colaborativa con dispositivos móviles, tales como: Stitching [Hinckley et al., 2014], Mobile Collocated Interactions [Lucero et al., 2015] y Collaboration Meets Interactive Surfaces [Campos et al., 2013], sin embargo, el soporte que ofrecen está limitado en ciertos aspectos, tales como no permitir realizar operaciones concurrentes, ausencia de soporte para dispositivos heterogéneos, la configuración del soporte es mínima o nula, así como con un número limitado de los mismos. Adicionalmente, cabe mencionar que ninguno de los prototipos no provee las herramientas necesarias para poder reproducir sus funcionalidades y resultados en otro tipo de aplicaciones, i.e., que las transiciones (e.g., individual a colaborativa o colaborativa a individual) están limitadas dentro de sus respectivos modelos de tareas y no pueden ser extendidos ni sustituidos.

La extensa variedad de dispositivos móviles que existe hoy en día, junto con las tecnologías que conforman a cada uno de sus componentes (tanto software como hardware) crean áreas

de oportunidad en diversos campos de investigación (e.g., Cómputo Ubicuo, Interacción Hombre Computadora, Trabajo Cooperativo Asistido por Computadora) que podrían ser explotadas para permitir a los usuarios verse envueltos en interacciones más dinámicas e intuitivas.

1.5. Objetivos

El objetivo general de la presente tesis consiste en la extensión de un marco de desarrollo para la plataforma Android y la creación de un marco de desarrollo para la plataforma iOS, para aplicaciones colaborativas, que facilite el acoplamiento de un espacio de trabajo individual a un espacio de trabajo colaborativo para dispositivos móviles que establecen una conexión por medio de una red inalámbrica local.

A continuación se enumeran los objetivos específicos de la extensión del soporte propuesto:

1. Diseñar un servicio de actualizaciones con el fin de notificar la presencia de los dispositivos existentes en una red para el sistema operativo iOS.
2. Diseñar un servicio de comunicación con el fin de unir dos o mas áreas de trabajo individual para el sistema operativo iOS.
3. Diseñar un servicio de descubrimiento con el fin de identificar otros dispositivos presentes en la misma red para el sistema operativo iOS.
4. Diseñar un servicio que verifique el estado de presencia de otros dispositivos que estén en una sesión colaborativa para el sistema operativo Android e iOS.
5. Diseñar e implementar una aplicación compatible los dispositivos Android e iOS, que permita validar la factibilidad y utilidad de la extensión del soporte propuesto.

1.6. Hipótesis de investigación

Mediante el diseño de un marco de desarrollo que relacione e incorpore las técnicas de homogeneidad, distribución, aplicaciones multi-plataforma y espacios de trabajo individual y colaborativo, establecer un procedimiento para llevar a cabo una adecuada adaptación e implementación de un marco de desarrollo con múltiples contextos de uso.

1.7. Organización de la tesis

El documento de tesis está organizado de la siguiente forma (ver Figura 1.7):

Capítulo 2. Marco teórico y trabajos relacionados: se describe la teoría requerida para desarrollar el marco de desarrollo propuesto, así como las técnicas mediante las cuales es posible establecer un acoplamiento para formar espacios de trabajo colaborativo. Así mismo, se describen algunos prototipos encontrados en la literatura que forman parte del trabajo relacionado y son representativos en este contexto.

Capítulo 3. Análisis y diseño del marco de desarrollo: corresponde a la etapa de análisis y diseño del soporte propuesto, contempla aspectos de diseño del modelo de arquitectura propuesto, así como diagramas de cada caso de uso de los servicios que ofrece el marco de desarrollo propuesto.

Capítulo 4. Implementación del marco de desarrollo: se describe la metodología de desarrollo para el marco de desarrollo propuesto, las herramientas de desarrollo usadas, la implementación de los servicios, la configuración del marco de desarrollo y la implementación de la aplicación de demostración.

Capítulo 5. Desarrollo de aplicaciones mediante el marco de desarrollo: se describe un ejemplo de una aplicación usando el marco de desarrollo propuesto y el proceso de interacción de los usuarios con la aplicación.

Capítulo 6. Pruebas y resultados: corresponde a la descripción de las pruebas de usabilidad realizadas a la aplicación desarrollada por el marco de desarrollo propuesto.

Capítulo 7. Conclusiones y trabajo a futuro: se presenta la recapitulación del problema resuelto en esta tesis, así como las conclusiones, las contribuciones y el trabajo futuro que se deriva de este.

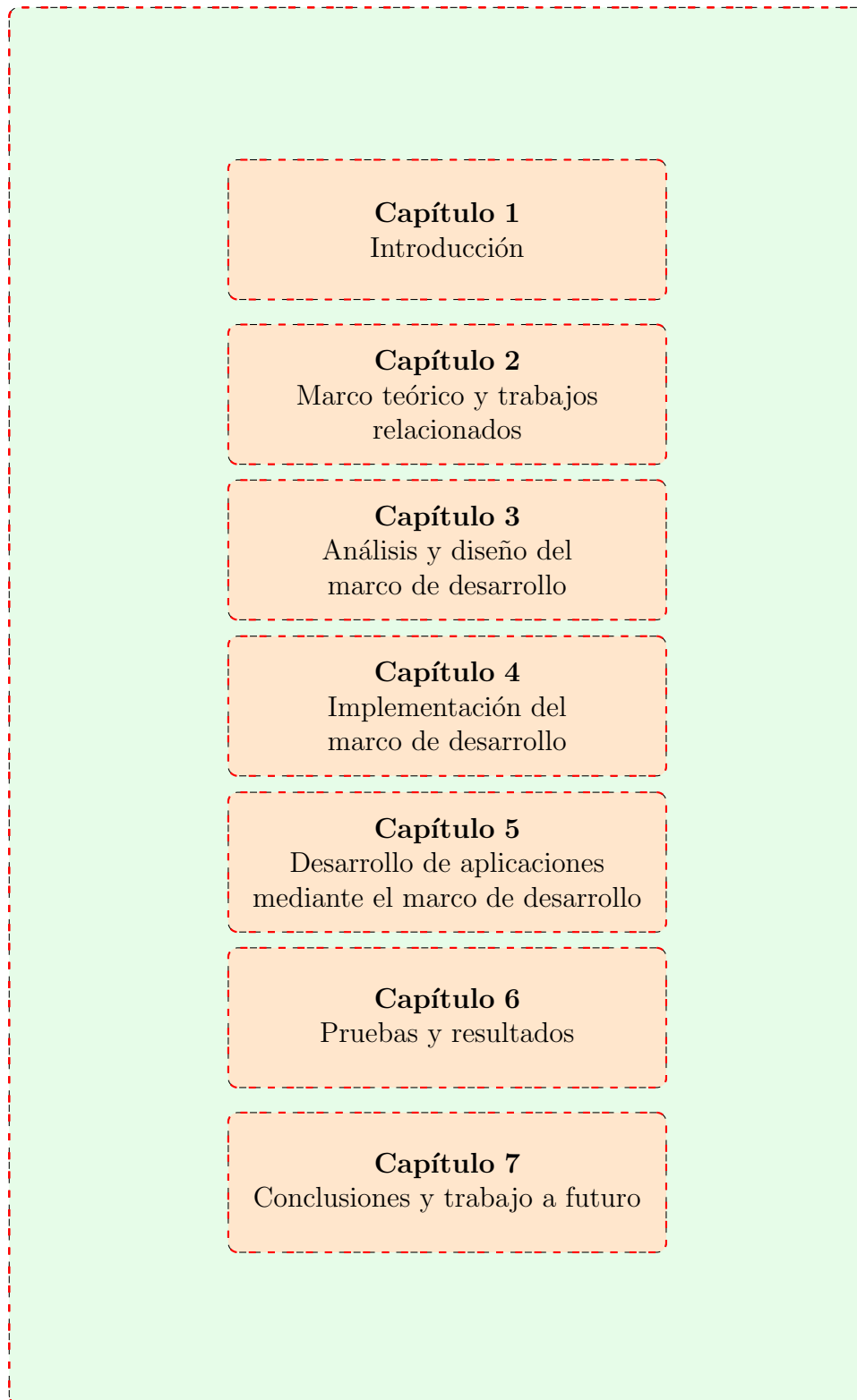


Figura 1.7: Organización del documento

Capítulo 2

Marco teórico y trabajos relacionados

El presente capítulo está estructurado en siete partes que cubren diferentes aspectos significativos para la presente tesis. La primera parte cubre algunas de definiciones importantes de colaboración y describe dos modelos de colaboración (Sección 2.1). La segunda parte explica el uso de los espacios colaborativos (Sección 2.2). La tercera parte describe la granularidad de distribución de los objetos en un espacio de trabajo (Sección 2.3). La cuarta parte explica las definiciones más importantes del concepto de “contexto de uso” (Sección 2.4). La quinta parte describe las arquitecturas posibles para el diseño del soporte (Sección 2.5). La sexta parte describe trabajos relacionados específicos de sistemas y aplicaciones que dan soporte a la colaboración que, en algunos casos, también son sensibles al contexto de uso (Sección 2.6). Por último, se presenta un análisis comparativo de los sistemas colaborativos estudiados en este capítulo, destacando sus principales características, ventajas y desventajas (Sección 2.7).

2.1. Definición y modelos de colaboración

La mayoría de las personas tienen una comprensión intuitiva de lo que significa colaborar. La palabra colaborar como tal, proviene del latín *collaborāre*, que significa “trabajar juntos”. London interpreto este significado como “trabajar juntos de forma sinérgica” [Scott London, 2016]. Gray define la colaboración como “un proceso de toma de decisiones entre los actores clave de un dominio del problema” [Gray, 1990]. Roberts y Bradley llamaron a la colaboración “un proceso interactivo que tiene un propósito compartido de transmutación” [Roberts and Bradley, 1991].

La gente usa la palabra colaboración en diversos contextos y de manera intercambiable con términos tales como la coordinación y la cooperación. Denning y Yaholkovsky mencionaron que la coordinación y la cooperación son formas más débiles de trabajar en grupo, sin embargo todas estas actividades requieren el intercambio de información con otros [Denning and Yaholkovsky, 2008]. Taylor-Powell añade el componente de contribución y llegó a la conclusión de que para una colaboración eficaz, cada miembro del grupo tiene que hacer una contribución a la colaboración [Taylor-Powell, 2008]. El uso de la comunicación, la contribución, la coordinación y la cooperación como pasos esenciales para la colaboración, requiere una forma más estricta de la integración, pasando de la simple comunicación a mayores niveles de coordinación, cooperación y colaboración.

A continuación se muestran dos modelos de colaboración propuestos por Taylor-Powell, el

primer modelo de colaboración se sintetiza y se presenta en la figura 2.1. Este modelo tiene cinco componentes: la comunicación (intercambio de información), la contribución, la coordinación, la cooperación y la colaboración. Muestra qué actividades se apoyan en otras. Por ejemplo, la coordinación es un subconjunto de la colaboración, lo que indica que para una colaboración significativa, tiene que haber coordinación de personas y eventos. Estos cinco componentes se describen a continuación:

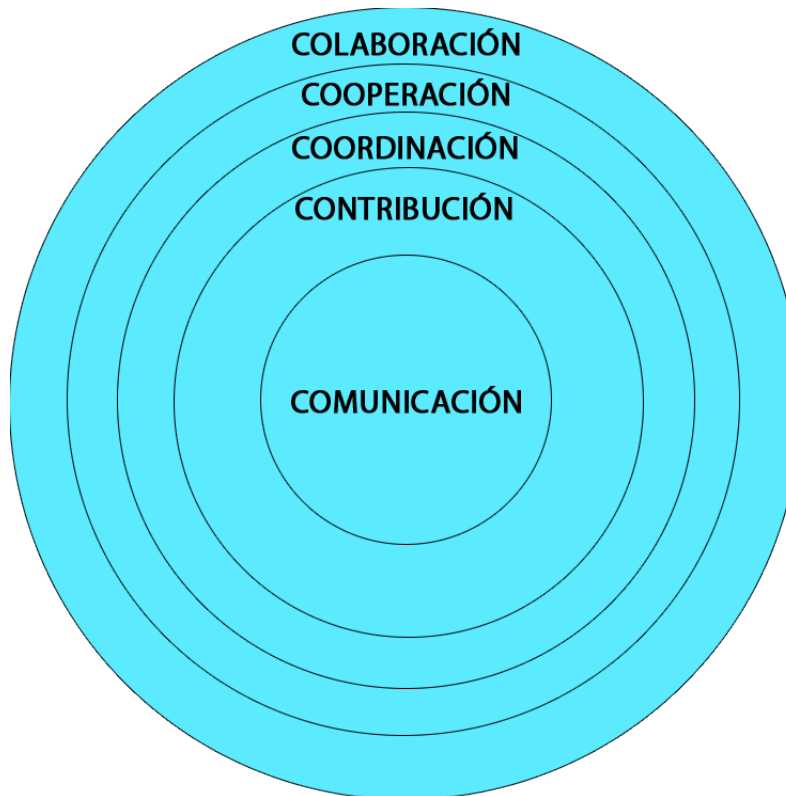


Figura 2.1: Modelo basado en el conjunto de la colaboración [Ellen Taylor-Powell, 2001]

- **Comunicación:** este es un proceso de envío o intercambio de información, que es uno de los requisitos básicos para la realización de la colaboración o el mantenimiento de cualquier tipo de relación productiva. Por ejemplo, un mensaje en un cartel informativo sobre una conferencia de computación es una manera para que los organizadores del evento puedan comunicarse con el público que se interese.
- **Contribución:** esta es una relación informal mediante la cual las personas se ayudan entre sí para lograr sus objetivos individuales.
- **Coordinación:** este es un proceso de conexión de los diferentes agentes por una acción armoniosa.
- **Cooperación:** esta es una relación en la que diferentes personas con intereses similares toman parte en las actividades de planificación, en las funciones de negociación y en el uso compartido de recursos para lograr objetivos comunes. Además de la coordinación, la cooperación implica seguir algunas reglas de interacción.

- **Colaboración:** este es un proceso que implica a varias personas que pueden ver diferentes aspectos de un problema. Se involucran en un proceso que va más allá de su propia experiencia y la visión individual para completar una tarea o proyecto. En contraste a la cooperación, la colaboración implica la creación de una solución o un producto que es más que la suma de la contribución de cada participante.

En la figura 2.2 se muestran las diferencias entre la comunicación, contribución, coordinación, cooperación y colaboración, mediante las variables interacción, intención, confianza, intervención humana, simetría de beneficios y nivel de conciencia, las cuales a continuación se describen:

- **Interacción:** mientras que la comunicación está en el centro de las otras actividades, es posible tener poca o ninguna interacción durante la comunicación. Por ejemplo, un administrador de un sistema envía un correo electrónico a un usuario particular y puede no requerir ninguna interacción adicional con el usuario. La colaboración, sin embargo, requiere un alto nivel de interacción entre los participantes.
- **Intención:** al igual que en la interacción, un proyecto de colaboración requiere la intención mucho más fuerte en comparación con aquellas tareas que simplemente necesitan coordinación de eventos o una entidad que coopera con otro.
- **Confianza:** para tener una colaboración efectiva y mutuamente beneficiosa, los participantes necesitan establecer confianza, la cual no es necesaria para coordinar o cooperar.
- **Intervención humana:** la comunicación puede no requerir mucha intervención humana. Por ejemplo, la publicación de un mensaje en un pizarrón de anuncios es un acto de comunicación, pero rara vez se requiere una comunicación interactiva o participación. La colaboración, por el contrario, requiere que los participantes estén en constante comunicación uno con el otro.
- **Simetría de beneficios:** el tipo de colaboración se considera que beneficia a todos los involucrados en el proceso. La cantidad de beneficio puede variar en función de los roles y responsabilidades de los participantes.
- **Nivel de conciencia:** para lograr una colaboración interactiva, intencional y de beneficio mutuo es imperativo que todos los participantes sean conscientes de las acciones y contribuciones de los demás. Esto también ayuda a establecer la confianza entre los participantes.

2.2. Espacios colaborativos

La estructura del espacio alrededor de nosotros guía nuestras acciones e interacciones. Con los años de experiencia, nosotros tenemos un grado de habilidad en la estructuración y la interpretación del espacio para nuestros propósitos individuales o colaborativos [Harrison and Dourish, 1996]. Algunos ejemplos se explican a continuación:

- Los objetos con los que trabajamos e interactuamos más a menudo se encuentran generalmente más cercanos a nosotros (e.g., dispositivo móvil, computadora, documentos, llaves), mientras que otros objetos se mantienen más lejos (e.g., gabinetes, armarios, librerías).

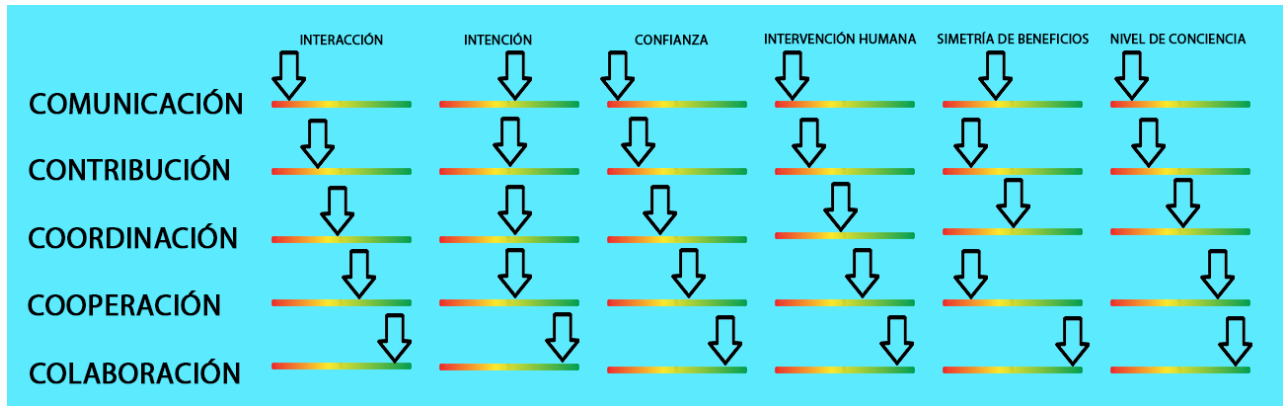


Figura 2.2: Diferentes niveles de comunicación, contribución, coordinación, cooperación y colaboración utilizando distintas variables [Ellen Taylor-Powell, 2001]

- Los espacios físicos están estructurados de acuerdo a los usos y necesidades de interacción. La puerta de una oficina se puede cerrar para dar independencia desde el espacio exterior, o dejar abierta para dejarnos ver a los transeúntes.
- Un dispositivo móvil puede dar cabida a un espacio virtual en el mundo físico, pero el espacio virtual es significativo a cualquier consideración de movilidad.

La observar las formas que tienen los objetos en relación con las acciones y las interacciones en el espacio, los diseñadores han utilizado modelos espaciales y metáforas en los sistemas colaborativos [Gaver et al., 1992]. Los sistemas mono-usuario se han extendido a una metáfora de escritorios, oficinas, pasillos y ciudades. Estos sistemas facilitan una colaboración natural, al explotar nuestro entendimiento del espacio y de las propiedades del mundo tridimensional en que vivimos e interactuamos todos los días. Según William [Gaver, 1992] llamamos “espacio” a un sentido de “lugar”, donde lugar es comprender toda la realidad.

El uso de metáforas espaciales y organización espacial se ha vuelto cada vez más popular en sistemas colaborativos en los últimos años. A continuación, se describen algunos sistemas (ver Sección 2.2.1) y luego se enuncian las propiedades que exhiben (Sección 2.2.2).

2.2.1. Sistemas basados en espacios virtuales

Algunas clasificaciones de sistemas de colaboración que se basan en espacios virtuales propuestos por Harrison y Dourish [Harrison et al., 1996] son los siguientes:

- **Realidad virtual colaborativa:** utiliza un modelo espacial de la interacción, en la que cada participante está consiente de las oportunidades que tiene para la interacción con otros participantes. Estas interacciones se gestionan a través de extensiones espaciales de la presencia de los participantes [Carlsson and Hagsand, 1993].
- **MUDs:** los MUDs (*Multi-User Dungeons*) [Curtis and Nichols, 1994] son programas multi-usuario basados en texto para entornos colaborativos. Los MUDs estructuran los espacios virtuales en espacios separados llamados “habitaciones” que permiten a los participantes

que puedan moverse de un lugar a otro, participando de forma selectiva en eventos, actividades y conversaciones. Los MUDs se usan principalmente en el desarrollo de videojuegos.

- **Comunicaciones multimedia:** estos sistemas se han basado en analogías con la organización espacial del mundo físico para estructurar los aspectos de la interacción de múltiples usuarios. Por ejemplo, el diseño del sistema de espacios multimedia [Root, 1988] utiliza una metáfora de “pasillos virtuales” como principio organizador para la interacción y la participación en un sistema de comunicación.

2.2.2. Características del espacio virtual

Hay muchos aspectos del “mundo real” (e.g., dispositivos móviles, espacios de trabajo) que pueden ser explotados como parte de un modelo espacial o virtual para la colaboración algunas características que pueden ser modeladas según Harrison y Dourish son las siguientes:

- **La orientación relacional y la reciprocidad:** es organización espacial del mundo es el mismo para todos los seres humanos. Por ejemplo: “abajo” es hacia el centro de la tierra, y “arriba” es hacia el cielo; podemos reconocer que es “delante” y “hacia atrás”, y entender lo que eso implica para nuestro campo de visión. Nuestra orientación común con el mundo físico es un recurso muy valioso para la presentación y el comportamiento de los objetos. Esto es lo que nos permite señalar a los objetos, o utilizar las descripciones espaciales para establecer una referencia [Harrison and Dourish, 1996].
- **La proximidad y la acción:** en el mundo cotidiano, actuamos (más o menos) donde estamos. Recogemos los objetos que están cerca de nosotros, no a distancia; hablamos con personas que nos rodean, porque nuestras voces sólo viajan a corta distancia; llevamos las cosas con nosotros; y nos acercamos a las cosas para ver con claridad. Propiedades similares son explotados en los espacios virtuales de colaboración. El entendimiento de la proximidad nos ayuda a relacionar a la gente con las actividades [Harrison and Dourish, 1996].

2.3. Granularidad de distribución

La granularidad es un concepto fundamental en la cognición humana. Un gránulo es un grupo de puntos (objetos) unidos por la indistinción, semejanza, proximidad o funcionalidad. El proceso de granulación es un proceso que consiste en dividir un universo en subconjuntos o la agrupación de sujetos individuales en grupos. Los gránulos se pueden visualizar como subconjuntos del universo [Hu and Xu, 2010].

La remodelación de la interfaz de usuario denota cualquier reconfiguración que es perceptible al usuario y que resulta de la aplicación de una o más transformaciones sobre toda o una parte de la interfaz de usuario [Vanderdonckt et al., 2008].

La redistribución denota la reasignación de los componentes de la interfaz de usuario de un sistema a diversos dispositivos de interacción [Coutaz and Calvary, 2012].

La granularidad de los componentes del espacio virtual de trabajo del usuario representa como la unidad mas pequeña que puede ser afectada por la remodelación o la redistribución [Vanderdonckt et al., 2008]:

- La remodelación o redistribución pueden ser totales (toda la interfaz de usuario se ve afectada) o parcial.
- Cuando es parcial, la remodelación o la redistribución pueden trabajar a nivel de espacio de trabajo, o en el nivel interactor.

A continuación se describen cuatro tipos de granularidad (total, espacio de trabajo, interactor, píxel) propuestos por Coutaz y Calvary [Coutaz and Calvary, 2012]:

- **Total:** cuando se tiene que modificar completamente la interfaz de usuario e.g., en MOY [Goh et al., 2014] la figura de la izquierda y la central se han cambiado en su totalidad (ver Figura 2.3.a).
- **Espacio de trabajo:** consiste en distribuir partes del espacio virtual de trabajo, e.g., en la Figura 2.3.b [Rekimoto, 1997] se puede observar que la paleta de colores, las imágenes y el teclado se encuentran en el dispositivo móvil, ofrece la distribución de la interfaz de usuario a nivel de espacio de trabajo.
- **Interactor:** es un caso especial del espacio de trabajo, donde la unidad de distribución es un interactor elemental e.g., en las superficies aumentadas Rekimoto [Rekimoto and Saitoh, 1999](ver Figura 2.3.c) se observa que dibujos de mesas, sillas o muebles pueden ser desplegadas en computadoras.
- **Pixel:** se refiere a cualquier componente u objeto del espacio de trabajo que puede ser dividido en múltiples superficies, e.g., en ConnecTables [Tandler et al., 2001] un objeto se puede desplegar sobre dos dispositivos móviles (ver Figura 2.3.d). Es importante hacer notar que esta granularidad solo concierne a la redistribución de la interfaz de usuario.

2.4. Contexto de uso

El “contexto” juega un papel importante en áreas como el de CSCW para identificar, aspectos que el diseñador considera útiles para el modelado y para describir el entorno que será desplegado y ejecutado [Baldauf et al., 2007]. Uno de los objetivos del contexto es que la naturaleza de los dispositivos móviles se refleja en el diseño de sistema o prototipo. Muchos autores presentan su propia definición de contexto. Schilit menciona que el servicio de contexto contiene información de direccionamiento que responde a preguntas como dónde estás, con quién estás, y qué recursos están cerca [Schilit et al., 1994]. Dey y Abowd definen este término como cualquier información que puede ser utilizada para caracterizar la situación de una entidad [Dey and Abowd, 2000]. Zimmermann menciona que los elementos para la describir la información de contexto se dividen en cinco categorías: individualidad, actividad, localización, tiempo y relaciones [Zimmermann et al., 2007].

- **Individualidad:** permite acceder a la información contextual de una entidad, la cual puede ser física o virtual.
- **Actividad:** puede ser representada por modelos de tareas (de dominio específico).

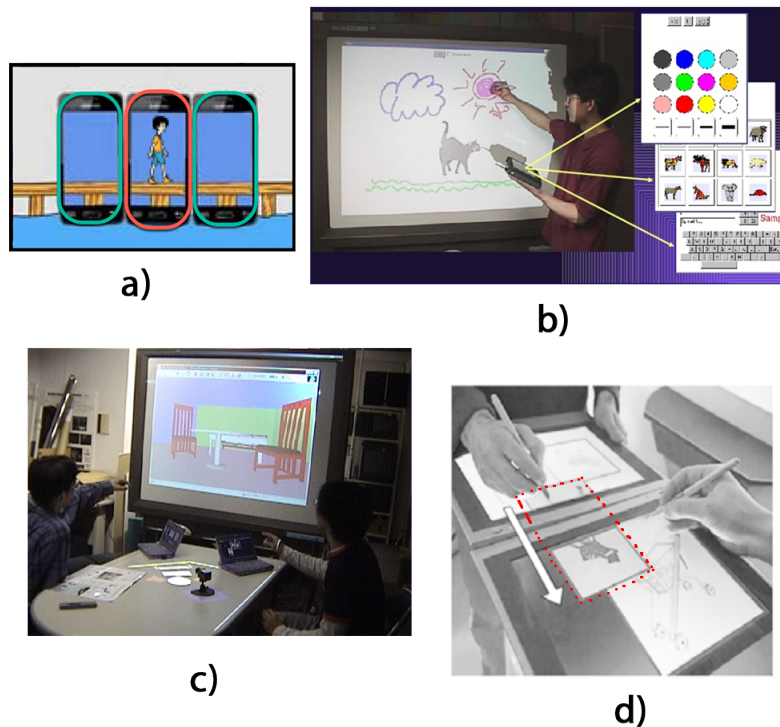


Figura 2.3: Ejemplos de granularidad

- **Localización:** describe los modelos de localización que clasifican al espacio físico o virtual como una entidad (e.g., la dirección IP como una posición dentro de una red informática o la orientación).
- **Tiempo:** es un aspecto vital para la comprensión humana y clasificación del contexto en el que la mayoría de los estados están relacionados con la dimensión temporal (e.g., la información de la zona horaria del usuario o la hora actual).
- **Relaciones:** una relación expresa una dependencia semántica entre dos entidades (i.e., pueden ser personas, cosas, dispositivos, servicios o información).

2.4.1. Dimensiones del contexto

Según Chen y Kotz [Chen and Kotz, 2000], el contexto es un espacio de cuatro dimensiones compuesta por: contexto de computación, contexto físico, el contexto del tiempo, y el contexto de usuario.

- **Contexto de computación:** se ocupa de todos los aspectos técnicos relacionados con las capacidades y recursos de cómputo. Esta categoría tiene dos objetivos. En primer objetivo expresa todas esas heterogeneidades que suelen estar presentes en entornos móviles. En segundo objetivo tiene en cuenta los diferentes recursos que posee un dispositivo móvil. Muchos de los sistemas existentes ya usan estos atributos para activar las funciones de gestión y

de adaptar los servicios. Por ejemplo, Google y Facebook pueden adaptarse dinámicamente a las características actuales de los dispositivos móviles, los clientes Web y conectividad (e.g., el ancho de banda disponible).

- **Contexto físico:** contempla todos aquellos aspectos que representan el mundo real y que se pueden acceder, mediante el uso de sensores (e.g., acelerómetro, giroscopio, magnetómetro y proximidad) y los recursos que provee de una red (e.g., local o móvil). La ubicación del dispositivo es un ejemplo básico del contexto físico; otros aspectos incluyen condiciones de tráfico, la velocidad de la gente, el nivel de ruido, la temperatura, y los datos de iluminación. Los modelos físicos y leyes, por ejemplo, las de la Mecánica para ayudar a predecir los estados futuros físicos del sistema, también son parte del contexto físico. Debido a su naturaleza, el contexto físico es intrínsecamente propenso a errores de medición (i.e., debido a varias fuentes de inexactitud, la imprecisión y la naturaleza estocástica de los procesos físicos).
- **Contexto de tiempo:** capta la dimensión de tiempo, como la hora en día, semana, mes y temporada del año, de cualquier actividad realizada en el sistema (i.e., sea en el mundo real o virtual). Estos elementos del contexto pueden ser principalmente de dos tipos: esporádicos y periódicos. Los esporádicos (inesperados) son ocurrencias que se desencadenan en ocasiones, incluso una sola vez. Los eventos periódicos describen eventos esperados que se presentan de manera repetida y predecible. Además, estos dos tipos también se pueden combinar para construir complejos acontecimientos de contexto basados tanto en una secuencia de eventos, como en el número de eventos en un segmento de tiempo particular. Por ejemplo, una actividad esporádica puede reducir automáticamente la calidad de video durante la congestión de la red, mientras que una actividad periódica puede cambiar automáticamente el tono de timbre del teléfono celular siempre a la misma hora del día. Además, una actividad compleja, basada en el seguimiento de los sucesos de congestión de la red, puede decidir cambiar entre diferentes interfaces de red en aras de lograr una mejor calidad de transmisión.
- **Contexto de usuario:** contiene aspectos de alto nivel relacionados con la dimensión social de los usuarios, como el perfil de los usuarios, las personas cercanas, y la situación social actual. De hecho, se consideran los sistemas móviles distribuidos que son el resultado de la agregación de múltiples dispositivos. Por lo tanto, cada contexto de usuario tiene tanto como una dimensión individual (como el perfil de usuario y preferencias) y una dimensión social, que son los aspectos que interactúan con la dimensión (como otras personas a su alrededor, la proximidad y las características de las personas).

El desarrollo de una amplia variedad de dispositivos de acceso ha creado nuevos requerimientos en el campo de investigación de la Interacción Hombre-Máquina, tales como la capacidad de los sistemas interactivos para ejecutarse en diferentes contextos de uso. Al tratarse de un desarrollo móvil, el número de contextos de uso en los cuales puede estar inmerso un sistema se incrementa.

En los sistemas móviles, como en otros ámbitos de la Interacción Hombre-Máquina, no es suficiente centrarse en la interfaz específica de un dispositivo. El dispositivo opera dentro de un contexto más amplio. Este contexto incluye la red, la infraestructura computacional, el dominio de aplicación y el entorno físico.

Cada uno de los diferentes contextos representa una parte diferente del espacio de diseño en el que los sistemas móviles deben ser abordados, como las características de la infraestructura (e.g., dispositivo móvil y red WiFi), el sistema operativo (e.g., Android o iOS), el dominio de un ambiente móvil.

El soporte propuesto en esta tesis se centra en la naturaleza contextual de dispositivos móviles para en el diseño y desarrollo de aplicaciones colaborativas móviles, delimitado por las características de hardware y software del dispositivo móvil en el cual se desarrolle el soporte propuesto. El diseño de un soporte para aplicaciones colaborativas móviles permitirá a los desarrolladores tener en cuenta las propiedades de los sistemas en construcción y la forma en que pueden estar relacionados con otras aplicaciones de carácter colaborativo y sistemas operativos con el fin de establecer una relación colaborativa.

2.4.2. Localización y espacio en el contexto

Claramente, la idea misma de localización exige una comprensión del lugar y uno de los aspectos únicos de los dispositivos móviles es que puedan tener un conocimiento de la ubicación dentro de la cual se están utilizando. Además, esta información de ubicación puede ser explotada como un medio para comprender el contexto general en el que se coloca el sistema. En esencia, la ubicación espacial en el que se encuentra un dispositivo móvil es útil para inferir el contexto general que influye en el diseño de una aplicación móvil. Con el fin de descubrir qué dispositivos están cerca (contexto del sistema), es necesario conocer la ubicación de cada uno de ellos en el espacio (contexto físico). Por otra parte, la necesidad de ser consiente del contexto de otras maneras (e.g., ubicación, efectos, eventos) depende en gran medida de la movilidad en el espacio de los dispositivos. Si los dispositivos están espacialmente estáticos, muchos aspectos de su medio ambiente también son estáticos o, como máximo, de variación lenta.

Cualquier noción de ubicación pone un dispositivo dentro de algún tipo de espacio, en el cual también puede contener otros dispositivos y usuarios con los que dicho dispositivo puede interactuar. Un dispositivo involucrado puede ser considerado como una entidad que:

- tiene ubicación en el espacio
- tiene un efecto sobre el espacio
- ésta sujeta a eventos que influyen desde el espacio

En esencia, los dispositivos que están situados en un espacio y su interacción es mediada a través de este espacio. En consecuencia, la comprensión de la naturaleza de su ubicación en ese espacio es clave para entender el diseño del sistema móvil que proporcione un medio de reflexión sobre el contexto.

Si los dispositivos móviles estuvieran en ubicaciones fijas, la naturaleza de estas interacciones sería solo una configuración de las características del lugar. Sin embargo, la naturaleza interesante y desafiante de las interfaces móviles es la naturaleza cambiante de estas relaciones. Así que, para tener un modelo global de la interacción en el espacio situado, tenemos que entender:

- ubicación en el espacio (del dispositivo y sus características de localización)
- movilidad a través del espacio (dispositivos móviles)

- tipos de dispositivos que interactúan en el espacio
- conocimiento del dispositivo y sus características

2.5. Arquitecturas de sistemas

Una arquitectura de software se refiere a la estructuración del sistema la cual representa al diseño de alto nivel donde se identifican los principales componentes que sirven como guía en el desarrollo [Bass et al., 2012]. A continuación se mencionan los tipos de arquitectura:

1. **Arquitectura centralizada:** según Tanenbaum y Steen “la comunicación entre un cliente y un servidor puede implementarse mediante un protocolo simple no orientado a conexión cuando la red subyacente es muy confiable, como sucede en muchas redes de área local” [Tanenbaum and Steen, 2008]. Esta arquitectura está organizada de tal forma que el servidor sea el “eje central” del sistema, que recibe la información del cliente, la analiza y envía una respuesta, según la configuración, o la información que recibe por parte del cliente. Pero como se menciono, esta arquitectura está basada en un protocolo no orientado a conexión. Entre sus principales características, este protocolo es susceptible a que los mensajes no lleguen al destinatario o simplemente sean corrompidos. En la Figura 2.4 se puede ver la interacción de un cliente y un servidor.

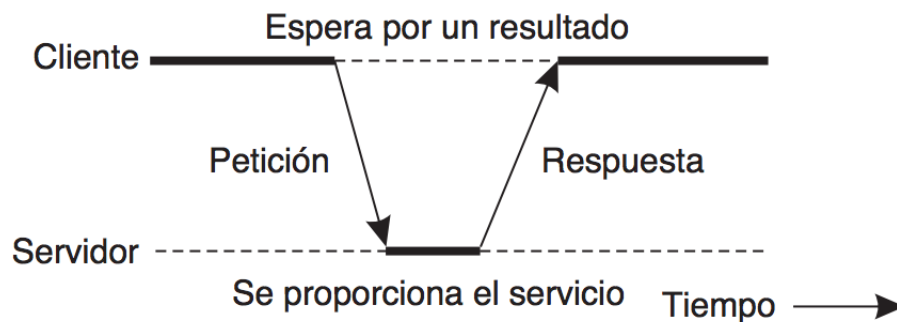


Figura 2.4: Interacción centralizada

2. **Arquitectura descentralizada:** Este tipo de arquitectura coexisten diferentes servidores que dependen de uno central. La caída de los servidores supone la desconexión uno o más servidores de la red, pero no afecta a toda la arquitectura [Baran, 1962] (ver Figura 2.5).
3. **Arquitectura distribuida:** en este tipo de arquitectura si algún par cliente o servidor se desconecta no afectaría a la red o algún otro par cliente servidor. Todos los pares clientes servidores se conectan entre sí sin que tengan que pasar necesariamente por uno o varios servidores locales (ver Figura 2.5).
4. **Arquitectura per to peer:** es a una red que no tiene clientes ni servidores fijos, sino una serie de nodos que se comportan simultáneamente como clientes y como servidores respecto de los demás nodos de la red (ver Figura 2.5).

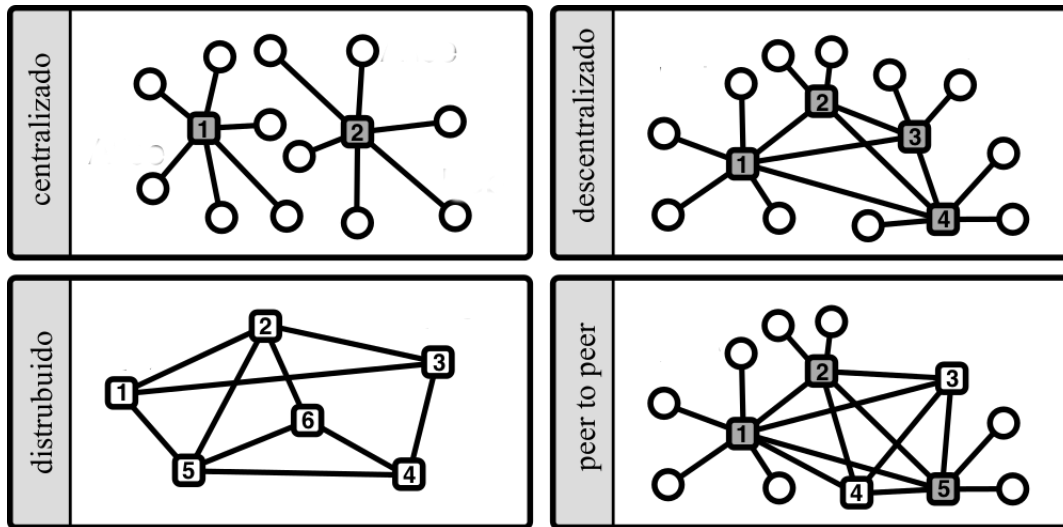


Figura 2.5: Tipos de arquitecturas [Baran, 1964]

2.6. Trabajos relacionados

A continuación, se describen aplicaciones y prototipos relacionados con el presente trabajo de investigación, los cuales hacen uso del acoplamiento de dispositivos.

2.6.1. Stitching

Stitching [Hinckley et al., 2014] es una técnica de interacción que permite a los usuarios acoplar dispositivos móviles (i.e., tabletas con sistema operativo Windows mobile) con acceso a redes WiFi, usando principalmente gestos con una pluma (*stylus pens*). Un usuario inicia moviendo la pluma en un dispositivo y cruza el borde de esta hasta iniciar con el borde del segundo dispositivo (ver Figura 2.6). Estos dispositivos se sincronizan mediante una comunicación WiFi. Esta técnica permite a los usuarios copiar imágenes de un dispositivo a otro, expandiendo el área de trabajo a múltiples pantallas. Así mismo, debido a su composición, solamente es posible unir pares de dispositivos móviles.

2.6.2. Mobicast

Mobicast [Kaheel et al., 2009] es un sistema que admite el acoplamiento entre dispositivos (i.e., teléfonos inteligentes y tabletas con sistema operativo Android). Este sistema permite a los usuarios establecer un evento (*video streaming*) con diferentes teléfonos inteligentes para proporcionar una mejor visualización combinada del evento mediante una red local. Además, si dos usuarios (A y B) están grabando un evento en el mismo ángulo de visión, el sistema puede dejar de grabar de forma automática para uno de estos dos usuarios. En Figura 2.7 se muestra la captura de un paisaje con tres teléfonos inteligentes en los cuales se está usando el sistema Mobicast, en a) se captura la parte izquierda de un paisaje con el primer dispositivo, en b) se captura la parte central del paisaje con un segundo dispositivo, en c) se captura la parte derecha del paisaje con un tercer dispositivo y en d) se muestra el área que visualizan al combinar las tres capturas en

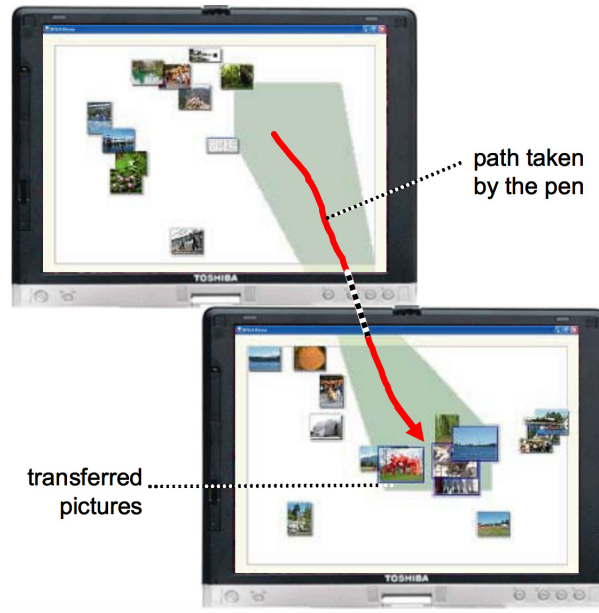


Figura 2.6: Ejemplo de interacción usando Stitching [Hinckley et al., 2014]

cada uno de los dispositivos.

2.6.3. MobiSurf

MobiSurf [Seifert et al., 2012] es un sistema capaz de extender el espacio de trabajo de un dispositivo móvil a pantallas externas, aprovechando así el espacio extra proporcionado por la pantalla. Para acoplar el dispositivo móvil con la pantalla, basta con acercarlo a uno de los bordes de la pantalla. La conexión se realiza a través de tags NFC y códigos grabados en éstos, los cuales están situados en las orillas de la pantalla. Para su funcionamiento, requiere de un servidor encargado de manipular el contenido de la pantalla a la cual el dispositivo está conectado. Además este sistema está implementado en el sistema operativo Android y como máximo admite dos dispositivos móviles por sesión. En Figura 2.8 se muestra cómo el contenido de un dispositivo móvil que usa el sistema MobiSurf puede trasladar los objetos a una pantalla externa, previamente el dispositivo móvil debe estar acoplado.

2.6.4. ConnecTable

ConnecTable [Tandler et al., 2001] es un dispositivo consciente de contexto, el cual brinda soporte para la transición del trabajo individual al colaborativo, i.e., permite expandir el espacio de trabajo, con el propósito de compartir los objetos entre tabletas idénticas, superando de esa forma las restricciones físicas que conlleva el uso de una sola pantalla para interacciones entre dos usuarios. ConnecTable hace uso de sensores dedicados (ya instalados) para el rastreo y acoplamiento con otras ConnecTables, por lo cual requiere de hardware especial. Así mismo, debido a su composición, solamente es posible unir ConnecTables en arreglos de dos elementos. Adicionalmente, no cuenta con soporte para el cambio de posiciones físicas de las mismas, ya que sólo permite el acoplamiento de los dispositivos tocando sus respectivas partes superiores.

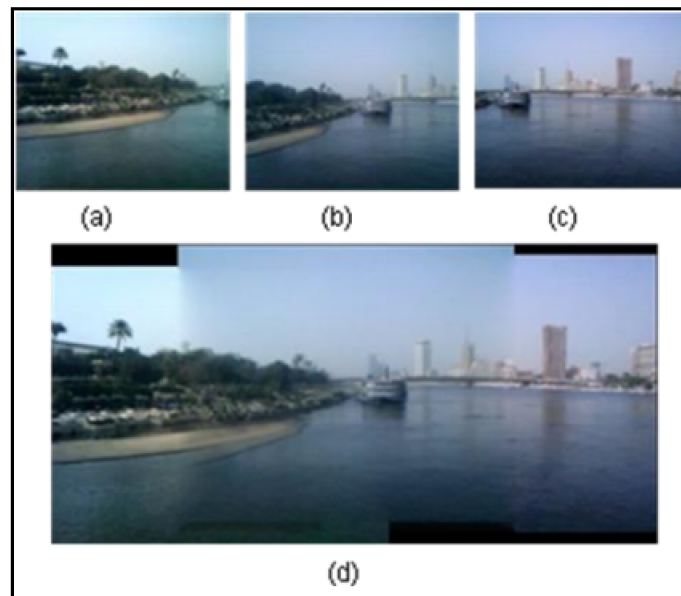


Figura 2.7: Ejemplo de interacción usando Mobicast con tres diferentes dispositivos [Kaheel et al., 2009]



Figura 2.8: Vista general de un prototipo utilizando MobiSurf [Seifert et al., 2012]

En la Figura 2.9 en a) se muestra el trabajo individual que se realiza en una tableta, en b) se muestra el acoplamiento entre dos tabletas que se colocan uno al lado del otro, en c) se muestra el intercambio de objetos entre las dos tabletas acopladas y en d) se muestra que los usuarios pueden tener sus propias vistas compartidas del mismo objeto.



Figura 2.9: Ejemplo de algunas configuraciones posibles con ConnectTables [Tandler et al., 2001]

2.6.5. Interactive Phone Call

Interactive Phone Call [Winkler et al., 2011] es un sistema capaz de permitir a los usuarios navegar, compartir y copiar datos personales y colaborar en tiempo real, durante la realización de una llamada telefónica. Este sistema es capaz de extender el área de trabajo del teléfono inteligente a una tableta con Windows Mobile o a una computadora personal con Windows. La conexión se realiza a través de servicios de terceros como Skype, Windows Meeting Space, Windows Communicator, Adobe Connect y Cisco. En la Figura 2.10 se muestra el área del teléfono inteligente en una tableta.

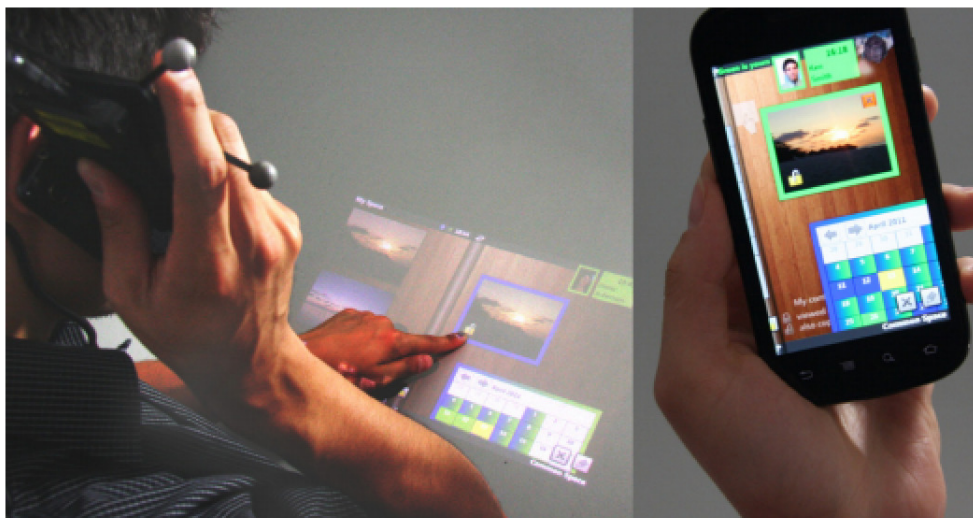


Figura 2.10: Ejemplo de interacción usando Interactive Phone Call para compartir recursos [Winkler et al., 2011]

2.6.6. Mindmap

Mindmap [Lucero et al., 2010] es una de las posibles aplicaciones que se pueden implementar mediante la plataforma SSI (*Social and Spatial Interactions*) propuesta por Nokia. En sí, es una herramienta para dispositivos móviles que da soporte a sesiones de lluvias de ideas, permitiendo a un grupo de trabajo crear, editar y visualizar notas virtuales. La idea original consiste en permitir a un número pequeño de personas realizar las acciones mencionadas, de manera colaborativa, sobre cualquier superficie plana, teniendo cada usuario su propio dispositivo y la posibilidad de poder utilizarlos y compartirlos de forma indistinta. Mindmap da soporte al despliegue del espacio de trabajo en múltiples dispositivos (arreglos máximo de 8 dispositivos) a través de un gesto (basado en mediciones del acelerómetro a una frecuencia de 10 Hz) en las pantallas que se desean unir y usa Linux Maemo 5v como sistema operativo. En la Figura 2.11 se muestran dos dispositivos acoplados en los cuales se modifican las notas que se visualizan en un tercer dispositivo.

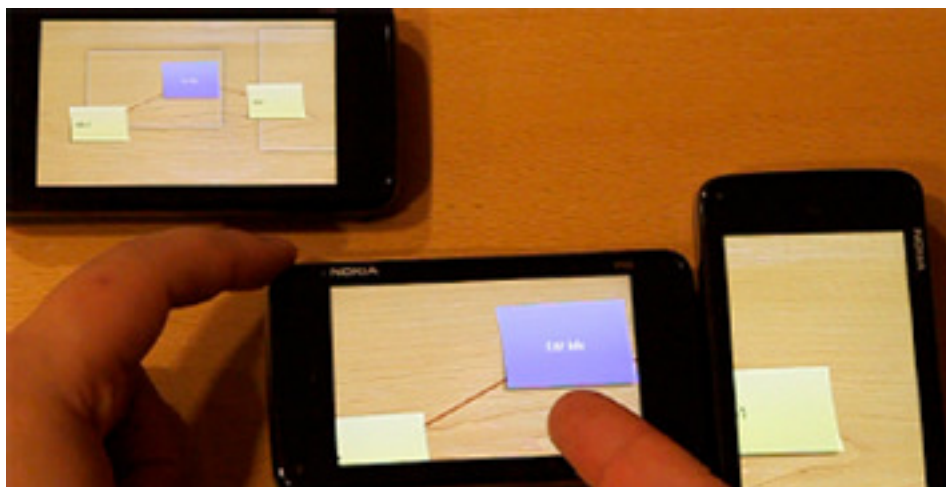


Figura 2.11: Vista general de Mindmap [Lucero et al., 2010]

2.6.7. MOY

MOY [Goh et al., 2014] es un soporte propuesto para el desarrollo multi-usuario de actividades relacionadas con juegos para niños, en entornos que combinan espacios privados y compartidos. Este soporte puede usarse en múltiples dispositivos móviles con un máximo de cuatro, en combinación con una mesa de juego interactiva. Los dispositivos incorporan un TAG visual que es reconocido por la superficie de la mesa de juego. Cada usuario ubica la posición y la orientación de los dispositivos. El soporte está desarrollado e implementado en el sistema operativo Android. Los dispositivos pueden comunicarse entre ellos y pueden compartir información e interactuar con los objetos que se encuentran en la superficie de la mesa (ver Figura 2.12).



Figura 2.12: Niños colaborando con sus dispositivos móviles para resolver un problema en la superficie de la mesa de juego [Goh et al., 2014]

2.7. Análisis comparativo

Como se pudo observar en la sección previa, existe una variedad de propuestas que se asemejan al tipo de prototipos o aplicaciones que pretendemos soportar. Partiendo de estos prototipos propuestos, es importante identificar las limitaciones que presentan para que el soporte desarrollado sea mucho más robusto y cubra los puntos deficientes que presentan. Como se puede apreciar en las tablas 2.1 y 2.2, mediante el marco de desarrollo se pretende cubrir las limitaciones de los prototipos propuestos, así como facilitar una forma flexible la transición del trabajo individual al trabajo colaborativo.

	Heterogeneidad de dispositivos	Tamaño del arreglo	Tipo de arreglo	Hardware externo
Stitching	No	No definido	Regular	Si
Mobicast	No	1 - 3	Regular	No
MobiSurf	No	1 - 2	Regular	No
ConnecTables	No	1 - 2	Regular	Si
Mindmap	No	1 - 8	Regular	No
Interactive phone call	No	1 - 2	Regular	No
MOY	No	1 - 4	Regular	No

Tabla 2.1: Análisis comparativo (parte 1)

	Soporte de desarrollo	Configuración de acoplamiento	Tipo de dispositivo
Stitching	No	No	Tabletas
Mobicast	No	-	Telefonos inteligentes
MobiSurf	No	-	Telefonos inteligentes / Tabletas
ConnecTables	No	No	Tabletas
Mindmap	No	-	Telefonos inteligentes
Interactive phone call	No	-	Telefonos inteligentes
MOY	Si	-	Telefonos inteligentes

Tabla 2.2: Análisis comparativo (parte 2)

Todos los prototipos analizados soportan únicamente arreglos regulares de dispositivos homogéneos e incluso requieren de hardware adicional, lo cual implica que sea más complicado que un usuario no experimentado pueda hacer uso de los prototipos. Adicionalmente, algunos soportan arreglos de tamaños muy limitados o simplemente no se especifica. Un punto muy importante es que ninguno de ellos proporciona un soporte de desarrollo reutilizable.

Capítulo 3

Análisis y diseño del marco de desarrollo

En el presente capítulo se describe la solución obtenida que provee a los desarrolladores las herramientas necesarias para habilitar transiciones entre espacios de trabajo individuales y espacios de colaboración, independientemente del modelo de tareas de la aplicación final. En la Sección 3.1 se describe el análisis del marco de desarrollo; así como algunos conceptos para la comprensión de la solución propuesta. En la Sección 3.2 se describe la arquitectura, así como los servicios que la componen. Finalmente, en las Secciones 3.3 a 3.6 se describe el diseño del marco de desarrollo propuesto, los casos de uso, los diagramas de despliegue, los diagramas de clases y los diagramas de secuencia.

3.1. Requerimientos

En la Sección 2.2 del Capítulo 2 se definieron los espacios colaborativos y sus características; así como los tipos de espacios virtuales propuestos en la literatura científica. Con base en dichas definiciones, en la presente tesis definimos un **espacio colaborativo virtual** como una representación del modo de trabajo colaborativo físico, i.e., el trabajo colaborativo realizado por medio de un modelo de tareas simplificadas y simuladas en una aplicación para dispositivos móviles. En esta tesis proponemos que un espacio colaborativo puede estar compuesta de sesiones individuales y sesiones colaborativas. A continuación describimos estos tipos de sesiones:

- **Sesión individual:** es el intercambio de información entre un usuario y un dispositivo móvil. Además, esta sesión es creada en el dispositivo móvil. Por otro lado, una sesión individual implica el uso exclusivo de una aplicación en el dispositivo sin tener que compartir los objetos virtuales existentes.
- **Sesión colaborativa:** es el intercambio de información entre dos o más usuarios mediante sus respectivos dispositivos móviles, además de tener un objetivo en común para los participantes de la sesión.

En el presente trabajo las sesiones colaborativas se llevan a cabo mediante la interacción cara a cara de los usuarios. Así mismo, estas sesiones son creadas y ejecutadas en arreglos de dispositivos móviles.

Uno de los aspectos más importantes de la presente tesis es el mapeo de un espacio colaborativo a un arreglo de dispositivos móviles. Sin embargo, este trabajo no se limita a arreglos homogéneos i.e., que un arreglo puede contener dispositivos con sistemas operativos como Android e iOS.

Para tener una definición general del diseño del marco de desarrollo, se retoman los aspectos de diseño que propuso [Castro, 2014], además agregaremos tres aspectos de diseño que son disponibilidad, conjunto de plataformas y entorno común.

- **Conectividad:** se refiere a la disponibilidad de iniciar y/o terminar una sesión colaborativa, además de poder establecer una comunicación con cada uno de los integrantes. Esta comunicación es realizada por el envío y la recepción de mensajes tanto por los emisores como los receptores.
- **Geometría:** es la determinación de posición relativa de uno o varios dispositivos en aplicaciones conscientes de la ubicación, sin la necesidad de hardware especial.
- **Acoplamiento:** un acoplamiento es una acción realizada por un usuario mediante gestos sobre la pantalla del dispositivo i.e., la creación una sesión colaborativa. Además la activación de los intentos de acoplamiento debe ser fácil e intuitiva para los usuarios de una aplicación [Castro, 2014]. Sin embargo estos intentos de acoplamiento no deben interferir con la tarea que se esté ejecutando.
- **Consciencia de grupo:** cuando una aplicación está en una sesión colaborativa, debe ser posible identificar las entidades que son partícipes de la sesión.
- **Disponibilidad:** se refiere a identificar e informar por medio de mensajes a los participantes de una sesión colaborativa sobre la presencia y estado de un dispositivo.
- **Conjunto de plataformas:** tiene como finalidad tener sesión homogénea entre dos o más plataformas de sistemas operativos móviles (e.g., Android y iOS).
- **Entorno común:** se refiere a los objetos que se tienen en un espacio de trabajo colaborativo, los cuales pueden enviarse a los demás participantes cuando exista una actualización de los objetos.

Estos aspectos describen un espacio de trabajo cuya finalidad es mejorar la interacción entre los miembros que están en una sesión colaborativa.

3.2. Arquitectura

En esta sección se describe el diseño de la arquitectura del marco propuesto. Dicha arquitectura esta dirigida particularmente a sistemas colaborativos que se ejecuten en arreglos de dispositivos móviles. Por lo tanto, se toman en cuenta algunos aspectos como el reconocimiento e identificación de dichos dispositivos, las políticas de acoplamiento, la comunicación con cada uno de los colaboradores que estén usando la aplicación, los recursos disponibles y otras variables típicas (e.g., objetos virtuales) en los grupos de trabajo. Primeramente se realiza una descripción de la arquitectura (ver Sección 3.2.1), posteriormente se explica detalladamente cada una de los servicios que integran la arquitectura.

3.2.1. Descripción general

La arquitectura propuesta peer to peer (P2P por su siglas en inglés) en capas para el soporte de desarrollo propuesto consta de cuatro servicios; descubrimiento, comunicación, actualización y de presencia (ver Figura 3.1). A continuación se describe la arquitectura:

- **Aplicación:** presenta el sistema creado por el marco de desarrollo propuesto al usuario, le comunica la información que se produce por los servicios que lo integran.
- **API:** contiene los métodos que ofrece el marco de desarrollo para que los desarrolladores puedan configurar algunos aspectos como:
 1. El tiempo de publicación de identificación (servicio de descubrimiento).
 2. La activación del gesto de arrastre o gesto de inclinación (servicio de comunicación).
 3. El envío de objetos del espacio de trabajo (servicio de actualización).
- **Servicios:** contiene los servicios como; descubrimiento, comunicación, actualización y presencia.
- **Red local:** es utilizada para comunicar los servicios (e.g., descubrimiento, comunicación, actualización y presencia).

Los servicios mencionados en la arquitectura se describen a continuación:

- **Servicio de descubrimiento:** es encargado de publicar y recolectar la identificación del dispositivo móvil (e.g., IP, puertos TCP y UDP, sistema operativo) (ver Sección 3.2.2).
- **Servicio de comunicación:** encargado de los eventos de acoplamiento y actualización de las listas de usuarios en una sesión colaborativa (ver Sección 3.2.3).
- **Servicio de actualización:** encargado de modificar los objetos de una sesión colaborativa (ver Sección 3.2.4).
- **Servicio de presencia:** encargado de verificar la presencia de los dispositivos que se encuentren en una sesión colaborativa (ver Sección 3.2.5).

Los servicios mencionados (servicio de descubrimiento, servicio de comunicación, servicio de actualización, servicio de presencia) contienen otros servicios (ver Sección 3.5). A continuación se describen estos servicios:

- **Servicio de descubrimiento:** este servicio contiene dos servicios que son (ver Sección 3.5):
 - **Publicación de identificación:** este servicio publica los datos de identificación a través de la red local (e.g., IP, puertos TCP y UDP, sistema operativo).
 - **Recolección de identificación:** este servicio recolecta los datos de identificación producidos por el servicio de publicación de identificación.
- **Servicio de comunicación:** este servicio contiene dos servicios que son (ver Sección 3.5):

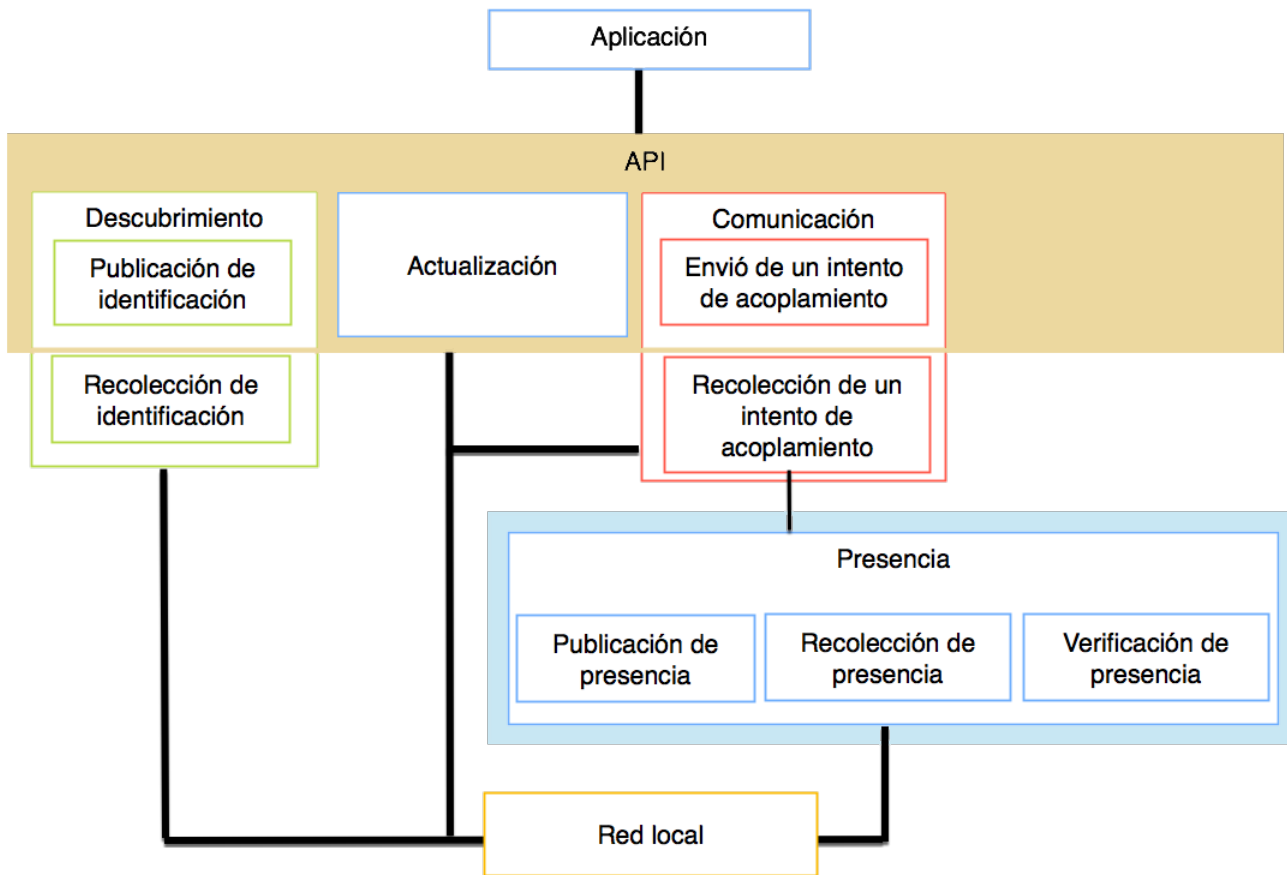


Figura 3.1: Arquitectura del marco de desarrollo propuesto

- **Envió de un intento de acoplamiento:** este servicio envía un intento de acoplamiento que realice el usuario a través de la red local.
- **Recolección de un intento de acoplamiento:** este servicio recolecta el intento de acoplamiento producido por el servicio de envió de un intento de acoplamiento.
- **Servicio de actualización:** este servicio solo contiene un servicio que es el encargado de modificar y actualizar los objetos de una sesión colaborativa (ver Sección 3.5).
- **Servicio de presencia:** este servicio tiene tres servicios que son (ver Sección 3.5):
 - **Publicación de presencia:** este servicio es el encargado de publicar el estado de presencia de un dispositivo que se encuentra en una sesión colaborativa.
 - **Recolección de presencia:** este servicio se encarga de recolectar el estado de presencia producido por el servicio de publicación de presencia.
 - **Verificación de presencia:** este servicio se encarga de de verificar el estado de presencia (i.e., determina si el tiempo de recepción del estado de presencia es mayor al tiempo del dispositivo) recolectado por el servicio de recolección de presencia.

A continuación se describe el flujo de ejecución de los servicios mencionados (descubrimiento, comunicación, presencia y actualización) (ver Figura 3.2):

- Pasos que sigue el flujo de la comunicación del servicio de descubrimiento:
 1. El servicio principal WorkTogether ejecuta el servicio de descubrimiento (número 1).
 2. El servicio de descubrimiento ejecuta al servicio de publicación de identificación (número 2).
 3. El servicio de publicación de identificación (par 1 izquierdo de la imagen) envía los datos de identificación (e.g., IP, puertos, disponibilidad, nombre y tipo de sistema operativo) al servicio de servicio de recolección de información (par 2 derecho de la imagen) (número 3).
 4. El servicio de recolección de información (par 2 derecho de la imagen) recibe los datos de identificación, actualiza la lista de descubrimiento y verifica la disponibilidad del dispositivo (número 4).
 5. El servicio de descubrimiento (par 2 derecho de la imagen) envía una notificación al servicio principal WorkTogether para informarle que un dispositivo se conecto a la red o esta próximo a desconectarse de la red (número 5).
- Pasos que sigue el servicio de comunicación:
 1. El servicio principal WorkTogether ejecuta al servicio de comunicación cuando el usuario realiza un intento de acoplamiento (número 6).
 2. El servicio de comunicación ejecuta al servicio de envió de un intento de acoplamiento (número 7).
 3. El servicio de envió de un intento de acoplamiento (par 1 izquierdo de la imagen) envía los datos para acoplarse con otro dispositivo (e.g., lado de acoplamiento, las dimensiones de la pantalla y interfaces de comunicación) al servicio de recolección de un intento de acoplamiento (par 2 derecho de la imagen) (número 8).
 4. El servicio de recolección de un intento de acoplamiento (par 2 derecho de la imagen) recibe los datos, los procesa y determina si el intento es valido o no para posteriormente crear una sesión colaborativa (número 9).
 5. El servicio de comunicación informa al servicio principal WorkTogether de un intento de acoplamiento (número 10).
- El servicio principal WorkTogether ejecuta al servicio de presencia cuando se crea una sesión colaborativa, los pasos del flujo de la comunicación son los siguientes:
 1. El servicio principal ejecuta al servicio de presencia (número 11).
 2. El servicio de presencia ejecuta al servicio de publicación de presencia (número 12).
 3. El servicio de publicación de presencia (par 1 izquierdo de la imagen) envía los datos (e.g., IP, fecha y los datos de los objetos que están en una parte del espacio de trabajo común) al servicio de recolección de presencia (par 2 derecho de la imagen) (número 13).

4. El servicio de recolección de presencia (par 2 derecho de la imagen) guarda los datos recibidos en la lista de presencia (número 14).
 5. El servicio de verificación de presencia (par 1 izquierdo y par 2 derecho) se encarga de revisar la lista de presencia y determinar si el tiempo de recepción es mayor a la del dispositivo y activar el mecanismo de recuperación de objetos (número 15).
 6. El servicio de presencia informa al servicio principal WorkTogether si se activo el mecanismo de recuperación de los objetos (número 16).
- El flujo de la comunicación del servicio de actualización es el siguiente:
 1. El servicio principal WorkTogether ejecuta al servicio de actualización (número 17).
 2. El servicio de actualización (par 1 izquierdo de la imagen) envía los datos de los objetos (e.g., color del objeto, texto y posición en los ejes X y Y) que se actualicen en la parte común del espacio de trabajo al servicio de actualización (par 2 derecho de la imagen) (número 18).
 3. El servicio de actualización (par derecho de la imagen) envía los datos de los objetos al servicio principal WorkTogether para que actualice los objetos que están en la parte del espacio de trabajo común (número 19).

Estos servicios están pensados y diseñados para que se ejecuten en conjunto con una arquitectura bajo una arquitectura, i.e., operan de forma independiente entre un dispositivo y otro, por lo tanto, aplicaciones que implementen el marco de desarrollo y se encuentren ejecutándose en diferentes dispositivos pueden hacer uso de estos servicios cuando sean requeridos, sin depender de alguna entidad externa. En este contexto, cada dispositivo se puede considerar como una unidad funcional integrada por dichos servicios, la cual es independiente de otras unidades funcionales, i.e., una aplicación estará operando sin restricciones en una sesión individual, a pesar de que no se utilicen los tres servicios disponibles que forman parte del marco de desarrollo (servicio de comunicación, servicio de actualización, servicio de presencia). Cada unidad funcional entonces alcanza su máximo nivel de efectividad cuando es utilizada en una sesión colaborativa, i.e., cuando se utiliza en conjunto con otras unidades funcionales al explotar los servicios proporcionados.

3.2.2. Servicio de descubrimiento

Este servicio es encargado de recibir y enviar información de identificación de otros dispositivos que estén conectados en la misma red local, así como difundir la información propia. La información que es enviada contiene un identificador generado por el soporte, puertos de servicio de comunicación, así como información básica del dispositivo (e.g., características del hardware).

La publicación de identificación se hace cada determinado tiempo dependiendo de la configuración que el desarrollador considere. La publicación de los mensajes de descubrimiento se realiza por medio del protocolo TCP a través de una difusión *broadcast*.

El formato de los mensajes de identificación para Android e iOS es el siguiente:

- **IP:** contiene la IP del dispositivo.

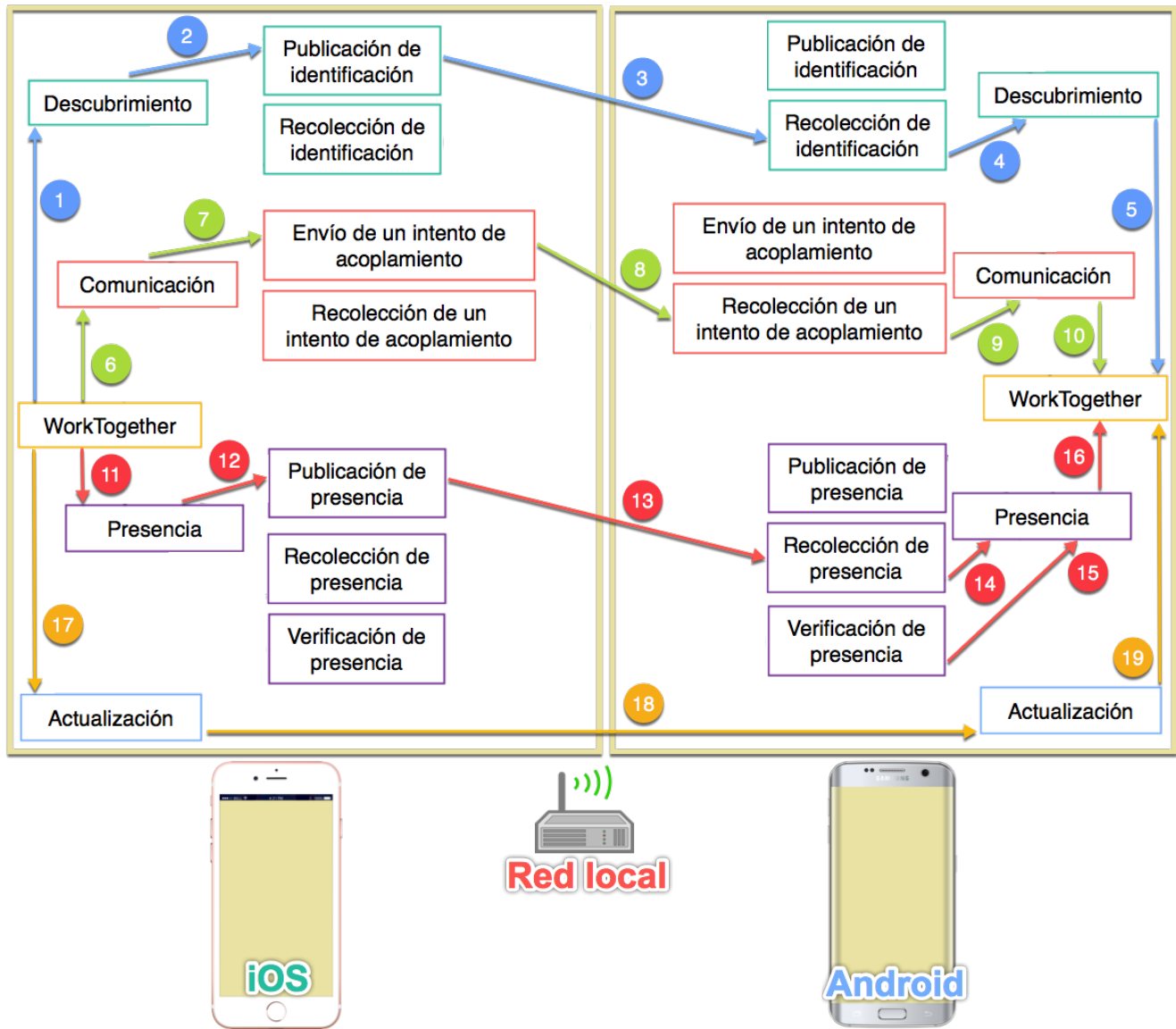


Figura 3.2: Flujo de la comunicación entre los servicios del marco de desarrollo

- **UP:** contiene un valor verdadero si el soporte se esta ejecutando.
- **TCP:** contiene el puerto TCP generado por el soporte.
- **UDP:** contiene el puerto UDP generado por el soporte.
- **DEVICE:** contiene el nombre del dispositivo.
- **TYPEDEVICE:** contiene el tipo de sistema operativo (Android o iOS).

3.2.3. Servicio de comunicación

Este servicio es encargado de realizar dos eventos, uno de estos tiene el propósito de iniciar una sesión colaborativa con otro dispositivo, al acoplar dos espacios de trabajo (e.g., individual o colaborativo) en uno solo a través de gestos de acoplamiento. Posteriormente, después de haber realizado un gesto de acoplamiento válido, la sesión colaborativa se crea uniendo los espacios de trabajo implicados. Este servicio también se encarga de recibir información referente a la sesión creada (e.g., id de la sesión, nombre de la sesión, puertos de conexión). El segundo de los eventos que realiza este servicio es el de actualizar las listas de los dispositivos en las que se ejecuta la sesión colaborativa (ver Sección 3.2). El intento de acoplamiento se envía a través del protocolo TCP por medio de una difusión *broadcast* en la red local.

Los gestos de acoplamiento son medios de selección que pueden ser entendidos y traducidos de forma correcta por los dispositivos. Además los gestos de acoplamiento hacen uso del hardware del dispositivo móvil como: *touch* y acelerómetro, los cuales se pueden usar de manera individual o combinada, ya que la finalidad es reconocer el lado de la pantalla por donde se quiere unir un espacio de trabajo con otro (ver Figura 3.3). A continuación mencionamos dos tipos de gestos implementados en este marco de desarrollo:

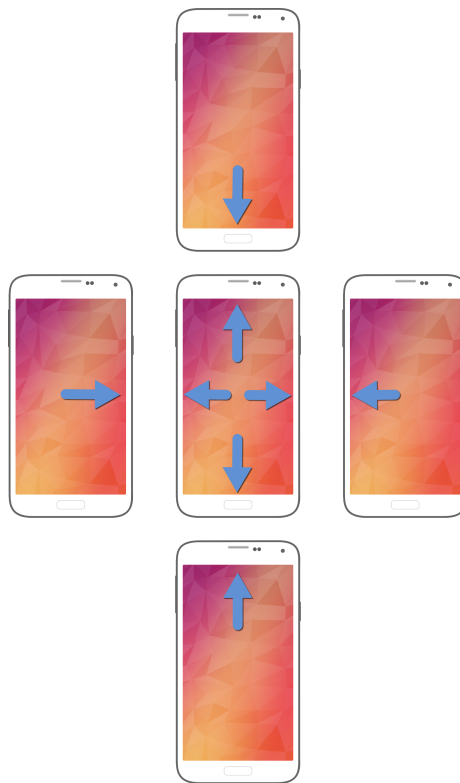


Figura 3.3: Direcciones de acoplamiento entre dispositivos

1. **Gesto basado en el *touch* del dispositivo:** este gesto consiste en arrastrar un dedo sobre la superficie de la pantalla. Este gesto requiere que las direcciones sean opuestas, de tal manera que se indique de qué lado de cada dispositivo se quiere unir los espacios de trabajo (ver Figura 3.4)

2. **Gesto basado en el acelerómetro:** en este gesto, se debe inclinar el dispositivo en un ángulo mínimo y máximo de grados durante un tiempo determinado, de tal forma que los lados por donde se desea unir los espacios de trabajo permanezcan sobre la superficie (ver Figura 3.5).

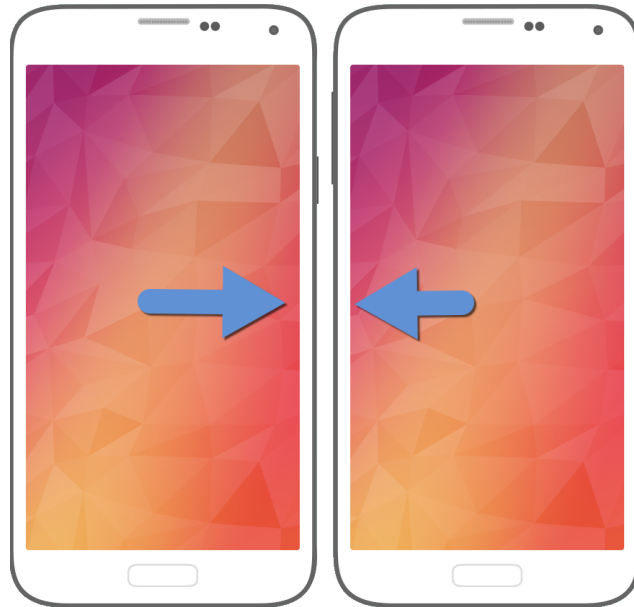


Figura 3.4: Gesto basado en el *touch* del dispositivo



Figura 3.5: Gesto de acoplamiento mediante acelerómetro

Como se mencionó anteriormente, una de las ventajas principales para el uso de gestos de acoplamiento es el aprovechamiento de los recursos que ofrece la tecnología actual de los dispositivos móviles, para poder formar tales como el tamaño y el espacio de las pantallas [Campos et al., 2013]. Por ejemplo, después de unir los dispositivos, se puede mostrar las secciones del espacio de trabajo en cada una de las pantallas de los dispositivos. Esto implica que se habiliten las interfaces de usuario que se muestran en cada dispositivo en un espacio colaborativo.

Debido a los dispositivos no siempre tiene el mismo hardware se puede hacer uso combinado de ambos gestos propuestos. Los usuarios son libres de hacer uso del gesto que más resulte agradable. Además, se hace uso de la háptica y de sonidos que retroalimentan la experiencia del usuario al tratar de hacer un acoplamiento y que este sea exitoso.

Un dispositivo (concretamente, la aplicación que está siendo ejecutada por dicho dispositivo) puede operar únicamente bajo un solo modo de trabajo en un determinado momento. Dependiendo de los modos de trabajo bajo los cuales se encontraban operando dos dispositivos recién acoplados, la sesión colaborativa resultante puede:

- **Ser creada:** para el caso en el que ninguno de los dispositivos involucrados fuera integrante de una sesión colaborativa antes del acoplamiento, i.e., ninguno de los dispositivos había sido acoplado anteriormente.
- **Ser combinada:** sucede cuando ambos dispositivos ya pertenecían a una sesión colaborativa distinta, resultando en la combinación de dichas sesiones, i.e., ambos dispositivos ya habían sido acoplados anteriormente.
- **Ser aumentada:** cuando se une un espacio de trabajo individual al espacio de trabajo colaborativo de una sesión colaborativa existente, i.e., solamente uno de los dispositivos ya había sido acoplado anteriormente.

El formato de los mensajes de acoplamiento para Android e iOS es el siguiente:

- **IP:** contiene la IP del dispositivo.
- **TCP:** contiene el puerto TCP generado por el soporte.
- **UDP:** contiene el puerto UDP generado por el soporte.
- **DIR:** contiene el lado de acoplamiento (e.g., arriba, abajo, izquierda o derecha).
- **SPECS:** contiene las características del dispositivo móvil (e.g., tamaño de la pantalla, tipos de sensores).

3.2.4. Servicio de actualización

El servicio de actualización es el encargado de recibir y enviar mensajes de los cambios producidos en los objetos (e.g., creados, modificados o eliminados) dentro del espacio de trabajo colaborativo. Estas modificaciones son realizadas por medio de mensajes, con el fin de poder reconocer las acciones que desean realizar, esto quiere decir que dichos mensajes pueden ser simples o complejos dependiendo la aplicación en que se esté ejecutando. La actualización de los objetos se realiza a través del protocolo TCP y el puerto de comunicación asignado por el soporte. Este servicio puede ser activado de dos formas distintas:

- **Explícita:** cuando se envía o recibe algún mensaje de algún dispositivo miembro de la sesión colaborativa o cuando se crea, modifica o elimina uno o más objetos pertenecientes al espacio de trabajo común.
- **Implícita:** sucede cuando la activación del servicio se da como resultado de un acoplamiento de dispositivos, e.g., difusión del nuevo conjunto de objetos después de acoplar dos dispositivos cuyos espacios de trabajo ya contenían uno o más objetos. El servicio de comunicación será quien le comunique dicho evento.

El formato de los mensajes de actualización para Android e iOS es el siguiente:

- **TYPE:** contiene el tipo de mensaje (e.g., actualizar objeto, actualizar fondo del espacio de trabajo).
- **OBJECT:** contiene los objetos que se actualizaran.

3.2.5. Servicio de presencia

Este servicio está encargado de monitorear la presencia de los dispositivos que se encuentren acoplados en una sesión colaborativa, mediante dos tipos de mensaje.

1. **Mensaje de presencia:** este mensaje tiene el fin de comprobar la permanencia de los demás participantes de la sesión colaborativa.
2. **Mensaje de ausencia:** este mensaje es producido cuando un dispositivo que forma parte de una sesión colaborativa no responde a los mensajes de presencia de otros dispositivos que participan en esta sesión.

Este servicio permanece en ejecución luego de haberse creado una sesión colaborativa. Después de haber comprobado la presencia de los dispositivos, si detecta que algún dispositivo no responde a estos mensajes, se inician dos mecanismos:

1. El primer mecanismo es eliminar los datos que hacen referencia al dispositivo en la lista de dispositivos (e.g., IP, puertos de conexión, sistema operativo, nombre del dispositivo móvil).
2. El segundo mecanismo es informar al usuario por medio un mensaje que le mostrará las acciones que se pueden realizar. Las acciones que se contemplan son recuperar los objetos virtuales del espacio de trabajo o descartarlos.

En envío de los mensajes de presencia y de recolección se realiza a través del protocolo TCP y un puerto asignado por el desarrollador. El formato de los mensajes de presencia para Android e iOS es el siguiente:

- **IP:** contiene la IP del dispositivo.
- **OBJECT:** contiene los objetos del espacio del trabajo del dispositivo.
- **STATUS:** contiene la fecha de emisión del mensaje (e.g., hora, minuto y segundo).

3.3. Casos de uso

A continuación, se presentan los diagramas de casos de uso generales y específicos más representativos, junto con sus respectivos formatos extendidos. Dichos diagramas guardan una estrecha relación con las definiciones mostradas gráficamente en la secciones anteriores.

3.3.1. Caso de uso: Usuario

A continuación, se presentan los diagramas de casos de uso de usuario, así como los formatos extendidos.

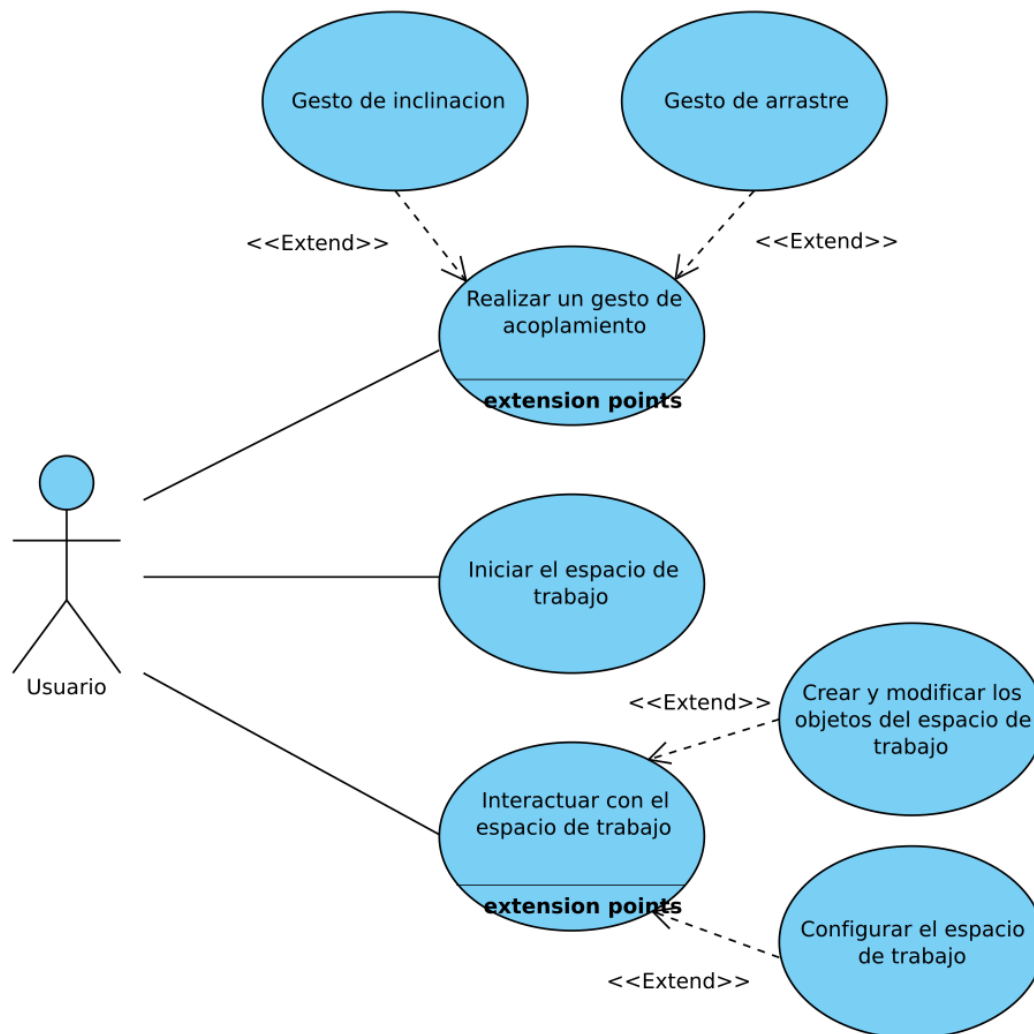


Figura 3.6: Diagrama de casos de uso de Usuario

Caso de uso	CU1: Realizar un gesto de acoplamiento
Versión:	1.0
Actor(es):	Usuario
Propósito:	Habilitar el dispositivo para aceptar y procesar gestos de acoplamiento
Resumen:	Crear una sesión colaborativa
Entradas:	Ninguna
Salidas:	Retroalimentación háptica y acústica
Pre-condiciones:	Haber configurado y seleccionado el gesto de acoplamiento.
Post-condiciones:	Ninguna
Módulo:	Usuario

Tabla 3.1: Caso de uso 1: Realizar un gesto de acoplamiento

Caso de uso	CU2: Realizar un gesto de inclinación
Versión:	1.0
Actor(es):	Usuario
Propósito:	Validar el gesto de inclinación realizado por el usuario
Resumen:	Crear una sesión colaborativa
Entradas:	Datos registrados por el acelerómetro del dispositivo móvil
Salidas:	Retroalimentación háptica y acústica
Pre-condiciones:	Haber realizado un click largo
Post-condiciones:	Ninguna
Módulo:	Usuario

Tabla 3.2: Caso de uso 2: Realizar un gesto de inclinación.

Caso de uso	CU3: Realizar un gesto de arrastre
Versión:	1.0
Actor(es):	Usuario
Propósito:	Validar el gesto de arrastre realizado por el usuario
Resumen:	Crear una sesión colaborativa
Entradas:	Datos registrados por el <i>touch</i> del dispositivo móvil
Salidas:	Retroalimentación háptica y acústica
Pre-condiciones:	Haber realizado un click largo
Post-condiciones:	Ninguna
Módulo:	Usuario

Tabla 3.3: Caso de uso 3: Realizar un gesto de arrastre.

Caso de uso	CU4: Iniciar el espacio de trabajo
Versión:	1.0
Actor(es):	Usuario
Propósito:	Iniciar la aplicación que implemente el soporte en un dispositivo compatible
Resumen:	iniciar el soporte para la creación de sesiones colaborativas
Entradas:	Ninguna
Salidas:	Ninguna
Pre-condiciones:	Ninguna
Post-condiciones:	Ninguna
Módulo:	Usuario

Tabla 3.4: Caso de uso 4: Iniciar el espacio de trabajo.

Caso de uso	CU5: Interactuar con el espacio de trabajo
Versión:	1.0
Actor(es):	Usuario
Propósito:	Crear y eliminar objetos en el espacio de trabajo, además de configurar el soporte e interactuar con los espacios de trabajo colaborativas
Resumen:	Interactuar con el espacio de trabajo en el dispositivo móvil
Entradas:	Ninguna
Salidas:	Ninguna
Pre-condiciones:	Haber iniciado la aplicación
Post-condiciones:	Ninguna
Módulo:	Usuario

Tabla 3.5: Caso de uso 5: Interactuar con el espacio de trabajo.

Caso de uso	CU6: Crear y modificar objetos
Versión:	1.0
Actor(es):	Usuario
Propósito:	Crear y eliminar objetos en el espacio de trabajo
Resumen:	Interactuar con los objetos del área de trabajo
Entradas:	Ingresa un texto para el objeto para su creación, seleccionar un objeto a eliminar
Salidas:	Ninguna
Pre-condiciones:	Haber creado un objeto antes de eliminarlo
Post-condiciones:	Ninguna
Módulo:	Usuario

Tabla 3.6: Caso de uso 6: Crear y modificar objetos.

Caso de uso	CU7: Configurar el espacio de trabajo
Versión:	1.0
Actor(es):	Usuario
Propósito:	Configurar los gestos de acoplamiento
Resumen:	Seleccionar el gesto de acoplamiento que se desea usar
Entradas:	Pulsar botón de configuración
Salidas:	Ninguna
Pre-condiciones:	Ninguna
Post-condiciones:	Ninguna
Módulo:	Usuario

Tabla 3.7: Caso de uso: Configurar el espacio de trabajo.

3.3.2. Caso de uso: Espacio de trabajo

A continuación, se presentan los diagramas de casos de uso del espacio de trabajo, así como los formatos extendidos.

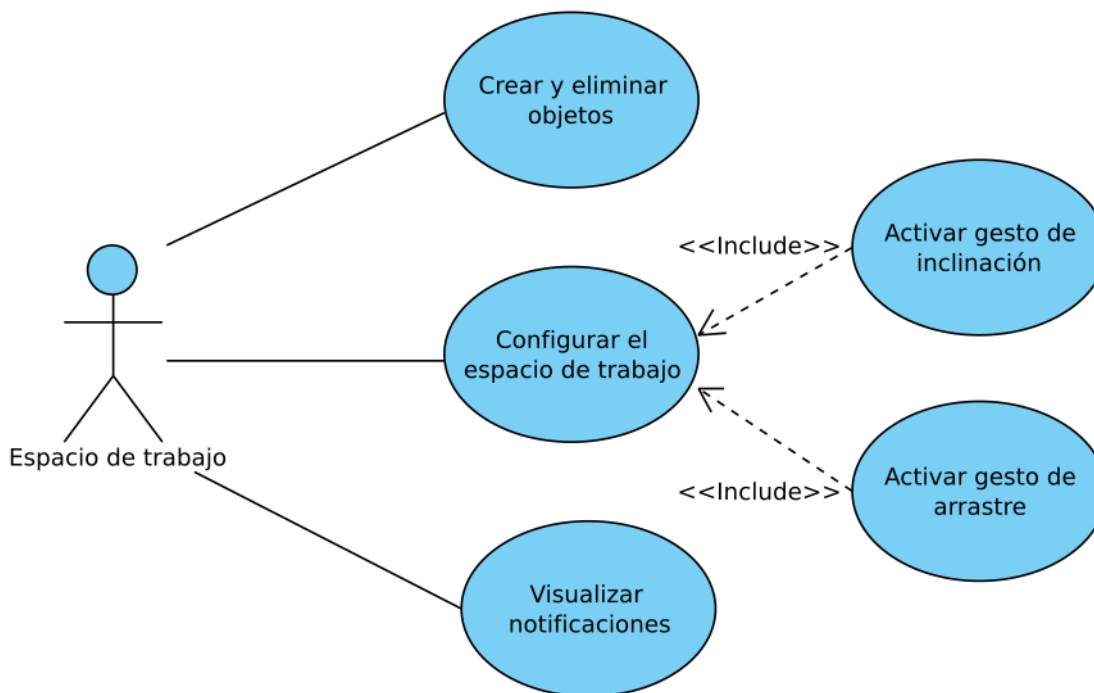


Figura 3.7: Diagrama de casos de uso del espacio de trabajo

Caso de uso	CU8: Interactuar con el espacio de trabajo
Versión:	1.0
Actor(es):	Usuario y Dispositivo
Propósito:	Interactuar con el espacio de trabajo
Resumen:	Permitir interactuar con el espacio de trabajo creando sesiones colaborativas
Entradas:	Ninguna
Salidas:	Ninguna
Pre-condiciones:	Ninguna
Post-condiciones:	Ninguna
Módulo:	Espacio de trabajo

Tabla 3.8: Caso de uso 8: Interactuar con el espacio de trabajo.

Caso de uso	CU9: Crear y eliminar objetos
Versión:	1.0
Actor(es):	Usuario y Dispositivo
Propósito:	Crear y eliminar objetos en el espacio de trabajo
Resumen:	Permitir crear y eliminar objetos en una sesión individual o colaborativa
Entradas:	Ingresa un texto para crear un objeto seleccionar un objeto para eliminar
Salidas:	Ninguna
Pre-condiciones:	Ninguna
Post-condiciones:	Ninguna
Módulo:	Espacio de trabajo

Tabla 3.9: Caso de uso 9: Crear y eliminar objetos.

Caso de uso	CU10: Configurar el espacio de trabajo
Versión:	1.0
Actor(es):	Usuario y Dispositivo
Propósito:	Acceder a la configuración del soporte
Resumen:	Seleccionar las configuraciones permitidas por el soporte
Entradas:	Ninguna
Salidas:	Ninguna
Pre-condiciones:	Ninguna
Post-condiciones:	Ninguna
Módulo:	Espacio de trabajo

Tabla 3.10: Caso de uso 10: Configurar el espacio de trabajo.

Caso de uso	CU11: Activar gesto de inclinación
Versión:	1.0
Actor(es):	Dispositivo
Propósito:	Seleccionar gesto de inclinación como predeterminado
Resumen:	Activar gesto de inclinación como predeterminado
Entradas:	Seleccionar gesto
Salidas:	Ninguna
Pre-condiciones:	Estar en el menú configuración
Post-condiciones:	Ninguna
Módulo:	Espacio de trabajo

Tabla 3.11: Caso de uso 11: Activar gesto de inclinación.

Caso de uso	CU12: Activar gesto de arrastre
Versión:	1.0
Actor(es):	Dispositivo
Propósito:	Seleccionar gesto de arrastre como predeterminado
Resumen:	Activar gesto de arrastre como predeterminado
Entradas:	Ninguna
Salidas:	Ninguna
Pre-condiciones:	Estar en el menú configuración
Post-condiciones:	Ninguna
Módulo:	Espacio de trabajo

Tabla 3.12: Caso de uso 12: Activar gesto de arrastre.

Caso de uso	CU13: Visualizar notificaciones
Versión:	1.0
Actor(es):	Dispositivo
Propósito:	Mostrar las notificaciones en en dispositivo
Resumen:	Visualizar las notificaciones de los cambios que se realicen en el soporte
Entradas:	Ninguna
Salidas:	Ninguna
Pre-condiciones:	Haber iniciado el espacio de trabajo
Post-condiciones:	Ninguna
Módulo:	Espacio de trabajo

Tabla 3.13: Caso de uso 13: Visualizar notificaciones.

3.3.3. Caso de uso: Servicio de descubrimiento

A continuación, se presentan los diagramas de casos de uso del servicio de descubrimiento, así como los formatos extendidos.

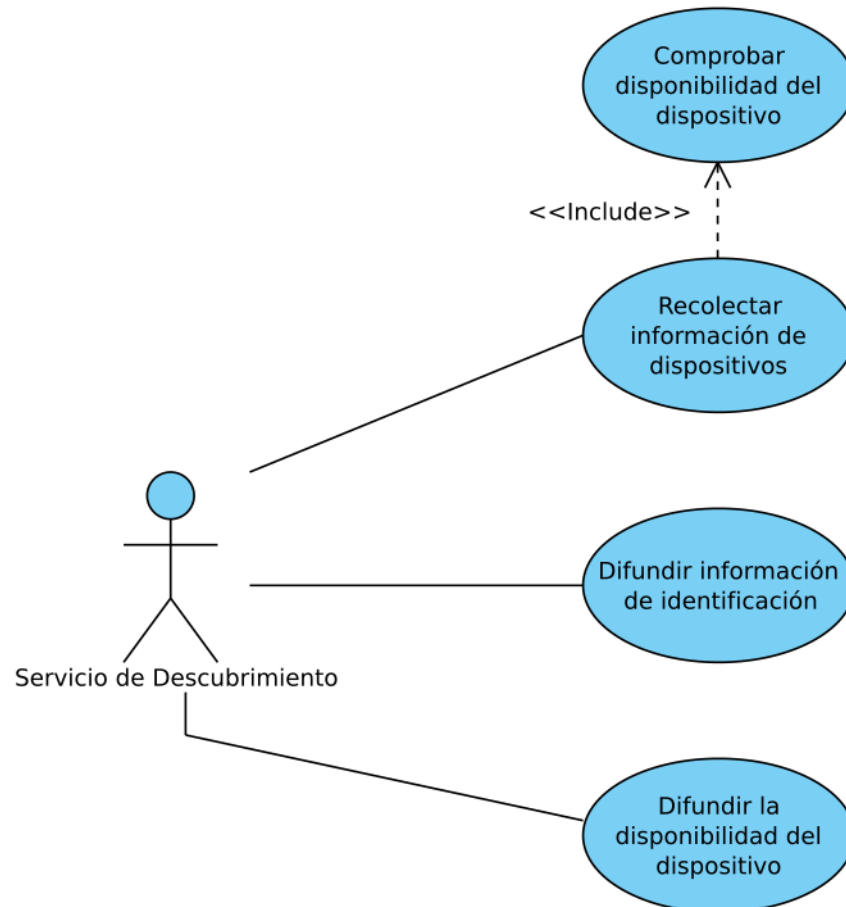


Figura 3.8: Diagrama de casos de uso del servicio de descubrimiento

Caso de uso	CU14: Recolectar información de dispositivos
Versión:	1.0
Actor(es):	Dispositivo
Propósito:	Permitir saber que dispositivos se encuentran ejecutando el soporte en una red local
Resumen:	Identificar las características de los dispositivos que se encuentran presentes en una red local y que estén ejecutando el soporte
Entradas:	<i>String</i> en formato JSON con las características del dispositivo como: IP, puertos, disponibilidad, nombre del SO, nombre del dispositivo
Salidas:	Ninguna
Pre-condiciones:	Estar en una sesión colaborativa
Post-condiciones:	Ninguna
Módulo:	Servicios

Tabla 3.14: Caso de uso 14: Recolectar información de dispositivos.

Caso de uso	CU15: Comprobar disponibilidad del dispositivo
Versión:	1.0
Actor(es):	Dispositivo
Propósito:	Verificar que la disponibilidad del dispositivo
Resumen:	Identificar si el dispositivo esta disponible o esta abandonando la aplicación
Entradas:	<i>String</i> en formato JSON con las características del dispositivo como: IP, puertos, disponibilidad, nombre del SO, nombre del dispositivo
Salidas:	Mensaje informativo
Pre-condiciones:	Estar en una sesión colaborativa
Post-condiciones:	Ninguna
Módulo:	Servicios

Tabla 3.15: Caso de uso 15: Comprobar disponibilidad del dispositivo.

Caso de uso	CU16: Difundir información de identificación
Versión:	1.0
Actor(es):	Dispositivo
Propósito:	Difundir la disponibilidad y presencia a través de red local
Resumen:	Publicar la información de identificación del dispositivo, así como la disponibilidad
Entradas:	<i>String</i> en formato JSON con las características del dispositivo como: IP, puertos, disponibilidad, nombre del SO, nombre del dispositivo
Salidas:	Mensaje informativo
Pre-condiciones:	Estar en una sesión colaborativa
Post-condiciones:	Ninguna
Módulo:	Servicios

Tabla 3.16: Caso de uso: Difundir información de identificación.

Caso de uso	CU17: Difundir la disponibilidad del dispositivo
Versión:	1.0
Actor(es):	Dispositivo
Propósito:	Difundir la disponibilidad del dispositivo
Resumen:	Difundir si el dispositivo esta disponible o esta abandonando la aplicación
Entradas:	<i>String</i> en formato JSON con las características del dispositivo como: IP, puertos, disponibilidad, nombre del SO, nombre del dispositivo
Salidas:	Mensaje informativo
Pre-condiciones:	Estar en una sesión colaborativa
Post-condiciones:	Ninguna
Módulo:	Servicios

Tabla 3.17: aso de uso 17: Difundir la disponibilidad del dispositivo.

3.3.4. Caso de uso: Servicio de comunicación

A continuación, se presentan los diagramas de casos de uso del servicio de comunicación, así como los formatos extendidos.

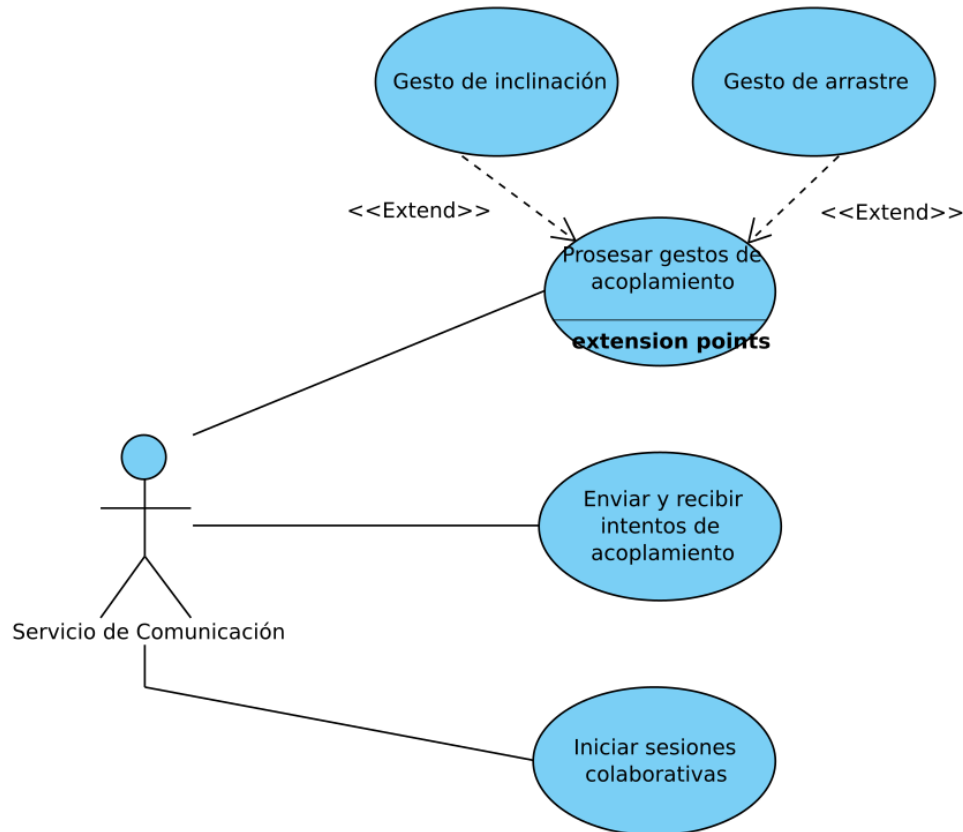


Figura 3.9: Diagrama de casos de uso del servicio de comunicación

Caso de uso	CU18: Procesar gestos de acoplamiento
Versión:	1.0
Actor(es):	Usuario y Dispositivo
Propósito:	Ejecutar y procesar gestos de acoplamiento
Resumen:	Procesar los gestos de acoplamiento que se realicen en la aplicación
Entradas:	Gestos de arrastre y de inclinación
Salidas:	Ninguna
Pre-condiciones:	Ninguna
Post-condiciones:	Gestos validos, retroalimentación háptica y acústica
Módulo:	Acoplamiento

Tabla 3.18: Caso de uso 18: Procesar gestos de acoplamiento.

Caso de uso	CU19: Procesar un gesto de inclinación
Versión:	1.0
Actor(es):	Dispositivo
Propósito:	Registrar las mediciones que se realicen con el gesto de inclinación
Resumen:	Obtener mediciones por medio del acelerómetro y verificar si el gesto de inclinación es valido
Entradas:	Ninguna
Salidas:	Gesto valido o invalido
Pre-condiciones:	Mediciones obtenidas a través del acelerómetro
Post-condiciones:	Ninguna
Módulo:	Acoplamiento

Tabla 3.19: Caso de uso 19: Procesar un gesto de inclinación.

Caso de uso	CU20: Procesar un gesto de arrastre
Versión:	1.0
Actor(es):	Dispositivo
Propósito:	Registrar las mediciones que se realicen con el gesto de arrastre
Resumen:	Obtener mediciones por medio del <i>touch</i> de la pantalla del dispositivo y verificar si el gesto de arrastre es valido
Entradas:	Ninguna
Salidas:	Gesto valido o invalido
Pre-condiciones:	Ninguna
Post-condiciones:	Ninguna
Módulo:	Acoplamiento

Tabla 3.20: Caso de uso 20: Procesar un gesto de arrastre.

Caso de uso	CU21: Enviar y recibir intentos de acoplamiento
Versión:	1.0
Actor(es):	Dispositivo
Propósito:	Difundir y recolectar un intento de acoplamiento
Resumen:	Difundir y recolectar datos de intento de acoplamiento de otro dispositivo que igual mente realizo un intento de acoplamiento
Entradas:	<i>String</i> en formato JSON con las siguientes características: IP, puertos, disponibilidad, nombre del SO, nombre del dispositivo, dirección de acoplamiento
Salidas:	Retroalimentación háptica y acústica
Pre-condiciones:	Ninguna
Post-condiciones:	Ninguna
Módulo:	Acoplamiento

Tabla 3.21: Caso de uso 21: Enviar y recibir intentos de acoplamiento.

Caso de uso	CU22: Iniciar sesiones colaborativas
Versión:	1.0
Actor(es):	Dispositivo
Propósito:	Crear una sesión colaborativa
Resumen:	Crear una sesión colaborativa, unir dos o mas espacios de trabajo individuales
Entradas:	Ninguna
Salidas:	Ninguna
Pre-condiciones:	Estar en una sesión colaborativa
Post-condiciones:	Ninguna
Módulo:	Acoplamiento

Tabla 3.22: Caso de uso 22: Iniciar sesiones colaborativas.

3.3.5. Caso de uso: Servicio de presencia

A continuación, se presentan los diagramas de casos de uso del servicio de presencia, así como los formatos extendidos.

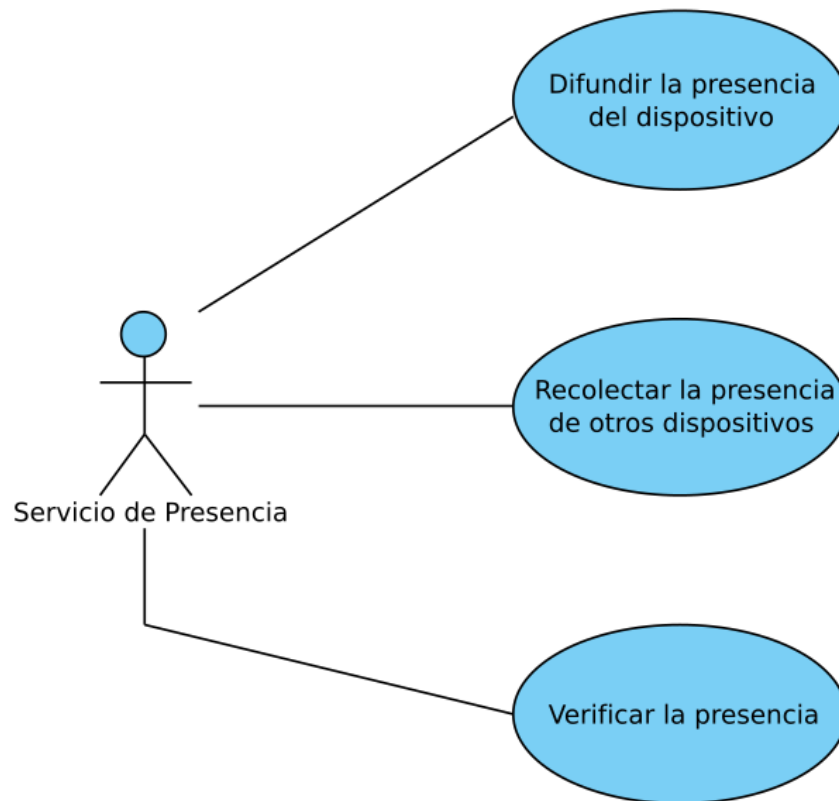


Figura 3.10: Diagrama de casos de uso del servicio de presencia

Caso de uso	CU23: Difundir la presencia del dispositivo
Versión:	1.0
Actor(es):	Dispositivo
Propósito:	Enviar mensajes de presencia en una sesión colaborativa
Resumen:	Enviar los mensajes de presencia
Entradas:	IP del dispositivo, objetos del espacio de trabajo y fecha
Salidas:	Mensaje de presencia
Pre-condiciones:	Ninguna
Post-condiciones:	Estar en una sesión colaborativa
Módulo:	Presencia

Tabla 3.23: Caso de uso 23: Difundir la presencia del dispositivo.

Caso de uso	CU24: Recolectar mensajes de presencia
Versión:	1.0
Actor(es):	Dispositivo
Propósito:	Recolectar los mensajes de presencia en una sesión colaborativa
Resumen:	Recolectar los mensajes de presencia y guardarlos para su posterior verificación
Entradas:	Ninguna
Salidas:	Ninguna
Pre-condiciones:	Estar en una sesión colaborativa
Post-condiciones:	Ninguna
Módulo:	Presencia

Tabla 3.24: Caso de uso 24: Recolectar mensajes de presencia.

Caso de uso	CU25: Verificación de los mensajes de presencia
Versión:	1.0
Actor(es):	Usuario y Dispositivo
Propósito:	Verificar los mensajes de presencia
Resumen:	Verificar el tiempo de los mensajes de presencia contra el tiempo del dispositivo
Entradas:	Ninguna
Salidas:	Recuperación de objetos en el espacio de trabajo
Pre-condiciones:	Ninguna
Post-condiciones:	Ninguna
Módulo:	Presencia

Tabla 3.25: Caso de uso 25: Verificación de los mensajes de presencia.

3.4. Diagrama de despliegue y de paquetes

En esta sección se describe la estructura interna del marco de desarrollo y la organización de cada uno de sus componentes, a través de una serie de diagramas cuyo propósito es ilustrar de forma más específica, la arquitectura y funcionalidades descritas en las secciones anteriores.

3.4.1. Diagrama de despliegue

En la Figura 3.11 se muestra de despliegue del marco de desarrollo, el cual contempla tres componentes que son: usuario, dispositivo móvil y red local. El diagrama de despliegue engloba los componentes que componen una aplicación creada con el marco de desarrollo. Dichos componentes se describen a continuación:

- **Usuario:** el participante de una sesión que interactúa con una aplicación desarrollada mediante el marco de desarrollo propuesto que se ejecuta en un dispositivo móvil.
- **Dispositivo móvil:** contiene los servicios diseñados con el marco de desarrollo propuesto y está encapsulado en un archivo (e.g., ipa para iOS o apk para Android), el cual se instala en un dispositivo móvil compatible.
- **Red local:** infraestructura que se usa para la para comunicación de los servicios diseñados por el marco de desarrollo propuesto con otros dispositivos móviles.

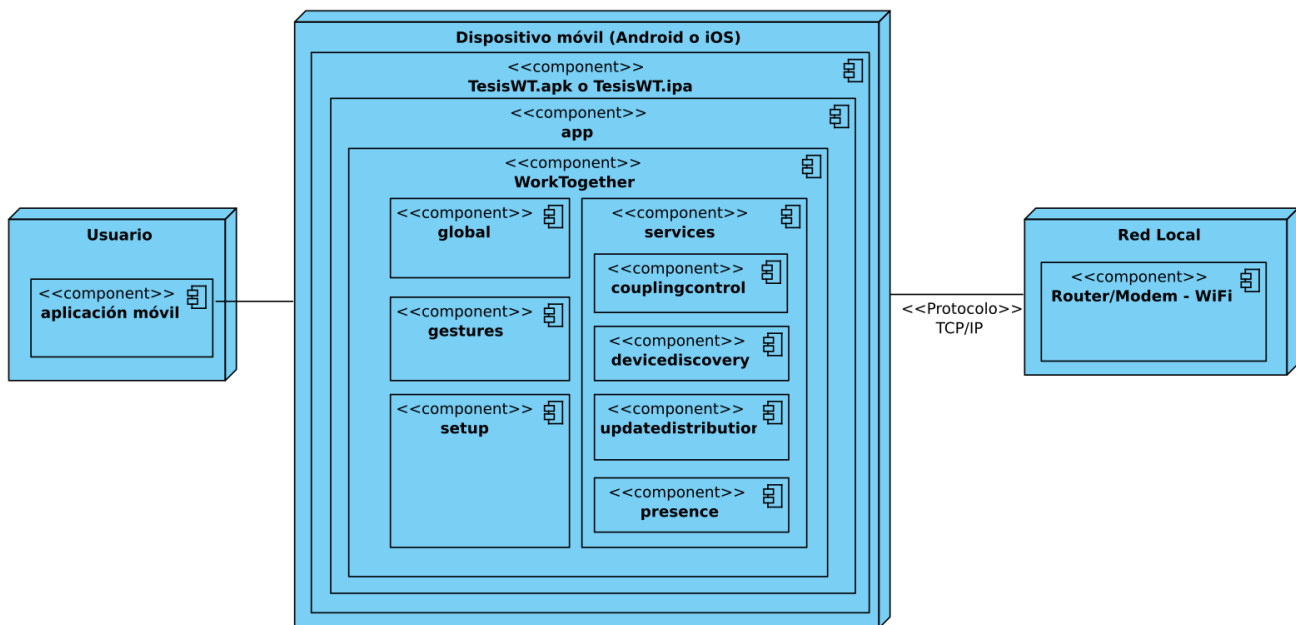


Figura 3.11: Diagrama de despliegue del soporte

3.4.2. Diagrama de paquetes

El diagrama de paquetes *WorkTogether* (ver Figura 3.12) está compuesto por otros cuatro paquetes relacionados entre sí, cuyo contenido engloba la configuración, gestos y servicios que permiten el funcionamiento del marco de desarrollo. A continuación se describen cada uno de estos paquetes:

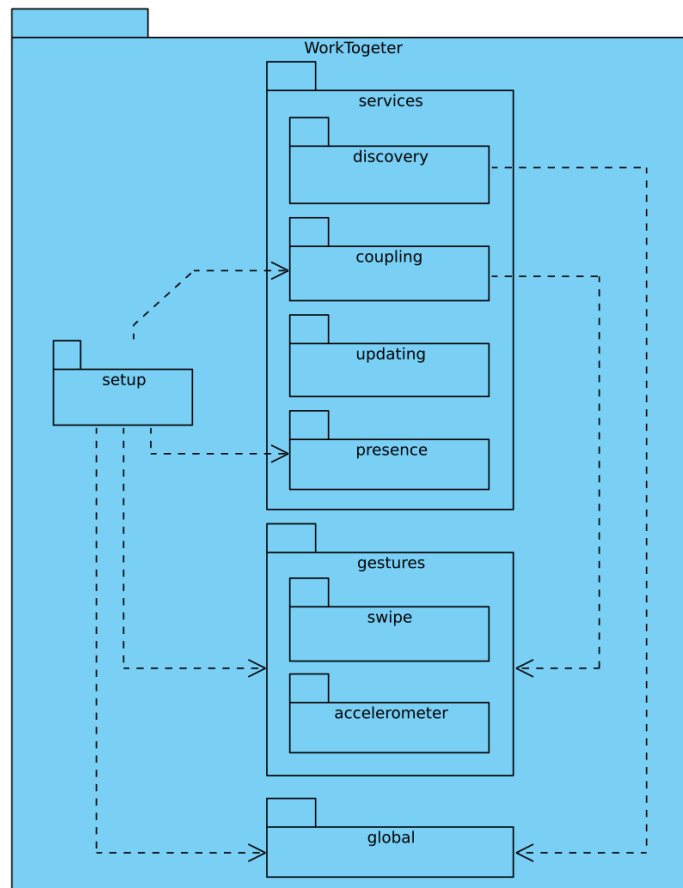


Figura 3.12: Diagrama de paquetes del soporte

- **setup**: este paquete proporciona una clase llamada `WorkTogetherSetup` que pretende facilitar la implementación del soporte propuesto, al brindar la configuración básica del soporte.
- **gestures**: este paquete contiene las configuraciones hechas por el paquete *setup* y los algoritmos que se utilizan para reconocer o validar los gestos de acoplamiento (ver Sección 3.2.3) cuyas definiciones se encuentran en los paquetes *swipe* y *accelerometer*.
- **global**: en este paquete están agrupados todos los recursos compartidos por cada uno de los cuatro servicios, tales como las listas de los dispositivos identificados de los dispositivos que están en una sesión colaborativa y de los dispositivos que comparten una sección del espacio de trabajo. Así mismo, en este paquete se encuentran definidas las funcionalidades necesarias para evitar problemas de concurrencia en dichos recursos.

- **services**: este paquete contiene los servicios que se utilizan para iniciar, utilizar y terminar los servicios de descubrimiento, comunicación, actualización y presencia. Está compuesto por los paquetes *discovery*, *coupling*, *updating* y *presence*.

3.5. Diagramas de clases

A continuación, se presentan los diagramas de cada una de las clases que conforman los paquetes del soporte (*setup*, *gestures*, *global*, *services*).

3.5.1. Clases del paquete *workSpace*

El diagrama de clases del paquete *workSpace* (ver Figura 3.13) está compuesto por las siguientes clases:

- Clase **WorkSpace**: esta clase se encarga de crear el espacio de trabajo que implementa el soporte propuesto. Además, esta clase configura el entorno, crea y elimina objetos y muestra mensajes y notificaciones del soporte.
- Clase **ResourcesWS**: esta clase es una auxiliar de la clase **WorkSpace**, la cual debe ser implementada por el desarrollador.
- Clase **WorkTogetherSetup**: esta clase es encargada de iniciar el soporte, así como los servicios y configuraciones propuestas.

3.5.2. Clases del paquete *setup*

El diagrama de clases del paquete *setup* (ver Figura 3.14) contiene la implementación del marco de desarrollo propuesto y está contenido por la siguiente clase:

- Clase **WorkTogetherSetup**: esta clase se encarga de inicializar las configuraciones del soporte (e.g., activar gesto de arrastre, activar gesto de inclinación), además de iniciar los servicios que conforman el soporte (e.g., descubrimiento, comunicación y actualización).

3.5.3. Clases del paquete *gestures*

El diagrama de clase del paquete *gestures* (ver Figura 3.15) contiene los mecanismos que se utilizan para reconocer o validar los gestos de acoplamiento y está compuesto por las siguientes clases:

- Clase **SwipeGesture**: esta clase se encarga de verificar los gestos que se realizan en la pantalla del dispositivo (i.e., eventos de tipo *touch*). Los datos recibidos con los cuales se contextualiza el gesto (i.e., los que permiten determinar hacia qué lado se arrastró el dedo del usuario) provienen del mismo objeto al cual se encuentra adherido el detector de clicks largos, definido en la clase **LongClickDetector**.

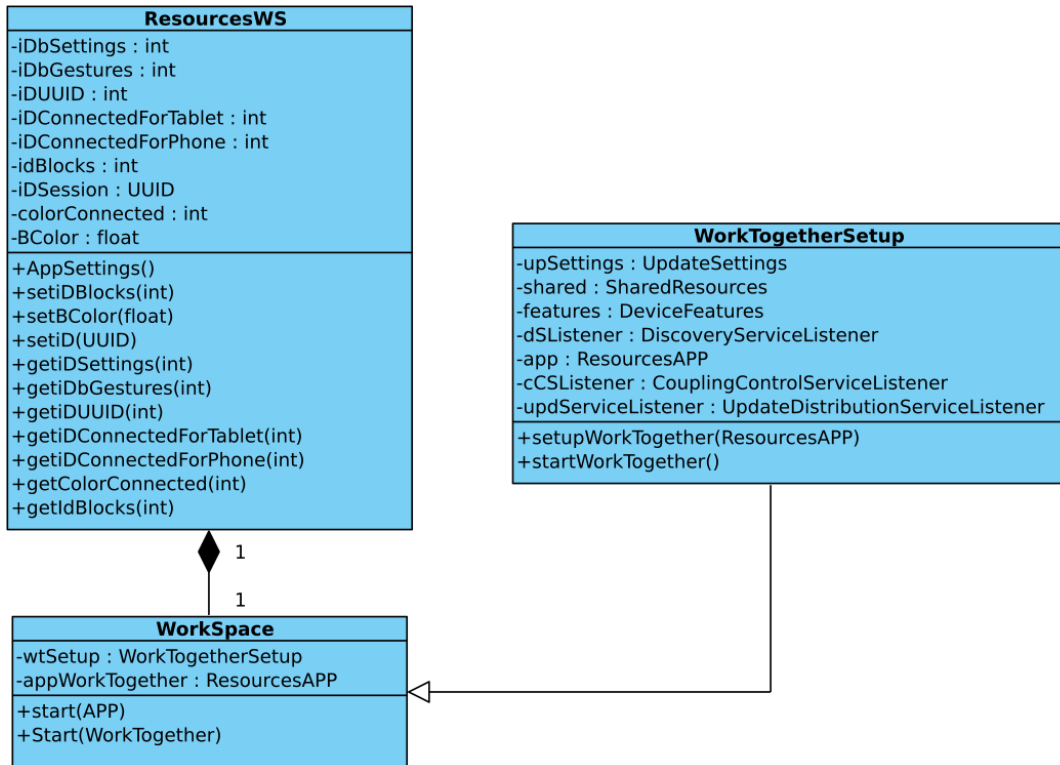


Figura 3.13: Diagrama de clases del paquete *Workspace*

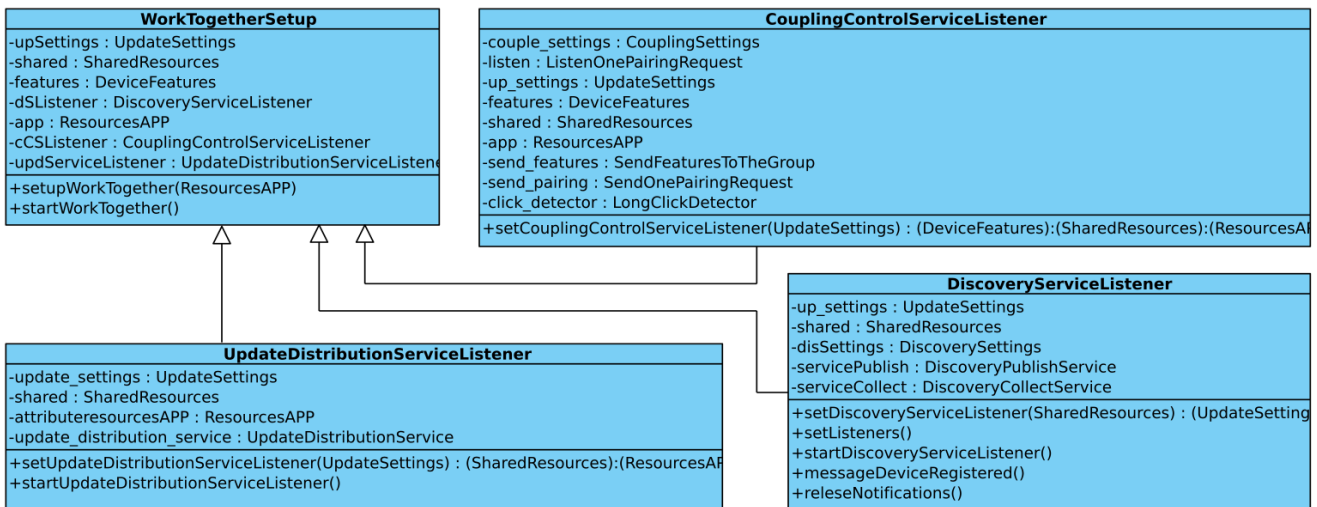
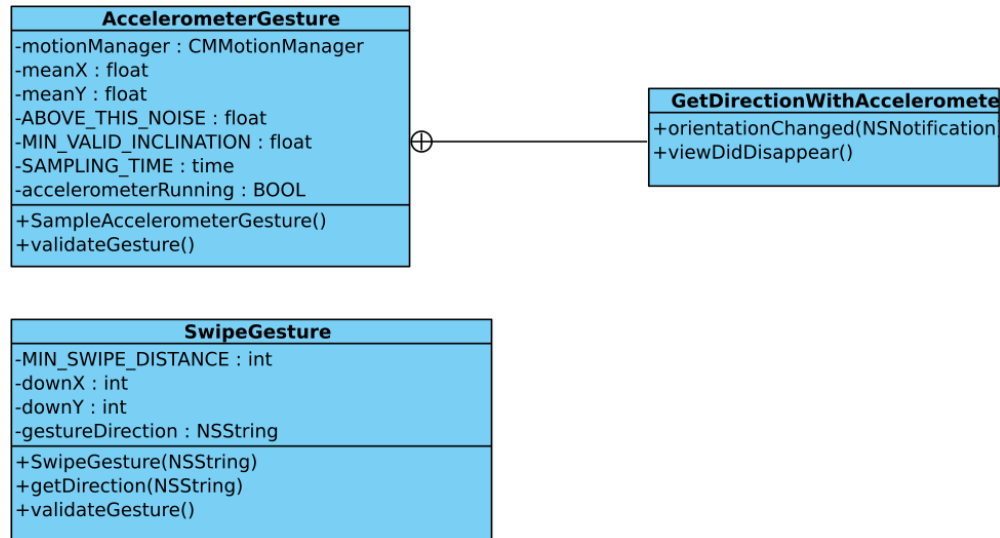


Figura 3.14: Diagramas de clases del paquete *setup*

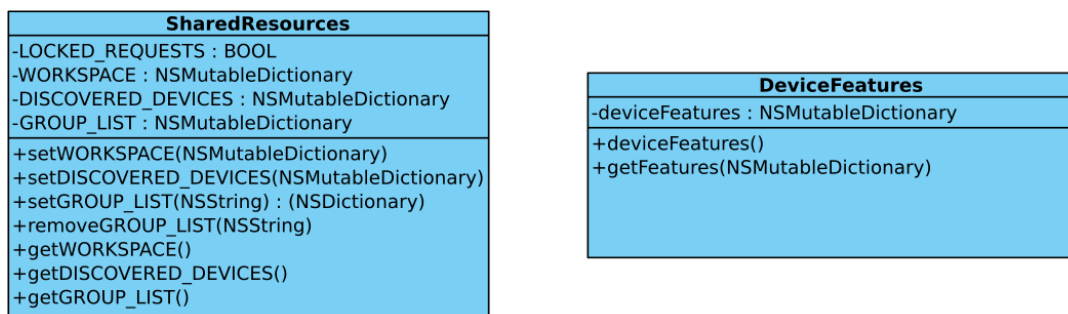
- Clase *AccelerometerGesture*: esta clase se encarga de procesar un gesto de inclinación el cual permite seleccionar algún lado del dispositivo para el proceso de acoplamiento, dependiendo de la inclinación constante del dispositivo durante un tiempo determinado.

Figura 3.15: Diagrama de clases del paquete *gestures*

3.5.4. Clases del paquete *global*

El diagrama de clase del paquete *global* (ver Figura 3.16) contiene las variables compartidas que son usadas por el marco de desarrollo y que a continuación se describen:

- Clase **SharedResources**: almacena los recursos compartidos del soporte (e.g., listas de dispositivos que están en una sesión colaborativa, lista de dispositivos que implementan el soporte y que están presentes en la red local). Dichos recursos son utilizados por cada uno de los servicios y clases ya mencionadas, razón por la cual deben ser inicializados antes que cualquier otro servicio. Debido a la concurrencia sucitada en tiempo de ejecución, estos recursos cuentan con soporte de control de concurrencia.
- Clase **DeviceFeatures**: se encarga de obtener y almacenar las características y funcionalidades de un dispositivo. Las clases **SendOnePairingRequest** y **SendFeaturesToTheGroup** hacen uso de esta clase durante su flujo de operaciones normal.

Figura 3.16: Diagrama de clases del paquete *global*

3.5.5. Clases del paquete *services*

El diagrama de clases del paquete *services.devicediscovery* (ver Figura 3.17) contiene la implementación del servicio de descubrimiento y está compuesto por las siguientes clases:

- Clase **DiscoveryPublishService**: publica datos de identificación continuamente en un lapso de tiempo establecido. Los datos que publica son los siguientes: IP, puertos TCP y UDP, estado de la aplicación (e.g., disponible o abandono de la aplicación), nombre del dispositivo (e.g., iPhone, Samsung o Sony) y nombre del sistema operativo (e.g., Android o iOS)
- Clase **DiscoveryCollectService**: recolecta la información difundida a través de la red. Dicha información que fue producida por otros dispositivos mediante la clase **DiscoveryPublishService**.
- Clase **DiscoverySettings**: guarda las variables para establecer una comunicación con otro dispositivo (e.g., puertos TCP y UDP, tiempo de espera de difusión de los datos de identificación y estado de la aplicación).
- Interfaz **DiscoveryServiceListener**: controla los eventos de las clases **DiscoveryPublishService** y **DiscoveryCollectService**, además de indicarle a la aplicación qué hacer cuando se detecta que un dispositivo externo tiene en funcionamiento el servicio de descubrimiento.

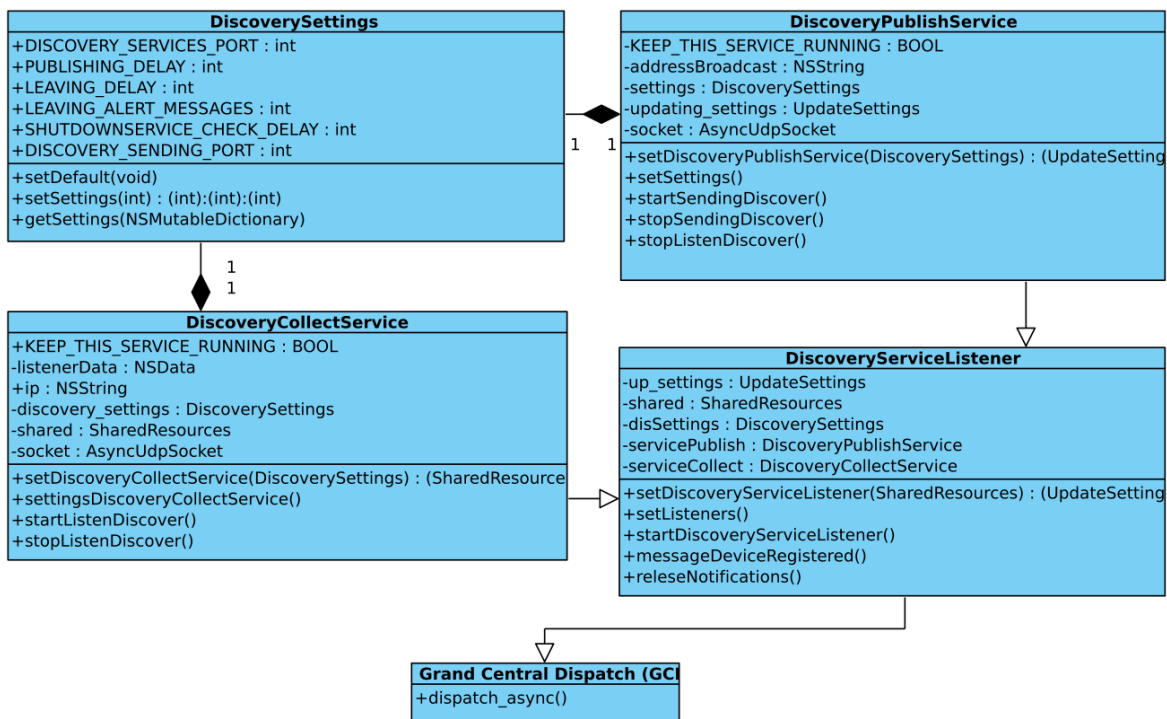


Figura 3.17: Diagrama de clases del paquete *discovery*

El diagrama de clases del paquete *services.coupling* que se muestra en la Figura 3.18, el cual contiene la implementación del servicio de comunicación. Las clases e interfaz que lo componen se describen a continuación:

- Clase **LongClickDetector**: esta clase está encargada de detectar un click largo para procesar un gesto de acoplamiento (e.g., gesto de arrastre o gesto de inclinación).
- Clase **SendOnePairingRequest**: esta clase se encarga de propagar los datos de intento de acoplamiento (e.g., IP, puertos TCP y UDP y el lado por el cual se requiere hacer el acoplamiento).
- Clase **ListenOnePairingRequest**: esta clase recibe un intento de acoplamiento propagado por la clase **SendOnePairingRequest** desde otro dispositivo.
- Clase **SendFeaturesToTheGroup**: esta clase está encargada de propagar a los integrantes de una sesión colaborativa, los cambios que se realicen en el dispositivo (e.g., la rotación del dispositivo).
- Clase **CouplingSettings**: guarda la configuración de las clases anteriores (e.g., los puertos TCP y UDP y el tiempo de espera).
- Interfaz **CouplingControlServiceListener**: controla y procesa los eventos *touch* detectados en la superficie de la pantalla.

El diagrama de clases del paquete *services.updatedistribution* (ver Figura 3.19) está compuesto por las siguientes tres clases que se describen a continuación:

- Clase **UpdateDistributionService**: esta clase tiene el objetivo enviar y recibir los objetos y actualizaciones cuando existe una sesión colaborativa.
- Clase **UpdateSettings**: guarda la configuración de comunicación del soporte (e.g., puertos TCP y UDP).
- Interfaz **DiscoveryServiceListener**: procesa los eventos recibidos de otros dispositivos que implementan el soporte (e.g., objetos, cambios o mensajes).

El diagrama de clases del paquete *services.presence* (ver Figura 3.20) está compuesto por las siguientes clases que se describen a continuación:

- Clase **PresencePublishService**: esta clase es iniciada cuando el soporte se encuentra en una sesión colaborativa y cuyo objetivo es verificar la presencia de los integrantes de la sesión (i.e., los integrantes de la sesión colaborativa siguen estando activos y no han abandonado la sesión).
- Clase **PresenceCollectService**: su principal función de esta clase es recolectar los mensajes de otros dispositivos que se encuentran en una misma sesión colaborativa. Los mensajes que se recolectan son producidos por la clase **PresencePublishService**.
- Clase **PresenceCheckService**: esta clase es encargada de verificar los mensajes de presencia que se reciben en la clase **PresenceCollectService**.

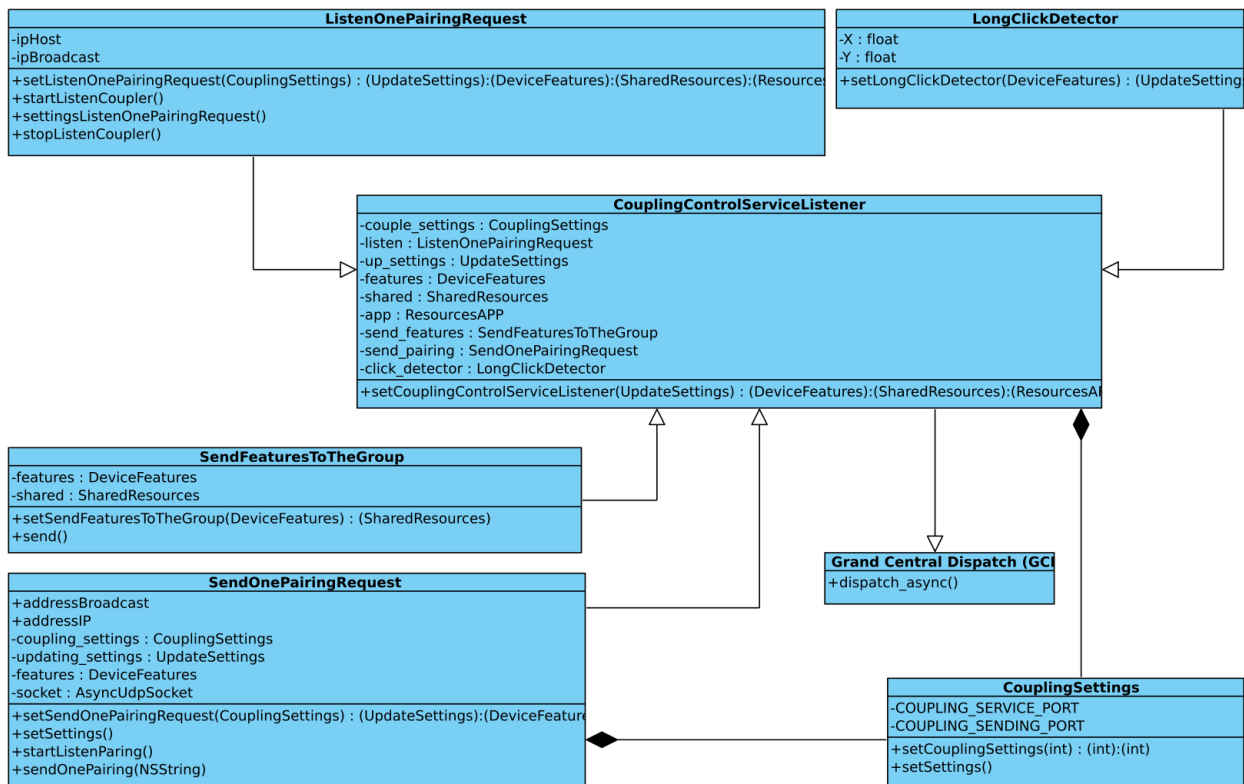


Figura 3.18: Diagrama de clases del paquete *coupling*

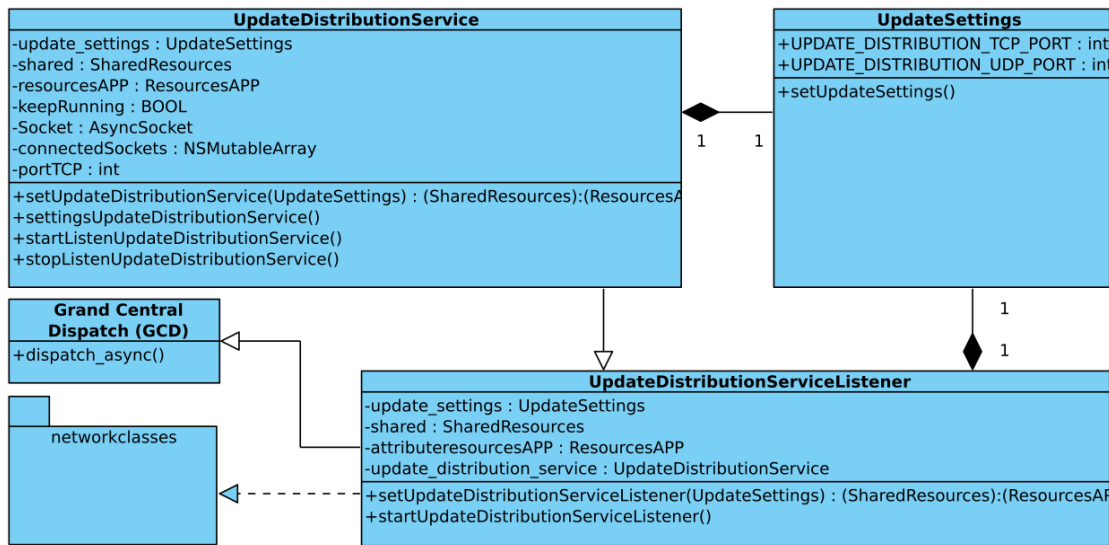
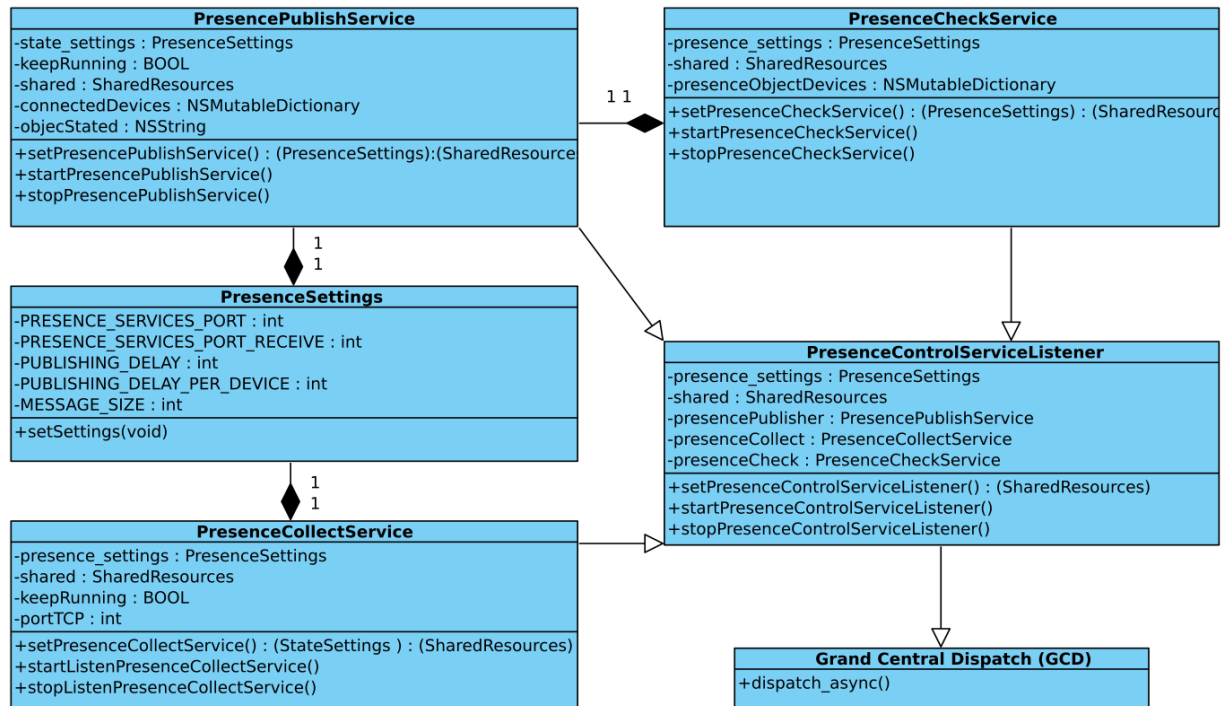


Figura 3.19: Diagrama de clases del paquete *updating*

3.6. Diagramas de secuencia

A continuación, en esta sección se describen las trayectorias de la sección 3.3.

Figura 3.20: Diagrama de clases del paquete *presence*

3.6.1. Servicio de descubrimiento

El diagrama 3.21 muestra la secuencia que realiza el servicio de descubrimiento. La difusión de mensajes de identificación es realizada por la clase `DiscoveryPublishService`. A continuación se describe la sucesión de la comunicación entre las clases de este servicio con la red local:

1. La clase `DiscoveryPublishService` difunde los datos de identificación del dispositivo a través de la red local.
2. La clase `DiscoveryPublishService` repite el proceso de difundir el mensaje de identificación por un determinado tiempo.

Si la aplicación creada por el marco de desarrollo esta por finalizar el mensaje de identificación difundido por la clase `DiscoveryPublishService` cambiara el estado de disponibilidad por falso.

La clase `DiscoveryCollectService` se encarga de recolectar la información producida por la clase `DiscoveryPublishService`. A continuación se describe la sucesión de la comunicación de los mensajes recibidos:

1. La clase `DiscoveryCollectService` captura información difundida a través de la red local por otros dispositivos que implementan el soporte.
2. La clase `DiscoveryCollectService` procesa la información obtenida.
3. La clase `DiscoveryCollectService` informa al usuario del evento procesado.

Si los datos extraídos del mensaje recibido por la clase `DiscoveryCollectService` con tiene el estado en falso, se eliminan los datos relacionados con el dispositivo que emitió el mensaje de

identificación (e.g., IP, puertos de conexión) e informa al usuario del evento procesado con un mensaje en el espacio de trabajo.

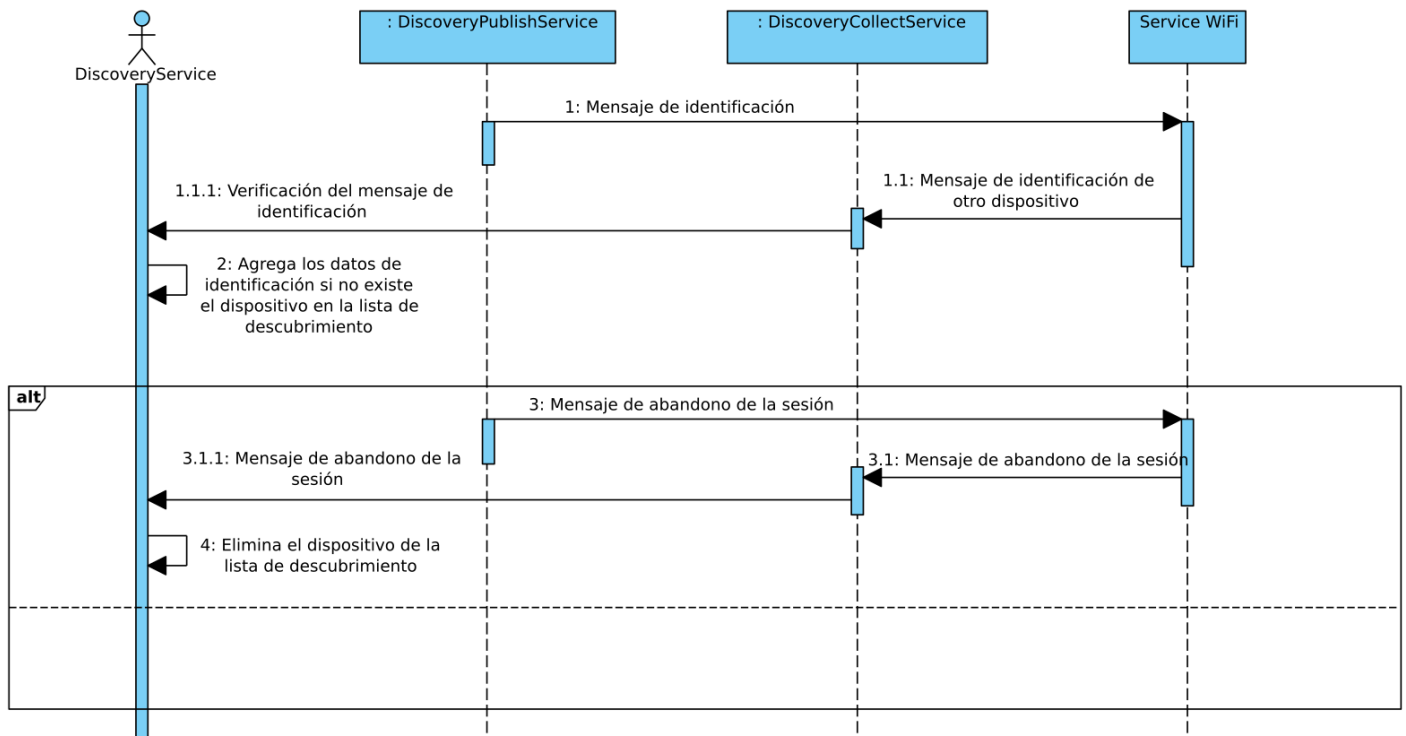


Figura 3.21: Diagrama de secuencia del servicio de descubrimiento

3.6.2. Servicio de comunicación

El diagrama 3.22 muestra la secuencia para realizar un intento de acoplamiento, así como recibir un intento y poder crear una sesión colaborativa. A continuación se describe la sucesión de la comunicación entre las clases de este servicio y la red:

1. El usuario realiza un gesto de acoplamiento.
2. La clase `LongClickDetector` valida gesto de acoplamiento
3. La clase `AccelerometerGesture` procesa el gesto de inclinación.
4. La clase `SwipeGesture` procesa el gesto de arrastre.

Una vez realizado el intento de acoplamiento este es difundido por la clase `SendOnePairingRequest` a través de la red local, el intento de acoplamiento es recibido por la clase `ListenOnePairingRequest`, si el intento de acoplamiento es valido se procede a crear una sesión colaborativa y actualiza las listas de usuarios de la sesión. Posteriormente se produce una retroalimentación háptica y acústica.

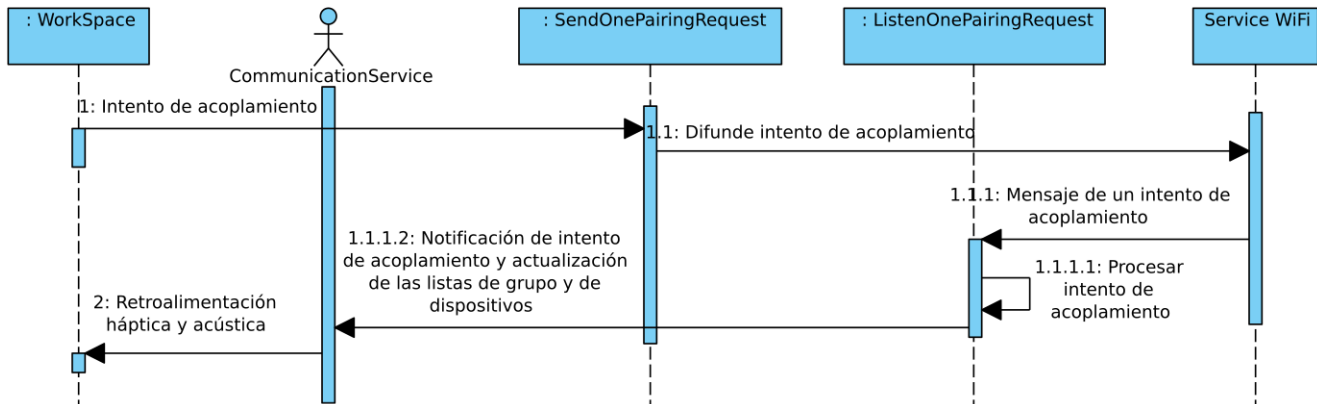


Figura 3.22: Diagrama de secuencia del servicio de comunicación

3.6.3. Servicio de presencia

El diagrama 3.23 muestra la secuencia de la difusión, recolección y verificación de la presencia de un dispositivo en una sesión colaborativa, el servicio de comunicación ejecuta el servicio de difusión de presencia que esta contenido en la clase `PresencePublishService`. A continuación se describe la sucesión de la comunicación entre las clases del servicio de presencia:

1. La clase `PresencePublishService` envía un mensaje de presencia a los dispositivos que pertenecen a una sesión colaborativa.
2. La clase `PresenceCollectService` guarda los mensajes de presencia en la lista `STATE_OBJECTS_DEVICES`.
3. La clase `PresenceCheckService` verifica el tiempo de recepción de los mensajes de presencia.
4. La clase `PresenceCheckService` verifica el tiempo del mensaje contra el tiempo del dispositivo es mayor, se procede a recuperar los objetos de la sesión.
5. La clase `PresenceCheckService` enviara un mensaje a la interfaz de usuario indicándole si desea recuperar los objetos de la sesión.

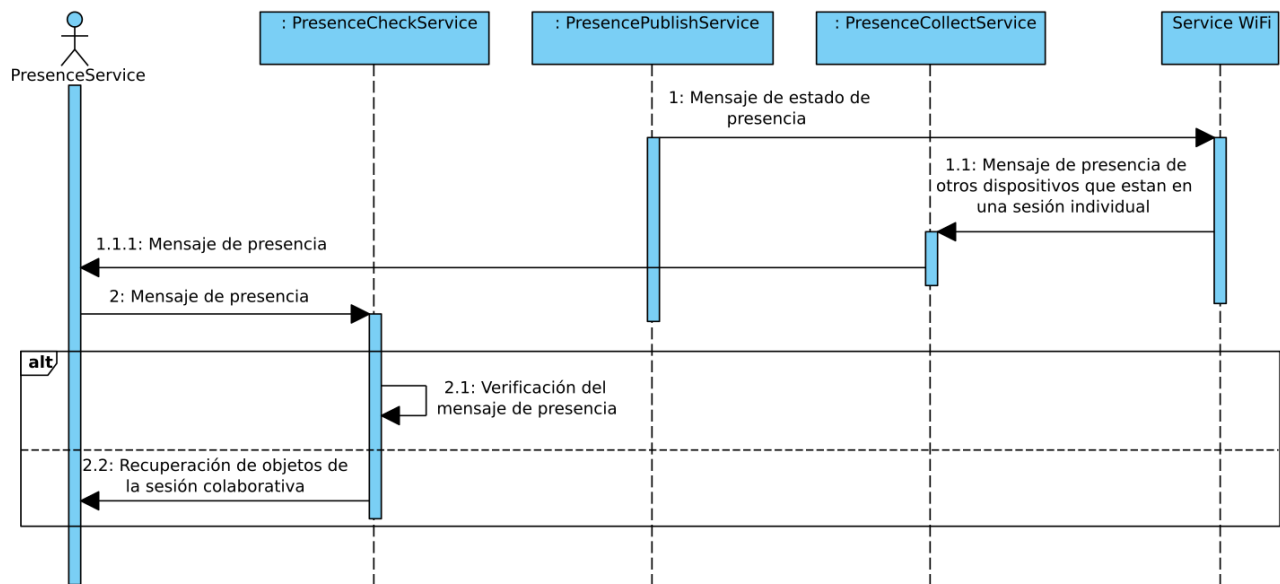


Figura 3.23: Diagrama de secuencia del servicio de presencia

Capítulo 4

Implementación del marco de desarrollo

En este capítulo se describe la implementación del marco de desarrollo, detallando cada uno de los servicios descritos en el capítulo 3. En la Sección 4.1 se presenta la metodología de desarrollo seguida a lo largo de la fase de implementación. En la Sección 4.2 se describen las herramientas de desarrollo, las plataformas móviles y el paradigma de programación. En la Sección 4.3 se aterriza el diseño de cada uno de los servicios, presentando cada uno de los algoritmos que los componen, las herramientas y bibliotecas utilizadas. En la Sección 4.4 se detalla la configuración del marco de desarrollo que está implementada en la clase principal llamada `WorkTogetherSetup`.

4.1. Metodología de desarrollo

El término metodología se refiere a los métodos de investigación en una área científica. Es la parte del proceso de investigación que sigue la propedéutica y permite sistematizar los métodos y las técnicas necesarias para llevarla a cabo.

Actualmente existen varias metodologías, modelos y métodos de desarrollo, algunos de ellos son los siguientes:

- Metodología en cascada
- Metodología en espiral
- Metodología de prototipos
- Modelo en V
- Desarrollo por etapas
- Proceso unificado

Debido a la naturaleza del soporte propuesto, se requirió de un modelo de desarrollo en V, para poder comprobar la efectividad de los avances, que se fueron desarrollando a lo largo de la presente tesis, en cada una de sus fases. Una de las ventajas más resaltables, como producto de la utilización de este modelo de desarrollo, radicó en la posibilidad de efectuar modificaciones en uno o varios servicios del soporte, sin afectar todo el sistema en general. A continuación, se

describe el modelo en V.

Modelo en V

Debido a que cada vez se tiene mayor información referente a las áreas involucradas en un proyecto, este tendría la capacidad de crecer, por lo que el proyecto se encontraría en una evolución continúa en la que incluso se podría llegar a tal grado que los usuarios pudieran aportar información después de un proceso de verificación.

Es por esta razón que para la realización de la presente tesis se optó por utilizar el modelo en V, ya que en nuestra opinión toma las mejores características de las metodologías iterativas como de un modelo de prototipos, condensando todo en una ruta de implementación y en una de verificación.

El modelo en V define ocho etapas principales [German Army, 1992] como son:

- **Ingeniería de requisitos:** se requiere saber cuál es la temática y para ello se recaba la suficiente información en los distintos medios, como artículos científicos.
- **Validación del sistema:** de la información recabada se hará una validación para saber si es correcta y se continuará con el siguiente módulo.
- **Diseño del sistema:** esta fase tiene como objetivo definir la estructura o arquitectura en la cual se pretende que el usuario interactúe con el sistema.
- **Implementación:** en esta fase se aplicarán las pruebas que se hicieron en la etapa del diseño del sistema.
- **Diseño del software:** en esta fase se hace hincapié en las tecnologías necesarias para el cumplimiento de los requisitos del sistema.
- **Verificación del software:** en esta fase se comprobará si las tecnologías utilizadas son lo suficientemente eficaces y óptimas para el correcto funcionamiento del software.
- **Codificación:** en esta fase se implementará el código fuente de los prototipos, también se hará el uso de la programación en un lenguaje ya definido.
- **Operación y mantenimiento:** del sistema ya terminado se comprobará si este funciona correctamente y si cumple con todos los requisitos.

Como puede verse en la Figura 4.1 el modelo V implica una forma de trabajo donde se disponen de pocos recursos, pero da la posibilidad de una correcta integración, verificación y, en su caso, una pronta corrección.

A diferencia de otras metodologías, tiene pocas iteraciones en realidad pero las mismas son bien planeadas y de calidad. El modelo, a pesar de ser compacto, cumple con todas las fases de trabajo de un proyecto serio y que tenga un tamaño desde pequeño hasta grande ya que cuando el proyecto crece demasiado (más de 500,000 líneas de código) debe cambiarse por el proceso unificado [Schach, 2005].

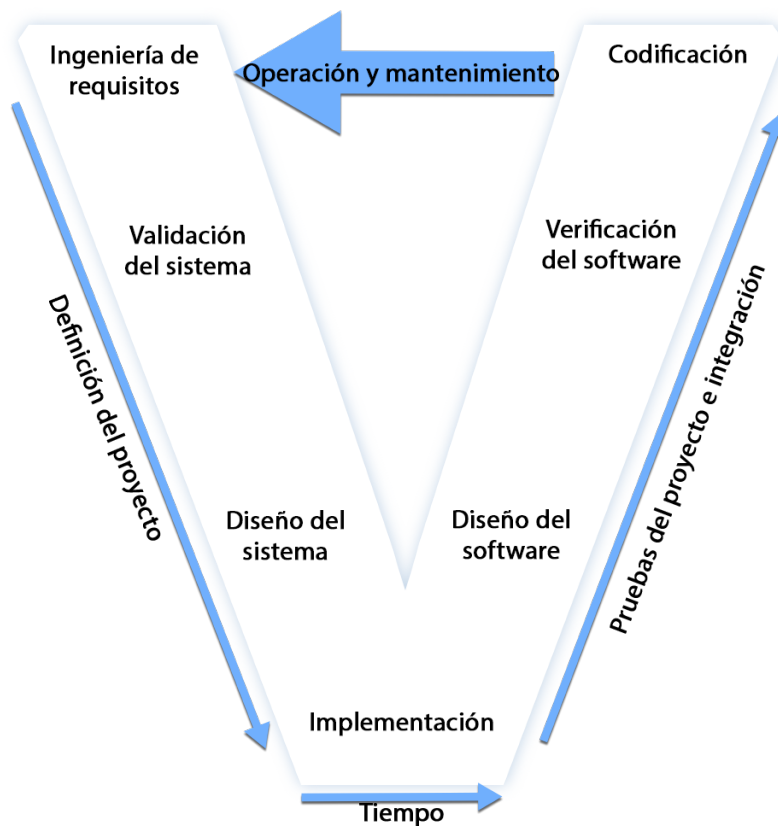


Figura 4.1: Modelo en V

4.2. Herramientas de desarrollo empleadas

En esta sección se describen las herramientas seleccionadas para llevar a cabo la implementación del marco de desarrollo. Primeramente, se describen las plataformas elegidas (ver Subsección 4.2.1), así como el lenguaje de programación utilizado (ver Subsección 4.2.2).

4.2.1. Plataformas móviles

Una plataforma es un sistema que sirve como base para hacer funcionar determinados módulos de hardware o de software con los que es compatible. Dichos sistemas son los programas encargados de la administración de los recursos de la computadora o de dispositivos móviles: CPU, memoria, y dispositivos de entrada y salida. Una característica principal en estos sistemas es que deben mantener un buen rendimiento. Esto es, rapidez y eficiencia deben tenerse presentes en su diseño, desarrollo y funcionamiento.

De acuerdo a un estudio realizado por la IDC (*International Data Corporation*) [IDC, 2016], hasta el segundo cuatrimestre del año 2016 las plataformas móviles Android (86.2%) y iOS (12.9%) (ver Figura 4.2) son las más usadas, abarcando el 98.9% del total de dispositivos móviles en el mercado (ver Tabla 4.1).

En la Figura 4.3 se puede ver la distribución de la cuota de mercado de Android hasta el mes de mayo de 2016. La versión de Android 4.4 (Kitkat) tenía una cuota de mercado del 32.5%.

Período	Android	iOS	Windows Phone	BlackBerry OS	Otros
2016Q2	86.2%	12.9%	0.6%	-	0.2%
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012Q2	69.3%	16.6%	3.1%	4.9%	6.1%

Tabla 4.1: Uso de sistemas operativos [IDC, 2016]

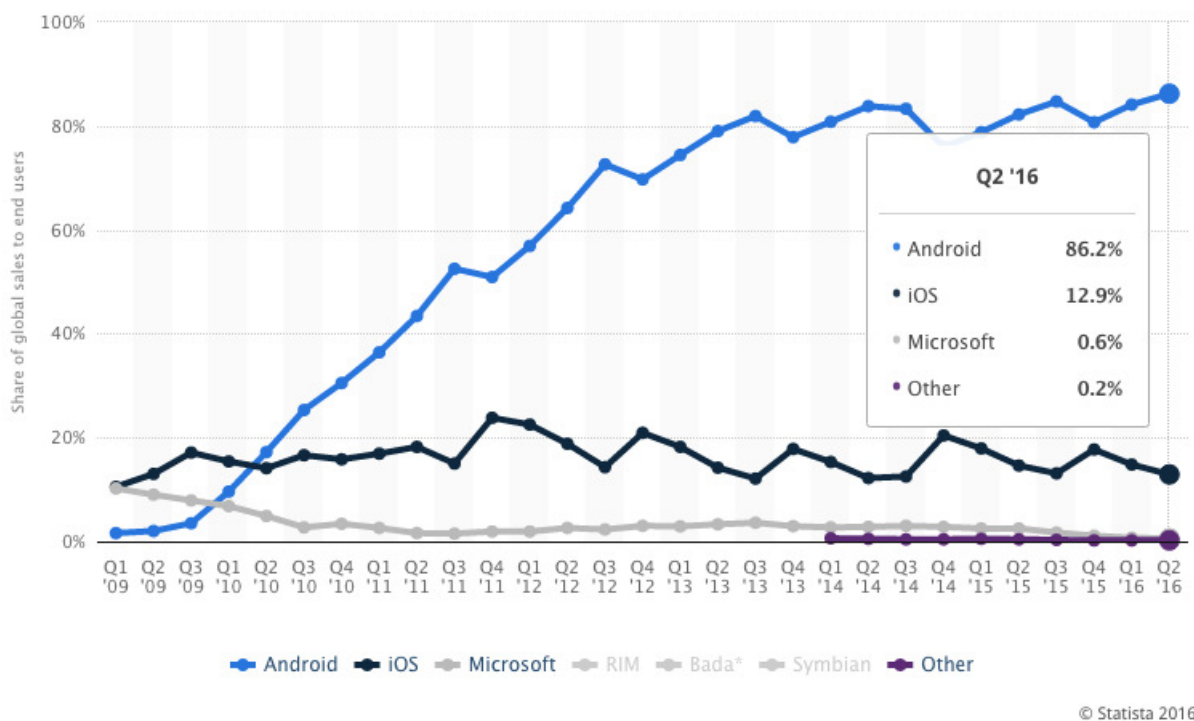


Figura 4.2: Gráfica de sistemas operativos por plataforma [Statista, 2016b]

Las cifras se basan en el número de dispositivos Android que han accedido a la tienda de Google Play.

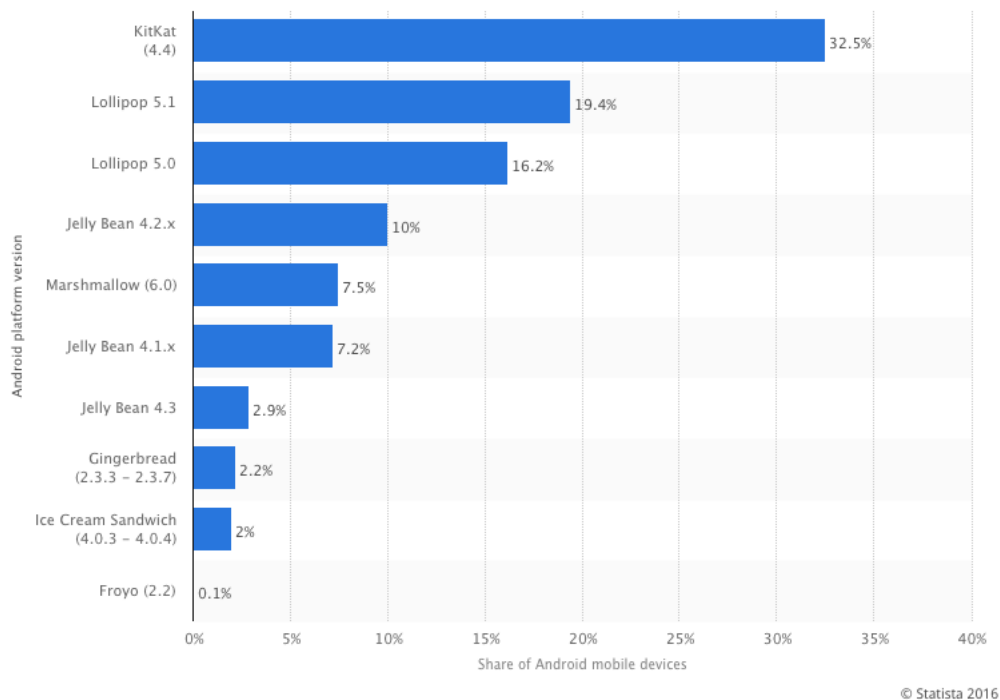


Figura 4.3: Gráfica del sistema operativo Android por versión [Statista, 2016a]

En la Figura 4.4 se muestra la participación de las versiones del sistema operativo iOS de Apple que se ejecutan en iPhones y iPads con el acceso a la tienda de aplicaciones de Apple, en mayo de 2016. iOS 9 se ejecuta en el 84 % de los dispositivos de Apple que accede a la tienda de aplicaciones de dicha compañía.

4.2.2. Paradigma de programación

Como se mencionó en el subsección 4.2.1, los dispositivos móviles que tienen más presencia en el mercado, son los que cuentan con sistemas operativos Android e iOS, para los cuales seleccionaremos el paradigma de programación que más se adecue a nuestras necesidades de desarrollo. En la Tabla 4.2 podemos observar los lenguajes en los que se puede llevar a cabo la implementación del marco de desarrollo.

Un estudio realizado por Developer Economics [Developer Economics, 2014] para desarrolladores explica los lenguajes de programación más utilizados por plataforma (sistema operativo móvil), pero en este caso podemos observar dos plataformas líderes en el mercado, Android que cuenta con un 42 % de la prioridad de todos los desarrolladores encuestados y iOS que cuenta con un 32 % contra 10 % de Windows Phone y el 3 % de BB10.

Podemos ver en la Figura 4.5 que Android concentra el 58 % para el lenguaje JAVA e iOS presenta el 53 % para Objective-C; esto nos da a entender que estos dos lenguajes de programación son los que mejor se adecuan para la implementación del marco de desarrollo. Cabe mencionar que también se tiene el 13 % para Android y 12 % para iOS de HTML/CSS/JavaScript. Sin

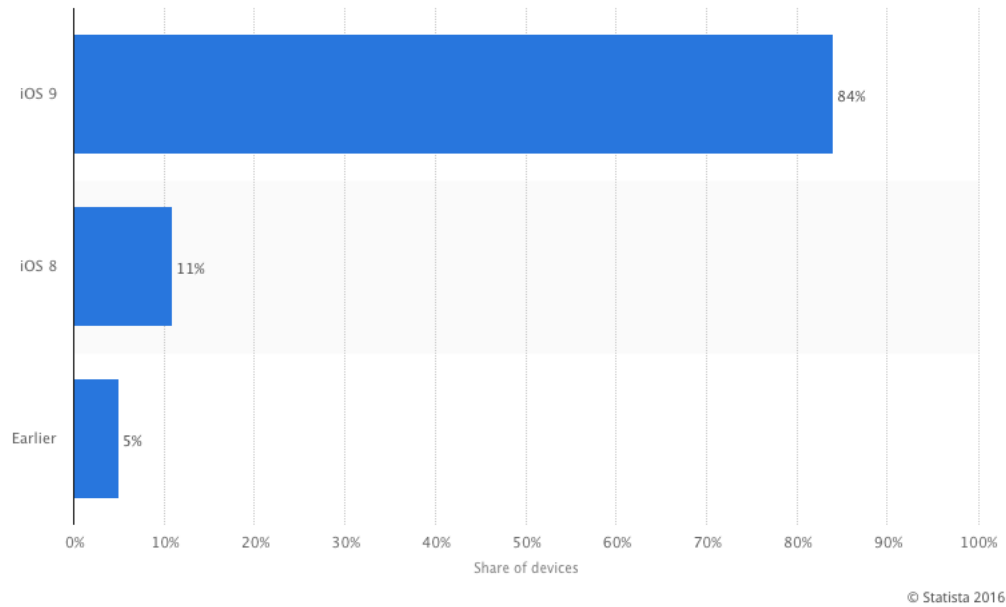


Figura 4.4: Gráfica del sistema operativo iOS por versión [Statista, 2016c]





Sistema Operativo	Logo	Lenguaje
Windows Phone		C#, C/C++, HTML/CSS/JavaScript
Android		Java, C/C++, HTML/CSS/JavaScript
iOS		Swift, Objective-C, C/C++, HTML/CSS/JavaScript
BlackBerry OS		Java, C/C++, HTML/CSS/JavaScript

Tabla 4.2: Lenguajes de programación móvil

embargo, ya que proponemos un sistema en el cual se trabajará con múltiples servicios, este lenguaje se limita a la ejecución de un proceso por aplicación. Por lo tanto, solo se utilizaron los lenguajes mencionados anteriormente en esta tesis.

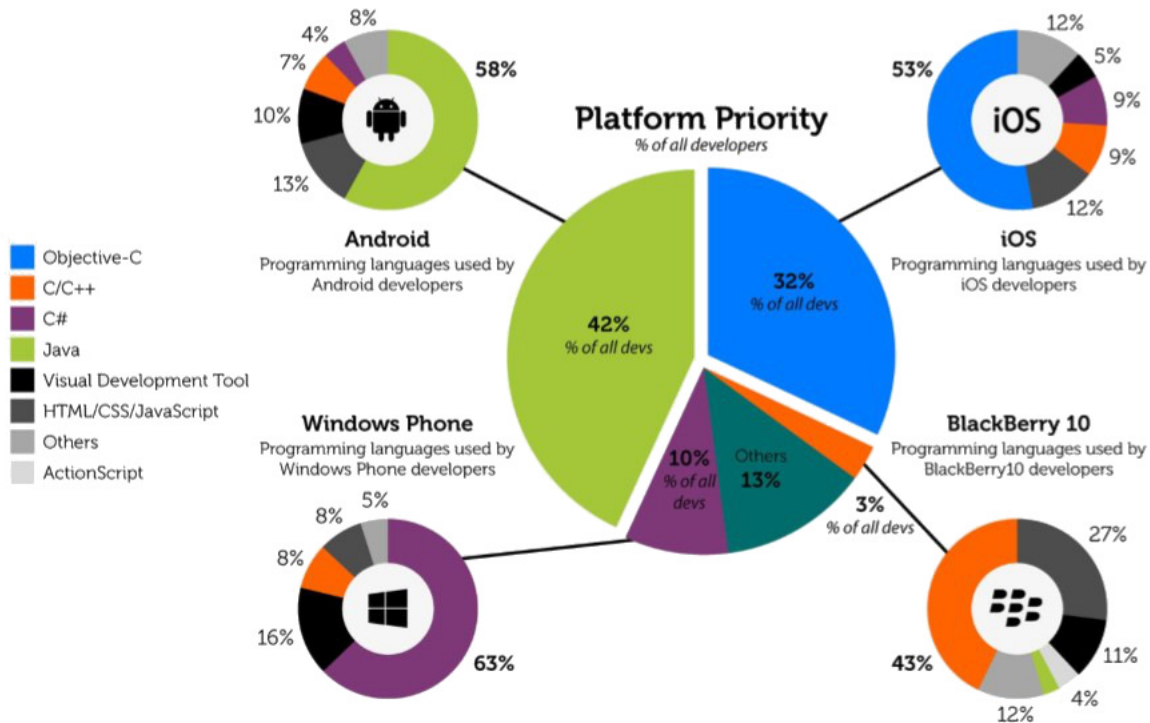


Figura 4.5: Gráfica de lenguajes de programación por plataforma [Developer Economics, 2014]

Estos dos lenguajes, Objective-C y Java, están basados en el paradigma de programación orientado a objetos (POO). Una definición de este paradigma es la siguiente: “Es pues una filosofía de desarrollo y empaquetamiento de programas que permite crear unidades funcionales extensibles y genéricas, de forma que el usuario las pueda utilizar según sus necesidades y de acuerdo con las especificaciones del problema a desarrollar” [Amo et al., 2005].

Entonces, la programación orientada a objetos es un método que permite modelar abstracciones de objetos, tanto físicos como intangibles, para ciertos problemas acotados por un conjunto de reglas de negocio. Con cada objeto que se modela, se define una serie de atributos para esa entidad y un comportamiento.

Para la implementación del marco de desarrollo propuesto, se requirieron conocimientos técnicos del lenguaje de programación Java para el sistema operativo Android y Objective-C para el sistema operativo iOS, además del aprovechamiento inteligente de las funciones y capacidades que proveen los sensores integrados en los dispositivos móviles. Esto nos permitió construir un sistema que cubriera cada uno de los objetivos especificados en la sección 1.5. Un desarrollo adecuado bajo estas plataformas siempre es posible, siguiendo las pautas y recomendaciones descritas en la guía de desarrolladores en línea de Android e iOS.

4.3. Implementación del marco de desarrollo

Aunque la arquitectura y el diseño descritos en el Capítulo 3 están implementados para la plataforma Android e iOS, es posible utilizar el mismo modelo para futuras implementaciones en otros lenguajes o dispositivos móviles. A continuación, presentamos los detalles de implementación de cada uno de los servicios que provee el marco de desarrollo, así como la forma en cómo se utilizan y diversos puntos clave que nos permitieron alcanzar los objetivos planteados.

4.3.1. Servicio de descubrimiento

Como se mencionó en la Sección 3.2.2, el **servicio de descubrimiento** es el encargado de reconocer los dispositivos que están ejecutando una sesión. La identificación se hace por medio de mensajes tipo de *broadcast* a través de la red local. El mensaje está compuesto por datos de identificación que son extraídos del dispositivo, tales como:

- **IP:** la dirección IP del dispositivo que está asignada en una red local.
- **Puertos TCP y UDP:** los puertos de comunicación libres para poder establecer una comunicación con los otros servicios dedicados.
- **Estado de la aplicación:** el estado en el que se encuentra la sesión (i.e., la sesión está activa o la sesión está por finalizar), ya sea que el usuario se encuentra activo o ha abandonado la sesión.
- **Nombre del dispositivo:** según sea el tipo de dispositivo se obtiene el nombre (e.g., iPhone, iPad, iPod Touch, Samsung, Sony).
- **Sistema Operativo:** el tipo de sistema operativo (Android o iOS según sea el caso) en el que se está ejecutando una sesión.

Este servicio tiene por dos principales funciones: **propagación** y **recolección** de datos. Para ambas funciones utilizamos la Notación de Objetos de JavaScript (mejor conocido como JSON por sus siglas en inglés) tanto para Java como para Objective-C, el cual es un formato ligero de intercambio de datos y ampliamente usado, que nos permite convertir objetos en texto para agilizar el envío de paquetes por la red.

La función de propagación tiene como objetivo informar la presencia del dispositivo a otros dispositivos móviles a través de la red local. El algoritmo implementado está en la clase `DiscoveryPublishService`, el cual se muestra en el Algoritmo 1.

Del algoritmo 1, `PUBLISHING_DELAY` y `LEAVING_DELAY` denotan el tiempo en milisegundos de propagación. La primera variable indica el tiempo en el que se deben propagar los datos por la red; de igual forma, la segunda variable indica el tiempo en el que se debe propagar el mensaje de abandono de la sesión. Estas dos variables pueden ser configuradas a decisión del desarrollador.

La variable `isAvailable` puede cambiar según sea el caso, si la sesión está en ejecución el valor que le corresponde es *true* y si la sesión ya no se ejecutará, antes de terminar su proceso esta instancia enviará un mensaje de abandono a los demás dispositivos y la variable cambiará a *false*.

Algoritmo 1 Propagación de los datos de identificación del dispositivo para Android e iOS

```

1: procedure IDENTIFICATIONDATADIFUSION(IP, puertoTCP, puertoUDP)
2:   Agregar (“IP”, IP) al objeto JSONObject
3:   Agregar (“TCP”, puertoTCP) al objeto JSONObject
4:   Agregar (“UDP”, puertoUDP) al objeto JSONObject
5:   Agregar (“isAvailable”, true) al objeto JSONObject
6:   Agregar (“Device”, nombreDispositivo) al objeto JSONObject
7:   Agregar (“typeDevice”, tipoDispositivo) al objeto JSONObject
8:
9:   while no se abandone la sesión do
10:     Propagar el objeto JSONObject mediante mensajes de tipo broadcast (UDP)
11:     Pausar el servicio PUBLISHING_DELAY milisegundos
12:   end while
13:
14:   Agregar (“isAvailable”, false) al objeto JSONObject
15:
16:   for i:=1 hasta LEAVING_ALERT_MESSAGES do
17:     Propagar el objeto JSONObject modificado mediante mensajes de tipo broadcast
18:     Pausar el servicio LEAVING_DELAY milisegundos
19:   end for
20: end procedure

```

El algoritmo de recolección (ver Algoritmo 2) permite a la instancia de la sesión obtener los datos de identificación que se transmiten por medio del algoritmo de propagación y tiene dos funciones principales:

- **Agregar dispositivo:** agrega los datos de identificación a lista de dispositivos DISCOVERED_DEVICES.
- **Eliminar dispositivo:** elimina los datos de identificación de las listas DISCOVERED_DEVICES, GROUP_LIST y CONNECTED_DEVICES.

A continuación describimos el contenido de las listas mencionadas anteriormente:

- **DISCOVERED_DEVICES:** ésta lista contiene la información de identificación de un dispositivo como los siguientes datos:
 - **IP:** del dispositivo móvil.
 - **UP:** si la sesión ésta activa el valor es *true* y si la sesión ha finalizado es *false*.
 - **TCP:** contiene el número del puerto de conexión TCP.
 - **UDP:** contiene el número del puerto de conexión UDP.
 - **DEVICE:** el nombre del dispositivo móvil (e.g., iPhone, Samsung, Sony).
 - **TYPEDEVICE:** es el tipo de sistema operativo usado por el dispositivo móvil (e.g., Android o iOS).

- **GROUP_LIST**: ésta lista contiene la información usada por el marco de desarrollo como los siguientes datos:
 - **IP**: del dispositivo móvil.
 - **TCP**: contiene el número del puerto de conexión TCP.
 - **UDP**: contiene el número del puerto de conexión UDP.
 - **SPECS**: contiene algunas características del dispositivo (e.g., tipo de sistema operativo, el modelo del dispositivo, el tamaño de la pantalla, el tipo de sensores).
- **CONNECTED_DEVICES**: ésta lista contiene la IP de cada dispositivo que este acoplado.

Algoritmo 2 Recolección de los datos de identificación del dispositivo para Android e iOS

```

1: procedure IDENTIFICATIONDATACOLLECTION(IP propia)
2:   while no se abandone la aplicación do
3:     Estar a la espera de paquetes de tipo broadcast (operación bloqueante de sockets)
4:
5:     if paquete recibido then
6:       Extraer los valores del objeto JSONObject recibido
7:       if “IP” not IP propia then
8:         if “isAvailable” = true then
9:           Agrupar los valores extraídos en un HashMap
10:          Agregar el HashMap en la lista DISCOVERED_DEVICES si es que
              no existía ya
11:          Notificar que un dispositivo ha sido reconocido y agregado
12:        else
13:          if dispositivo se encuentra en DISCOVERED_DEVICES then
14:            Eliminarlo de DISCOVERED_DEVICES
15:          else if dispositivo se encuentra en GROUP_LIST then
16:            Eliminarlo de GROUP_LIST
17:          else if dispositivo se encuentra en CONNECTED_DEVICES then
18:            Eliminarlo de CONNECTED_DEVICES
19:          end if
20:          Notificar que un dispositivo ha abandonado la sesión y no está
              disponible
21:        end if
22:      end if
23:    end if
24:  end while
25: end procedure

```

El algoritmo de recolección de los datos de identificación hace uso de la clase `SharedResources` (ver Sección 3.5) la cual almacena los datos de identificación en una lista ordenada (i.e., agrupados en objetos `HashMap` para Java y para Objective-C `NSMutableDictionary`), donde cada entrada es representada por la información del dispositivo que se ha identificado.

Como se mencionó anteriormente, en el Algoritmo 2 de recolección de los datos de identificación ésta a la espera de la recepción de los paquetes de información. A continuación se describen los pasos que sigue este algoritmo:

- Cuando se reciben los datos (línea 5) se extraen del objeto JSON.
- Se comprueba que los datos recibidos no sean del mismo emisor por medio de la IP del dispositivo (línea 7).
- Se verifica la disponibilidad (línea 8) y agrega los datos recibidos a la lista `DISCOVERED_DEVICES` si no esta.
- Las líneas 11 y 20 corresponden al método *listener* para Android y *NSNotification* para iOS, el cual se encarga de notificar al usuario de la presencia de un dispositivo o el abandono.
- Las líneas 14, 16 y 18 corresponden a las listas donde se puede encontrar almacenada la identificación de un dispositivo (`DISCOVERED_DEVICES`, `GROUP_PLIST`, `CONNECTED_DEVICES`).

Cabe mencionar que las clases `DiscoveryPublishService` y `DiscoveryCollectService` heredan de la clase abstracta `AsyncTask` para el caso de Android y para iOS *Grand Central Dispatch* (GCD por sus siglas en inglés) que es una tecnología desarrollada por Apple, para optimizar el soporte de las aplicaciones para procesadores de varios núcleos, i.e., ambas tareas (identificación y reconocimiento) se ejecutan en hilos secundarios del hilo principal. La razón primordial de ejecutarlas en diferentes hilos se debe a que son susceptibles de tener operaciones bloqueantes (e.g., las funciones `sleep()` y `usleep()`), las cuales tendrían un efecto de retardo que afectaría el rendimiento. Como alternativa al uso de operaciones bloqueantes, se pueden utilizar clase que implementen la interfaz `Runnable` para Android y `Dispatch` para iOS.

4.3.2. Servicio de comunicación

Este servicio tiene las siguientes funciones: crear sesiones colaborativas, remodelar y actualizar el espacio de trabajo como resultado del acoplamiento de los dispositivos móviles. Este servicio esta dividido en las siguientes fases: gestos de acoplamiento, configuración de gestos de acoplamiento, propagación de un intento de acoplamiento y captura de un intento de acoplamiento.

Como se mencionó en la Sección 3.2.3, un gesto puede hacer uso del hardware del dispositivo (e.g., *touch*, acelerómetro) que es el medio a través del cual un usuario puede acoplar dos espacios de trabajo. Esto le indica a la sesión que realice la acción de acoplamiento, por medio de la interacción de las siguientes clases: `LongGesture`, `SwipeGesture` y `AccelerometerGesture`.

- **LongGesture:** esta clase se encarga de determinar cuándo el usuario ha realizado un click largo sobre el espacio del trabajo por medio del *touch* (ver Figura 4.6). Para ello, basta con llamar el método `verifyLongClick(MotionEvent or UIGestureRecognizer)` en el evento `onTouch(View, MotionEvent)` para Android y para iOS con `NSNotification(UIGestureRecognizer)` del objeto que representa al espacio de trabajo. El objeto que se envía como parámetro (`MotionEvent or UIGestureRecognizer`) contiene la información requerida para contextualizar un gesto, i.e., para determinar si un click largo es válido o no. Mediante el algoritmo 3 se muestra parte de la clase `LongGesture`.

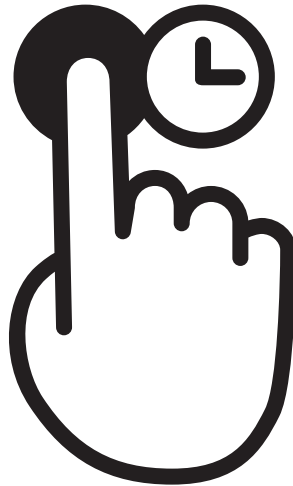


Figura 4.6: Click largo

- **SwipeGesture**: esta clase se encarga de la evaluación del gesto de arrastre, comparando el primer y último punto de contacto del dedo con la pantalla; si la diferencia entre ambos es lo suficientemente grande, indicará que el gesto se realizó correctamente. El gesto de arrastre puede ser en las direcciones arriba, abajo, derecha e izquierda como se muestra en la Figura 4.7. Mediante el Algoritmo 4 se muestra parte de la clase **SwipeGesture**.
- **AccelerometerGesture**: esta clase se encarga de la evaluación del gesto de inclinación usando el acelerómetro que incorporan los dispositivos móviles, obteniendo solo los valores de los ejes X y Y. La medición puede ser entre los valores positivos y negativos de los ejes como se muestra en la Figura 4.8. El Algoritmo 5 muestra parte de la clase **AccelerometerGesture**.

Como se mencionó anteriormente en el Algoritmo 3, la clase **LongGesture** se encuentra a la espera de una acción del usuario al presionar la pantalla del dispositivo. A continuación se describen los pasos que sigue este algoritmo:

- Se extraen los eventos del parámetro **MotionEvent** para Android o **UIGestureRecognizer** para iOS (línea 2), los cuales brindan información acerca de los puntos de los ejes X o Y referentes a la posición del dedo del usuario en la pantalla (líneas 4 y 5).
- El registro de acontecimientos de tipo *touch* provee múltiples flujos constantes de eventos de movimiento [Android, 2016]. No obstante, solamente tres de ellos son relevantes en el contexto de la detección de un click largo, los cuales están denotados por las siguientes constantes: **ACTION_DOWN** (cuando un dedo entró en contacto con la pantalla), **ACTION_MOVE** (cuando un dedo es arrastrado por la pantalla) y **ACTION_UP** (cuando un dedo se ha levantado de la pantalla). Para el caso de iOS existen seis estados que pueden ser de ayuda para detectar los cambios de un click largo, por medio de la clase **UIGestureRecognizerState**. De estos estados solo usaremos tres y están denotados por las siguientes constantes: *Began* (cuando un dedo entró en contacto con la pantalla), *Changed* (cuando un dedo es arrastrado por la pantalla), *Ended* (cuando un dedo se ha levantado de la pantalla) (líneas 3, 8 y 12).

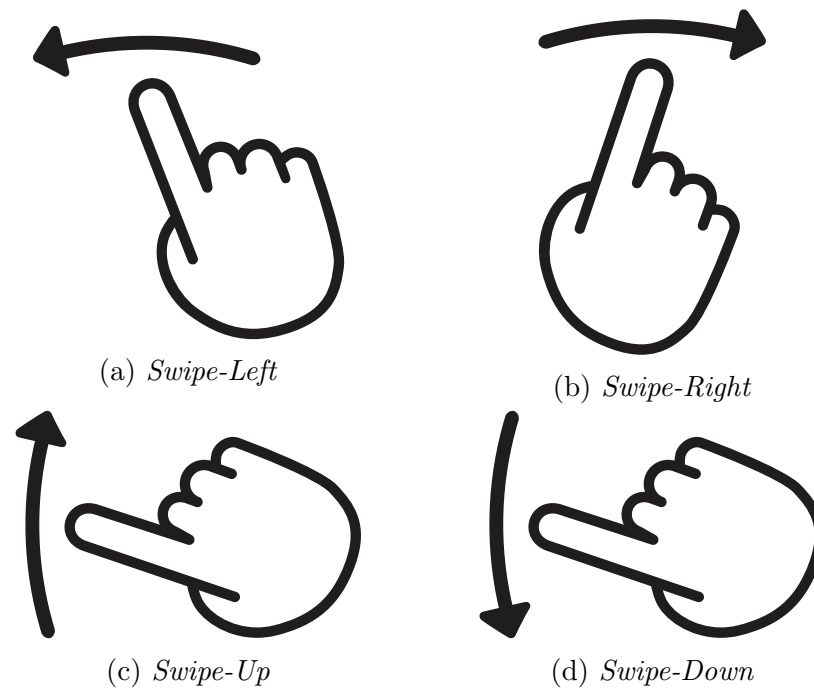


Figura 4.7: Gestos de arrastre

Algoritmo 3 Detección de clicks largos para Android e iOS

```

1: function VERIFYLONGCLICK(MotionEvent or UIGestureRecognizer)
2:   event ← Extraer evento del parámetro MotionEvent (Android) o
                                     UIGestureRecognizer (iOS)
3:   if event fue ACTION_DOWN o BEGAN then
4:     x ← Posición x del evento (obtenida de MotionEvent or UIGestureRecognizer)
5:     y ← Posición y del evento (obtenida de MotionEvent or UIGestureRecognizer)
6:     longClickDetected ← falso
7:     Poner un hilo en espera durante LONG_CLICK_PROC_TIMER milisegundos
8:   else if event fue ACTION_MOVE o CHANGED then
9:     if Si el dedo se movió en x o en y más de MOVEMENT_GAP pixeles then
10:      Cancelar el hilo que se puso en espera en la línea 7
11:    end if
12:   else if event fue ACTION_UP o ENDED then
13:     Cancelar el hilo que se puso en espera en la línea 7
14:   end if
15:   return longClickDetected
16: end function
17: procedure TAREAS DEL HILO DE LA LÍNEA 7
18:   longClickDetected ← verdadero
19:   Dar retroalimentación acústica y háptica
20: end procedure

```

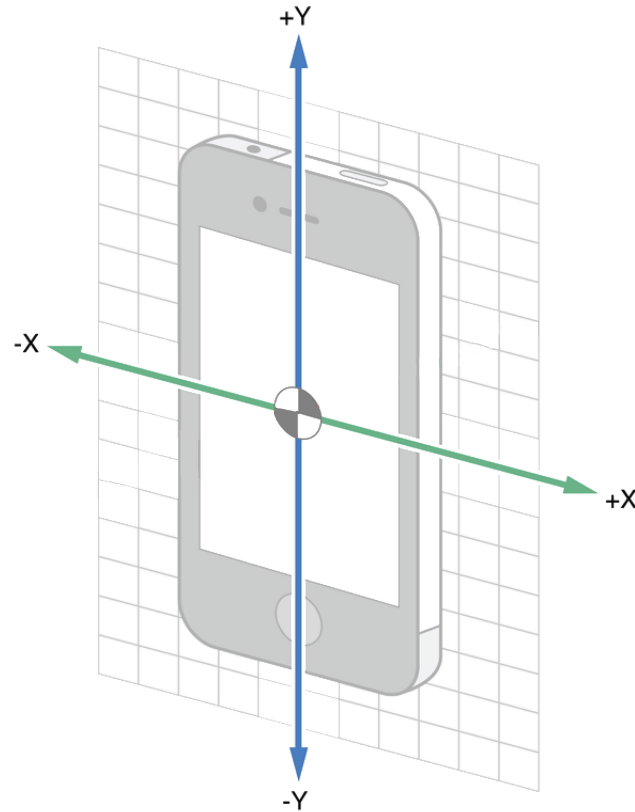


Figura 4.8: Ejes X y Y del acelerómetro de un dispositivo móvil

- Para que se pueda considerar un click largo correcto, se debe validar durante un tiempo (`LONG_CLICK_PROC_TIMER` milisegundos), por medio de un hilo que se pone en espera y que se muestra en la línea 7. Si el hilo se ejecuta, entonces quiere decir que el usuario mantuvo su dedo presionado en la pantalla del dispositivo por `LONG_CLICK_PROC_TIMER` milisegundos (línea 7) y que no realizó un arrastre con su dedo en los ejes X ni Y más de `MOVEMENT_GAP` pixeles (línea 9), además de que no levantó el dedo (línea 12), i.e., que dejó su dedo prácticamente en el mismo lugar durante un tiempo determinado. Una vez arrancado el hilo, este hilo avisa al usuario mediante retroalimentación háptica y acústica que el click largo ha sido reconocido y aceptado (línea 19).
- La línea 6 reinicia el reconocimiento de un click largo cada vez que se coloca un dedo sobre la pantalla (i.e., cuando se reconoce un evento de tipo `ACTION_DOWN` o `ENDED`).
- La línea 13 cancela el hilo siempre que se levante el dedo de la pantalla. Dicha cancelación solamente tendrá efecto si el hilo no ha sido lanzado, i.e., si no han transcurrido más de `LONG_CLICK_PROC_TIMER` milisegundos.
- La línea 9 verifica que el usuario no arrastre su dedo desde la posición en la que comenzó el gesto. La constante `MOVEMENT_GAP` establece un margen de movimiento límite, ya que al mantener un dedo sobre la pantalla del dispositivo, el usuario estará tocando varios puntos al mismo tiempo.

Algoritmo 4 Detección de un gesto de arrastre para Android e iOS

```

1: function VALIDATEGESTURE(MotionEvent or UIGestureRecognizer, validLongClick)
2:   event ← Extraer evento del parámetro MotionEvent
3:   if event fue ACTION_DOWN o BEGAN then
4:     x ← Posición x del evento (obtenida de MotionEvent (Android) or
                                     UIGestureRecognizer (iOS))
5:     y ← Posición y del evento (obtenida de MotionEvent (Android) or
                                     UIGestureRecognizer (iOS))
6:   else if event fue ACTION_UP o ENDED and validLongClick = verdadero then
7:     direction ← "null"
8:     deltaX ← x - posición x del evento (obtenida de MotionEvent (Android) or
                                           UIGestureRecognizer (iOS))
9:     deltaY ← y - posición y del evento (obtenida de MotionEvent (Android) or
                                           UIGestureRecognizer (iOS))
10:    if |deltaX| > MIN_SWIPE_PX_DISTANCE then
11:      if deltaX < 0 then
12:        direction ← "Derecha"
13:      else
14:        direction ← "Izquierda"
15:      end if
16:    else if |deltaY| > MIN_SWIPE_PY_DISTANCE then
17:      if deltaY < 0 then
18:        direction ← "Abajo"
19:      else
20:        direction ← "Arriba"
21:      end if
22:    end if
23:    if direction no está vacía and LOCKED_REQUESTS = falso then
24:      LOCKED_REQUESTS ← verdadero
25:      Estar a la escucha de un intento de acoplamiento externo
26:      Difundir intento de acoplamiento
27:      return verdadero
28:    end if
29:  end if
30:  return falso
31: end function

```

El algoritmo 4 valida un gesto de arrastre mediante movimientos que el usuario hace al interactuar con el *touch* de la pantalla del dispositivo. Estos movimientos pueden ser en cualquier dirección sobre la pantalla, no obstante solamente se consideran los movimientos lineales como válidos para nuestro marco de desarrollo. A continuación se describen los pasos que sigue el algoritmo:

- En la línea 3 se obtienen los eventos `ACTION_DOWN` o `BEGAN` cuando el usuario inicia el gesto de arrastre, ya sea en el eje `x` o en el `y` y se almacenan los valores de los eventos mencionados en estas variables (líneas 4 y 5).
- En la línea 6 se obtienen los eventos `ACTION_UP` o `ENDED` cuando el usuario termina el gesto de arrastre, ya sea en el eje `x` o en el `y`. Después se hace una resta entre el valor del punto final obtenido y del punto inicial obtenido en el eje `x` o en el `y` y se almacenan los valores en las variables `deltaX` y `deltaY` (líneas 8 y 9).
- Posteriormente, en la línea 10 se valida si la distancia obtenida en `deltaX` es mayor a `MIN_SWIPE_PX_DISTANCE`. Si se cumple esta condición y si el valor de `deltaX` es negativo, entonces significa que el gesto de arrastre es hacia la “Derecha” (línea 12); en caso contrario es hacia la “Izquierda” (línea 14).
- En la línea 16 se valida si la distancia obtenida en `deltaY` es mayor a `MIN_SWIPE_PY_DISTANCE`. Si se cumple esta condición y si el valor de `deltaY` es negativo, significa que el gesto de arrastre es hacia “Abajo” (línea 18); en caso contrario es hacia la “Arriba” (línea 20).
- En la línea 24 se introduce el recurso compartido `LOCKED_REQUESTS`, el cual básicamente se comporta como un candado para evitar que se difunda otro intento de acoplamiento, antes de que se termine de procesar el que está en curso. Este candado es indispensable, ya que un usuario podría realizar dos gestos de acoplamiento correctos, consecutivos y con direcciones opuestas, sin dar la oportunidad de que se procese un intento externo (línea 25).

El Algoritmo 5 valida un gesto de inclinación mediante movimientos de inclinación que el usuario realiza en los ejes `x`, `y` y `z` del dispositivo móvil. Estos movimientos pueden ser en cualquier dirección (e.g., `X`, `-X`, `Y`, `-Y`). No obstante, solamente se consideran los movimientos en ejes `X` o `Y`, ya que el eje `Z` no brinda información relevante para poder seleccionar un lado del dispositivo. A continuación se describen los pasos que sigue este algoritmo:

- En la línea 5, se deshabilita la rotación automática, de la pantalla previamente al muestreo del acelerómetro, ya que de no hacerlo, el usuario podría provocar que la pantalla rote al inclinar el dispositivo, afectando el resultado final del gesto.
- En la línea 7 se habilita el acelerómetro y se toman muestras durante `SAMPLING_TIME` milisegundos.
- En la línea 13 se evalúa si las muestras tomadas de los ejes `X` y `Y` del acelerómetro. Dichas muestras de las aceleraciones registradas no deberán sobrepasar el valor de la gravedad cuando el dispositivo se mantiene inclinado e inmóvil. Por lo tanto, debemos

mantener un valor de muestras coherente cuando se realicen movimientos bruscos por accidente. Se descartan todas aquellas muestras que sobrepasen la cota de aceleración máxima `ABOVE_THIS_IS_NOISE` m/s^2 (líneas 16 y 17), cuyo valor es de $9.8 m/s^2$ por omisión.

- Si `meanX` es menor que `MIN_VALID_INCLINATION`, entonces significa que el dispositivo permaneció inclinado más tiempo hacia la posición “Derecha” (línea 23); en caso contrario la posición es a la “Izquierda” (línea 25).
- Si `meanY` es menor que `MIN_VALID_INCLINATION`, entonces indica que el dispositivo permaneció inclinado más tiempo hacia la posición “Abajo” (línea 29); en caso contrario la posición es a la “Arriba” (línea 31).
- Una vez que se ha seleccionado algún lado del dispositivo, se procede a difundir un intento de acoplamiento (línea 38).

Configuración de gestos de acoplamiento

Para poder hacer uso de un gesto de acoplamiento específico, se puede configurar en la clase `WorkTogetherSetup` (Algoritmo 6), esta clase recibe dos parámetros que habilitan los gestos:

- `enableSwipeGesture`: si el valor de esta variable es positivo, se activará el gesto de arrastre (línea 7).
- `enableAccelerometerGesture`: si el valor de esta variable es positivo, se activará el gesto de inclinación (línea 10).

Propagación de un intento de acoplamiento

Una vez que se ha realizado una combinación de un gesto en el dispositivo (e.g., un click largo y gesto de arrastre o un click largo y gesto de inclinación) se puede obtener el lado en el que queremos procesar un intento de acoplamiento. Mediante el Algoritmo 7, mostramos como está compuesto dicho intento de acoplamiento.

Un intento de acoplamiento está compuesto por la dirección IP del dispositivo, los puertos TCP y UDP en donde se encuentra operando el servicio actualizaciones, el lado seleccionado a través de un gesto de acoplamiento y las especificaciones del dispositivo extraídas del parámetro `features` (líneas 2-6). Dichas especificaciones se obtienen a través de la clase `DeviceFeatures` y deben ser recabadas al inicio de la aplicación y cuando se rota la pantalla (a través del método `onConfigurationChanged`). El método `setAllFeatures` es el encargado de recabar los siguientes datos:

- Dimensiones y densidad de pixeles de la pantalla.
- Sensores disponibles.
- Interfaces de comunicación.
- Modelo del dispositivo.

Algoritmo 5 Gesto de inclinación para Android e iOS

```

1: function VALIDATEGESTURE(MotionEvent or UIGestureRecognizer, validLongClick)
2:   event ← Extraer evento del parámetro MotionEvent (Android) or
                                     UIGestureRecognizer (iOS)
3:   if event fue ACTION_UP or ENDED and
       validLongClick = verdadero and isRunning = falso then
4:     isRunning ← verdadero
5:     Deshabilitar la rotación automática de la pantalla
6:     rotation ← Rotación de la pantalla
7:     Habilitar el acelerómetro y muestrear durante SAMPLING_TIME milisegundos
8:     meanX ← Aceleración en el eje de las x de la primera muestra
9:     meanY ← Aceleración en el eje de las y de la primera muestra
10:    for all muestras do
11:      x ← Aceleración en el eje de las x
12:      y ← Aceleración en el eje de las y
13:      if |x| > ABOVE_THIS_IS_NOISE or |y| > ABOVE_THIS_IS_NOISE then
14:        Ignorar esta muestra
15:      else
16:        meanX ← (meanX + x)/2
17:        meanY ← (meanY + y)/2
18:      end if
19:    end for
20:    direction ← "null"
21:    if |meanX| > |meanY| then
22:      if meanX < -MIN_VALID_INCLINATION then
23:        direction ← "Derecha"
24:      else if meanX > MIN_VALID_INCLINATION then
25:        direction ← "Izquierda"
26:      end if
27:    else
28:      if meanY < -MIN_VALID_INCLINATION then
29:        direction ← "Abajo"
30:      else if meanY > MIN_VALID_INCLINATION then
31:        direction ← "Arriba"
32:      end if
33:    end if
34:    Habilitar rotación automática de la pantalla
35:    if direction no está vacía and LOCKED_REQUESTS = falso then
36:      LOCKED_REQUESTS ← verdadero
37:      Estar a la escucha de un intento de acoplamiento externo
38:      Difundir intento de acoplamiento
39:      isRunning ← falso
40:      return verdadero

```

```

41:     end if
42: end if
43: isRunning ← falso
44: return falso
45: end function

```

Algoritmo 6 Configuración de gestos de acoplamiento para Android e iOS

```

1: procedure SETUP(enableSwipeGesture, enableAccelerometerGesture)
2:   swipeGesture ← Instancia de la clase SwipeGesture
3:   accelerometerGesture ← Instancia de la clase AccelerometerGesture
4:   procedure ONTOUCH(View, MotionEvent or UIGestureRecognizer)
5:     validLongClick ← Evaluar si el evento es un click largo con verifyLongClick
6:     validAccelerometer ← Evaluar si el evento es usado por el acelerómetro con
                           verifyAccelerometer
7:     if validLongClick = verdadero and enableSwipeGesture = verdadero then
8:       validSwipeGesture ← Evaluar si es un gesto de arrastre con
                           swipeGesture.validateGesture
9:     end if
10:    if validAccelerometer = verdadero and
                           enableAccelerometerGesture = verdadero then
11:      validAccelerometerGesture ← Evaluar si es un gesto de arrastre con
                           accelerometerGesture.validateGesture
12:    end if
13:    if validLongClick = falso then
14:      Notificar que se realizó un evento de tipo touch sobre la pantalla
15:    end if
16:    if validAccelerometer = falso then
17:      Notificar que no se obtuvo el ángulo mínimo requerido en el eje X o Y
18:    end if
19:  end procedure
20: end procedure

```

- Versión del sistema operativo.
- Funcionalidades adicionales (e.g., si tiene soporte para múltiples ventanas, soporte para *multitouch* y protocolos de comunicación disponibles).

Por lo tanto, un intento de acoplamiento se puede considerar como una representación lógica de un dispositivo, a través de la cual este puede ser identificado de forma única en la red. La difusión del intento se lleva a cabo una sola vez y de forma similar a la difusión de los mensajes realizada por el servicio de descubrimiento.

Algoritmo 7 Propagación de un intento de acoplamiento para Android e iOS

```

1: class SendOnePairingRequest (IP, puertoTCP, puertoUDP, direction, features)
2:   Agregar (“IP”, IP) al objeto JSONObject
3:   Agregar (“TCP”, puertoTCP) al objeto JSONObject
4:   Agregar (“UDP”, puertoUDP) al objeto JSONObject
5:   Agregar (“DIR”, direction) al objeto JSONObject
6:   Agregar (“SPECS”, features) al objeto JSONObject
7:   Propagar el objeto JSONObject mediante un mensaje de tipo broadcast (UDP)
8:   Dar retroalimentación audible
9: end class

```

Captura de un intento de acoplamiento

Esta tarea es llevada a cabo cuando ya se realizó un intento de acoplamiento. Su propósito principal es capturar los datos de acoplamiento difundidos por algún otro dispositivo. La implementación se muestra en la clase `ListenOnePairingRequest`, la cual se muestra mediante el Algoritmo 8 y se describe a continuación:

- Para procesar un intento de acoplamiento, se debe esperar `COUPLING_WAITING_TIME` milisegundos por un intento externo (línea 2).
- Si el intento proviene de un dispositivo externo (línea 4), se procede a analizar los datos recibidos (línea 6).
- Se compara la dirección de conexión recibida con la realizada (e.g., “derecha” con “izquierda”, “izquierda” con “derecha”, “arriba” con “abajo”, “abajo” con “arriba”) (línea 7).
- Si la lista `GROUP_LIST` esta vacía, entonces se tiene que agregar el mensaje de acoplamiento pero con valores propios (línea 11).
- Se agregan los datos del dispositivo externo a las listas `GROUP_LIST` y `CONNECTED_DEVICES` (líneas 13 y 14) y se notifica al usuario mediante un mensaje en pantalla o una retroalimentación acústica (línea 15).
- Posteriormente, se actualizan los elementos de la sesión colaborativa (e.g., identificador de la sesión, lista de colaboradores) si es necesario (línea 16).

- Si el intento de acoplamiento no es válido, se procede a notificar al usuario mediante un mensaje en pantalla o retroalimentación acústica (línea 20).

Algoritmo 8 Captura de un intento de acoplamiento para Android e iOS

```

1: class ListenOnePairingRequest (direction, features)
2:   Esperar por un intento externo durante COUPLING_WAITING_TIME milisegundos
3:   if Intento recibido then
4:     if Intento no proviene de este mismo dispositivo then
5:       newDevice ← Crear un objeto de tipo HashMap
6:       Obtener de message, los valores asociados a las llaves "IP", "TCP", "UDP",
       "DIR", "SPECS" y almacenarlos en newDevice
7:       if direction es contraria a la dirección recibida and direction no está en
       CONNECTED_DEVICES and GROUP_LIST no contiene el objeto newDevice then
8:         if GROUP_LIST no contiene elementos then
9:           ownDevice ← Crear un objeto de tipo HashMap
10:          ownDevice ← Las mismas llaves que en newDevice pero con los valores
              propios
11:          Agregar ownDevice a GROUP_LIST
12:        end if
13:        Agregar newDevice a GROUP_LIST
14:        Agregar newDevice a CONNECTED_DEVICES
15:        Dar retroalimentación audible de que el acoplamiento fue exitoso
16:        Actualización de la GROUP_LIST y el espacio de trabajo si es necesario
17:      end if
18:    end if
19:  else
20:    Dar retroalimentación para indicar que no se recibió un intento
21:    LOCKED_REQUESTS ← falso
22:  end if
23: end class

```

4.3.3. Servicio de actualización

El servicio de actualización se implementa principalmente en la clase `UpdateDistributionService`. Este servicio se encarga de procesar las acciones que realice el usuario dentro del espacio de trabajo. Para ello, este servicio cuenta con dos métodos (`sendObject` y `spreadObject`) y con un servidor local definido en la clase `UpdateDistributionTCPSTerver`, el cual se encarga de recibir y procesar los objetos transmitidos desde otros dispositivos a través de este servicio. La diferencia principal entre `sendObject` y `spreadObject` radica básicamente en que la primera permite enviar objetos únicamente a un solo dispositivo y la segunda envía objetos automáticamente a todo el grupo de trabajo, i.e., a cada uno de los dispositivos registrados en `GROUP_LIST`. A continuación se describen los parámetros que recibe envío a un solo dispositivo que se muestra en el Algoritmo 9.

- El parámetro `target` contiene la IP del dispositivo.
- El parámetro `target_object` guarda el puerto.
- El parámetro `object` almacena los elementos en formato `JsonObject`.
- El parámetro `protocol` mantiene el protocolo que se usará (TCP o UDP).

Algoritmo 9 Envío de objetos a un solo dispositivo para Android o iOS

```

1: procedure SENDOBJECT(target, target_object, object, protocol, waitForAnswer)
2:   jsonObject = object en formato de cadena de caracteres
3:   if protocol = "TCP" then
4:     Instanciar WorkerThread para enviar el objeto JsonObject mediante el protocolo
       TCP (ver Algoritmo 11)
5:   else
6:     Instanciar WorkerThread para enviar el objeto JsonObject mediante el protocolo
       UDP (ver Algoritmo 11)
7:   end if
8: end procedure

```

Algoritmo 10 Envío de objetos a múltiples dispositivos Android e iOS

```

1: procedure SPREADOBJECT(object, protocol)
2:   jsonObject = object en formato de cadena de caracteres
3:   GROUP_LIST_COPY = generar una copia de GROUP_LIST para iterar sobre ella
4:   for each device in GROUP_LIST_COPY do
5:     if protocol = "TCP" then
6:       Instanciar WorkerThread para enviar el objeto JsonObject mediante el
         protocolo TCP (ver algoritmo 11)
7:     else
8:       Instanciar WorkerThread para enviar el objeto JsonObject mediante el
         protocolo UDP (ver algoritmo 11)
9:     end if
10:  end for
11: end procedure

```

Como es posible apreciar, los Algoritmos 9 y 10 son parecidos. Ambos transforman el objeto a enviar `object` en otro que esté en formato JSON y preparan, dependiendo del protocolo `protocol` requerido, el hilo (o los hilos, para el caso de `spreadObject`) en segundo plano que estará(n) a cargo del envío o distribución del objeto transformado `JsonObject`. Ya que el soporte admite comunicación heterogénea entre los datos que se reciben (Android o iOS) y arreglos heterogéneos de dispositivos móviles (Android e iOS) no se puede instanciar los objetos con sus respectivas clases, ya que la serialización y deserialización de objetos enviados y recibidos por la red es diferente entre Android e iOS.

Una vez preparado el objeto a transmitir, se inicia el hilo que estará a cargo de dicha tarea. Para el caso del Algoritmo 10, se ejecutan tantos hilos como dispositivos estén registrados en `GROUP_LIST`. Este procedimiento requiere iterar sobre el recurso compartido `GROUP_LIST`.

`WorkerThread` es el hilo encargado de enviar el objeto y recibir respuestas sobre dichos envíos (un hilo por objeto para el caso de `spreadObject`). Para llevar a cabo el envío, cada instancia de `WorkerThread` obtiene los datos de conexión del parámetro `target` (el cual es básicamente una entrada de la lista de grupo, i.e., un objeto de tipo `HashMap` para Android o `NSMutableDictionary` para iOS con los datos de conexión almacenados en formato llave-valor). Después de obtener los datos necesarios, se realiza el envío dependiendo del protocolo `protocol`. Adicionalmente, es posible esperar una respuesta del servidor local en el dispositivo `target`, si el protocolo utilizado fue TCP y si así lo requiere el modelo de tareas de una aplicación (definido por los desarrolladores) (ver Algoritmo 11).

Algoritmo 11 Hilo en segundo plano encargado de enviar un objeto en formato JSON para Android e iOS

```

1: class WorkerThread (target, jsonObject, protocol, waitForAnswer)
2:   Extraer los datos de target para la conexión con el otro dispositivo
3:   Preparar el envío de jsonObject con los datos de conexión extraídos de acuerdo a
   protocol
4:   Enviar el objeto a target de acuerdo a protocol
5:   if protocol = "TCP" and waitForAnswer = verdadero then
6:     Esperar por la respuesta del otro dispositivo y procesarla
7:   end if
8: end class

```

Todo objeto enviado por los métodos `sendObject` y `spreadObject`, desde otros dispositivos, es procesado por el servidor local en la clase `UpdateDistributionTCPSTerver`. La lógica de dicho servidor se muestra en el Algoritmo 12.

Al igual que el servicio de descubrimiento de dispositivos, el servidor local permanece en ejecución en segundo plano desde que se inicia la sesión hasta que es terminada. Este comportamiento es requerido, ya que dicho servidor es utilizado durante el acoplamiento de dispositivos cuando un intento es válido. De esta forma, aunque un intento difundido y válido no sea capturado (i.e., que un dispositivo no sea consciente de que ya es partícipe de un acoplamiento exitoso) ambos dispositivos (aquel que difundió su intento) tendrán la capacidad de recibir y procesar los objetos.

Una vez que se recibe un objeto, este se convierte de una cadena de caracteres a un objeto en formato JSON. Posteriormente, se determina qué función procesará la información de `object`. Dependiendo del tipo de información que contenga el objeto recibido, el Algoritmo 12 realizará las siguientes acciones:

- Intento de acoplamiento
- Actualización del identificador de sesión
- Actualización de la lista de dispositivos acoplados
- Actualización del espacio de trabajo

Algoritmo 12 Servidor local encargado de recibir y procesar objetos para Android e iOS

```

1: class UpdateDistributionTCPSTerver
2:   while no se abandone la sesión do
3:     Estar a la espera de objetos externos (operación bloqueante de sockets)
4:     if objeto recibido then
5:       object = conversión del objeto recibo al objeto en formato string
6:       Determinar la función a la cuál pertenece object
7:       if object es utilizado por la sesión then
8:         Procesarlo y regresar a la línea 3
9:       else
10:        Notificar que se recibió un objeto que no es utilizado por el marco de desarrollo
        y será procesado por la aplicación desarrollada
11:      end if
12:    end if
13:  end while
14: end class

```

4.3.4. Servicio de presencia

El servicio de presencia está implementado en las clases `PresencePublishService`, `PresenceCollectService` y `PresenceCheckService`. Una vez que la sesión colaborativa se encuentra activa (i.e., que se formó un espacio de trabajo común a partir de dos espacios de trabajo individuales) es necesario contar con un mecanismo de verificación, el cual se encarga de comprobar si se encuentran activos los dispositivos que pertenecen a una sesión colaborativa, ya que si algún dispositivo se abandona de la sesión sin previo aviso o notificación (i.e., que puede existir una falla en la red o que el dispositivo se desconecte de la red), los demás dispositivos creerán erróneamente que la sesión está completa y esto puede causar que los objetos no se reciban en el dispositivo correcto ya que la sesión en ese dispositivo ha finalizado.

El Algoritmo 13 se encarga de enviar mensajes de presencia a los dispositivos que se encuentran en la lista `GROUP_LIST` y están activos este Algoritmo se encuentra en la clase `PresencePublishService`. A continuación se describe el algoritmo propuesto:

- Obtener las listas de los dispositivos que se encuentran activos en una sesión colaborativa (líneas 2 y 3); listas que se encuentran en la clase `SharedResources`. Este procedimiento requiere iterar sobre los recursos compartidos `GROUP_LIST` y `DEVICE_LIST`. Es por ello que primero se crea una copia de cada lista de forma segura (forma conocida en inglés como *thread safe*) y después se itera sobre cada copia. Es necesario crear una copia por cada lista de estos recursos ya que, al ser compartidos, cualquier hilo podría alterar sus contenidos mientras se realizan las iteraciones, provocando comportamientos indeseados y excepciones relacionadas con accesos concurrentes. Los métodos de *thread safe* que se usan para este soporte son: `synchronized()` para Android y `dispatch_async()` para iOS.
- Mientras el servicio esté activo (línea 4) y si las listas `GROUP_LIST` y `DEVICE_LIST` no están vacías (línea 5) se procede a extraer la información de cada dispositivo que este presente en la lista `GROUP_LIST`, lista que contiene los parámetros (e.g., “IP”, “TCP”, “UDP”,

“*DIR*”, “*SPECS*”).

- Se itera sobre la lista `GROUP_LIST`, la cual contiene la “*IP*” del dispositivo y al que se enviara el mensaje de presencia (línea 8).
- Se agregan los valores *IP*, *object* y *date* al objeto `JSONObject` (líneas 9, 10 y 11) y se envía el mensaje al dispositivo que se encuentra en la lista `GROUP_LIST` (línea 12).
- Se queda en espera por un tiempo `PRESENCE_CHECK_TIME` en milisegundos por cada elemento de la lista `GROUP_LIST` (línea 13).

Algoritmo 13 Difusión de la presencia del dispositivo para Android e iOS

```

1: procedure SENDPRESENCE(enableState)
2:   ListA ← Obtener Group_List de la clase SharedResources
3:   ListB ← Obtener Device_List de la clase SharedResources
4:   if enableState = verdadero then
5:     if ListA != Lista vacía and ListB != Lista vacía then
6:       Extraer datos de ListA
7:       while enableState do
8:         for all los dispositivos en ListA do
9:           Agregar (“IP”, IP) al objeto JSONObject
10:          Agregar (“OBJECT”, objeto) al objeto JSONObject
11:          Agregar (“DATE”, date) al objeto JSONObject
12:          Enviar mensaje de tipo JSONObject de estado de presencia
13:          Esperar PRESENCE_CHECK_TIME en milisegundos
14:        end for
15:      end while
16:    end if
17:  end if
18: end procedure

```

El Algoritmo 14 se encarga de recolectar los mensajes de presencia y guardarlos en la lista `STATE_OBJECTS_DEVICES`, este Algoritmo se encuentra en la clase `PresenceCollectService`. A continuación se describe el algoritmo propuesto:

- Se extraen los valores de mensaje de presencia del objeto `JSONObject` (e.g., “*IP*”, “*OBJECT*”, “*DATE*”) (línea 2).
- Mientras el servicio esté activo (línea 6) se procede a guardar los objetos recibidos en la lista `STATE_OBJECTS_DEVICES` (línea 7).

El Algoritmo 15 se encarga de comprobar si los dispositivos que se encuentran en la lista `GROUP_LIST` y están activos este Algoritmo se encuentra en la clase `PresenceCheckService`. A continuación se describe el algoritmo propuesto:

Algoritmo 14 Recolección de mensajes de presencia para Android e iOS

```

1: procedure RECEIVEPRESENCE(enableState, presenceMessage)
2:   Extraer los valores del objeto JSONObject recibido
3:   IP ← Obtener de JSONObject la IP del mensaje de presencia
4:   object ← Obtener de JSONObject los objetos de recuperación
5:   date ← Obtener de JSONObject la fecha
6:   if enableState = verdadero then
7:     Guardar la IP, object y date en la lista STATE.OBJECTS.DEVICES de la clase
       SharedResources
8:   end if
9: end procedure

```

- Obtener las listas de los dispositivos que se encuentran activos en una sesión colaborativa (líneas 2 y 3); listas que se encuentran en la clase `SharedResources` y obtener los objetos de presencia (línea 4). Este procedimiento requiere iterar sobre los recursos compartidos `GROUP_LIST`, `DEVICE_LIST` y `STATE.OBJECTS.DEVICES`. Es por ello que primero se crea una copia de cada lista de forma segura (forma conocida en inglés como *thread safe*) y después se itera sobre cada copia. Es necesario crear una copia por cada lista de estos recursos ya que, al ser compartidos, cualquier hilo podría alterar sus contenidos mientras se realizan las iteraciones, provocando comportamientos indeseados y excepciones relacionadas con accesos concurrentes. Los métodos de *thread safe* que se usan para este soporte son: `synchronized()` para Android y `dispatch_async()` para iOS.
- Mientras el servicio esté activo (línea 8) y si las listas `GROUP_LIST` y `DEVICE_LIST` no están vacías (línea 6) se procede a extraer la información de cada dispositivo que este presente en la lista `GROUP_LIST`, lista que contiene los parámetros (e.g., “*IP*”, “*TCP*”, “*UDP*”, “*DIR*”, “*SPECS*”).
- Se itera sobre la lista `GROUP_LIST`, el cual contiene la “*IP*” del dispositivo, el cual se verificara el estado de presencia (línea 9).
- Se extraen los objetos de la lista `STATE.OBJECTS.DEVICES`, el cual contiene los objetos de presencia (e.g., “*IP*”, “*OBJECT*”, “*DATE*”) (línea 10).
- En la línea 15 se se verifica si el tiempo del dispositivo contra el recibido es mayor a `PRESENCE_CHECK_TIME` en milisegundos. Si por causas ajenas a la sesión (e.g., el usuario apago el servicio de WiFi o fallas en la red local) se procede a eliminar los datos del dispositivos de las listas `GROUP_LIST` y `DEVICE_LIST` (línea 16).
- En la línea 18 se muestran algunas opciones de remodelación del espacio de trabajo común, las cuales consisten en recuperar los objetos perdidos del dispositivo que se ha desconectado como son:
 - Recuperar solo los objetos que tuvieron alguna iteración entre las partes del espacio de trabajo común.
 - Eliminar los objetos que se encuentran temporalmente en una lista o archivo.

- Se queda en espera por un tiempo `PRESENCE_CHECK_TIME` en milisegundos por cada elemento de la lista `GROUP_LIST` (línea 20).

Algoritmo 15 Verificación del estado de presencia para Android e iOS

```

1: procedure CHECKPRESENCE(enableState)
2:   ListA ← Obtener Group_List de la clase SharedResources
3:   ListB ← Obtener Device_List de la clase SharedResources
4:   ListC ← Obtener STATE_OBJECTS_DEVICES de la clase SharedResources
5:   if enableState = verdadero then
6:     if ListA != Lista vacía and ListB != Lista vacía then
7:       Extraer datos de ListA
8:       while enableState do
9:         for all los dispositivos en ListA do
10:          Extraer datos de ListC
11:          IP ← Obtener de ListC la IP
12:          object ← Obtener de ListC los objetos de recuperación
13:          date ← Obtener de ListC la fecha
14:          ownDate ← Obtener la fecha del dispositivo
15:          if date > ownDate por TIME_PRESENCE en milisegundos then
16:            Eliminar dispositivo de Group_List y Device_List
17:            Mostrar mensaje de desconexión al usuario
18:            Mostrar opciones de recuperación de objetos al usuario
19:          end if
20:          Esperar PRESENCE_CHECK_TIME en milisegundos
21:        end for
22:      end while
23:    end if
24:  end if
25: end procedure

```

4.4. Configuración del soporte

Como se mencionó en la sección 3.4.2, junto con los servicios se proporciona una clase que brinda una configuración básica para la sesión. Dicha clase lleva por nombre `WorkTogetherSetup` y se encuentra en el paquete `global`. Dentro de `WorkTogetherSetup`, se puede apreciar a modo de ejemplo, el orden en el cual se inicializan o liberan los recursos y servicios proporcionados. La lógica de la configuración básica se muestra en el Algoritmo 16.

A continuación describimos cada uno de los pasos definidos por el Algoritmo 16:

- Ejecutar el método `init_SharedResources` de la clase `SharedResources` (línea 2) que inicializara los recursos compartidos como: `GROUP_LIST`, `DEVICE_LIST` o `CONNECTED_DEVICES`.
- Crear un objeto de las clases `DiscoverySettings`, `CouplingSettings` y `UpdateSettings` (línea 3).

Algoritmo 16 Configuración básica del soporte para Android e iOS

```

1: class WorkTogetherSetup
2:   Inicializar recursos compartidos por los servicios
3:   Inicializar las configuraciones de los servicios
4:   Inicializar los objetos de cada servicio
5:   Arrancar los servicios
6:   Agregar los notficadores para los servicios de descubrimiento y comunicación
7:   Inicializar el detector de clicks largos
8:   Inicializar los objetos de los gestos de acoplamiento
9:   Adherir los objetos de los gestos de acoplamiento al espacio de trabajo
10:  Habilitar la actualización de características cuando se rota el dispositivo
11:  Habilitar la finalización de los servicios cuando se cierra la sesión
12: end class

```

- Crear una instancia de las clases `UpdateDistributionService`, `DiscoveryPublishService` y `DiscoveryCollectService` (línea 4).
- Ejecutar el servicio de distribución de actualizaciones con el método `startService` (línea 5).
- Agregar los *listeners* de tipo `DiscoveryServiceListener` y `UpdateDistributionServiceListener`, en los cuales se procesan los datos u objetos recibidos de acuerdo a lo descrito en las secciones 4.3.1 y 4.3.3, respectivamente (línea 6).
- Crear un objeto de la clase `LongClickDetector`(línea 7).
- Crear un objeto de cada gesto de acoplamiento (clases `SwipeGesture` y `AccelerometerGesture`), si es que se desea habilitar ambos. De lo contrario crear sólo el objeto requerido (línea 8).
- Agregar los validadores (línea 9) del click largo (función `verifyLongClick`) y los gestos de acoplamiento (función `validateGesture` de cada objeto creado en la línea 8) dentro del evento `onTouch` de la clase `View` cuyo objeto representa el espacio de trabajo.
- Crear un hilo dentro del evento `onConfigurationChanged()` de la actividad principal, cuyo código a ejecutar es precisamente el definido en la clase `SendFeaturesToTheGroup` (línea 10).
- Ejecutar los métodos `stopService()` de cada uno de los servicios dentro del evento `onStop()` de la actividad principal, así como el método `freeResources()` de la clase `SharedResources` (línea 11).

Capítulo 5

Desarrollo de aplicaciones mediante el marco de desarrollo

En este capítulo se muestra la implementación de una aplicación de ejemplo con el soporte propuesto. En la Sección 5.1, se describe los métodos proporcionados por la API del marco de desarrollo propuesto. En la Sección 5.2, se describe un ejemplo del uso del marco de desarrollo propuesto. En la Sección 5.3, se describen los componentes que integran el espacio de trabajo individual y colaborativo, así como las acciones que se pueden realizar sobre dichos espacios. En la Sección 5.4 se describe la interfaz gráfica de usuario, las clases de apoyo de la aplicación de ejemplo. En la Sección 5.5 se muestra el mecanismo de recuperación que se implementa. En la Sección 5.6 se muestran algunos de los problemas encontrados durante la implementación de la aplicación de ejemplo.

5.1. Descripción de la API proporcionada por el soporte

Una *Application Programming Interface* (mejor conocida como API por sus siglas en inglés) es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software. A continuación se mencionan los métodos principales que un desarrollador puede comunicar la aplicación creada mediante el soporte de desarrollo propuesto:

- `WorkTogetherSetup(Activity, View, DiscoveryServiceListener, WorkspaceTouchEventListeners, UpdateDistributionServiceListener, CustomListener)`: recibe los parámetros siguientes:
 - **Activity**: recibe como parámetro una instancia a la actividad principal.
 - **View**: recibe como parámetro una instancia del espacio de trabajo.
 - **DiscoveryServiceListener**: recibe como parámetro un *listener* que se encarga de notificar los cambios que se producen en la clase `DiscoveryCollectService`.
 - **WorkspaceTouchEventListeners**: recibe como parámetro un *listener* que se encarga de notificar los cambios que se producen en la clase `WorkTogetherSetup`.

- `UpdateDistributionServiceListener`: recibe como parámetro un *listener* que se encarga de notificar los cambios que se producen en la clase `UpdateDistributionService`.
 - `CustomListener`: recibe como parámetro un *listener* que se encarga de notificar los cambios que se producen en la clase `PresenceService`.
- `void stopWorkTogether()`: este método detiene el soporte propuesto.
 - `List<HashMap<String, Object>>getConected()`: este método regresa una lista con los dispositivos conectados.
 - `HashMap<String, String>getOwnInfo()`: este método regresa una lista con la información del dispositivo (e.g., IP, puertos de conexión, nombre del dispositivo y el tipo de sistema operativo).
 - `HashMap<String, String>getInfoConfig()`: este método regresa una lista con la configuración del soporte.
 - `setupCoupling(Boolean, Boolean)`: este método recibe dos parámetros de tipo *Boolean* para activar los gestos de acoplamiento.
 - `setPresenceObjects(String)`: este método recibe como parámetro un *string* con los objetos que se encuentran en el espacio de trabajo.

Para el sistema operativo iOS la a API es la siguiente:

- `startWorkTogether`: este método inicia el soporte.
- `stopWorkTogether`: este método detiene la ejecución del soporte.
- `(NSMutableDictionary*) getConnList`: este método regresa una lista con los dispositivos conectados.
- `(NSMutableDictionary*) getOwnInfo`: este método regresa una lista con la información del dispositivo (e.g., IP, puertos de conexión, nombre del dispositivo y el tipo de sistema operativo).
- `(NSMutableDictionary*) getInfoConfig`: este método egresa una lista con la configuración del soporte.
- `(void) setObjectState:(NSString*)obj`: este método recibe como parámetro un *string* con los objetos que se encuentran en el espacio de trabajo.
- `(NSString*)validateSwipeGesture:(BOOL)longClick`: este método recibe como parámetro un objeto de tipo *BOOL* para validar el gesto de arrastre.
- `(NSString*)validateAccelerometerGesture:(BOOL)longClick`: este método recibe como parámetro un objeto de tipo *BOOL* para validar el gesto de inclinación.

5.2. Aplicación de demostración

Esta sección, se brinda un ejemplo para el uso del marco de desarrollo propuesto, junto con los servicios que se proporcionan facilitan el trabajo colaborativo. Un ejemplo se encuentra en la clase `ExampleWorkTogether`, la cual contiene algunos métodos para procesar los objetos que se reciben o envían a través de los servicios que proporciona el marco de desarrollo. En la clase `Workspace` se muestra un ejemplo de la aplicación de demostración. En el algoritmo 17 se muestra el proceso de ejecución e interacción del marco de desarrollo con la aplicación de demostración.

Como se puede observar el Algoritmo 17 el proceso de configuración es muy fácil, ya que solo requieren dos simples pasos que a continuación se describen:

- Al inicializar la clase `Workspace` se construye el espacio de trabajo que se mostrará gráficamente al usuario (línea 2).
- Se inicializan los servicios contenidos en la clase `WorkTogetherSetup` (línea 3).

Algoritmo 17 Configuración de la aplicación de ejemplo para Android e iOS

```

1: class ExampleWorkTogether
2:   Inicializar la clase Workspace
3:   Inicializar el soporte WorkTogetherSetup
4: end class

```

Además la clase `ExampleWorkTogether` incluye algunos métodos los cuales procesan los objetos recibidos o enviados por el soporte que son diseñados por el desarrollador. A continuación describe cada uno de los pasos definidos por el Algoritmo 18:

Algoritmo 18 Métodos de ejemplo de la clase `ExampleWorkTogether` para Android e iOS

```

1: class ExampleWorkTogether
2:   procedure DISCOVERYSERVICEDEVICEJOINED(object)
3:     Ejecutar un hilo y procesar el objeto object recibido
4:   end procedure
5:   procedure DISCOVERYSERVICEDEVICELEFT(object)
6:     Ejecutar un hilo y procesar el objeto object recibido
7:   end procedure
8:   procedure CHECKOBJECTRECEIVED(object)
9:     Ejecutar un hilo y procesar el objeto object recibido
10:  end procedure
11: end class

```

- El método `DiscoveryServiceDeviceJoined`(línea 2) recibe la notificación de otros dispositivos que se conectan a la red y que están ejecutando nuestro soporte, el `object` recibido es en formato JSON y contiene los valores “*IP*”, “*TCP*”, “*UDP*”, “*DEVICE*”, “*TYPEDEVICE*”.

- El método `DiscoveryServiceDeviceLeft` (línea 5) recibe la notificación de otros dispositivos que se desconectan de la red y que están ejecutando nuestro soporte, el `object` recibido es en formato JSON y contiene los valores “*IP*”, “*TCP*”, “*UDP*”, “*DEVICE*”, “*TYPEDEVICE*”.
- El método `checkObjectReceived` (línea 8) recibe los objetos de otros dispositivos que están en una sesión colaborativa y que están ejecutando nuestro soporte, el `object` recibido es en formato JSON y puede contener la remodelación de los objetos que se encuentran en el área de trabajo.

En la clase `Workspace` contiene un ejemplo de aplicación construida a partir del marco de desarrollo propuesto. Esta clase crea objetos que se colocan en el espacio de trabajo. Estos objetos pueden ser trasladados en cualquier parte de la pantalla. Si existe una sesión colaborativa los objetos pueden ser trasladados hacia otras partes del espacio de trabajo común distribuido en varios dispositivos móviles. A continuación se muestran los Algoritmos 19 y 20 que conforman esta clase.

Algoritmo 19 Creación de un objeto en el espacio de trabajo para Android e iOS

```

1: procedure ADDBLOCKPARAMS(text)
2:   color ← generar un color RGB aleatorio entre 0 y 256
3:   image ← crear una imagen del tamaño de text y color
4:   agregar a List_Block un id, text, color, posición en X y Y
5:   agregar image al espacio de trabajo
6:   agregar un listener a image
7: end procedure

```

A continuación se describe el Algoritmo 19:

- Se genera un valor aleatorio RGB (línea 2).
- Se crea una imagen (línea 3) con el texto que se recibe como parámetro y el color generado en la línea (2).
- Se agregan los valores (línea 4) a la lista `List_Block`, la cual se usa por el servicio de presencia para recuperar los objetos de una sesión colaborativa.

Algoritmo 20 Creación de un objeto externo en el espacio de trabajo para Android e iOS

```

1: procedure ADDBLOCKPARAMSEXTERN(externID,externX,externY,externText,externColor)
2:   image ← crear una imagen del tamaño de externText y externColor
3:   agregar a List_Block los parámetros externID, externText, externColor, posición en externX y externY
4:   agregar image al espacio de trabajo
5:   agregar un listener a image
6: end procedure

```

El Algoritmo 20 se usa cuando existe una sesión colaborativa, los objetos que se reciben de otros dispositivos y que son parte de la sesión son procesados dependiendo de los datos que se reciben (e.g., id, texto, color, posición en x, posición en y). A continuación se describe el Algoritmo 20:

- Se crea una imagen dependiendo del color y el texto que se recibe (línea 2).
- Se agregan los valores (línea 3) a la lista `List_Block`, la cual se usa por el servicio de presencia para recuperar los objetos de una sesión colaborativa.

5.3. Espacios de trabajo de la aplicación de ejemplo

El objetivo de la aplicación de demostración es validar el marco de desarrollo propuesto y la facilidad con la cual se pueden crear aplicaciones que los desarrolladores propongan. La aplicación ofrece dos tipos de espacios: el espacio individual y el espacio colaborativo.

5.3.1. Espacio de trabajo individual

Al iniciar la aplicación de ejemplo, se crea un espacio de trabajo y una sesión individual. El espacio de trabajo tiene asociado un identificador, botones de configuración, objetos y un color de fondo aleatorio (ver Figura 5.1).

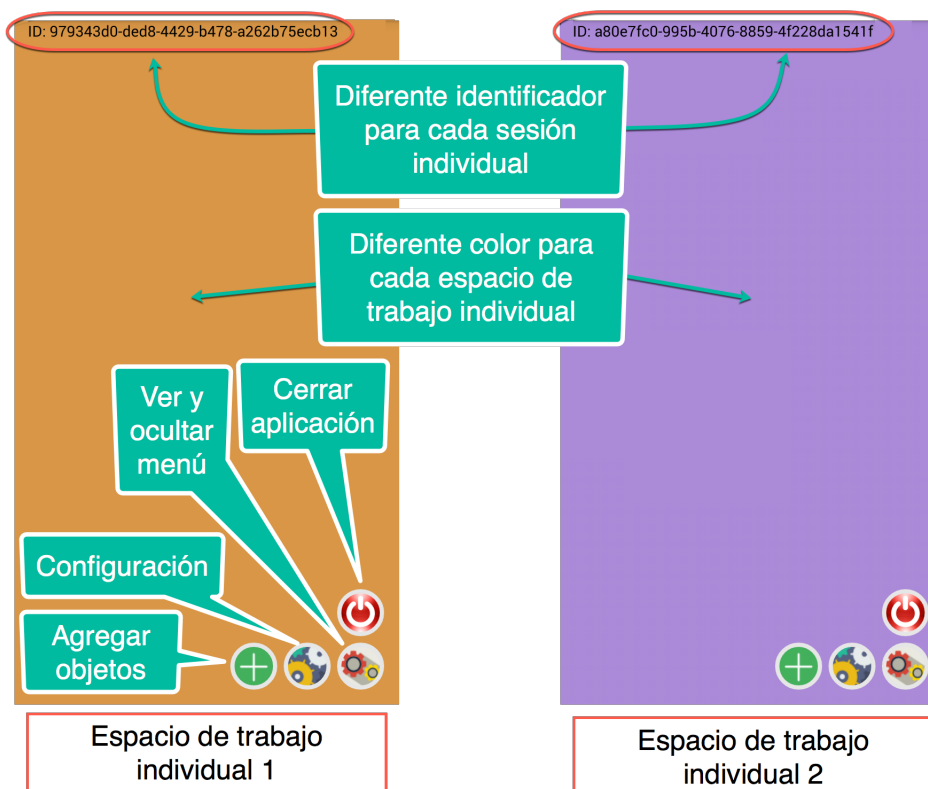


Figura 5.1: Espacio de trabajo individual y botones de configuración

El identificador de sesión es generado aleatoriamente y sirve como identificación de la sesión asociada, a pesar de que aún no se haya creado una sesión colaborativa. Dicho identificador determina en un futuro que dispositivos pertenecen a la misma sesión.

Los botones de configuración son creados al iniciar la aplicación y su propósito principal es la configuración de la sesión. A continuación se describe cada una de las funciones de los botones (ver Figura 5.1):

- **Botón de cerrar aplicación:** su función principal es detener todos los servicios del soporte incluyendo la aplicación de ejemplo (ver Sección 3.1) y el cierre de la aplicación.
- **Botón ver y ocultar menú:** este botón muestra u oculta los botones de configuración y agregar objeto.
- **Botón de configuración:** muestra cinco categorías de configuración e información (ver Figura 5.2).
 - ***Coupling settings:*** en esta categoría se puede configurar el gesto de acoplamiento que desea emplear el usuario. Además, se puede ver la información de los dispositivos que se encuentren acoplados con este dispositivo(ver Figura 5.3).
 - ***Mobile info:*** muestra la información básica del dispositivo como nombre del dispositivo, sistema operativo, dirección IP y puertos de conexión (ver Figura 5.4.b).
 - ***Framework configuration:*** muestra algunos parámetros de los servicios que pueden ser modificados por el usuario (e.g., la IP, los puertos TCP y UDP, el tiempo de ejecución de los servicios de descubrimiento y de presencia, el gesto de acoplamiento activo) (ver Figura 5.4.a).
 - ***Change color screen:*** cambia el color del área de trabajo aleatoriamente.
- **Botón agregar objeto:** despliega una ventana en el cual se ingresa un texto, el objeto creado se agrega al espacio de trabajo.

Los objetos son creados cuando el usuario hace click en el botón añadir objeto (ver Figura 5.5). Algunas características de los objetos se mencionan a continuación:

- puede ser trasladado en todo el espacio de trabajo mediante el arrastre con el dedo del usuario.
- puede eliminarse mediante un doble click o *touch*.
- el objeto puede contener un texto que el usuario ingresa al momento de su creación.

Posteriormente, la sesión individual se queda en espera de peticiones de acoplamiento para crear una sesión colaborativa. Cabe mencionar que al iniciar una sesión individual los servicios de descubrimiento (ver Sección 3.2.2) y comunicación (ver Sección 3.2.3) propuestos son iniciados, con el fin de identificar el dispositivo a través de la red local y recibir peticiones de acoplamiento respectivamente.

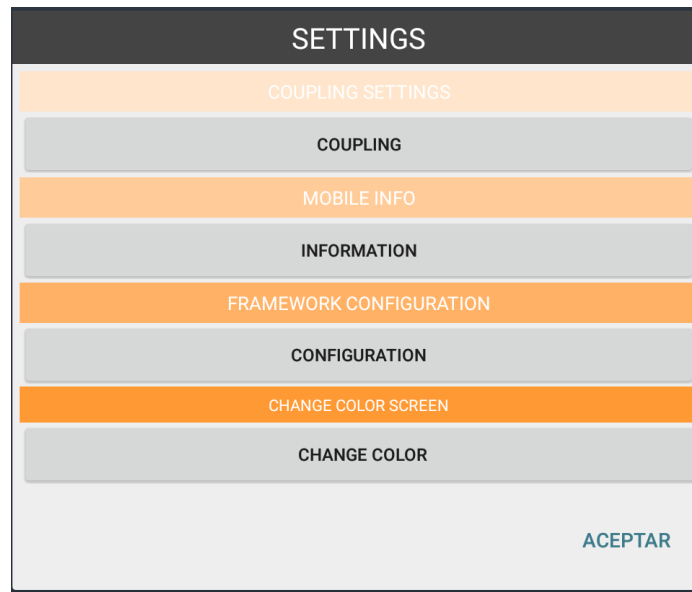


Figura 5.2: Menú del botón de configuración

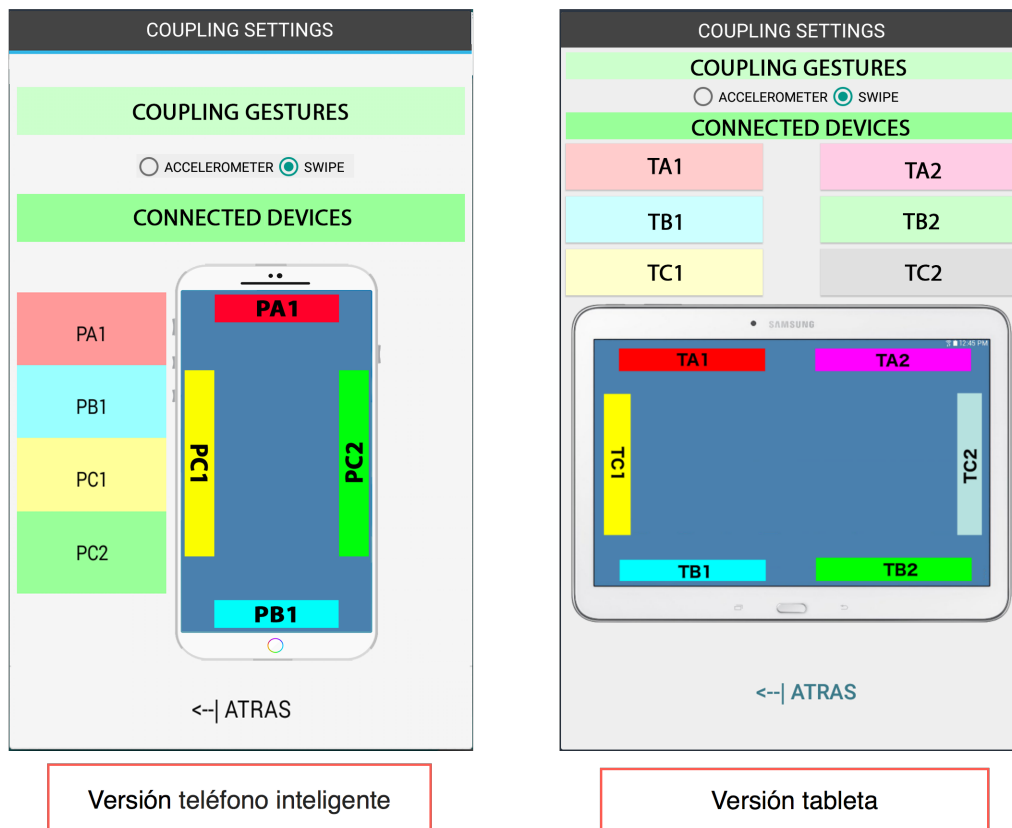


Figura 5.3: Menú de la sección *Coupling settings* del botón de configuración

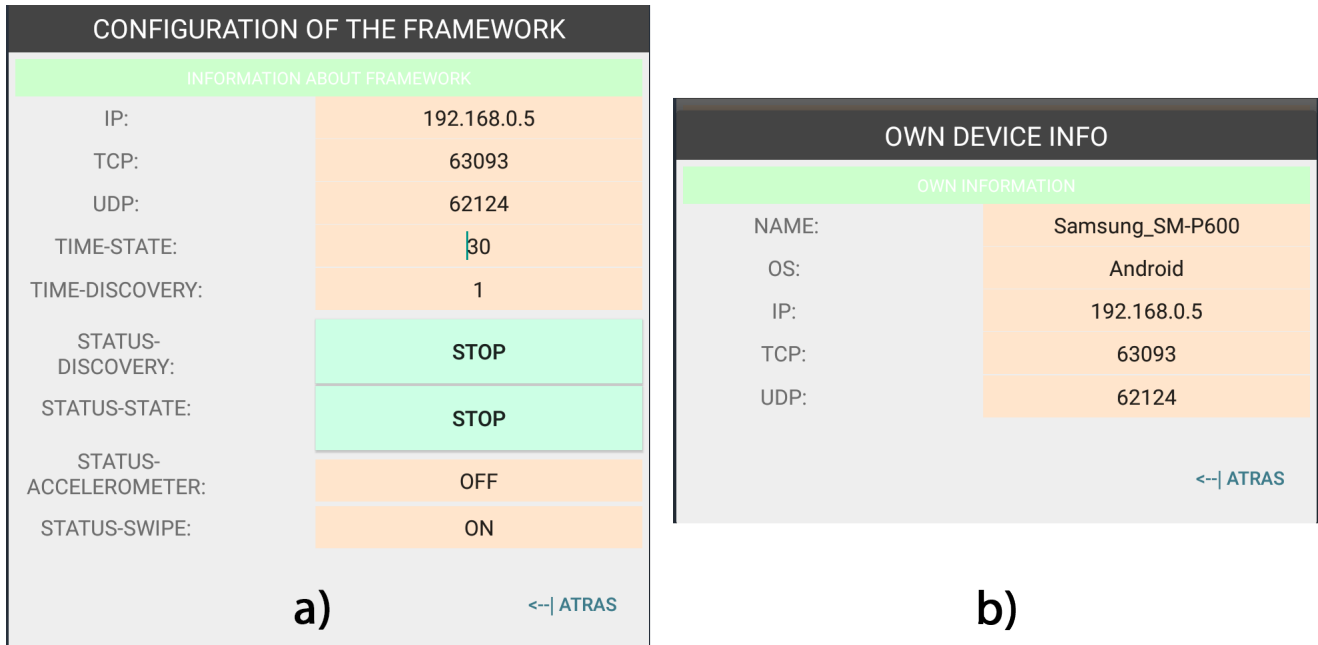


Figura 5.4: Menú de la sección *Framework configuration* y *Mobile info* del botón de configuración

Agregar Bloque

Ingresar un texto:

Maximo de 10 caracteres

CANCEL ACEPTAR

Figura 5.5: Menú del botón de agregar objeto

5.3.2. Espacio de trabajo colaborativo

Tras iniciar la sesión individual y al menos otro dispositivo ha iniciado otra sesión individual creadas ambas mediante el marco de desarrollo propuesto, se puede crear una sesión colaborativa, la cual se crea mediante un intento exitoso de acoplamiento de los dispositivos móviles donde se están ejecutando dichas sesiones individuales (ver Figura 5.6). A continuación se mencionan los eventos que se desencadenan para crear una sesión colaborativa:

- Dependiendo de que lado del dispositivo se hizo el intento de acoplamiento se mostrará una barra de color que indica referencia que dispositivo se acoplo correctamente. Esta barra desaparece cuando el dispositivo que se acoplo cierra la aplicación de ejemplo o se desconecta de la red WiFi. Asimismo, cabe mencionar que dependiendo del dispositivo (teléfono inteligente o tableta) se pueden mostrar hasta seis barras de color (ver Figura 5.7 y Figura 5.8).
- EL identificador de la se sesión es actualizado para tener un único identificador de la sesión colaborativa (ver Figura 5.6).

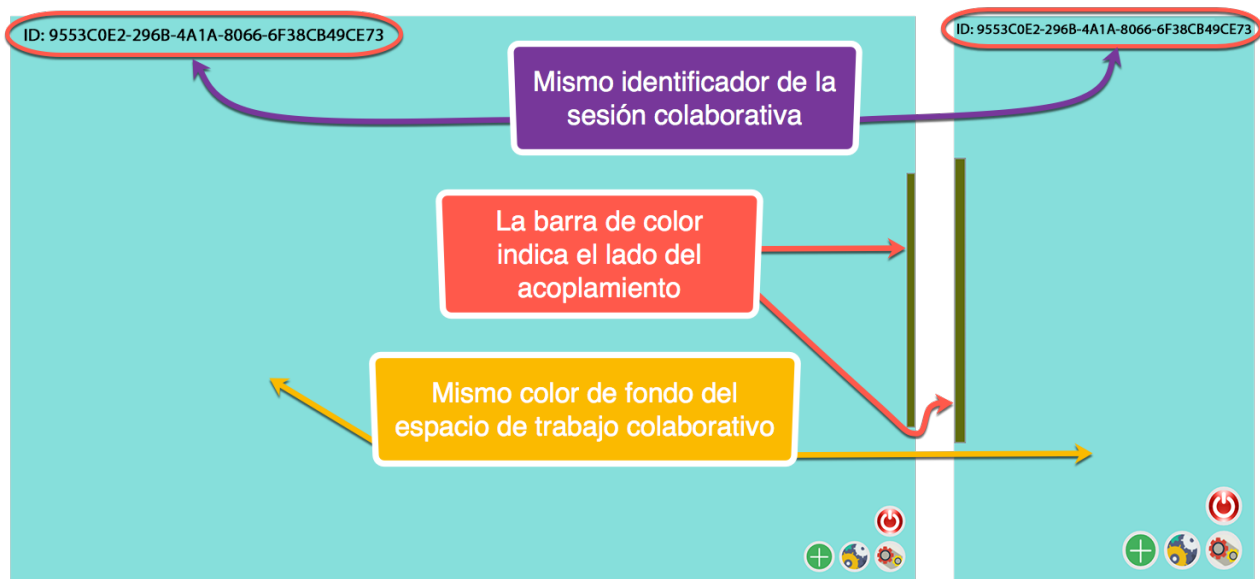


Figura 5.6: Elementos del espacio de trabajo colaborativo

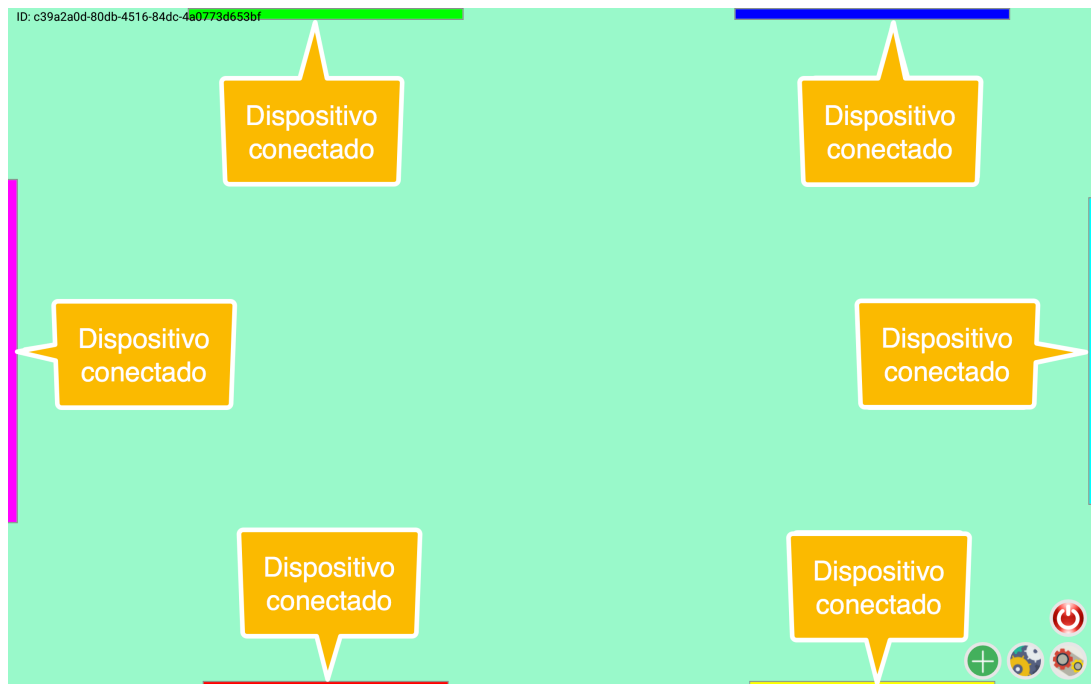


Figura 5.7: Posibles lados de acoplamiento para una tableta

- Los objetos creados en una parte del espacio trabajo colaborativo pueden trasladarse de un dispositivo a otro o eliminarse (ver Figura 5.9). Estos objetos solo se pueden trasladar por el lado de acoplamiento.
- El color de fondo del espacio de trabajo colaborativo es el mismo en todos los dispositivos que lo conforman, Este color de fondo puede cambiarse mediante el menú configuración

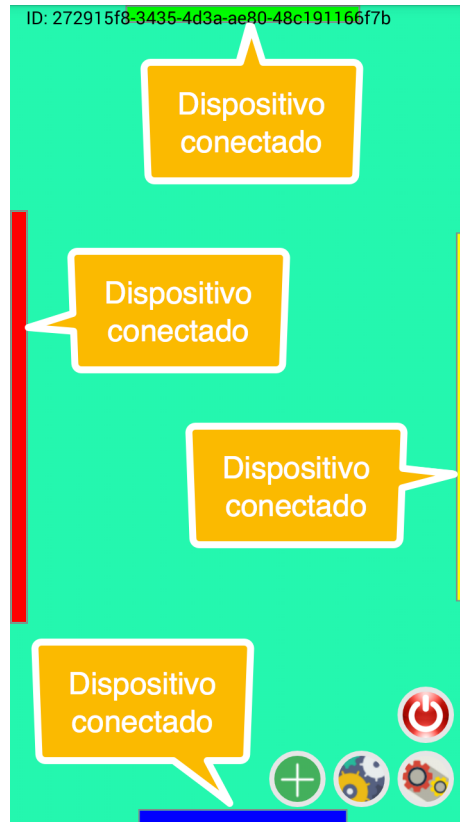


Figura 5.8: Posibles lados de acoplamiento para un teléfono inteligente

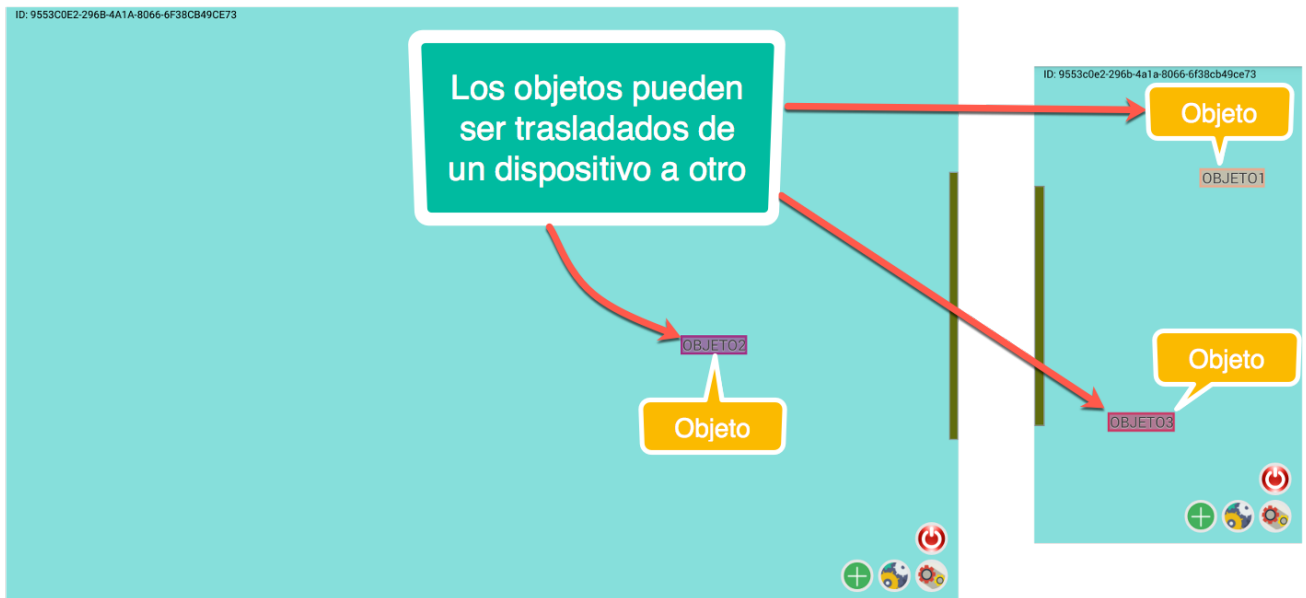


Figura 5.9: Objetos dentro del área colaborativa

(ver Figura 5.2).

Además, solo se puede enviar los objetos de una parte del espacio común a otro si los lados del dispositivo están acoplados de otra manera el objeto seguirá estando en esa parte del espacio de trabajo común.

5.3.3. Eventos producidos por los servicios

Los mensajes que se despliegan en la interfaz gráfica de usuario son producto de los eventos que se producen por los servicios propuestos (ver Sección 4.3), son tres eventos que se generan tras iniciar la sesión colaborativa:

- **Mensaje de registro:** este mensaje se muestra tras iniciar la sesión individual y tiene tres elementos que a continuación se mencionan (ver Figura 5.10):
 1. tipo de mensaje, representado por un color azul.
 2. tipo de sistema operativo móvil en el que se ejecuta la sesión individual representado por un logo.
 3. nombre del dispositivo móvil.
- **Mensaje de conexión:** este mensaje es mostrado cuando otro dispositivo está conectado a la misma red local y ejecuta una sesión individual. Sirve para identificar el dispositivo guardando algunos datos, e.g., la dirección IP, puertos de conexión y disponibilidad (ver Sección 4.3.1) que sirven para establecer una sesión colaborativa. A continuación, se mencionan los elementos que componen el mensaje (ver Figura 5.11):
 1. tipo de mensaje, representado por un color verde.
 2. tipo de sistema operativo móvil en el que se ejecuta la sesión (individual o colaborativa) representado por un logo.
 3. nombre del dispositivo móvil.
- **Mensaje de desconexión:** este mensaje es mostrado cuando un dispositivo ha cerrado la sesión en curso (individual o colaborativa). Sirve para avisar el abandono de una sesión colaborativa o el cierre de la aplicación. Además, los datos que se reciben por medio del servicio descubrimiento (ver Sección 4.3.1) eliminan cualquier intento de acoplamiento realizado. A continuación, se muestran los elementos que componen este mensaje (ver Figura 5.12):
 1. tipo de mensaje, representado por un color rojo.
 2. tipo de sistema operativo móvil en el que se ejecuta la sesión en curso representado por un logo.
 3. nombre del dispositivo móvil.



Figura 5.10: Mensaje de registro

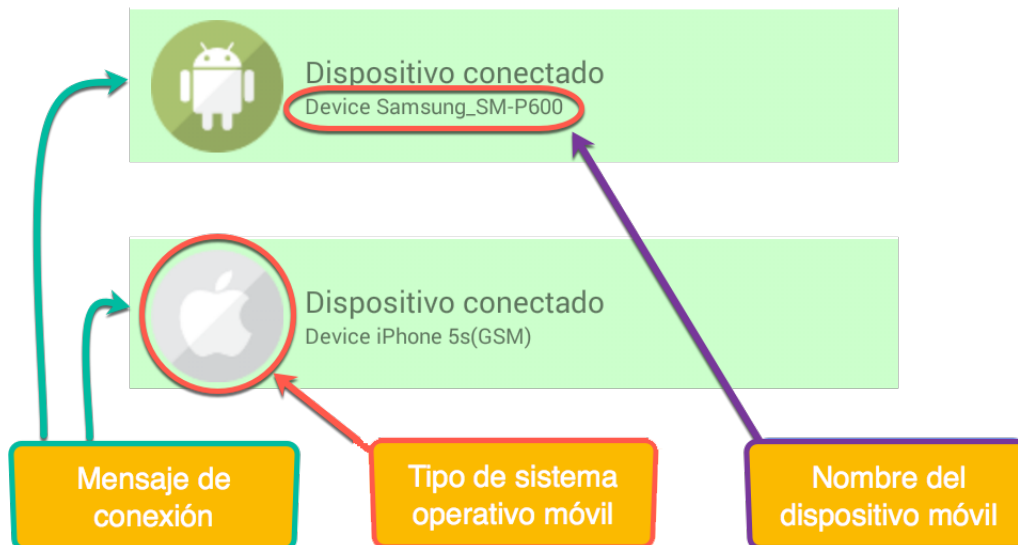


Figura 5.11: Mensaje de conexión

5.4. Aspectos técnicos de la aplicación de ejemplo

Aunque el tipo de aplicación descrita en la sección 5.3 es uno de tantos ejemplos que se pueden implementar con el marco de desarrollo propuesto, la creación de esta aplicación de ejemplo proporciona las bases para personalizar los servicios proporcionados por el marco de desarrollo. En la Subsección 5.4.1 se describe a detalle cada uno de los aspectos técnicos y puntos clave detectados para la construcción de la aplicación. En la Subsección 5.4.2 se detallan las clases de apoyo para la aplicación de ejemplo.

5.4.1. Interfaz gráfica de usuario

El espacio de trabajo es representado por la clase `View` tanto para Android como iOS. Los objetos que se muestran sobre esta clase pueden ser e.g., botones, texto, imágenes. El objeto de la clase `View` utilizado como espacio de trabajo abarca la totalidad de la pantalla del dispositivo en todo

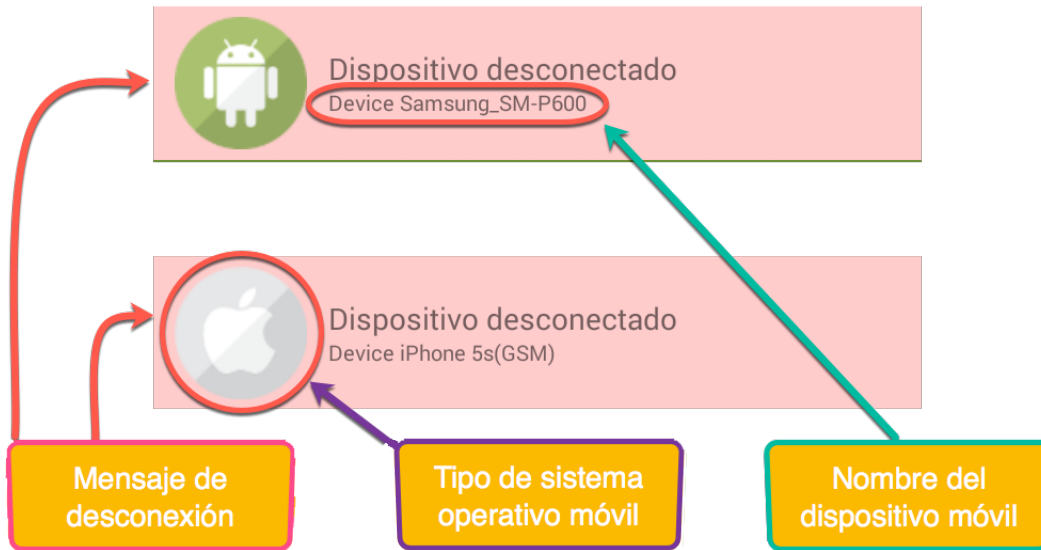


Figura 5.12: Mensaje de desconexión

momento. Sobre dicho objeto, se pueden crear los siguientes elementos:

- identificador de sesión
- barras de color que indican el lado de acoplamiento
- botones que tienen diferentes acciones
- objetos que pueden interactuar con el usuario, los cuales están compuestos de un texto y un color aleatorio
- fondo de color que se genera aleatoriamente al iniciar la sesión individual

5.4.2. Clases de apoyo

Para el desarrollo de la aplicación de validación se crearon dos clases de apoyo, así como la clase principal. Las clases de apoyo tienen como propósito crear los recursos y objetos que se emplean en la clase principal. La clase principal configura el soporte y permite ajustar los servicios para procesar correctamente los eventos que se produzcan. Cabe mencionar que en la sección 5.2 se muestra la implementación de la clase principal `ExampleWorkTogether` de la aplicación de validación.

A continuación se describe las clases de apoyo que forman parte de la lógica de la aplicación de ejemplo:

- **Workspace:** dentro de esta clase se implementan los siguientes métodos:
 - **Menú:** este método brinda la configuración básica para los servicios propuestos, además de brindar información sobre el dispositivo que ejecuta la aplicación.
 - **Crear objetos:** este método tiene como objetivo crear una imagen con un texto y un color.

- **Crear objetos externos:** cuando existe una sesión colaborativa este método recibe los objetos creados en otros dispositivos que implementan los servicios y los agrega al espacio de trabajo.
 - **Cambiar fondo del espacio de trabajo:** este método cambia el color del área de trabajo y si existe una sesión colaborativa envía un mensaje a los demás integrantes de la sesión para que actualicen el fondo del área de trabajo.
 - **Mensajes de información:** este método muestra los mensajes recibidos por el servicio de descubrimiento (mensaje de conexión y mensaje de desconexión).
- `WorkspaceSettings`: esta clase se encarga de guardar los ajustes y configuraciones de la clase `Workspace` (e.g., identificador de las barras de conexión, colores).

En el Algoritmo 21 se muestra el funcionamiento de los métodos contenidos de la clase `Workspace`.

5.5. Mecanismo de recuperación de objetos

El servicio de presencia implementa un mecanismo de recuperación de objetos propuesto (ver Sección 4.3.4). Este servicio es ejecutado cuando existe una sesión colaborativa, el servicio es ejecutado cada 30 segundos; este tiempo puede ser modificado por el desarrollador si es necesario. Los objetos que son creados en el espacio colaborativo son replicados por este servicio a todos los dispositivos que se encuentren acoplados y pertenezcan a la misma sesión colaborativa, el mensaje de presencia contiene los siguientes objetos:

- Texto asignado por el usuario
- Color generado aleatoriamente

En la Figura 5.13 se muestran dos dispositivos, uno que ejecuta una instancia de la aplicación propuesta en el sistema operativo Android y el segundo que ejecuta una instancia de dicha aplicación en el sistema operativo iOS. De esta figura se puede observar tres aspectos:

- Los dispositivos se encuentran en la misma sesión colaborativa que se distingue por el identificador de sesión.
- Las barras de conexión denotan el lado de acoplamiento de los dispositivos.
- El fondo del espacio de trabajo colaborativo es el mismo para toda la sesión.

Cuando el usuario inicia la interacción creando objetos (ver Figura 5.14) en el espacio de trabajo, el servicio de presencia actualiza y envía los nuevos elementos a los demás dispositivos acoplados. Si por alguna causa el dispositivo se desconecta de la red local (ver Figura 5.15), el servicio de presencia verifica la última vez de la recepción de los elementos recibidos comparando el tiempo de recepción con el tiempo local del dispositivo.

El marco de desarrollo propuesto requiere una constante conexión con la red local, si la sesión detecta que no hay red, la aplicación es forzada a terminar su ejecución. Mientras que en los

Algoritmo 21 Lógica de la clase `WorkSpace` para Android e iOS

```
1: class WorkSpace
2:   Generar color aleatorio y establecerlo como fondo del espacio de trabajo
3:   sessionID ← Generar un identificador de sesión
4:
5:   procedure DOMENU()
6:     Generar botón Salir
7:     Generar botón Ocultar Menú
8:     Generar botón Configuración
9:     Generar botón Agregar Objeto
10:    Agregar listeners a los botones
11:  end procedure
12:
13:  procedure CREAROBJETO(texto)
14:    Color ← Generar color aleatorio
15:    Imagen ← Crear imagen con el color generado y el texto
16:    Agregar listener a Imagen
17:    if existe un dispositivo acoplado then
18:      Mensaje ← Agregar color y texto
19:      Enviar mensaje
20:      Eliminar imagen
21:    end if
22:  end procedure
23:
24:  procedure CREAROBJETOEXTERNO(texto, color)
25:    Imagen ← Crear imagen con el color y el texto recibido
26:    Agregar listener a la imagen
27:    if existe un dispositivo acoplado then
28:      Mensaje ← Agregar color y texto
29:      Enviar mensaje
30:      Eliminar imagen
31:    end if
32:  end procedure
33:
34:  procedure GENERARCOLORFONDO()
35:    Color ← Generar color aleatorio
36:    Establecer color de fondo del espacio de trabajo
37:    if existe una sesión colaborativa then
38:      Enviar color de fondo a los integrantes de la sesión colaborativa
39:    end if
40:  end procedure
41: end class
```

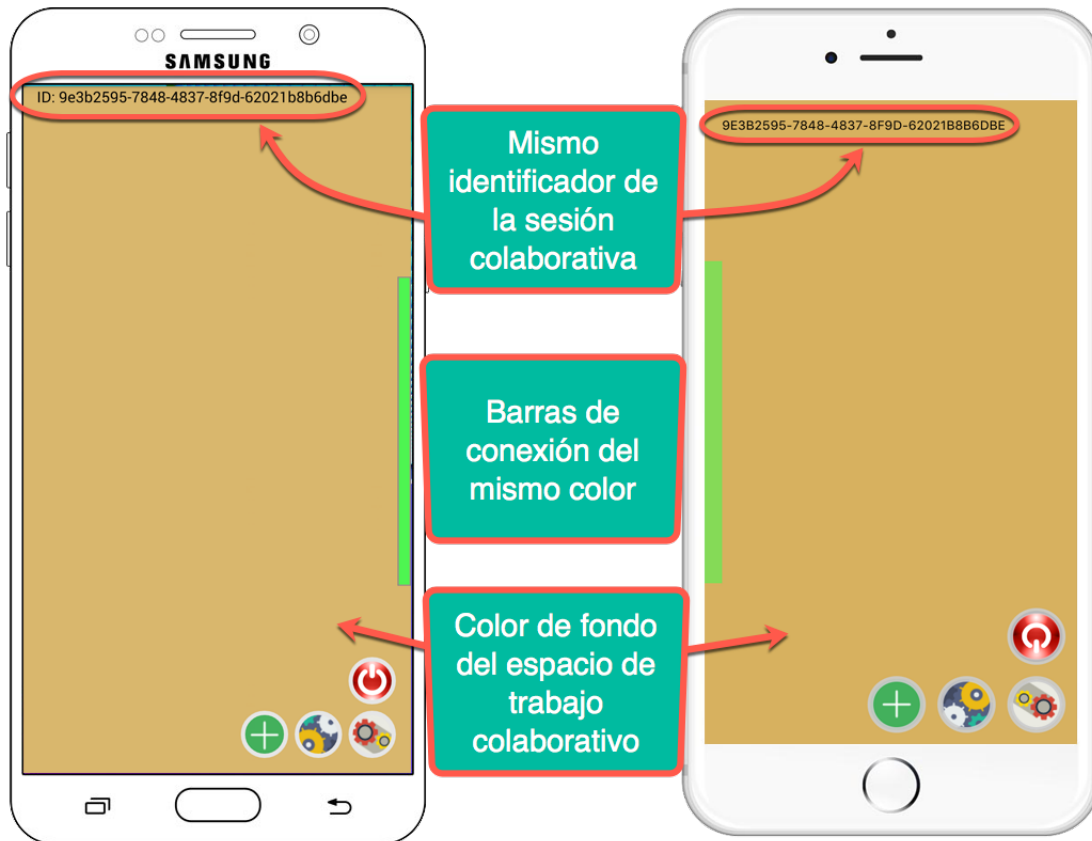


Figura 5.13: Estado inicial de una sesión colaborativa

demás dispositivos se mostrara un mensaje de recuperación de los objetos que se encontraban en el dispositivo que se desconecto (ver Figura 5.16). El usuario tiene la ultima decisión si quiere recuperar los objetos. Si el usuario quiere recuperar los objetos bastara hacer click en la opción *YES* del mensaje. Los objetos se mostrarán en el espacio de trabajo colaborativo y la aplicación continuara con su ejecución normal (ver Figura 5.17).

5.6. Problemas en el desarrollo

Algunos problemas encontrados al proponer la aplicación de validación fueron los siguientes:

- Los colores que se le asignan a las barras de conexión que indican que existe un dispositivo acoplado, a veces al generar el color, este se puede repetir y puede dar origen a una confusión por parte del usuario.
- Los colores que se generan para los objetos, barras de acoplamiento, fondo de el espacio de trabajo y mensajes pueden variar de dispositivo a dispositivo, dependiendo del tipo de pantalla que implementen (e.g., AMOLED, Retina, OLED, IPS, LCD).
- Los objetos generados toman la tipografía del sistema operativo en el que se implementa la aplicación, es decir que el texto que se muestra en el sistema operativo Android es distinto al



Figura 5.14: Interacción en el espacio de trabajo colaborativo

del sistema operativo iOS (i.e., en Android la tipografía default es Sans, Serif, Monospace y en iOS son Helvetica, San Francisco).

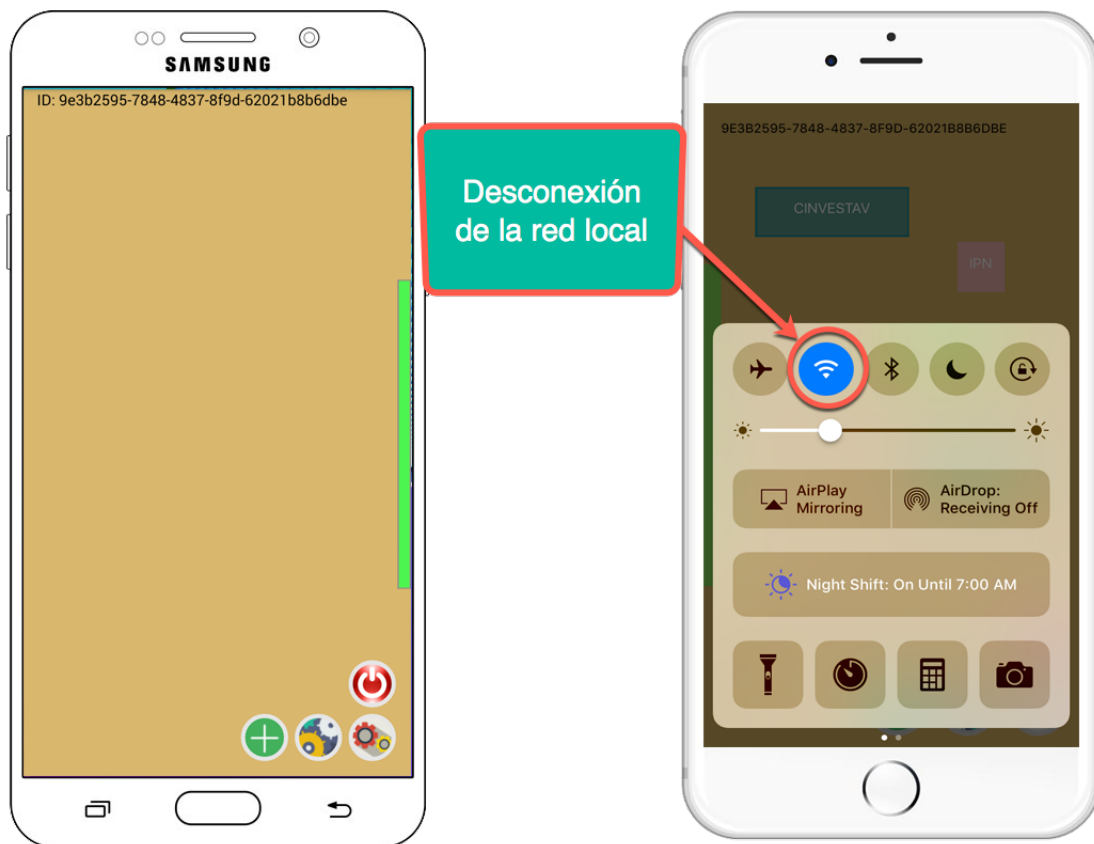


Figura 5.15: Servicio de presencia verificando la sesión colaborativa

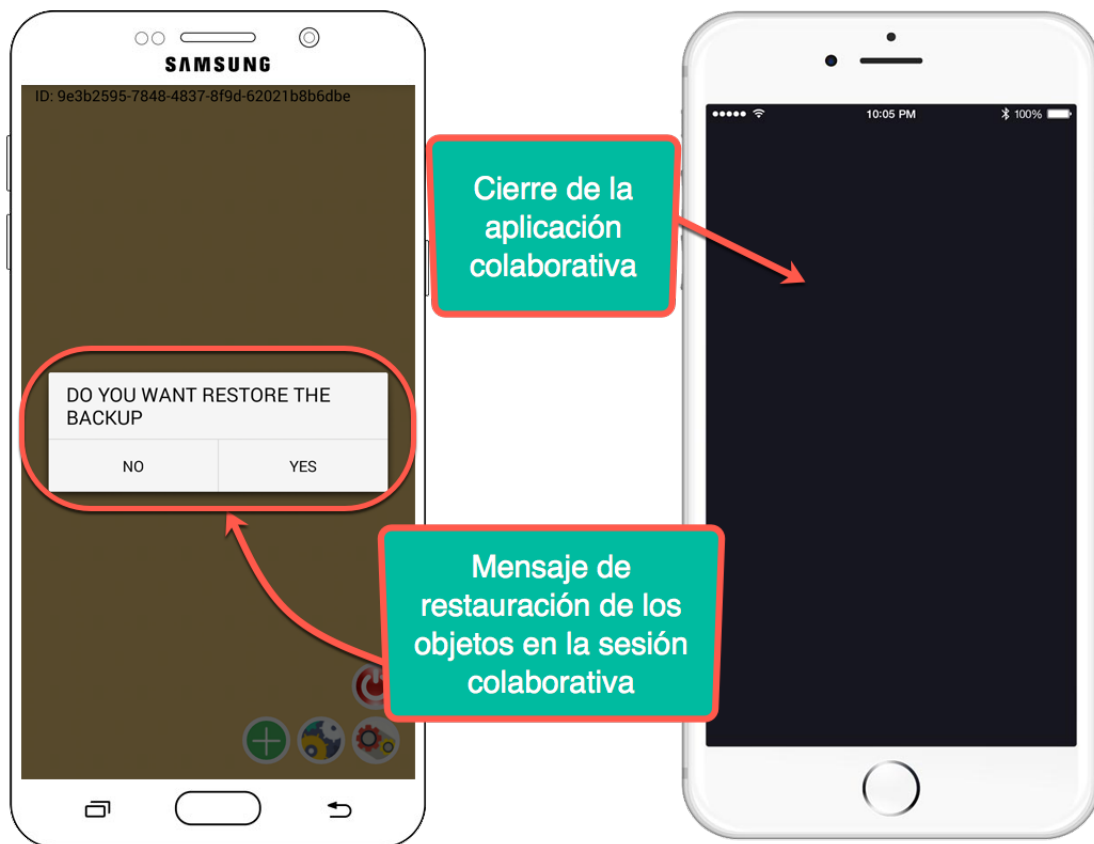


Figura 5.16: Mecanismo de recuperación de objetos en una sesión colaborativa

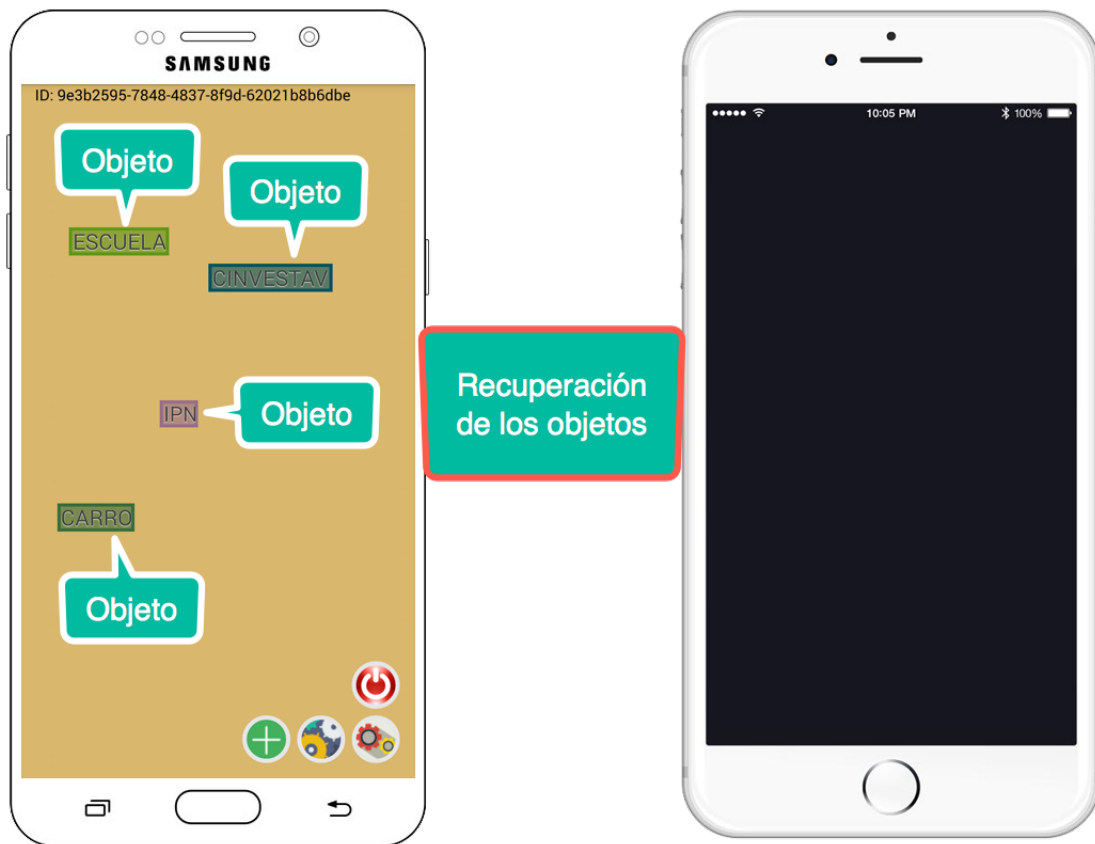


Figura 5.17: Recuperación de los objetos en una sesión colaborativa

Capítulo 6

Pruebas y resultados

En este capítulo se describen las pruebas realizadas al soporte propuesto en esta tesis. Por medio de estas pruebas se verifico primeramente el uso de la aplicación desarrollada (ver Capítulo 5) sujeta a las siguientes evaluaciones: 1) cuando la aplicación se encuentra en una sesión individual; y 2) cuando la aplicación se encuentra en una sesión colaborativa. Los resultados son representados por medio de gráficas junto con sus respectivas interpretaciones.

6.1. Pruebas finales

Con el fin de evaluar diversos aspectos del soporte, así como identificar posibles mejoras respecto a los métodos de interacción y las interfaces de usuario que se muestran en la aplicación de ejemplo. A continuación iremos detallando cada una de las pruebas realizadas, junto con los resultados obtenidos.

6.1.1. Tipos de dispositivos usados en las pruebas

Las pruebas se hicieron utilizando los dispositivos móviles como los que se muestran en la tabla 6.1.

Marca de dispositivo	Tipo de dispositivo	Sistema Operativo	Versión	Resolución de pantalla
Samsung galaxy tab 2	tableta	Android	5.1.1	800 x 1280 pixels
Samsung galaxy s3	<i>smartphone</i>	Android	4.3	720 x 1280 pixels
Samsung galaxy s3 mini	<i>smartphone</i>	Android	4.1.2	480 x 800 pixels
BLU studio c mini	<i>smartphone</i>	Android	4.4.2	480 x 800 pixels
iPad 2	tableta	iOS	9.3.5	1024 x 768 pixels
iPhone 5s	<i>smartphone</i>	iOS	10.1.1	640 x 1136 pixels
iPhone 4	<i>smartphone</i>	iOS	7.1.2	960 x 640 pixels

Tabla 6.1: Dispositivos usados en las pruebas

Como podemos observar en la Tabla 6.1, se muestran las principales características de los

dispositivos móviles para las pruebas que realizamos como: el tipo de sistema operativo y la versión que tienen instalados, la resolución de pantalla y el tipo de dispositivo móvil (teléfono inteligente y tableta).

6.1.2. Evaluación de la sesión individual

Las pruebas para evaluar una sesión individual, los criterios a evaluar son los siguientes:

- Recepción de mensajes de identificación.
- Recepción de mensajes de abandono.

Para esta prueba se utilizó un programa para el análisis y monitoreo de una red local llamado *Wireshark* [Wireshark, 2016] y se usaron 7 dispositivos móviles de los cuales 3 ejecutan el sistema operativo iOS y 4 el sistema operativo Android.

En la figura 6.1 se muestra la captura de tráfico a través de una red local, en la cual se puede observar el filtro que se usa en el programa *Wireshark*, para obtener solo los resultados de tipo *broadcast*, la IP del dispositivo que emite el mensaje, la IP del dispositivo que recibe el mensaje, el protocolo, el puerto de destino, así como el mensaje que se transmite. En la figura 6.2 se muestra el porcentaje del 99.07% de mensajes que si llegan al destinatario.

The screenshot shows the Wireshark interface with the following components:

- Filter:** (eth.dst[0]&1)
- Packet List Table:**

No.	Time	Source	Destination	Protocol	Length	Info
426	68.588649	192.168.0.3	192.168.0.255	ADwin Conf...	146	
427	68.590472	192.168.0.14	192.168.0.255	UDP	148	47786-11001 Len=106
430	68.690995	192.168.0.7	192.168.0.255	UDP	148	60863-11001 Len=106
431	68.998266	192.168.0.15	192.168.0.255	UDP	148	58486-11001 Len=106
432	69.203429	192.168.0.4	192.168.0.255	UDP	147	39350-11001 Len=105
433	69.612576	192.168.0.3	192.168.0.255	ADwin Conf...	146	
434	69.714913	192.168.0.14	192.168.0.255	UDP	148	47786-11001 Len=106
435	69.919932	192.168.0.7	192.168.0.255	UDP	148	60863-11001 Len=106
436	70.125559	192.168.0.15	192.168.0.255	UDP	148	58486-11001 Len=106
437	70.226936	192.168.0.4	192.168.0.255	UDP	147	39350-11001 Len=105
438	70.636584	192.168.0.3	192.168.0.255	ADwin Conf...	146	
439	70.943768	192.168.0.14	192.168.0.255	UDP	148	60863-11001 Len=106
440	71.046239	192.168.0.7	192.168.0.255	UDP	148	60863-11001 Len=106
441	71.091064	192.168.0.2	255.255.255.255	DB-LS		ppbox LAN sync Dis
442	71.091473	192.168.0.2	192.168.0.255	DB-LS		ppbox LAN sync Dis
443	71.251032	192.168.0.15	192.168.0.255	UDP	148	58486-11001 Len=106
444	71.252686	192.168.0.4	192.168.0.255	UDP	147	39350-11001 Len=105
445	71.353370	192.168.0.8	192.168.0.255	UDP	143	10001-11001 Len=101
446	71.660643	192.168.0.8	192.168.0.255	ADwin Conf...	146	
447	72.070245	192.168.0.8	192.168.0.255	UDP	148	47786-11001 Len=106
448	72.072052	192.168.0.8	192.168.0.255	UDP	148	60863-11001 Len=106
- Packet Details (Frame 444):**
 - Ethernet II, Src: MurataMa_c9:8a:3f (5c:f8:a1:c9:8a:3f), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
 - Internet Protocol Version 4, Src: 192.168.0.4, Dst: 192.168.0.255
 - User Datagram Protocol, Src Port: 39350, Dst Port: 11001
 - Data (105 bytes)
- Raw Data:**

```

0000 ff ff ff ff ff ff 5c f8 a1 c9 8a 3f 08 00 45 00
0010 00 85 00 00 40 00 00 11 b8 14 c0 a8 00 04 c0 a8
0020 00 ff 99 b6 2a f9 00 71 6b dd 7b 22 54 59 50 45
0030 44 45 56 49 43 45 22 3a 22 41 4e 44 52 4f 49 44
0040 22 2c 22 44 45 56 49 43 45 22 3a 22 53 61 6d 73
0050 75 6e 67 5f 47 54 2d 4e 37 31 30 30 22 2c 22 55
0060 44 50 22 3a 36 34 39 39 39 2c 22 54 43 50 22 3a
0070 36 32 33 34 30 2c 22 55 50 22 3a 74 72 75 65 2c
0080 22 49 50 22 3a 22 31 39 32 2e 31 36 38 2e 30 2e
0090 34 22 7d

```

Figura 6.1: Filtrado de los mensajes de identificación

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
200.77.168.222	4				0.0000	0.61%	0.0200	4.756
TCP					0.0000	100.00%	0.0200	4.756
80					0.0000	100.00%	0.0200	4.756
200.77.168.173					0.0000	0.61%	0.0400	5.207
TCP					0.0000	100.00%	0.0400	5.207
443	4				0.0000	100.00%	0.0400	5.207
192.168.0.255	323				0.0032	49.54%	0.0300	39.712
UDP	323				0.0032	100.00%	0.0300	39.712
17500	3				0.0000	0.93%	0.0100	11.004
11001	320				0.0032	99.07%	0.0300	39.712
192.168.0.2	138				0.0014	21.17%	0.1500	77.685
UDP					0.0001	7.25%	0.0300	9.697
64695					0.0000	10.00%	0.0100	18.239
55929					0.0000	10.00%	0.0100	77.665
52916					0.0000	10.00%	0.0100	17.696
52521	1				0.0000	10.00%	0.0100	77.665
51664	4				0.0000	40.00%	0.0300	9.697
40004	1				0.0000	10.00%	0.0300	18.239

Figura 6.2: Porcentaje de mensajes de identificación recibidos

En la figura 6.3 se muestra la captura de los mensajes de abandono a través de una red local, en la cual se puede observar el filtro que se usa en el programa *Wireshark*, para obtener solo los resultados de tipo *broadcast*, la IP del dispositivo que emite el mensaje, la IP del dispositivo que recibe el mensaje, el protocolo, el puerto de destino, así como el mensaje que se transmite. En la figura 6.4 se muestra el porcentaje del 98.52% de mensajes que si llegan al destinatario.

6.1.3. Evaluación de la sesión colaborativa

Para la sesión colaborativa se hicieron dos tipos de pruebas, la primera prueba consiste en verificar los mensajes enviados en la sesión colaborativa como: actualización de objetos (servicio de actualización), cambio de color del fondo del espacio de trabajo (servicio de actualización), mensaje de presencia (servicio de presencia), actualización de listas de grupo (servicio de comunicación). A continuación detallamos estas pruebas:

- **Actualización de objetos:** esta prueba consiste en verificar la integridad de los mensajes de actualización de los objetos, en la Figura 6.5.a se observa el mensaje de actualización y el contenido como: texto y color.
- **Cambio de color del fondo del espacio de trabajo:** esta prueba consiste en verificar la integridad de los mensajes de cambio de color del fondo de trabajo colaborativo, en la Figura 6.5.b se observa el mensaje de cambio de color en formato RGB.
- **Mensaje de presencia:** esta prueba consiste en verificar la integridad de los mensajes de presencia, en la Figura 6.5.c se observa el mensaje de presencia y el contenido como: IP, objeto y el estado del mensaje.

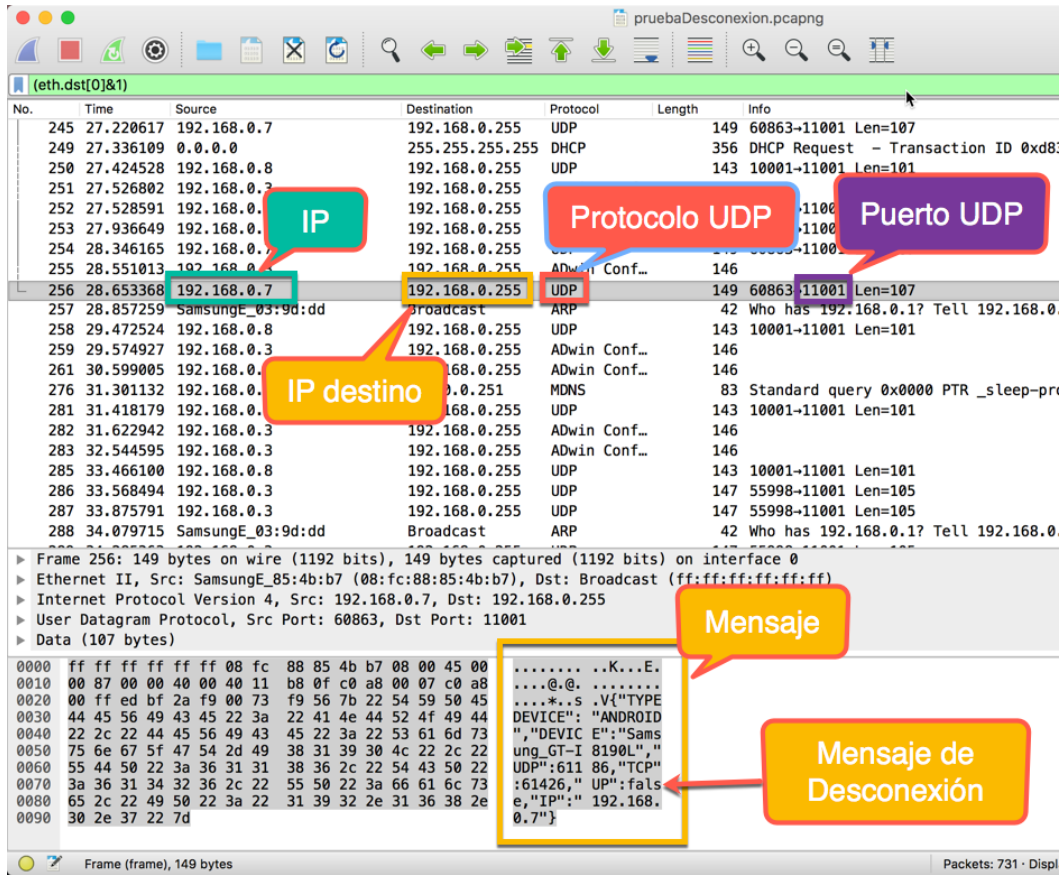


Figura 6.3: Filtrado de los mensajes de abandono

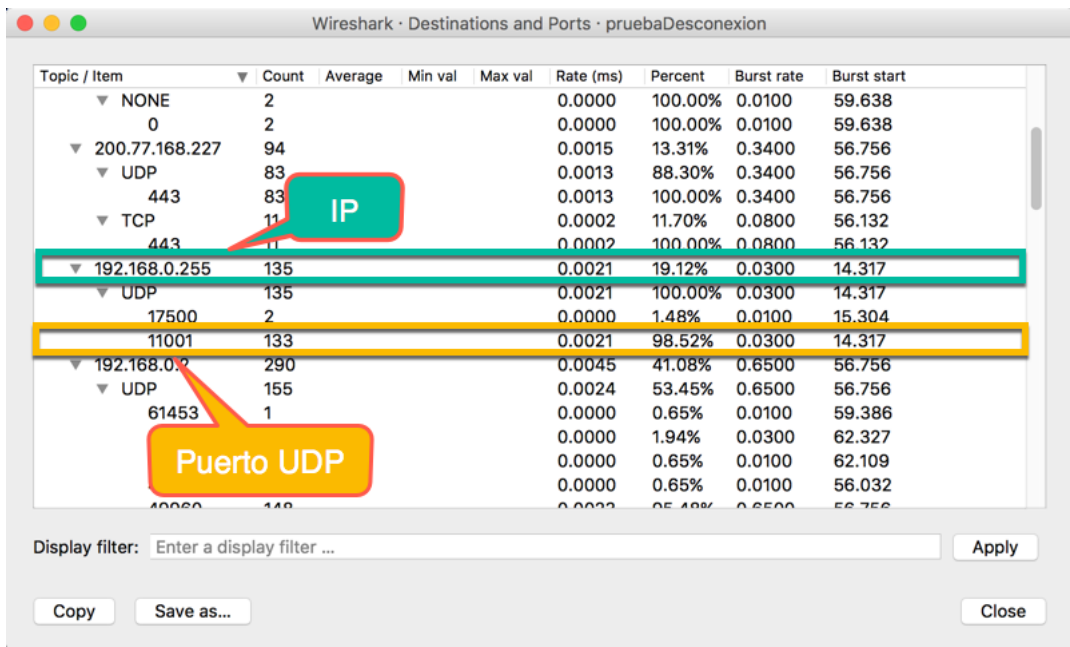


Figura 6.4: Porcentaje de mensajes de identificación abandono

- **Actualización de la lista de grupo:** esta prueba consiste en verificar la integridad de los mensajes de actualización de la lista del grupo, en la Figura 6.5.d se observa el mensaje de actualización de la lista del grupo y el contenido como:

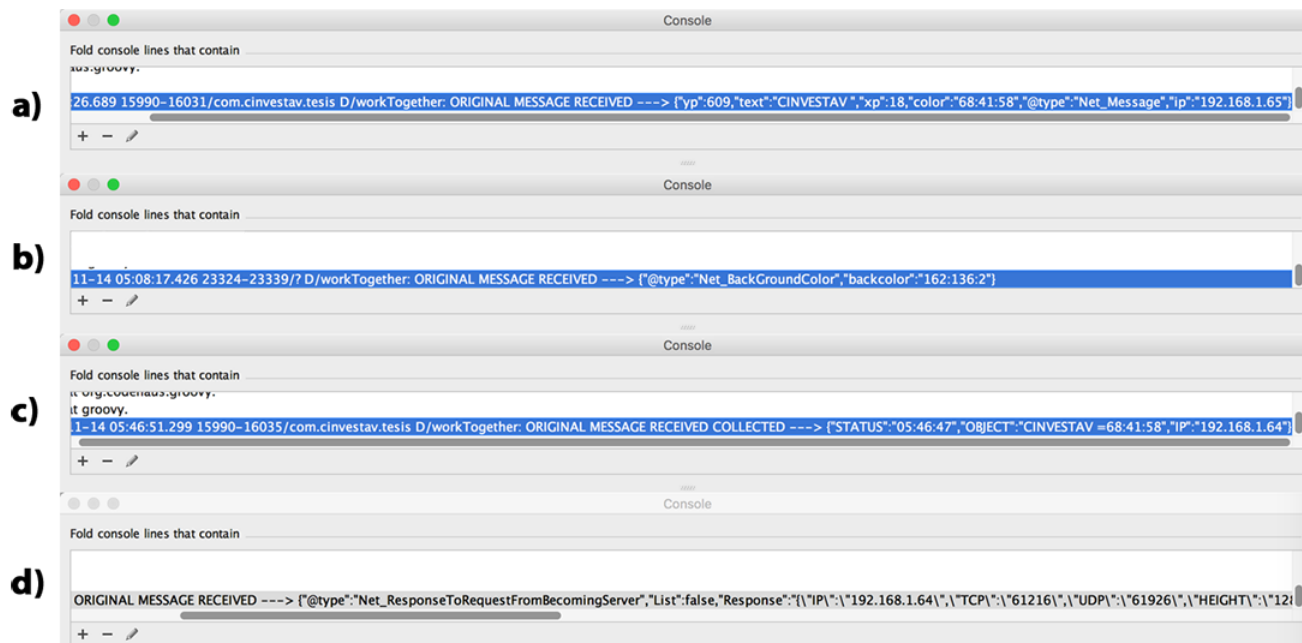


Figura 6.5: Mensajes de actualización, color de fondo, presencia y actualización de listas

La segunda prueba consiste en verificar el acoplamiento de los dispositivos. Para esta prueba se usaron 7 dispositivos, los cuales están enumerados del 1 al 7 (ver Figura 6.6). La configuración para el acoplamiento para el gesto de arrastre, inclinación y combinado (e.g., arrastre y inclinación o inclinación y arrastre) es la siguiente:

- dispositivo 1 del lado izquierdo con el dispositivo 2 del lado derecho.
- dispositivo 1 del lado derecho con el dispositivo 3 del lado izquierdo.
- dispositivo 1 del lado superior con el dispositivo 4 del lado inferior.
- dispositivo 1 del lado superior con el dispositivo 5 del lado inferior.
- dispositivo 1 del lado inferior con el dispositivo 6 del lado superior.
- dispositivo 1 del lado inferior con el dispositivo 7 del lado superior.

Cabe mencionar que la configuración usada anterior es solo para el dispositivo 1 y puede cambiar para los demás dispositivos si se desea acoplar más dispositivos.

En la Figura 6.7 se muestra la gráfica con el número de intentos fallidos para acoplar un dispositivo con el gesto de arrastre, cabe mencionar que el dispositivo 1 se encuentra en la parte central (ver Figura 6.6) por lo que en este dispositivo se realizaron la mayor parte de acoplamientos.

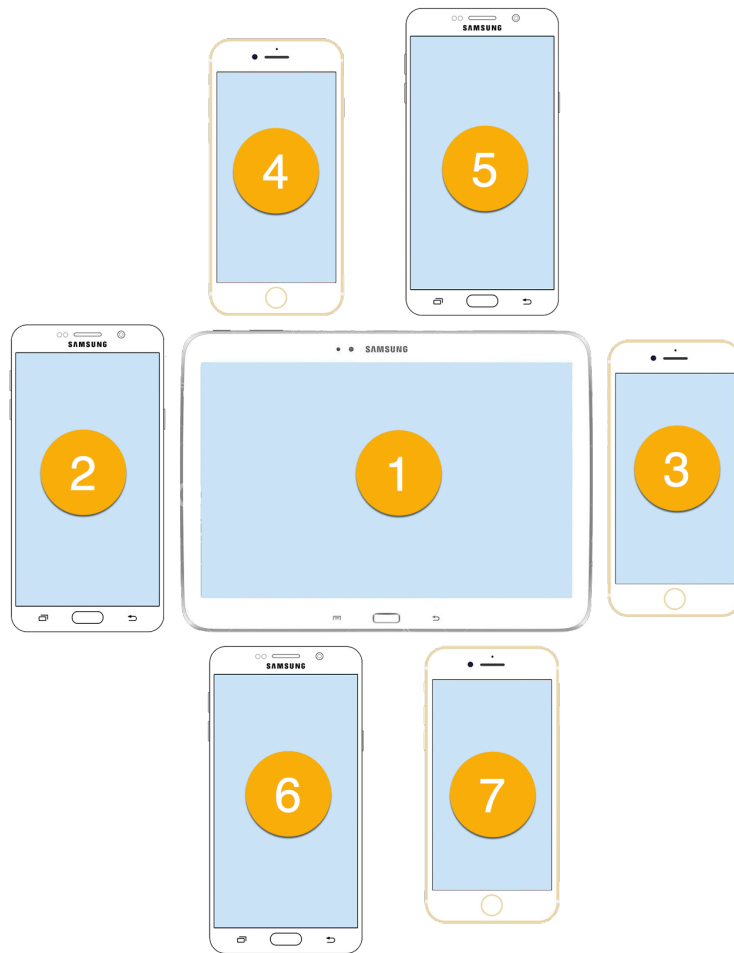


Figura 6.6: Configuración de dispositivos



Figura 6.7: Intentos de acoplamiento usando el gesto de arrastre



Figura 6.8: Intentos de acoplamiento usando el gesto de inclinación

En la Figura 6.8 se muestra la gráfica con el número de intentos fallidos para acoplar un dispositivo con el gesto de inclinación.

En la Figura 6.9 se muestra la gráfica con el número de intentos fallidos para acoplar un dispositivo con ambos gestos (e.g., arrastre e inclinación o inclinación e arrastre).

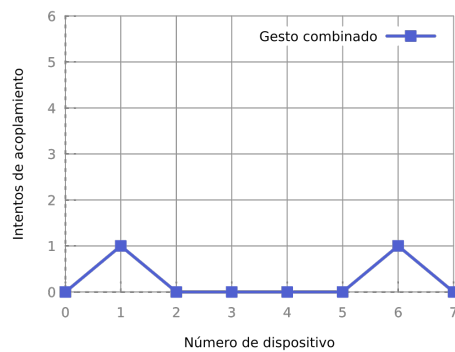


Figura 6.9: Intentos de acoplamiento usando ambos gestos (arrastre e inclinación)

Como se puede observar en las Figuras 6.7, 6.8 y 6.9, el número de errores es mínimo debido a que los mensajes se pueden perder en la red o en la recepción de estos mensajes en los dispositivos llegarán a estar incompletos.

Capítulo 7

Conclusiones y trabajo a futuro

Para concluir el presente trabajo, haremos una recapitulación de la problemática resuelta en esta tesis en la Sección 7.1. Después, se presentan las conclusiones a las que se llegó como resultado del desarrollo de este proyecto ver Sección 7.2. Finalmente, se describe el trabajo que queda por desarrollar, haciendo énfasis en algunas limitaciones, actualizaciones y extensiones que se pueden implementar al trabajo realizado en la Sección 7.3.

7.1. Resumen de la proplemática

El presente tema tesis se inscribe en el campo de investigación denominado *Computer Supported Cooperative Work* (CSCW), el cual estudia tanto los aspectos sociales de las actividades individuales y colectivas, como los aspectos tecnológicos de la información, con el objetivo de construir aplicaciones que den soporte a la colaboración entre personas. El interés de la tesis propuesta se centra en los aspectos tecnológicos del CSCW, con el fin de diseñar e implementar un marco de desarrollo que permita crear aplicaciones colaborativas conscientes de contexto, mediante las cuales los miembros de un grupo interactúen cara a cara y colaboren a través de sus dispositivos móviles.

Uno de los objetivos de la presente tesis consistía precisamente en tomar ventaja de la tecnología desde el punto de vista de los dispositivos móviles, utilizándolos como un medio a través del cual sea posible llevar a cabo interacciones más dinámicas y espontáneas. Concretamente, la forma en cómo se ha manifestado el desarrollo tecnológico en dichos dispositivos (sensores, adaptadores de red, poder de procesamiento) y su creciente disponibilidad han provocado un interés por desarrollar aplicaciones de tipo comercial y de investigación, con formas de interacción más complejas e intuitivas.

El ofrecer herramientas que permitan realizar sesiones explotando las capacidades tecnológicas de los dispositivos móviles para brindar formas de interacción novedosas y soporte para la mayor parte de los dispositivos, sigue siendo un reto. Actualmente, los ecosistemas de dispositivos, así como sus respectivas aplicaciones, se han vuelto más complejos y heterogéneos, abriendo una brecha de oportunidades para diversos campos de investigación, tales como: el Cómputo Ubicuo, el Cómputo Móvil, los Sistemas Colaborativos, las Interacciones Hombre-Máquina, entre otros más.

7.2. Conclusiones

La principal contribución de esta tesis es el diseño e implementación del marco de desarrollo para aplicaciones móviles, el cual facilita la implementación de aplicaciones conscientes del contexto que soportan el trabajo colaborativo cara a cara. Los componentes de este marco han sido validados, mediante la implementación de una aplicación destinada a crear sesiones colaborativas. La aplicación resultante es capaz de ejecutarse en arreglos de dispositivos móviles y en plataformas como Android e iOS.

En particular, en este trabajo de investigación describimos un marco de desarrollo heterogéneo y multi-plataforma para dispositivos móviles (teléfonos inteligentes y tabletas), que facilita la creación de sesiones colaborativas en espacios de trabajo. El diseño se realizó de tal forma, que pueda ser replicado en otras plataformas de sistemas operativos móviles. Adicionalmente el marco de desarrollo se entrega como un proyecto de tipo API para Android e iOS. Dicha API (ver Sección 5.1) cuenta con la documentación necesaria para poder crear nuevas aplicaciones de tipo colaborativo.

El marco de desarrollo cuenta con cuatro servicios descubrimiento, comunicación, actualización y de presencia, los cuales pueden ser utilizados, si así se requiriera, de forma individual para aumentar la funcionalidad de cualquier tipo de aplicaciones, cuyo modelo de tareas pueda beneficiarse por el uso de alguno de los servicios.

En cuanto a los dos tipos de espacios (individual y colaborativo) que se estudiaron en esta tesis pueden ser experimentados por las aplicaciones creadas por el marco de desarrollo conscientes del contexto, el marco de desarrollo permite implementar estos dos espacios en diferentes situaciones en que los desarrolladores consideren adecuado.

Otra contribución es que algunos de los problemas de implementación solucionados pueden ser tomados como ejemplo para desarrollos similares, e.g., el uso de métodos similares para el desarrollo del soporte de forma tal que se tenga compatibilidad con distintos dispositivos (Android e iOS). Estos ejemplos se pueden tomar tanto a nivel de código fuente como de diseño.

Una contribución secundaria y más abstracta es la arquitectura propuesta para este marco de desarrollo, ya que se basó en el análisis de los requerimientos particulares de los sistemas móviles. Esta arquitectura fue propuesta para desarrollar la biblioteca *WorkTogether*, la cual fue implementada y probada por medio de una aplicación de prueba.

7.3. Trabajo a futuro

Durante el diseño y la implementación del marco de desarrollo, se identificaron múltiples extensiones que lo podrán mejorar. A continuación se enumeran las posibles mejoras:

- Es posible tener más sesiones colaborativas mediante la implementación de un mecanismo que permita seleccionar una sesión colaborativa activa, así mismo poder asignar un nombre a la sesión para poderla diferenciar de otras, además de aumentar el número de sesiones colaborativas para tener una mayor diversidad de aplicaciones creadas con el marco de desarrollo propuesto ejecutándose en una misma red local.
- Con el objetivo de fortalecer el marco de desarrollo, agregar un módulo de seguridad que implemente algún método de cifrado (e.g., simétricos, asimétricos, RSA), con el fin de

ofrecer una mayor seguridad en el envío y recepción de mensajes a través de la red local.

- Dada la amplia adopción de los teléfonos inteligentes y tabletas, el tamaño de pantalla que ofrecen se obtienen mayores beneficios como la usabilidad, eficacia y eficiencia [Raptis et al., 2013], puede ser aprovechado para mejorar el marco de desarrollo al agregar un método en el cual se pueda emplear al máximo el área de pantalla y no estén sujetos al acoplar dispositivos (cuatro para smartphones y seis para tabletas).
- Si se implementa algún módulo que se encargue de verificar cual es la mejor ubicación de la redistribución de los objetos de configuración del soporte cuando existe una sesión colaborativa e.g., la aplicación de ejemplo desarrollada con el marco de desarrollo propuesto en ésta tesis, implementa un menú de configuración, este menú se visualiza en cada uno de dispositivos que se encuentran en una sesión colaborativa, si se implementara este módulo se tendría solo un menú para toda la sesión colaborativa.
- Con el modelo propuesto (ver Sección 3.2) para este marco de desarrollo, se pueden agregar más plataformas de sistemas operativos móviles para cubrir la mayor parte de la cuota de los dispositivos móviles que se encuentran actualmente en el mercado.
- Es posible extender el método de recuperación cuando algún dispositivo que se encuentra en una sesión colaborativa y por alguna circunstancia este no avisa a los demás dispositivos que abandona la sesión colaborativa, se puede implementar mas métodos de recuperación como: *logs* o *backups*.
- Si se puede implementar un método de desconexión de una sesión colaborativa sin tener que cerrar la aplicación e.g., cuando un dispositivo se encuentra en una sesión colaborativa y ya no le interesa pertenecer a esa sesión.

Bibliografía

- [ACM, 2016] ACM (2016). The ACM Computing Classification System (CCS). <http://www.acm.org/about/class/class/2012>.
- [Amo et al., 2005] Amo, F. A., Normand, L. M., and Pérez, F. J. S. (2005). *Introducción a la ingeniería de software*. Delta Publicaciones, 1ra edition.
- [Android, 2016] Android (2016). Android API level-24 MotionEvent. <https://developer.android.com/reference/android/view/MotionEvent.html>.
- [Baldauf et al., 2007] Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, pages 263–277.
- [Baran, 1962] Baran, P. (1962). On distributed communications networks. pages 3–5.
- [Baran, 1964] Baran, P. (1964). On distributed communications networks. *IEEE Transactions on Communications Systems*, 12:1–9.
- [Bass et al., 2012] Bass, L., Clements, P., and Kazman, R. (2012). *Software Architecture in Practice*. Addison-Wesley Professional, 3ra edition.
- [Campos et al., 2013] Campos, P., Ferreira, A., and Lucero, A. (2013). Collaboration meets interactive surfaces: walls, tables, tablets, and phones. *International conference on Interactive tabletops and surfaces*.
- [Card et al., 1983] Card, S. K., Newelland, A., and Moran, T. P. (1983). *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum, 1st edition.
- [Carlsson and Hagsand, 1993] Carlsson, C. and Hagsand, O. (1993). Dive a multi-user virtual reality system. *IEEE Virtual Reality Annual International Symposium*, pages 394–400.
- [Castro, 2014] Castro, M. (2014). Transición del trabajo individual al colaborativo mediante técnicas de remodelación y redistribución plásticas. Master’s thesis, Centro de Investigación y de Estudios Avanzados del IPN, Departamento de Computación, México D.F.
- [Chen and Kotz, 2000] Chen, G. and Kotz, D. (2000). A survey of context-aware mobile computing research. *Technical Report A Survey of Context-Aware Mobile Computing Research*.
- [Coutaz and Calvary, 2012] Coutaz, J. and Calvary, G. (2012). Hci and software engineering for user interface plasticity. In *The Human-Computer Handbook - Fundamentals, Evolving*

- Technologies and Emerging Applications*, pages 1195–1220. CRC Press Taylor and Francis Group.
- [Curtis and Nichols, 1994] Curtis, P. and Nichols, D. (1994). Muds grow up: Social virtual reality in the real world. *IEEE Xplore Conference*, pages 139–200.
- [Denning and Yaholkovsky, 2008] Denning, P. and Yaholkovsky, P. (2008). Getting to we. *Communications of the ACM*, 51:19–24.
- [Developer Economics, 2014] Developer Economics (2014). The State of the Developer Nation. <https://www.developereconomics.com/reports>.
- [Dey and Abowd, 2000] Dey, A. K. and Abowd, G. D. (2000). Towards a better understanding of context and context-awareness. *Proceedings of the Workshop on The What, Who, Where, When, and How of Context-Awareness within*, pages 1–12.
- [Ellen Taylor-Powell, 2001] Ellen Taylor-Powell (2001). Colaboration. http://www.earlychildhoodiowa.org/files/annual_reports/2001/Collaboration.pdf.
- [Elmqvist, 2011] Elmqvist, N. (2011). Distributed user interfaces: State of the art. *Workshop on Distributed User Interfaces 2011*.
- [Gaver et al., 1992] Gaver, B., Moran, T., MacLean, A., Lovstrand, L., Dourish, P., Carter, K., and Buxton, B. (1992). Realizing a video environment: Europarc’s rave system. *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 207–217.
- [Gaver, 1992] Gaver, W. W. (1992). The affordances of media spaces for collaboration. *ACM conference on Computer-supported cooperative work*, pages 17–24.
- [German Army, 1992] German Army (1992). V-Modell XT. http://www.cio.bund.de/Web/DE/Architekturen-und-Standards/V-Modell-XT/vmodell_xt_node.html.
- [Goh et al., 2014] Goh, W. B., Chen, M., Trinh, C. H., Tan, J., and Shou, W. (2014). The moy framework for collaborative play design in integrated shared and private interactive spaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 391–400.
- [Gray, 1990] Gray, B. (1990). Collaborating: Finding common ground for multiparty problems. *The Academy of Management Review*, 15:545–547.
- [Grudin, 1994] Grudin, J. (1994). *Computer-Supported Cooperative Work: History and Focus*. IEEE.
- [Harrison et al., 1996] Harrison, S., , and Dourish, P. (1996). Re-place-ing space: The roles of place and space in collaborative systems. *Computer supported cooperative work*, pages 67–76.
- [Harrison and Dourish, 1996] Harrison, S. and Dourish, P. (1996). Re-place-ing space: the roles of place and space in collaborative systems. *Proceedings of Conference on Computer Supported Coopeartive Work*, pages 67–76.

- [Hinckley et al., 2014] Hinckley, K., Ramos, G., Guimbretiere, F., Baudisch, P., and Smith, M. (2014). Stitching pen gestures that span multiple displays. *Proceedings of the working conference on Advanced visual interfaces*.
- [Hu and Xu, 2010] Hu, C. and Xu, C. (2010). Multi-granularity business process descriptions and intelligent task distribution. *Computer Supported Cooperative Work in Design*, pages 190–194.
- [IDC, 2016] IDC (2016). Smartphone OS Market Share, 2015 Q2. <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>.
- [Kaheel et al., 2009] Kaheel, A., El-Saban, M., Refaat, M., and Ezz, M. (2009). Mobicast: A system for collaborative event casting using mobile phones. *International Conference on Mobile and Ubiquitous Multimedia*.
- [Lucero et al., 2015] Lucero, A., Clawson, J., Lyons, K., Fischer, J. E., Ashbrook, D., and Robinson, S. (2015). Mobile collocated interactions: from smartphones to wearables. *International Conference on Human-Computer Interaction*.
- [Lucero et al., 2010] Lucero, A., Holopainen, J., and Jokela, T. (2010). Collaborative use of mobile phones for brainstorming. In de Sá, M., Carriço, L., and Correia, N., editors, *In Proceedings of the 12th Conference on Human Computer Interaction with Mobile Devices and Services*, pages 337–340, Lisbon, Portugal. ACM Press.
- [Malone and Crowston, 1990] Malone, T. W. and Crowston, K. (1990). What is coordination theory and how can it help design cooperative work systems. *Proceedings of Conference on Computer Supported Cooperative Work*, pages 157–370.
- [Nelson, 1994] Nelson, C. (1994). A forum for fitting the task. *IEEE Computer Society Press*, pages 104–109.
- [Paul, 1991] Paul, W. (1991). Computer supported cooperative work: An introduction. *European Conference on Computer Supported Cooperative Work*.
- [Raptis et al., 2013] Raptis, D., Tselios, N., Kjeldskov, J., and Skov, M. (2013). Does size matter?: investigating the impact of mobile phone screen size on users’ perceived usability, effectiveness and efficiency. *Human-computer interaction with mobile devices and services*, pages 127–136.
- [Rekimoto, 1997] Rekimoto, J. (1997). Pick-and-drop: A direct manipulation technique for multiple computer environments. *User interface software and technology*, pages 31–39.
- [Rekimoto and Saitoh, 1999] Rekimoto, J. and Saitoh, M. (1999). Augmented surfaces: A spatially continuous work space for hybrid computing environments. *Human Factors in Computing Systems*, pages 378–385.
- [Roberts and Bradley, 1991] Roberts, N. and Bradley, R. T. (1991). Stakeholder collaboration and innovation: A study of public policy initiation at the state level. *The Journal of Applied Behavioral Science*, 27:209–227.

- [Root, 1988] Root, R. (1988). Design of a multi-media vehicle for social browsing. *ACM conference on Computer supported cooperative work*, pages 25–38.
- [Schach, 2005] Schach, S. R. (2005). *Análisis y diseño orientado a objetos con UML y el proceso unificado*. McGraw-Hill, 1ra edition.
- [Schilit et al., 1994] Schilit, B., Adams, N., and Want, R. (1994). Context-aware computing applications. *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 95–90.
- [Schmidt, 1991] Schmidt, K. (1991). Riding a tiger or computer supported cooperative work. *Proceedings European of Conference on Computer Supported Cooperative Work*, pages 1–16.
- [Scott London, 2016] Scott London (2016). Building Collaborative Communities. <http://www.scottlondon.com/articles/oncollaboration.html>.
- [Seifert et al., 2012] Seifert, J., Simeone, A. L., Schmidt, D., Reinartz, C., Holleis, P., Wagner, M., Gellersen, H., and Rukzio, E. (2012). Mobisurf: Improving co-located collaboration through integrating mobile devices and interactive surfaces. *International conference on Interactive tabletops and surfaces*.
- [Statista, 2016a] Statista (2016a). Android version market share 2016. <https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/>.
- [Statista, 2016b] Statista (2016b). Global mobile OS market share 2009-2016. <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/>.
- [Statista, 2016c] Statista (2016c). iOS version share of Apple devices worldwide 2016. <https://www.statista.com/statistics/565270/apple-devices-ios-version-share-worldwide/>.
- [Talukder et al., 2010] Talukder, A. K., Ahmed, H., and Yavagal, R. R. (2010). *Mobile Computing - Technology, Applications and Service Creation*. Tata McGraw Hill, 2nd edition.
- [Tandler et al., 2001] Tandler, P., Prante, T., Müller Tomfelde, C., Streitz, N. A., and Steinmetz, R. (2001). Connectables: dynamic coupling of displays for the flexible creation of shared workspaces. In *Proceedings of the The 14th Annual ACM Symposium on User Interface Software and Technology*, pages 11–20, Orlando, Florida, USA. ACM Press.
- [Tanenbaum and Steen, 2008] Tanenbaum, A. S. and Steen, M. V. (2008). *Sistemas Distribuidos – Principios y Paradigmas*. Pearson, 2nd edition.
- [Taylor-Powell, 2008] Taylor-Powell, E. (2008). Evaluating collaboratives: Reaching the potential. *Cooperative Extension*.
- [Vanderdonckt et al., 2008] Vanderdonckt, J., Calvary, G., Coutaz, J., and Stanculescu, A. (2008). Multimodality for plastic user interfaces: Models, methods, and principles. *Signals and Communication Technologies*, pages 64–84.

-
- [Weiser, 1993] Weiser, M. (1993). Some computer science issues in ubiquitous computing. *Communications of the ACM*.
- [Winkler et al., 2011] Winkler, C., Reinartz, C., Nowacka, D., and Rukzio, E. (2011). Interactive phone call: Synchronous remote collaboration and projected interactive surfaces. *International Conference on Interactive Tabletops and Surfaces*.
- [Wireshark, 2016] Wireshark (2016). wireshark. <https://www.wireshark.org/>.
- [Zimmermann et al., 2007] Zimmermann, A., Lorenz, A., and Oppermann, R. (2007). An operational definition of context. *Proceedings of the 6th International and Interdisciplinary Conference on Modeling and using Context*, pages 558—571.