Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA (SEGOVIA)

Grado en Ingeniería Informática de Servicios y Aplicaciones

Estudio de WebRTC y su implementación con J2EE/JavaEE

Alumno: Jose Vicente Rodrigo Delgado

Tutor: Fernando Diaz Gómez

Índice

Índice

Resumen pág. 11

Abstract pág. 12

Índice de tablas, figuras y códigos pág. 15

1.- Introducción pág. 27

1.1.- Motivación pág. 27

1.2.- Organización del documento pág. 28

2.- Objetivos y alcance del sistema pág. 33

2.1.- Objetivo principal pág. 33

2.2.- Estado del arte pág. 33

2.3.- Árbol de características pág. 35

2.3.1.- Diagrama del árbol de características pág. 36

2.3.2- Características principales del proyecto pág. 37

2.4.- Descripción del producto a entregar pág. 38

2.5.- Contenido del CD-ROM pág. 38

3. Web de Comunicacion Real de Transmisión (WebRTC) pág. 41

3.1.- ¿Qué es WebRTC? pág. 41

3.2.- Ventajas e inconvenientes de utilizar WebRTC pág. 42

3.3.- Codecs utilizados por WebRTC pág. 43

3.4.- Navegadores que soportan actualmente la API WebRTC pág. 44

3.5.- La arquitectura de WebRTC pág. 45

3.6.- API JavaScript de WebRTC pág. 46

3.6.1.- getUserMedia (MediaStream) pág. 46

3.6.2.- RTCPeerConnection pág. 47

3.6.1.1.- Propiedades de la Interfaz pág. 47

ÍNDICE

3.6.1.2.- Los manejadores (handler) de eventos pág. 48

3.6.1.3.- Los métodos de API WebRTC pág. 49

3.6.3.- RTCDataChannel pág. 51

3.7.- Tecnologías involucradas en WebRTC pág. 52

3.7.1.- Real time transport protocol (RTP) pág. 52

3.7.2.- User datagram protocol (UDP) pág. 52

3.7.3.- Network address translation (NAT) pág. 53

3.7.4.- Session transversal utilities for NAT (STUN) / Transversal using relays around NAT (TURN) pág. 53

3.7.5.- Interactive connectivity establishement (ICE) pág. 55

3.7.6.- Datagram transport layer security (DTLS) pág. 56

3.7.7.- Session description protocol (SDP) pág. 56

4.- Tecnologías Web pág. 61

4.1.- Html5 pág. 61

4.1.1.- Etiqueta <video> pág. 62

4.1.2.- Etiqueta <canvas> pág. 64

4.2.- JavaScript pág. 65

4.2.1.- Objeto `XMLHttpRequest` pág. 67

4.2.2.- Objeto `JSON` pág. 67

4.3.- CSS pág. 69

4.3.1.- Responsive design pág. 71

4.4.- Java pág. 72

4.5.- J2EE/JavaEE pág. 73

4.6.- WebSocket pág. 74

4.7.- Tomcat pág. 79

4.8.- Jetty pág. 81

5.- Gestión del proyecto pág. 85

5.1.- Estimación de costes económicos y temporales pág. 85
5.1.1.- Estimación por puntos de función pág. 85
5.1.2.- Estimación mediante COCOMO pág. 100
5.1.3.- Presupuesto pág. 104
5.1.4.- Planificación temporal pág. 107
6.- Análisis del sistema pág. 111
6.1.- Introducción pág. 111
6.2.- Actores del sistema pág. 112
6.3.- Casos de uso pág. 112
6.3.1.- Introducción pág. 112
6.3.2.- Diagrama de casos de uso pág. 113
6.3.3.- Especificación pág. 114
6.4.- Requisitos pág. 115
6.4.1.- Requisitos de negocio pág. 115
6.4.2.- Requisitos funcionales pág. 116
6.4.3.- Requesitos no funcionales pág. 119
6.4.4.- Requisitos de información pág. 120
7.- Diseño del software pág. 123
7.1.- Introducción pág. 123
7.2.- Arquitectura física pág. 123
7.2.1.- Modelo cliente-servidor pág. 123
7.2.2.- Arquitectura física Sala Principal pág. 124
7.2.3.- Arquitectura física Habitación WebRTC pág. 125
7.3.- Arquitectura lógica pág. 127
7.3.1.- Diagrama de componentes pág. 127
7.3.2.- Diagrama de clases del servidor (BackEnd) pág. 128
7.3.3.- Diagramas de secuencia pág. 129

ÍNDICE

7.4.- Modelo de datos pág. 135

7.5.- Diagrama de despliegue pág. 136

8.- Implementación del proyecto pág. 139

8.1.- Introducción pág. 139

8.2.- Implementación del WebSocket (BackEnd) pág. 140

8.2.1.- WebRTCServerServletContextListener.java pág. 140

8.2.2.- SignalingWebSocket.java pág. 142

8.3.- Página de Bienvenida y Sala Principal pág. 145

8.3.1.- Eventos Cliente (FrontEnd) y Servidor (BackEnd) pág. 146

8.3.2.- La Sala Principal (FrontEnd) pág. 147

8.3.3.- La Sala Principal (BackEnd) pág. 149

8.4.- Habitación WebRTC Videollamada pág. 150

8.4.1.- Eventos Cliente (FrontEnd) pág. 150

8.4.2.- Habitación WebRTC videollamada (FrontEnd) pág. 151

8.4.3.- Habitación WebRTC videollamada (BackEnd) pág. 159

8.4.4.- Compartir archivos (FrontEnd) pág. 162

9.- Pruebas pág. 169

9.1.- Resultados pág. 169

10.- Manual de instalación y uso pág. 181

10.1.- Manual de instalación y despliegue de la aplicación pág. 181

10.1.1.- Crear el archivo WAR de la aplicación pág. 181

10.1.2.- OpenShift pág. 181

Herramienta Git pág. 183

Herramienta Client Tool pág. 186

Acceso al servidor OpenShift (SSH) pág. 187

10.2.- Manual de usuario pág. 188
10.2.1.- Introducción pág. 188
10.2.2.- Página de Inicio pág. 188
10.2.3.- Sala Principal pág. 189
10.2.4.- Habitación WebRTC pág. 191
11.- Conclusiones y trabajos futuros pág. 197
11.1.- Conclusiones pág. 197
11.2.- Trabajos futuros pág. 197
Bibliografía y referencias pág. 201
Anexo A.- Casos de uso pág. 207
Anexo B.- Requisitos funcionales pág. 231
Anexo C.- Diagrama de dependencias pág. 253
Anexo D.- Glosario de términos Pág. 257

Resumen

Este proyecto tiene como objetivo el estudio de la tecnología **WebRTC**, como implantarla en una plataforma **J2EE/JavaEE** y las herramientas necesarias para este propósito. El proyecto abarca el análisis, diseño, gestión, implementación y desarrollo de una aplicación software para gestionar y realizar videollamadas entre pares con WebRTC, grabar y hacer fotos de la videollamada, comunicación por mensajes de texto (chat) y compartir archivos en tiempo real. El sistema se basa en la señalización bidireccional cliente-servidor/servidor-cliente con un **socket** para el envío y recepción de datos.

Abstract

This project aims the study of **WebRTC** technology, such as implement it in a **J2EE/JavaEE** platform and tools needed for this purpose. The project includes analysis, design, management, implementation and development of a software application to manage and make video calls peer-to-peer with WebRTC, record and take pictures of the video call, communication text messages (chat) and share files in real time. The system is based on the bidirectional signaling client-server/server-client with a **socket** for sending and receiving data.

Índice
de tablas, figuras y códigos

Índice de tablas, figuras y códigos

Figura 2.1.- Ejemplo de aplicación con videollamada. Skype pág. 34

Figura 2.2.- Ejemplo de aplicación con videollamada. Hangouts pág. 34

Figura 2.3.- Árbol de características del proyecto pág. 36

Figura 3.1.- Arquitectura de WebRTC pág. 45

Figura 3.2.- Diagrama servidor STUN/TURN en WebRTC pág. 55

Figura 4.1.- Diagrama ejemplo de un Websocket para la comunicación entre usuarios pág. 74

Figura 4.2.- Trazas de ejemplo capturadas de la actividad un Websocket con WireShark pág. 78

Figura 4.3.- Diagrama simple del servidor Apache Tomcat. Cliente-Servidor pág. 79

Figura 4.4.- Diagrama de la arquitectura del servidor Tomcat con Jetty pág. 81

Figura 5.1.- Diagrama planificación temporal pág. 107

Figura 6.1.- Notación de casos de uso pág. 113

Figura 6.2.- Diagrama casos de uso pág. 113

Figura 7.1.- Diagrama arquitectura Cliente-Servidor pág. 123

Figura 7.2.- Diagrama de la arquitectura física de la Sala Principal pág. 124

Figura 7.3.- Diagrama de la arquitectura física de la Habitación WebRTC pág. 125

Figura 7.4.- Diagrama de componentes del proyecto pág. 127

Figura 7.5.- Diagrama de clases del Servidor (BackEnd) pág. 128

Figura 7.6.- Diagrama de secuencia Sala Principal pág. 129

Figura 7.7.- Diagrama de secuencia conexión Habitación WebRTC pág. 130

Figura 7.8.- Diagrama de secuencia compartir archivos pág. 131

Figura 7.9.- Diagrama de secuencia hacer foto pág. 132

Figura 7.10.- Diagrama de secuencia grabar vídeo pág. 133

Figura 7.11.- Diagrama de enviar/recibir mensajes por Chat pág. 134

Figura 7.12.- Diagrama de despliegue del proyecto pág. 136

Figura 8.1.- Esquema compartir archivos en tiempo real pág. 162

Figura 10.1.- Captura de pantalla de la página de inicio de OpenShift.com pág. 181

Figura 10.2.- Captura de pantalla del formulario de registro de OpenShift pág. 182

Figura 10.3.- Captura de pantalla del formulario de acceso de OpenShift pág. 182

Figura 10.4.- Captura de pantalla del listado de servidores operativos en OpenShift pág. 183

Figura 10.5.- Captura de pantalla de la aplicación en OpenShift pág. 184

Figura 10.6.- Manual de usuario: Página de Inicio pág. 189

Figura 10.7.- Manual de usuario: Sala Principal pág. 190

Figura 10.8.- Manual de usuario: Habitación WebRTC pág. 192

Figura 10.9.- Manual de usuario: Compartir dispositivos. Versión navegador de escritorio y móvil (Firefox) pág. 192

Figura 10.10.- Manual de usuario: Habitación Ocupada pág. 193

Figura C.1.- Diagrama de dependencias pág. 253

Tabla 2.1.- Ventajas y desventajas de un árbol de características pág. 35

Tabla 3.1.- Ventajas e inconvenientes de utilizar WebRTC pág. 42

Tabla 3.2.- Navegadores web de escritorio que soportan WebRTC pág. 44

Tabla 3.3.- Navegadores web movil que soportan WebRTC pág. 44

Tabla 3.4.- Sintaxis del método navigator.getUserMedia(). Lenguaje JavaScript pág. 46

Tabla 3.5.- Constructor del objeto RTCPeerConnection. Lenguaje JavaScript pág. 47

Tabla 3.6.- Formato del protocolo UDP pág. 53

Tabla 3.7.- Ejemplo de los candidatos (ICE) enviados y recibidos para configurar WebRTC pág. 56

Tabla 3.8.- Formato del Protocolo de descripción de sesión pág. 57

Tabla 4.1.- Códecs de video y audio soportados por los navegadores pág. 63

Tabla 4.2.- Trazas de ejemplo configuración de WebSocket ClientEndPoint pág. 77

Tabla 5.1.- Clasificación de las Entradas Externas pág. 87

Tabla 5.2.- Clasificación de las Salidas Externas y Consultas Externas pág. 88

Tabla 5.3.- Clasificación de los Archivos Lógicos Internos y Archivos de Interfaz Externos pág. 88

Tabla 5.4.- Asignación de los valores numéricos pág. 88

Tabla 5.5.- Valoración de los puntos de función del proyecto pág. 89

Tabla 5.6.- Características y preguntas para el cálculo de los puntos de función ajustados pág. 90

Tabla 5.7.- Valoración: Comunicación de datos pág. 91

Tabla 5.8.- Valoración: Procesamiento distribuido de datos pág. 91

Tabla 5.9.- Valoración: Rendimiento pág. 92

Tabla 5.10.- Valoración: Configuraciones fuertemente utilizadas pág. 92

Tabla 5.11.- Valoración: Frecuencia de transacciones pág. 93

Tabla 5.12.- Valoración: Entrada de datos on-line pág. 93

Tabla 5.13.- Parámetros para la valoración de la eficiencia del usuario final pág. 94

Tabla 5.14.- Valoración: Eficiencia del usuario final pág. 94

Tabla 5.15.- Valoración: Actualización on-line pág. 95

Tabla 5.16.- Parámetros para la valoración del procesamiento complejo pág. 95

Tabla 5.17.- Valoración: Procesamiento complejo pág. 95

Tabla 5.18.- Valoración: Reusabilidad pág. 96

Tabla 5.19.- Valoración: Facilidad de instalación pág. 96

Tabla 5.20.- Valoración: Facilidad de operación pág. 97

Tabla 5.21.- Valoración: Instalación en distintos lugares pág. 97

Tabla 5.22.- Parámetros para la valoración de la facilidad de cambios pág. 98

Tabla 5.23.- Valoración: Facilidad de cambios pág. 98

Tabla 5.24.- Cálculo de los puntos de función ajustados pág. 99

Tabla 5.25.- Modelos de COCOMO pág. 100

Tabla 5.26.- Valor de los factores COCOMO pág. 101

Tabla 5.27.- Desglose del presupuesto hardware pág. 104

Tabla 5.28.- Desglose del presupuesto software pág. 104

Tabla 5.29.- Desglose del presupuesto personal pág. 105

Tabla 5.30.- Desglose del presupuesto total pág. 106

Tabla 6.1.- ACT – 01: Usuario pág. 112

Tabla 6.2.- RQNF – 01: Interfaz de usuario simple e intuitiva pág. 119

Tabla 6.3.- RQNF – 02: Interfaz de usuario responsiva pág. 119

Tabla 6.4.- RQNF – 03: Conexión a Internet pág. 120

Tabla 6.5.- RQNF – 04: Ejecución correcta de la aplicación en varios navegadores pág. 120

Tabla 6.6.- RI – 01: Fichero de logs del sistema pág. 120

Tabla 8.1.- Diccionario de datos: Evento tipo → *connect* (Sala Principal) pág. 146

Tabla 8.2.- Diccionario de datos: Evento tipo → *newuser* (Sala Principal) pág. 146

Tabla 8.3.- Diccionario de datos: Evento tipo → *calling* (Sala Principal) pág. 146

Tabla 8.4.- Diccionario de datos: Evento tipo → *deleteuser* (Sala Principal) pág. 147

Tabla 8.5.- Diccionario de datos: Evento tipo → *offer* (Habitación WebRTC) pág. 150

Tabla 8.6.- Diccionario de datos: Evento tipo → *answer* (Habitación WebRTC) pág. 150

Tabla 8.7.- Diccionario de datos: Evento tipo → *candidate* (Habitación WebRTC) pág. 150

Tabla 9.1.- CP – 01: Registrar usuario (Página de Bienvenida) pág. 170

Tabla 9.2.- CP – 02: Listar usuarios (Sala Principal) pág. 170

Tabla 9.3.- CP – 03: Añadir usuario a la lista de usuarios (Sala Principal) pág. 171

Tabla 9.4.- CP – 04: Eliminar usuario de la lista de usuarios (Sala Principal) pág. 171

Tabla 9.5.- CP – 05: Realizar y notificar llamada (Sala Principal) pág. 172

Tabla 9.6.- CP – 06: Añadir usuario a la lista de usuarios pág. 172

Tabla 9.7.- CP – 07: Realizar videollamada entre pares (Habitación WebRTC) pág. 173

Tabla 9.8.- CP – 08: Compartir archivos (Habitación WebRTC) pág. 173

Tabla 9.9.- CP – 09: Cancelar la transferencia al compartir un archivo (Habitación WebRTC) pág. 174

Tabla 9.10.- CP – 10: Pantalla completa On/Off (Habitación WebRTC) pág. 174

Tabla 9.11.- CP – 11: Hacer foto de la videollamada (Habitación WebRTC) pág. 175

Tabla 9.12.- CP – 12: Grabar videollamada (Habitación WebRTC) pág. 175

Tabla 9.13.- CP – 13: Audio On/Off (Habitación WebRTC) pág. 176

Tabla 9.14.- CP – 14: Chat On/Off (Habitación WebRTC) pág. 176

Tabla 9.15.- CP – 15: Enviar, recibir y listar mensajes por Chat (Habitación WebRTC) pág. 177

Tabla 9.16.- CP – 15: Volver a la Sala Principal (Habitación WebRTC) pág. 177

Tabla A.1.- CU – 01: Registrar usuario pág. 208

Tabla A.2.- CU – 02: Listar los usuarios (Sala Principal) pág. 209

Tabla A.3.- CU – 03: Añadir usuario (Sala Principal) pág. 210

Tabla A.4.- CU – 04: Eliminar usuario (Sala Principal) pág. 211

Tabla A.5.- CU – 05: Realizar llamada (Sala Principal) pág. 212

Tabla A.6.- CU – 06: Recibir llamadas (Sala Principal) pág. 213

Tabla A.7.- CU – 07: Aceptar llamada (Sala Principal) pág. 214

Tabla A.8.- CU – 08 - Rechazar llamada (Sala Principal) pág. 215

Tabla A.9.- CU – 09: Realizar videollamada entre pares (Habitación WebRTC) pág. 216

Tabla A.10.- CU – 10: Enviar archivos pág. 218

Tabla A.11.- CU – 11: Cancelar transferir archivos entre pares pág. 219

Tabla A.12.- CU – 12: Grabar videollamada pág. 220

Tabla A.13.- CU – 13: Parar grabar video pág. 221

Tabla A.14.- CU – 14: Chat On/Off pág. 222

Tabla A.15.- CU – 15: Enviar mensaje (Chat) pág. 223

Tabla A.16.- CU – 16: Listar mensajes (Chat) pág. 224

Tabla A.17.- CU – 17: Hacer foto pág. 225

Tabla A.18.- CU – 18: Sonido On/Off pág. 226

Tabla A.19.- CU – 19: Pantalla completa On/Off pág. 227

Tabla A.20.- CU – 20: Volver a la Sala Principal pág. 228

Tabla B.0.- RQF – 00: Mostrar página de inicio y bienvenida pág. 231

Tabla B.1.- RQF – 01: Registrar usuario pág. 231

Tabla B.2.- RQF – 02: Validar usuario pág. 231

Tabla B.3.- RQF – 03: Asignar un token al usuario pág. 232

Tabla B.4.- RQF – 04: Redirigir al usuario a la Sala Principal pág. 232

Tabla B.5.- RQF – 05: Establecer un socket de conexión (Sala Principal) pág. 233

Tabla B.6.- RQF – 06: Listar usuarios (Sala Principal) pág. 233

Tabla B.7.- RQF – 07: Añadir usuario (Sala Principal) pág. 233

Tabla B.8.- RQF – 08: Eliminar usuario (Sala Principal) pág. 234

Tabla B.9.- RQF – 09: Realizar llamada pág. 234

Tabla B.10.- RQF – 10: Notificar llamada pág. 234

Tabla B.11.- RQF – 11: Rechazar llamada pág. 235

Tabla B.12.- RQF – 12: Aceptar llamada pág. 235

Tabla B.13.- RQF – 13: Crear la Habitación para la videollamada entre pares (WebRTC) pág. 236

Tabla B.14.- RQF – 14: Redirigir a la Habitación (WebRTC) pág. 236

Tabla B.15.- RQF – 15: Securizar la Habitación pág. 237

Tabla B.16.- RQF – 16: Establecer un socket de conexión (Habitación) pág. 237

Tabla B.17.- RQF – 17: Obtener los recursos de la máquina (audio y vídeo) *getUserMedia()* pág. 237

Tabla B.18.- RQF – 18: Sincronizar el par de usuarios pág. 238

Tabla B.19.- RQF – 19: Configurar la Conexión entre Pares. WebRTC pág. 238

Tabla B.20.- RQF – 20: Obtener y establecer SDP (Descripctión de la Sesión) Offer/Answer pág. 238

Tabla B.21.- RQF – 21: Enviar SDP (Descripción de la Sesión) Offer/Answer pág. 239

Tabla B.22.- RQF – 22: Establecer la SDP Remota (Descripción de la Sesión) Offer/Answer pág. 239

Tabla B.23.- RQF – 23: Obtener y establecer los ICECandidates pág. 239

Tabla B.24.- RQF – 24: Enviar los ICECandidates pág. 240

Tabla B.25.- RQF – 25: Recibir los ICECandidates Remotos pág. 240

Tabla B.26.- RQF – 26: Añadir los ICECandidates Remotos pág. 241

Tabla B.27.- RQF – 27: Establecer y mantener la conexión con WebRTC entre pares pág. 241

Tabla B.28.- RQF – 28: Cerrar la conexión con WebRTC entre pares pág. 242

Tabla B.29.- RQF – 29: Adaptar la vista para la videollamada Pág. 242

Tabla B.30.- RQF – 30: Menú GUI pág. 243

Tabla B.31.- RQF – 31: Pantalla Completa On/Off pág. 243

Tabla B.32.- RQF – 32: Hacer Foto pág. 244

Tabla B.33.- RQF – 33: Grabar vídeo pág. 244

Tabla B.34.- RQF – 34: Parar grabar vídeo pág. 244

Tabla B.35.- RQF – 35: Crear vídeo pág. 245

Tabla B.36.- RQF – 36: Transferir archivos pág. 245

Tabla B.37.- RQF – 37: Notificar estado de la transferencia del archivo pág. 245

Tabla B.38.- RQF – 38: Cancelar envío del archivo pág. 246

Tabla B.39.- RQF – 39: Cancelar recepción del archivo pág. 246

Tabla B.40.- RQF – 40: Sonido On/Off pág. 246

Tabla B.41.- RQF – 41: Chat On/Off pág. 247

Tabla B.42.- RQF – 42: Crear Chat pág. 247

Tabla B.43.- RQF – 43: Enviar mensaje (Chat) pág. 247

Tabla B.44.- RQF – 44: Recibir mensaje (Chat) pág. 248

Tabla B.45.- RQF – 45: Listar mensajes (Chat) pág. 248

Tabla B.46.- RQF – 46: Volver a la Sala Principal pág. 249

Código 4.1.- Esquema básico de un documento HTML5 pág. 62

Código 4.2.- Ejemplo etiqueta `<video>` en HTML5 pág. 62

Código 4.3.- Ejemplo etiqueta `<canvas>` en HTML5 pág. 64

Código 4.4.-. Ejemplo de una función con callBacks. Lenguaje JavaScript pág. 65

Código 4.5.- Ejemplo como obtener un elemento del DOM HTML con `getElementById()`. Lenguaje JavaScript pág. 66

Código 4.6.- Ejemplo declarar una instancia del objeto `XMLHttpRequest`. Lenguaje JavaScript pág. 67

Código 4.7.- Ejemplo estructura de un objeto `JSON`. Lenguaje JavaScript pág. 67

Código 4.8.- Ejemplo de objeto `JSON` y objeto `XMLHttpRequest`. Lenguaje JavaScript pág. 68

Código 4.9.- Ejemplo de código CSS básico pág. 69

Código 4.10.- Declaración en el documento HTML un fichero externo CSS pág. 69

Código 4.11.- Declaración 01 en el documento HTML la etiqueta `<style>` CSS pág. 70

Código 4.12.- Declaración 02 en el documento HTML la etiqueta `<style>` CSS pág. 70

Código 4.13.- Ejemplo responsive design CSS pág. 71

Código 4.14.- Ejemplo de los métodos de un WebSocket ServerEndPoint. Lenguaje Java pág. 75

Código 4.15.- Ejemplo configuración de un WebSocket ClientEndPoint. Lenguaje JavaScript pág. 76

Código 8.1.- Configuración del archivo *web.xml* para el listener del WebSocket pág. 141

Código 8.2.- Implementación del WebSocket Listener `ServletContextListener`. Lenguaje Java pág. 142

Código 8.3.- Implementación de la clase `SignalingWebSocket`. Lenguaje Java pág. 145

Código 8.4.- Implementación del la Sala Principal (FronEnd). Lenguaje JavaScript pág. 149

Código 8.5.- Implementación del la Sala Principal (BackEnd). Lenguaje Java pág. 149

Código 8.6.- Variables globales utilizadas para realizar conexión entre pares. Lenguaje JavaScript pág. 152

Código 8.7.- Variables globales para instanciar el streaming de vídeo. Lenguaje JavaScript pág. 152

Código 8.8.- Ejemplo para iniciar el algoritmo para crear la conexión entre pares. Lenguaje JavaScript pág. 153

Código 8.9.- Constructor e interfaz del WebSocket en el cliente(FrontEnd). Lenguaje JavaScript pág. 153

Código 8.10.- Función getUserMedia y success cliente (FrontEnd). Lenguaje JavaScript pág. 154

Código 8.11.- Objeto `RTCPeerConnection` y sus métodos, cliente (FrontEnd). Lenguaje JavaScript pág. 155

Código 8.12.- Función para crear la oferta(*offer*) y enviarla desde el cliente (FrontEnd). Lenguaje JavaScript pág. 156

Código 8.13.- Función `processSignalingMessage`, cliente (FrontEnd). Lenguaje JavaScript pág. 157

Código 8.14.- Método `onIceCandidate` implementado para enviar los candidatos y sus IPs (privadas y públicas). Lenguaje JavaScript pág. 157

Código 8.15.- Función `sendMessage` con objeto `XMLHttpRequest` a un Servlet Java. Lenguaje JavaScript pág. 158

Código 8.16.- Punto de entrada `MainPageServlet.java`. Lenguaje Java pág. 160

Código 8.17.- Implementación de la lógica de la Habitación WebRTC. Lenguaje Java pág. 161

Código 8.18.- Objetos HTML `input y button` para compartir archivos pág. 163

Código 8.19.- Ejemplo para obtener y leer un archivo. Lenguaje JavaScript pág. 163

Código 8.20.- Función `onReadAsDataURL` para enviar un archivo en tiempo real. Lenguaje JavaScript pág. 164

Código 8.21.- Eventos al recibir y/o enviar un archivo compartido. Lenguaje JavaScript pág. 165

Código 10.1.- Comandos por consola Ubuntu para instalar Git pág. 183

Código 10.2.- Comandos básicos de Git pág. 184

Código 10.3.- Comando Git clone pág. 185

Código 10.4.- Configuración del archivo *server.xml* pág. 185

Código 10.5.- Comando de instalación de la herramienta Client Tool pág. 186

Código 10.6.- Comando para ver los puertos abiertos en OpenShift pág. 186

Código 10.7.- Vista consola linux, puertos abiertos en OpenShift pág. 186

Código 10.8.- Comando de acceso a la aplicación en OpenShift por consola linux pág. 187

Código 10.9.- Estructura de los directorios de la aplicación en el servidor OpenShift pág. 187

Código 10.10.- Comando para ver las trazas de la aplicación en el servidor OpenShift por consola Ubuntu pág. 188

Capítulo 1
Introducción

1.- Introducción

1.1.- Motivación

La motivación y la finalidad de este Trabajo de Fin de Grado ha sido la aplicación de conocimientos adquiridos en la carrera de Informatica en Servicios y Aplicaciones, la investigación, estudio y documentación de nuevas tecnologías que están siendo desarrolladas en este momento y están en plena evolución.

Las ventajas del uso de las tecnologías en la sociedad son diversas; permite la comunicación e interacción de forma digital entre personas desde cualquier punto del mundo, fomenta el desarrollo de nuevas habilidades sociales, el desarrollo comercial y cientifico, fomenta la productividad, abre nuevas oportunidades de negocio, así como nos acerca más como sociedad.

La comunicación entre los seres humanos por medio de las nuevas tecnologías está derivando en multiples vertientes según su finalidad, por ejemplo si queremos enviar un mensaje de texto para que sea leido inmediatamente utilizamos una aplicación movil de mensajería o un SMS, si queremos que el mensaje enviado sea leido en cualquier momento o quede constancia del mismo podemos enviar un email, si queremos comunicación directa utilizamos una llamada de teléfono, etc. es decir las nuevas tecnologías nos brinda un abanico amplio de vias de comunicación e información según nuestras necesidades y gustos.

En este trabajo se ha dado mucha importancia a las vias de comunicación que las nuevas tecnologías nos brinda y a su evolución. El tipo de comunicación entre personas utilizando la tecnología son multiples y variadas, pero las más destacadas son: las redes sociales, aplicaciones moviles de mensajería instantanea, el correo electrónico, las llamadas de teléfono y las videollamadas. Por medio de la redes sociales, las aplicaciones moviles y el correo electrónico podemos compartir mensajes, fotos, publicaciones, archivos, etc. Con las llamadas de teléfono y las videoconferencias también se puede compartir mensajes pero con una metainformación añadida en el mensaje como puede ser los sentimientos, expresiones, tonos de voz , gestos,... esta tendencia apunta hacia la creación de una aplicación que abarque sistema completo donde la base de la comunicación es el contacto directo entre las personas, puedan verse, oirse y compartir información en tiempo real.

La comunicación en tiempo real es una rama de las nuevas tecnologías que está en pleno evolución y desarrollo, se aplica en muchos campos y profesiones, como hoy en día en el terreno de la medicina se puede operar a un paciente que fisicamente no está en la misma habitación que el cirujano o el equipo médico, en el terreno aeroespacial se puede controlar un satelite en tiempo real o un dron desde una distancia considerable. La comunicación en tiempo real cobra una gran importancia en la comunicación, servicios y aplicaciones que las nuevas tecnologías nos ofrece.

La videoconferencia permite la comunicación entre dos o más localizaciones en tiempo real, mediante imagen y sonido sincronizados. Esta posibilidad se da tanto en el caso de comunicaciones uno a uno (video teléfono), como en el caso de muchos a muchos (multiconferencia). En una

empresa el uso de las videoconferencias tiene muchas ventajas al ahorrar costos en transporte, ahorra tiempo, mejora la calidad de vida, disminuye riesgos laborales, puede ofrecer ganancias estrategicas (mejor servicio al cliente, expansión de negocio) y aumentar la producción.

1.2.- Organización del documento

La estructura de este documento está organizada en varios capitulos:

Capítulo 1. Resumen. El capítulo 2 recoge la motivación y finalidad del proyecto. Introduce el mundo de las videollamadas, la comunicación entre los seres humanos utilizando las nuevas tecnologías.

Capítulo 2. Introducción. El capitulo 2 se compone de una pequeña introducción, los objetivos y el alcance de este proyecto. Además incluye el árbol de características para tener una visión general y el estado del arte del proyecto comparandolo con los sistemas que hay actualmente en el mercado.

Capítulo 3. Web de Comunicación Real de Transmisión (WebRTC). El capítulo 3 introduce la tecnología WebRTC sus ventajas e inconvenientes. Realiza un estudio sobre la tecnología WebRTC, características, componentes, su API para desarrolladores y las tecnologías involucradas.

Capítulo 4. Tecnologías Web. El capítulo 4 incluye las tecnologías web que son necesarias para implementar y implantar la aplicación del proyecto, recoge las tecnologías relacionadas en el contexto web con J2EE/JavaEE, protocolos, lenguajes de programación, herramientas e interfaces de programación.

Capítulo 5. Gestión del proyecto. El capítulo 5 tiene como finalidad realizar un presupuesto estimando los costes temporales y económicos según los puntos de función del proyecto.

Capítulo 6. Análisis del sistema. El capítulo 6 contiene el análisis del sistema, identificando los actores del sistema, los casos de uso y los requisitos del sistema.

Capítulo 7. Diseño del software. El capítulo 7 muestra la arquitectura lógica y física de la aplicación web. Este capítulo se estructura en 2 partes: Diseño del BackEnd y Diseño del FrontEnd.

Capítulo 8. Implementación del proyecto. El capítulo 8 está diseñado para desarrolladores, en este capítulo se incluyen tutoriales con ejemplos sobre programación en Java y en JavaScript para implementar este sistema. Este capítulo se estructura en 2 partes: BackEnd y FronEnd del sistema.

Capítulo 9. Pruebas. El capítulo 9 es un documento QA (Quality Assurance) de las pruebas realizadas en la aplicación sobre los requisitos recogidos y la funcionalidad propia del software.

Capítulo 10. Manual de instalación y uso. En este capítulo se explica como desplegar la aplicación en un servidor público (OpenShift) Java y el manual de usuario.

Capítulo 11. Conclusiones y trabajos futuros. Este capítulo muestra las conclusiones y trabajos futuros que se pueden realizar con la tecnología WebRTC, JavaEE y otras tecnologías.

Bibliografía y referencias.

Anexo A. Casos de uso. Este anexo contiene los casos de uso identificados en el sistema y su respectiva descripción.

Anexo B. Requisitos funcionales. Este anexo incluye los requisitos funcionales del sistema y su respectiva descripción.

Anexo C. Diagrama de dependencias. Este anexo incluye el diagrama de dependencias de los requisitos funcionales del sistema.

Anexo D. Glosario de terminos. Este anexo contiene un lista de los terminos utilizados en este proyecto.

Capítulo 2
Objetivos y alcance del sistema

2.- Objetivos y alcance del sistema

2.1.- Objetivo principal

El objetivo principal de este proyecto es el estudio de **WebRTC**, de sus componentes, funciones, tecnologías involucradas para acercar y dar a conocer una herramienta que actualmente están en pleno desarrollo; este proyecto se materializa en la realización de una documentación y una aplicación que sirva de ejemplo de como implementar, implantar y desarrollar un sistema de comunicación dentro de un contexto web con WebRTC en un entorno **J2EE/JavaEE**.

Este proyecto recoge como crear una aplicación software que permita realizar videollamadas, llamadas de voz, chat, compartición de escritorio remoto, transmisión de archivos en tiempo real, grabar las sesiones y realizar fotos, para que pueda ser usada desde cualquier ordenador con acceso a internet desde cualquier parte del mundo. La aplicación es una ventana al mundo que acerca a los usuarios a tener una comunicación interactiva en tiempo real.

El alcance de este sistema es priorizar la comunicación entre personas por medio de la videollamada y que la aplicación se pueda utilizar desde cualquier ordenador con un navegador con acceso a internet compatible con WebRTC (más adelante se entrará en detalle de los navegadores que soportan WebRTC). La usabilidad, la experiencia del usuario es un requisito importante para que tenga éxito esta forma de comunicación.

2.2.- Estado del arte

Haciendo un estudio y análisis de las tecnologías que hay en el mercado actualmente para realizar videollamadas nos podemos encontrar con Skype de Microsoft por ejemplo. La a compra de Skype por parte de Microsoft costó unos 5.920 millones de euros, tiene servicio de videollamada, llamada de voz y mensajería, Skype es un programa software que se puede instalar en varios sistemas operativos (pc o movil) y actualmente se está desarrollando una fase beta para navegadores web. Otra aplicación que existe en el mercado es Google Hangouts en sus versiones de app movil, ordenador y navegador web, la propiedad de este software pertenece a Google y con esta aplicación se puede realizar videollamadas, llamadas de voz, enviar mensajes y archivos. Google Hangouts introdujo la tecnología WebRTC en su sistema en Junio de 2014, siendo esta tecnología novedosa y en fase de desarrollo. Existen más aplicaciones software de escritorio, móviles y web para realizar videollamadas pero todas nos ofrecen los mismos servicios y algunas utilizan la tecnología WebRTC para cumplir su funcionalidad, por este motivo el objetivo de este proyecto es el estudio de la tecnología WebRTC y el cómo implementarla en un entorno web.

Figura 2.1.- Ejemplo de aplicación con videollamada. Skype

Otra tecnología que permiten la realización de videollamadas son las llamadas VoIP con el protocolo SIP, pero el sistema con SIP es más complejo y el coste es mayor porque habría que utilizar software adicional para poder implementar las conexiones en un sistema. Este no está implementado en un navegador, etc., aunque WebRTC y SIP comparten muchas similitudes como los medios de comunicación en tiempo real, los códecs de voz y la seguridad de los medios pero la tendencia está siendo el uso de WebRTC por su fácil implementación.

Este proyecto abarca desde el mismo estudio de WebRTC y sus tecnologías relacionadas como son los protocolos, servidores, frameworks, etc., y otras tecnologías de apoyo software para que este funcione y cómo implementarlo. Todo el software utilizado en este proyecto ha sido software libre y abierto.

Figura 2.2.- Ejemplo de aplicación con videollamada. Hangouts

Este proyecto y la aplicación realizada está enfocada desde dos puntos de vista; desde el punto de vista del usuario cumple un requisito fundamental de usabilidad, es fácil de utilizar y el usuario puede disfrutar de ella sin conocimientos previos, sin instalación de programas adicionales ni plugins. La aplicación funciona en navegadores web y móvil más populares y más utilizados, funciona en múltiples plataformas con un único el código software (código unitario) siendo el mismo para todas. Para la realización de este trabajo se hace hincapié en que todo el software utilizado es libre y abierto, para que cualquier desarrollador pueda utilizar este proyecto y pueda implementarlo en cualquier contexto o aplicación web.

2.3.- Árbol del características

El árbol de características de un proyecto software es un documento que recoge las características principales de un sistema organizándolas en grupos para formar un diagrama en árbol con ellas, facilitando de esta manera al lector una sencilla interpretación del sistema desde un alto nivel y su alcance.

Este tipo de modelo contiene varios tipos de características:

Características funcionales: Características relacionadas con el hardware y software del sistema.

Características no funcionales: Características relacionadas con el rendimiento, mantenimiento y disponibilidad de un sistema.

Parámetros: Niveles de detalle de una característica, alcance, integración y dependencia.

Ventajas y desventajas de utilizar un árbol de características:

Ventajas	Desventajas
El diagrama de árbol de características es sencillo de leer.	Recoge de forma general las características de un sistema sin entrar en detalles.
Estructura las características en grupos y subgrupos facilitando la organización de las características de un sistema.	Puede estar carente de información.
Recoge las dependencias y jerarquías de un sistema.	En sistemas complejos puede ser difícil de leer.
Es útil para definir el alcance de un sistema.	Es solo un documento.

Tabla 2.1.- Ventajas y desventajas de un árbol de características

2.3.1.- Diagrama del árbol de características

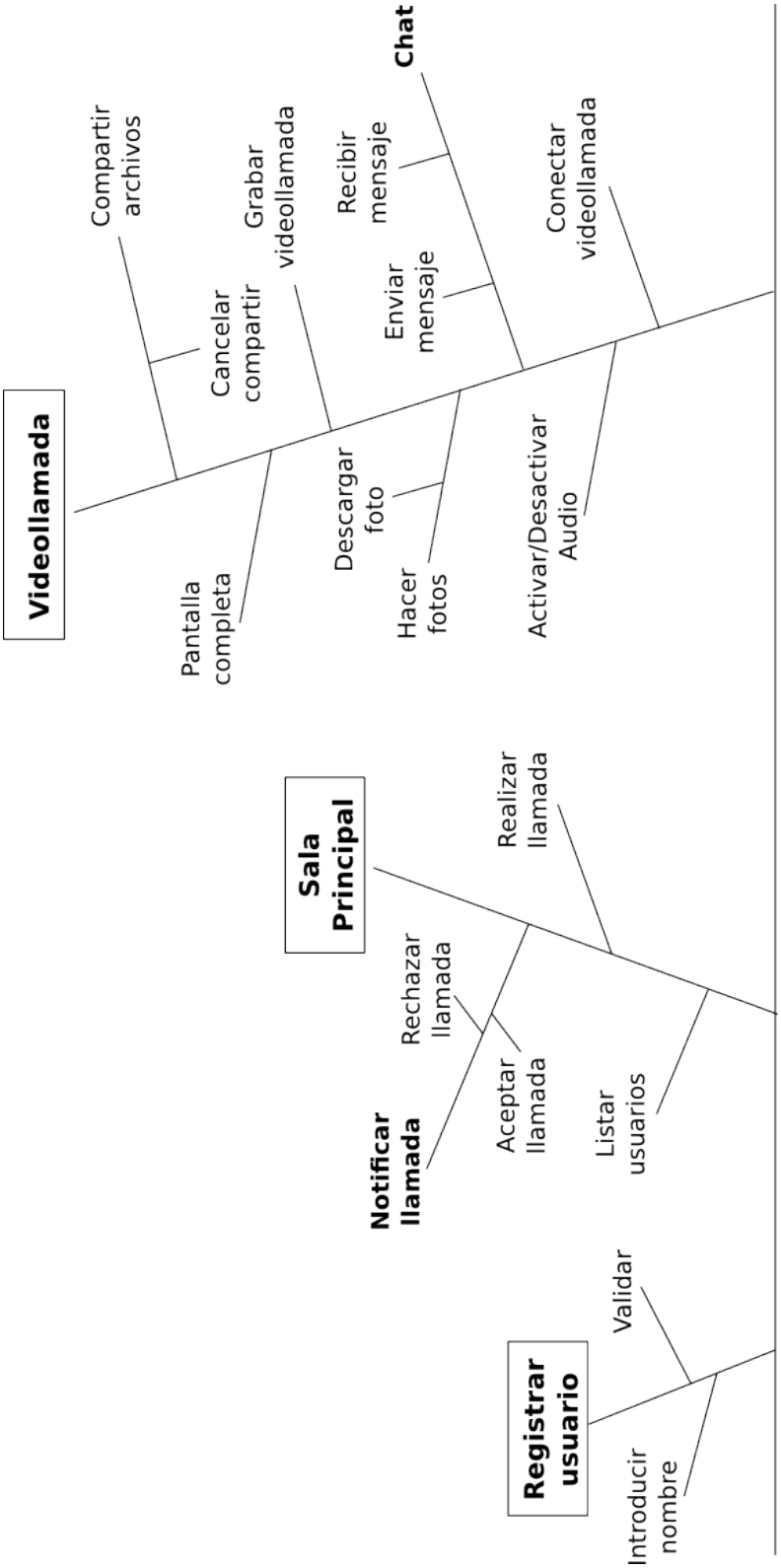

Figura 2.3.- Árbol de características del proyecto

2.3.2.- Características Principales del proyecto

1.- Registrar usuario: Esta característica se refiere al proceso de registrar el usuario en la aplicación.

- *Introducir nombre*: El usuario podrá registrarse en la aplicación por medio de un formulario compuesto por el campo de texto para introducir el nombre del usuario y un botón para enviar el formulario.

- *Validar*: El usuario podrá ser notificado a través de mensaje de información si la condición de validación no se cumple y no podrá enviar el formulario de registro.

2.- Conexión con entre muchos (Sala Principal): Esta característica se refiere a la conexión entre los usuarios en una Sala Principal para poder realizar llamadas.

- *Listar usuarios*: Lista de los usuarios que están conectados en la Sala Principal

- *Llamar*: El usuario podrá realizar llamadas a otro usuario pulsando sobre el nombre de un usuario de la lista de usuarios conectados en la Sala Principal.

- *Notificar llamada*: Cuando un usuario está siendo llamado por otro usuario, se le notifica por medio de un mensaje de llamada.

- *Aceptar llamada*: El usuario podrá aceptar la llamada de otro usuario.

- *Rechazar llamada*: El usuario podrá rechazar la llamada de otro usuario.

3.- Videollamada entre pares (Habitación): Esta característica se refiere

- *Realizar videollamada*: El usuario podrá realizar videollamadas entre pares con otro usuario.

- *Pantalla completa*: El usuario podrá adaptar la interfaz de la videollamada a pantalla completa.

- *Hacer foto*: El usuario podrá realizar fotos de la videollamada.

- *Grabar vídeo:* El usuario podrá grabar la videollamada.

- *Compartir archivos*: El usuario podrá compartir archivos en tiempo real con otro usuario.

- *Cancelar compartir archivo*: El usuario podrá cancelar el envío o recepción del archivo compartido en tiempo real.

- *Activar/Desactivar audio*: El usuario podrá activar o desactivar el audio de la videollamada.

- Enviar mensajes por Chat: El usuario podrá enviar mensajes por chat.

- *Recibir mensajes por chat*: El usuario podrá recibir mensajes por chat.

2.4.- Descripción del producto a entregar

El producto a entregar es una copia de esta memoria en formato papel y digital dentro de un disco compacto (CD) con el código fuente de la aplicación. Además se incluye un manual de usuario donde se indican los pasos para utilizar la aplicación. Esta memoria también incluye un manual, para desarrolladores, de despliegue de la aplicación en un entorno JavaEE en un servidor web público válido para producción y pruebas. Por último, en la memoria se realiza una explicación del sistema, del código fuente y de sus archivos con el fin de que un desarrollador tenga las herramientas y conocimientos necesarios para adaptar el sistema a las necesidades del negocio.

2.5.- Contenido del CD-ROM

El contenido del CD-ROM entregado contiene una copia de este documento (memoria) en varios formatos y el código fuente de la aplicación.

El CR-ROM tiene los siguiente directorios y archivos:

- **memoria**: Esta carpeta contiene una copia en varios formatos de la memoria de este proyecto.

- **codigo_fuente**: Esta carpeta contiene el código fuente de la aplicación.

- **war**: Esta carpeta contiene el archivo exportado Java war de la aplicación "***webrtc.war***".

- **figuras_imagenes:** Esta carpeta contiene todos los archivos de imagenes y figuras incluidos en el proyecto como archivos vectoriales, diagramas de secuencia y varios. Esta carpeta está dividida en subcarpetas por capítulos.

- **licencias**: Esta carpeta contiene las licencias GNU de las librerías utilizadas en la aplicación del proyecto.

Capítulo 3

Web de Comunicación Real de Transmisión (WebRTC)

3.- Web de Comunicación Real de Transmisión (WebRTC)

3.1.- ¿Qué es WebRTC?

WebRTC (Web Real Transmision Comunication) es un proyecto de código abierto elaborado por la World Wide Web Consortium (**W3C**) dirigida a dotar a la web una comunicación en tiempo real. El proyecto WebRTC fue publicado Open Source por **Google** en el 2010, estandarizando los protocolos pertinentes de la **IETF** y el **API** del navegador en la W3C. El API se basa en el trabajo previo realizado por la **WHATWG**. Actualmente la iniciativa de este proyecto es soportada por Google, Mozilla y Opera, entre otros.

El código fuente de este proyecto está disponible en el repositorio: https://chromium.googlesource.com/external/webrtc. Este proyecto tiene una *licencia BSD* (Berkeley Software Distribution) con algunas restricciones, como que las redistribuciones del código fuente deben conservar el copyright anterior y ni el nombre de Google ni los nombres de sus colaboradores podrán usarse para respaldar o promocionar productos derivados de este software sin el permiso previo por escrito.

Este software permite comunicaciones en tiempo real ya sea vídeo, audio o datos entre navegadores web, siendo esta herramienta una gran ventaja en dos sentidos, por lado del usuario que utiliza la aplicación realizada con WebRTC obtiene una satisfacción al no tener que instalar ningún programa ni plug-in adicional para realizar videollamadas o transferir archivos y puede hacerlo desde cualquier ordenador con conexión a internet de manera fácil y sencilla. Y por el lado del programador que utilice WebRTC para de desarrollar aplicaciones, dispone de una API en lenguaje **JavaScript** junto con etiquetas de **HTML5** para el cliente (FrontEnd) con de las herramientas necesarias para desarrollar aplicaciones para transmitir datos, realizar videollamadas, vídeos bajo demanda, etc. de manera sencilla y utilizando poco código.

WebRTC se basa en una API que todavía está en fase de desarrollo a través de los esfuerzos de WHATWG, W3C y el IETF. Los desarrolladores de este proyecto pretenden llegar a una API estable cuando pocos proveedores de navegadores tengan implementaciones listas para la prueba. Una vez que la API sea estable, según los desarrolladores de este proyecto el objetivo es ofrecer compatibilidad y la interoperabilidad a la capa API WebRTC. Los componentes menores pueden ser modificados para mejorar la calidad, rendimiento y conjunto de características.

WebRTC se compone de varios elementos y software libre como el motor de voz, el motor de vídeo, el ecualizador de red, cancelación de echo acústico, etc. También utiliza varios **códecs** para el procesamiento de audio y vídeo.

Las aplicaciones con WebRTC necesitan hacer varias cosas:

• Obtener streaming de audio, vídeo u otros datos.

• Obtener información de la red, como direcciones IP y puertos, e intercambiar esto con otros clientes WebRTC (conocidos como pares) para permitir la conexión, incluso a través de NAT y firewalls.

• Coordinar la comunicación de señalización para informar de errores e iniciar o cerrar las sesiones.

• Intercambiar información sobre los medios de comunicación y capacidad de cliente, como la resolución y códecs.

• Enviar el streaming de audio, vídeo o datos.

3.2.- Ventajas e inconvenientes de utilizar WebRTC

Ventajas	Inconvenientes
El código fuente de WebRTC tanto la parte web como la móvil es abierto y libre.	La API web solo está disponible para utilizarla en lenguaje JavaScript.
No necesita plugins ni programas adicionales para utilizar esta tecnología.	El navegador web del usuario tiene que ser compatible con la tecnología WebRTC.
Esta tecnología está en continua evolución y desarrollo, apoyado por una gran comunidad de organizaciones y programadores de todos los países del mundo.	Al estar esta tecnología en continuo desarrollo y evolución los sistemas tienen que ser adaptados a posibles cambios.
Ofrece una API documentada y pública para desarrolladores.	
Es fácil de implantar y configurar por un programador o equipo de desarrolladores en cualquier sistema web o aplicación móvil.	
Utiliza los estándares y normas establecidas por la IETF, W3C y RFC.	
Las conexiones entre usuarios y aplicaciones con esta tecnología se crea punto a punto, sin necesidad de pasar la información o los datos por un servidor intermedio.	

Tabla 3.1.- Ventajas e inconvenientes de utilizar WebRTC

3.3.- Códecs utilizados por WebRTC

Los códecs que utiliza actualmente WebRTC para vídeo y audio son varios aunque estos pueden cambiar en un futuro.

• Para el **vídeo** se utiliza:

- VP8: códec de vídeo publicado en RFC6386, fue a sus inicios desarrollado por On2 Technologies y adquirido por Google en el 2009, siendo este liberado el 19 de Mayo de 2010. Este códec es parte del proyecto WebM. Incluye componentes para ocultar paquetes perdidos, limpiar las imágenes de ruido, capacidades de captura y reproducción a través de múltiples plataformas. Google está desarrollando la siguiente versión de este códec el V9 para integrarlo en su navegador Chrome y en WebRTC.

- H.264-MPEG: es un códec de alta compresión de vídeo desarrollado por el ITU-T Video Coding Experts Group (VCEG) y el ISO/IEC Moving Picture Experts Group (MPEG). Es un formato muy extendido y en reproductores de vídeo de todo tipo. El códec H.264 se puede combinar con los códecs de audio ACC o MP3 dentro del contenedor MPEG-4 (conocido como MP4).

• Para **audio** se incluye varios códecs como:

- G.711: es un estándar del Sector de Normalización de las Telecomunicaciones (ITU-T) de la Unión Internacional de Telecomunicaciones (ITU) para la codificación de audio. Este estándar es usado principalmente en telefonía, y fue liberado para su uso en el año 1972.

- G.722: es un códec de audio estándar ITU-T 7 KHz de banda ancha que opera a 48, 56 y 64 Kbit / s. Fue aprobado por la UIT-T en noviembre de 1988.

- iSAC: es un robusto adaptador de ancho de banda y códec de voz de banda y súper banda ancha desarrollada por Global IP Solutions utilizados en muchas aplicaciones de Voz sobre IP (VoIP) y de streaming de audio. iSAC es utilizado por los líderes de la industria en cientos de millones de puntos finales de VoIP.

- iLBC: es un códec libre de voz de banda estrecha, fue desarrollado por Global IP Solutions utilizado en muchas aplicaciones de Voz sobre IP (VoIP) y audio streaming. En 2004, las versiones finales IETF RFC de la especificación códec iLBC y la iLBC RTP Perfil proyecto estuvieron disponibles. Este códec se incluye como parte del proyecto WebRTC.

• Para la transmisión de datos utiliza buffers de desviación (jitter) dinámicos y técnicas de ocultación de error para procesar el audio y el vídeo ayudando a mitigar los efectos de la pérdida de paquetes y las redes no confiables.

Como se muestra en la sección HTML5 la tabla X se observa los códecs que cada navegador soporta, cada navegador integra al menos un códec de audio y vídeo. En la siguiente sección se muestran los navegadores que tiene implementada la tecnología WebRTC no confundir con los códecs que soporta cada navegador.

3.4.- Navegadores que soportan actualmente la API WebRTC

WebRTC al ser una herramienta con pocos años de vida todavía no es compatible con todos los navegadores web más conocidos actualmente en el mercado, pero la idea es que conforme se avance con este proyecto se irá implantando en más navegadores web, he aquí un resumen de los navegadores de escritorio y móvil que soportan WebRTC.

Navegadores web de escritorio:

Navegador	Chrome	Firefox (Gecko)	Internet explorer	Opera	Safari
Soporte básico	Soportado	Soportado	No soportado	Soportado	No soportado

Tabla 3.2.- Navegadores web de escritorio que soportan WebRTC

Los navegadores web móvil que soportan WebRTC se recogen en la siguiente tabla:

Navegadores web móvil:

Navegador	Android WebView	Chrome for Android	Firefox Mobile/OS	IE Phone	Opera Mobile	Safari WebMobile
Soporte básico	Soportado	Soportado	Soportado	No soportado	Soportado	No soportado

Tabla 3.3.- Navegadores web movil que soportan WebRTC

Como se puede observar, WebRTC esta actualmente implementado en varios navegadores web muy conocidos tanto de móvil como de escritorio. Como se puede observar en estas tablas las ventajas significativas que esta tecnología puede ofrece tanto en movilidad como en la usabilidad e integración de la misma en una aplicación web. Desde el punto de vista del programador existe una gran ventaja porque con único código, la misma aplicación es compatible con múltiples plataformas y sistemas operativos, en un futuro se espera integrar esta tecnología en más navegadores web. Y desde el punto de vista del usuario, puede acceder a su plataforma web que haga uso de WebRTC desde distintos navegadores, aunque actualmente los navegadores implementados en Android son más numerosos que en los navegadores de distinta plataforma como Windows Phone e IOS.

3.5.- La arquitectura de WebRTC

La arquitectura en WebRTC se compone de tres partes fundamentales:

1.- API para desarrolladores web: Es la interfaz o API que ofrece el navegador web para poder hacer aplicaciones con WebRTC. Esta interfaz ofrece los métodos necesarios para ser implementada en lenguaje JavaScript para poder usar WebRTC y así poder obtener los recursos de la máquina, configurar la conexión entre pares, abrir un canal de transmisión de datos, obtener los candidatos, etc.

2.- API para desarrolladores de navegadores: Es una interfaz o API para que los desarrolladores de navegadores puedan realizar las conexiones entre pares utilizando la tecnología WebRTC.

3.- Software reemplazable por desarrolladores de navegadores: Es el software que puede ser remplazado por los desarrolladores de navegadores como el los códecs de voz, los de vídeo o los de conexión, aunque este último es desarrollado por la IETF.

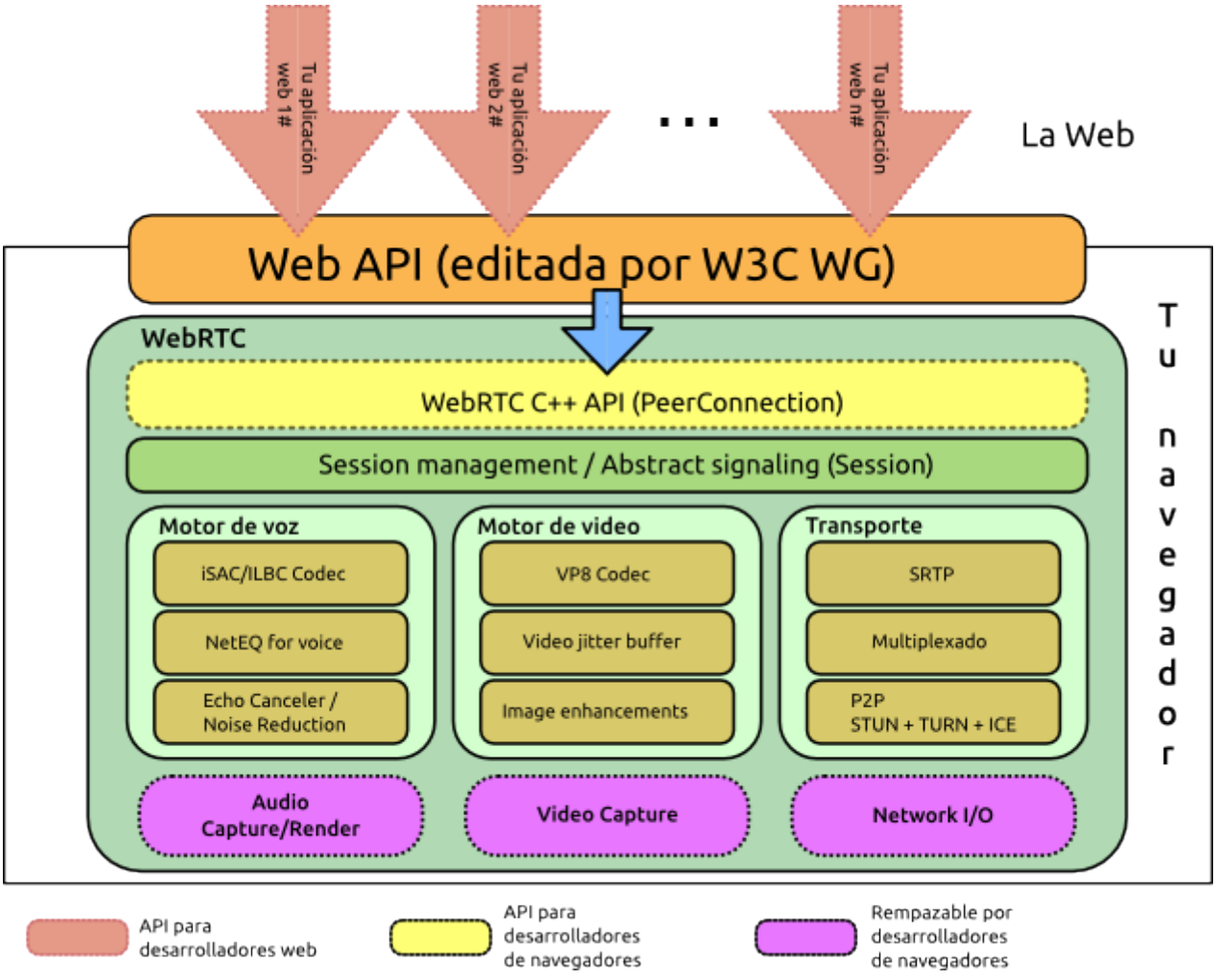

Figura 3.1.- Arquitectura de **WebRTC**

3.6.- API *JavaScript* de WebRTC

Por medio de esta **API** en **JavaScript** se puede acceder a la cámara web y al micrófono de un equipo para realizar conexiones entre pares o entre múltiples usuarios, la API proporciona al desarrollador varios una interfaz implementar los métodos y poder acceder a los recursos de la máquina. Para adquirir y comunicar datos de streaming, **WebRTC** implementa las siguientes API:

• **`RTCPeerConnection`**: audio o vídeo llamada, con módulos para el cifrado y gestión de ancho de banda.

• **`RTCDataChannel`**: comunicación entre pares para datos genéricos.

• **`getUserMedia`** `(MediaStream)`: obtener acceso a flujos de datos como la cámara y el micrófono del usuario.

3.6.1.- `getUserMedia (MediaStream):`

El método `getUserMedia` pide al usuario permiso para usar el vídeo y/o el dispositivo de entrada de audio, como una cámara, una pantalla compartida, o un micrófono. Si el usuario proporciona el permiso, entonces el éxito **(`success`)** se invoca con el objeto `MediaStream` resultante como su argumento. Si el usuario deniega el permiso o el medio no está disponible, entonces el **error** se llama con `PermissionDeniedError` o `NotFoundError` respectivamente. Tenga en cuenta que es posible que ni la devolución de llamada finalización de ser llamado, ya que no se requiere que el usuario pueda tomar una decisión.

Nota: El método `navigator.getUserMedia()` esta en desuso “deprecated” se recomienda utilizar el método `mediaDevices.getUserMedia()`.

```
navigator.getUserMedia(constraints, success, error);
```

Tabla 3.4.- Sintaxis del método `navigator.getUserMedia()`. Lenguaje JavaScript

Los posibles errores que este método puede mostrar es rechazo de la promesa de regresar con un objeto `MediaStreamError` que se inspira en `DOMException` como `PermissionDeniedError` porque el permiso para usar un dispositivo de medios fue negada por el usuario o el sistema o la excepción `NotFoundError` porque no se encontraron los recursos multimedia del tipo especificado que satisfacen las restricciones especificadas o bien no soporta esta función.

3.6.2.- `RTCPeerConnection`

Esta interfaz representa una conexión **WebRTC** entre el equipo local y un par remoto. Se utiliza para manejar la transmisión eficiente de datos entre los dos pares.

El uso básico del objeto `RTCPeerConnection` implica negociar una conexión entre el equipo local y otro equipo remoto mediante el Protocolo de Inicio de Sesión, actualmente redactado en el RFC 4566 en el Protocolo de Descripción de Sesión para intercambiar entre pares. La persona que llama inicia el proceso mediante el envío de una oferta para la máquina remota la cual responde aceptando o rechazando la llamada. Ambas partes (la persona que llama y la persona llamada) necesitan establecer sus propias instancias `RTCPeerConnection` para representar a su extremo de la conexión entre pares (peer-to-peer), he aquí un ejemplo actual de su sintaxis:

```
var peerConnectionConfig = {'iceServers' : [ {'urls' :
'stun:stun.services.mozilla.com'},...]};

new RTCPeerConnection(RTCConfiguration peerConnectionConfig,
optional MediaConstraints constraints);
```

Tabla 3.5.- Constructor del objeto `RTCPeerConnection`. Lenguaje JavaScript

3.6.1.1.- Propiedades de la Interfaz

El objeto `RTCPeerConnection` incluido en su objeto padre "window", necesita tener la configuración de los servidores TURN/STUN (ver apartado 3.7.4.- STUN/TURN) que va utilizar para realizar la conexión entre pares. Antes de conseguir la conexión entre los pares es necesario un intercambio previo de mensajes para acordar los parámetros de la comunicación. Tiene varios métodos en su interfaz para poder ser implementada en lenguaje JavaScript. Los métodos que incluyen su interfaz y son heredados de la interfaz de su padre `EventTarget` (`EventTarget` es una interfaz implementada por objetos que reciben eventos y pueden tener listeners para estar a la espera escucha de otros eventos o acciones) son:

• `RTCPeerConnection.iceConnectionState`: Esta función devuelve el estado de la conexión ICE, sus posibles valores son: "`new`", "`checking`", "`connected`", "`completed`", "`failed`", "`disconnected`", "`closed`".

• `RTCPeerConnection.iceGatheringState`: Devuelve un valor predefinido del tipo `RTCIceGatheringState` que describe el estado de la conexión, este valor puede ser: "`new`", "`gathering`", "`complete`".

• `RTCPeerConnection.localDescription`: Devuelve la descripción de la sesión de la máquina local del cliente (ClientEndPoint). Esta sesión que se enviará al otro cliente para establecer la descripción.

• RTCPeerConnection.peerIdentity: Si la conexión entre pares se ha realizado correctamente devuelve la identidad de la confirmación de la conexión entre pares, esta compuesto de dos valores: el nombre de dominio (ipd) y el nombre (name).

• RTCPeerConnection.remoteDescription: Devuelve la descripción de sesión de la maquina remota (ClientEndPoint) de la conexión. Si no se realiza la conexión devuelve nnull.

• RTCPeerConnection.signalingState: Devuelve el estado de la señal de la conexión local. Este estado describe la sesión de descripción de sesión (SDP) de la oferta, que define la configuración de las conexiones como el objeto MediaStream asociado, las opciones del códec/RTP/RTCP, los candidatos (ICE) unidos al agente ICE. Los estados que este método puede devolver son:

- "stable": no hay ofertas/contestaciones (offer/answer) de intercambio en el progreso. Es el estado inicial de la conexión.

- "have-local-offer": la oferta local de la máquina (ClientEndPoint) se aplica a la oferta SDP.

- "have-remote-offer": la descripción de la conexión remota (ClientEndPoint) se aplica a la oferta SDP local.

- "have-local-answer": la sesión SDP remota se ha aplicado y una SDP answer se aplica localmente.

"have-remote-answer": la sesión local SDP se ha aplicado y la sesión SPD answer se ha aplicado en la máquina (ClientEndPoint).

3.6.2.2- Los manejadores (handler) de eventos

• RTCPeerConnection.onaddstream: este evento es llamado cuando se recibe el evento stream (addstream). Tal como este evento es enviado cuando un MediaStream es añadido a la conexión por par remoto. El evento es enviado inmediatamente después de llamar a RTCPeerConnection.setRemoteDescription() y no espera por el resultado de la negociación SDP.

• RTCPeerConnection.ondatachannel: cuando el evento "datachannel" se recibe este handler lo maneja. Tal como el evento es enviado cuando el objeto RTCDataChannel es añadida a la conexión.

• RTCPeerConnection.onicecandidate: Este evento es llamado cuando un evento "icecandidate" es recibido. Tal como el evento se envía un objeto RTCICECandidate es añadido al script.

• RTCPeerConnection.onicecandidatestatechange: Este evento es llamado cuando un evento de tipo "iceconnectionstatechange" es recibido. Tal como el evento se

envía entonces el valor de iceConnectionState cambia.

• RTCPeerConnection.onidentityresult(): Este evento se llama cuando un evento de tipo "identityresult" es recibido. Este evento es enviado tal como un "identity assertion" es generada o durante la creación de una oferta o una respuesta.

• RTCPeerConnection.onidpassertionerror: Este evento es llamado cuando el evento "idpassertionerror" es recibido. Este evento es enviado cuando la identidad asociada del proveedor (IdP) encuentra un error generando una aserción de identidad.

• RTCPeerConnection.onidpvalidationerror: Este handler es ejecutado cuando se recibe el evento "idpassertionerror". Este evento se envía cuando la identidad asociada del proveedor (IdP) encuentra un error mientras valida la identidad de la afirmación.

• RTCPeerConnection.onnegotiationneeded: Este evento es llamado cuando el evento "negotiationneeded" se recibe, el navegador envía este evento para informar que negociación se requerirá en un futuro.

• RTCPeerConnection.onpeeridentity: Este evento es ejecutado cuando se recibe un evento de tipo "peeridentity", se envía la entidad al par cuando se ha realizado y verificado la conexión.

• RTCPeerConnection.onremovestream: Este evento es llamado cuando un evento de tipo "removestream" es recibido, se envía cunado el objeto "MediaStream" es eliminado de la conexión.

• RTCPeerConnection.onsignalingstatechange: Este evento se ejecuta cuando se recibe un evento "signalingstatechange", envía el valor de estado de la señal.

3.6.2.3.- Los métodos de API WebRTC

• RTCPeerConnection(): Es el constructor de la conexión entre pares.

• RTCPeerConnection.createOffer(): Este método crea la oferta solicitada para encontrar el par remoto con una configuración especifica. Los parámetros de entrada son el primero la configuración como puede ser de audio/vídeo, y los dos siguientes son los callBacks que se ejecuta según el éxito o el error de la ejecución de este método.

• RTCPeerConnection.createAnswer(): Este método crea la respuesta para la oferta recibida del par remoto, los parámetros de entrada son el primero la configuración como puede ser de audio/vídeo, y los dos siguientes son los callBacks que se ejecuta según el éxito o el error de la ejecución de este método.

• RTCPeerConnection.setLocalDescription(): Esta función cambia la descripción local asociada con la conexión. La descripción define las propiedades de la conexión como puede ser el códec. Este método tiene tres parámetros, un objeto RTCSessionDescription y los dos callBacks.

• RTCPeerConnection.setRemoteDescription(): Esta función cambia la descripción remota asociada con la conexión. Este método tiene tres parámetros de entrada, un objeto RTCSessionDescription y los dos callBacks.

• RTCPeerConnection.updateIce(): Este método actualiza la configuración de los ICECandidates.

• RTCPeerConnection.addIceCandidate(): Este método añade los ICECandidates válidos para realizar la conexión.

• RTCPeerConnection.getLocalStreams(): Devuelve un array de objetos de tipo MediaStream asociados con el punto final de la conexión local. Si está vacío devuelve null.

• RTCPeerConnection.getRemoteStreams(): Devuelve un array de objetos de tipo MediaStream asociados con el punto final de la conexión remota. Si está vacío devuelve null.

• RTCPeerConnection.getStreamById(): Devuelve el objeto MediaStream asociado a un id con la conexión con el punto final de la conexión remota y local.

• RTCPeerConnection.addStream(): Esta función añade un objeto MediaStream como recurso local de audio o vídeo. El recurso que se añade puede ser el stream remoto o local.

• RTCPeerConnection.removeStream(): Elimina un objeto MediaStream como recurso local de audio o vídeo.

• RTCPeerConnection.close(): Cierra la conexión.

• RTCPeerConnection.createDataChannel(): Crea un objeto RTCDataChannel asociada a la conexión. El método obtiene la configuración requerida como su fiabilidad.

• RTCPeerConnection.createDTMFSender(): Crea un objeto RTCDTMSender asociado a un especifico MediaStreamTrack que es válido para enviar la señalización por DTMF a través de la conexión.

• RTCPeerConnection.getStats(): Crea un objeto RTCStatsReport que contiene y permite el acceso a las estadísticas relativas a la conexión.

• RTCPeerConnection.setIdentityProvider(): Asigna la identidad del proveedor (IdP) al triplete dado en el parámetro: su nombre, el protocolo utilizado para comunicarse con él (opcional) y un nombre de usuario opcional. La IdP se utilizará sólo cuando se necesita una afirmación.

• RTCPeerConnection.getIdentityAssertion(): Inicia la reunión de una afirmación de identidad. Esto sólo tiene efecto si el signalingState no está "cerrado". No se espera para la aplicación tratar con el RTCPeerConnection: esto se hace de forma automática; una llamada explícita sólo permite anticipar la necesidad.

3.6.3.- `RTCDataChannel`

La interfaz `RTCDataChannel` de **WebRTC** representa un canal de datos bidireccional entre pares de una conexión. Los objetos de este tipo se pueden crear utilizando `RTCPeerConnection.createDataChannel()`, o se reciben en un evento de tipo `RTCDataChannel.datachannel`. Evento en una `RTCPeerConnection` existente.

Para utilizar esta característica del **API** de **WebRTC** es necesario establecer una conexión con un par (*peer*), se configura exactamente igual que para realizar una videollamada y se envían los mensajes por medio del objeto `RTCPeerConnection` creando una instancia de un nuevo canal, se utiliza el método de la interfaz `createDataChannel()` para crear el canal y luego se implementa sus métodos muy parecidos a los que utiliza un Socket, que más adelante veremos:

• `RTCPeerConnection.createDataChannel()`: Crea un objeto `RTCDataChannel` asociado a la conexión. El método obtiene la configuración requerida como su fiabilidad.

• `RTCPeerConnection.ondataChannel(event)`: Instancia una función de la interfaz del objeto `RTCPeerConnection` y se tienen que implementar los métodos de la interfaz de la instancia: `event.onopen, event.onclose, event.onmessage`.

3.7.- Tecnologías involucradas en WebRTC

3.7.1.- Real time transport protocol (RTP)

El protocolo de transporte de tiempo real (RTP), es un protocolo de nivel de sesión utilizado para la transmisión de información en tiempo real, como por ejemplo audio y vídeo en una vídeo-conferencia. Está desarrollado por el grupo de trabajo de transporte de Audio y Vídeo del IETF, publicado por primera vez como estándar en 1996 como el RFC 1889, y actualizado posteriormente en 2003 en el RFC 3550, que constituye el estándar de Internet STD 64.

Inicialmente se publicó como protocolo multicast, aunque se ha usado en varias aplicaciones unicast. Se usa frecuentemente en sistemas de streaming, junto a RTSP, videoconferencia y sistemas push to talk (en conjunción con H.323 o SIP). Representa también la base de la industria de VoIP.

El RFC 1890 ha quedado obsoleto por el RFC 3551 (STD 65), el cual define un perfil para conferencias de audio y vídeo con control mínimo. El RFC 3711, por otro lado, define SRTP (Secure Real-time Transport Protocol), una extensión del perfil de RTP para conferencias de audio y vídeo que puede usarse opcionalmente para proporcionar confidencialidad, autenticación de mensajes y protección de reenvío para flujos de audio y vídeo. Este protocolo utiliza el protocolo UDP para la transmisión de datos.

3.7.2.- User datagram protocol (UDP)

El protocolo de datagramas de usuario es un protocolo de nivel de transporte (encapsulado en la capa 4 del Modelo OSI) que dispone de un modo de comunicación informático basado en la conmutación de paquetes o datagramas dentro del entorno de un conjunto interconectado de redes de ordenadores. Este protocolo asume que el protocolo de internet (IP) se utiliza como protocolo subyacente.

Este protocolo proporciona un procedimiento para los programas de aplicación para enviar mensajes a otros programas con un mínimo de mecanismo de protocolo. El protocolo está orientado a la transacción, pero la entrega y la duplicación en la protección no están garantizados. Las aplicaciones que requieren la entrega confiable ordenada de flujos de datos debe utilizar el Protocolo de Control de Transmisión (TCP). Es decir, este protocolo (UDP) no posee un estado de los paquetes o datagramas enviados, sin esperar una confirmación de entrega, ni control de flujo, por lo que algunos paquetes pueden adelantarse a otros.

User Datagram Protocol (UDP) es un protocolo mínimo de nivel de transporte orientado a mensajes documentado en el RFC 768 de la IETF. Este protocolo es el más utilizado para transmitir vídeo o audio. El formato de este protocolo se recoge en la siguiente tabla:

Bits	Bits 0 – 15	Bits 16 – 31
0	Puerto origen	Puerto destino
32	Longitud del mensaje	Suma de verificación
64	Datos	

Tabla 3.6.- Formato del protocolo UDP

3.7.3.- Network address translation (NAT)

La traducción de direcciones de red, es un método por el cual las direcciones IP son mapeadas de un extremo a otro, en un intento de proporcionar un enrutamiento transparente a los anfitriones. Tradicionalmente, los dispositivos NAT se utilizan para conectar un dominio de direcciones aisladas con direcciones privadas no registradas a un ámbito externo con direcciones registradas a nivel mundial únicos. Esta terminología se recoge en el RFC 2663.

Las direcciones privadas son rangos especiales de direcciones IP que se reservan para ser utilizadas en redes locales, y se llaman privadas (o no enrutables) porque no pueden ser utilizadas en Internet ya que los routers intermedios que componen internet, no "entienden" este tipo de direcciones y no las encaminan. Esto da una gran flexibilidad para configurar redes locales, ya que por ejemplo, yo puedo tener en mi red local direcciones del tipo 192.168.0.0, y mi vecino también, pero como esas direcciones no salen de la red local no hay ningún conflicto. Lo que hace NAT es traducir las IPs privadas de la red en una IP pública para que la red pueda enviar paquetes al exterior; y traducir luego esa IP pública, de nuevo a la IP privada del pc que envió el paquete, para que pueda recibirlo una vez llega la respuesta.

3.7.4.- Session transversal utilities for NAT (STUN) / Transversal using relays around NAT (TURN)

La utilidades de sesión transversal para NAT (**STUN**) recogido en RFC 538, es un protocolo de red del tipo cliente-servidor que sirve como herramienta para otros protocolos para tratar con el traductor de direcciones de red (NAT) de manera transversal.

STUN proporciona una configuración para una punto final para determinar la dirección IP y el puerto asignado por un NAT que corresponde a su dirección IP privada y el puerto. También se puede utilizar para comprobar la conectividad entre dos puntos extremos, y como un protocolo de mantenimiento de conexión para mantener los enlaces de NAT o para retransmitir paquetes entre dos puntos finales. Esta información es utilizada para configurar una comunicación UDP entre dos hosts que se encuentren tras enrutadores NAT.

STUN funciona con muchos NAT existentes, y no requiere ningún comportamiento especial de ellos. STUN no es una solución NAT transversal por sí mismo. Más bien, es una herramienta para ser utilizada en el contexto de una solución de NAT transversal. Este es un cambio importante

respecto a la versión anterior de esta especificación (RFC 3489 obsoleta), que presentó STUN como una solución completa.

El uso de relés de recorrido alrededor de NAT (**TURN**) es un protocolo que asiste en el recorrido de traductores de direcciones de red (NAT) o servidores de seguridad para aplicaciones multimedia. Puede ser utilizado con el Protocolo de Control de Transmisión (TCP) y el Protocolo de datagramas de usuario (UDP). Es muy útil para los clientes en redes enmascaradas por los dispositivos NAT simétricos. TURN no ayuda en el funcionamiento de los servidores en los puertos bien conocidos en la red privada a través de un NAT; que soporta la conexión de un usuario detrás de una NAT que sólo un único par, como en la telefonía, por ejemplo. TURN se especifica en el *RFC 5766*. Una actualización de giro para IPv6 se especifica en el *RFC 6156*. El TURN esquema URI está documentado en el *RFC 7065*.

El proceso del protocolo TURN comienza cuando un equipo cliente quiere comunicarse con un equipo del mismo nivel para una transacción de datos, pero no puede hacerlo debido a ambos, cliente y los compañeros, al estar detrás sus respectivos NATs. Si STUN no es una opción porque uno de los NATs sea un NAT simétrica (un tipo de NAT que se sabe que no es compatible STUN), TURN debe ser utilizado.

En primer lugar, el cliente contacta con un servidor TURN con una solicitud de "Asignar". La solicitud Asignar pide al servidor TURN para asignar parte de sus recursos para el cliente para que pueda ponerse en contacto con un par. Si la asignación es posible, el servidor asigna una dirección para que el cliente utilice como un relé, y la envía al cliente una "asignación exitosa" respuesta, la cual contiene una "dirección de transporte transmitida asignada", ubicado en el servidor TURN.

En segundo lugar, el cliente envía una solicitud al servidor `CreatePermissions` TURN para crear un sistema de permisos para las comunicaciones entre pares-servidor de comprobación. En otras palabras, cuando un compañero es finalmente contactado y envía la información al servidor TURN para ser transmitida a cliente, el servidor TURN utiliza los permisos para verificar que el servidor de comunicación entre pares TURN es válida. Una vez creados los permisos, el cliente tiene dos opciones para el envío de los datos reales, que puede utilizar el mecanismo de envío, o se puede reservar un canal utilizando la solicitud `ChannelBind`. El mecanismo de envío es más sencillo, pero contiene un encabezado más grande, 36 bytes, que puede aumentar sustancialmente el ancho de banda en una conversación TURN retransmitido. Por el contrario, el método `ChannelBind` es más ligero: la cabecera se encuentra a sólo 4 bytes, pero requiere un canal para ser reservados que no necesita ser refrescada periódicamente, entre otras consideraciones.

El uso de cualquier método, enviar o unión de canal, el servidor recibe a su vez los datos del cliente y lo transmite a los pares usando datagramas UDP, que contienen como su Dirección Fuente de la "dirección asignada Retransmisión de transporte". El par recibe los datos y responde, de nuevo utilizando un datagrama UDP como protocolo de transporte, el envío de los datagramas UDP a la dirección del relé en el servidor TURN. El servidor TURN recibe el datagrama UDP entre iguales, comprueba los permisos y si son válidas, lo envía de vuelta al cliente. Este proceso recibe alrededor incluso NATs simétricos porque tanto el cliente como pares por lo menos hablar posible por separar al servidor, que se ha asignado una dirección IP para la comunicación de relé. Mientras que a su vez

es más robusto y realiza el recorrido en más tipos de NAT, una comunicación TURN transmite toda la comunicación a través del servidor que requiere mucho más ancho de banda que el protocolo STUN, que normalmente sólo se resuelve la dirección IP del lado público y retransmite la información a clientes y compañeros para que los utilice en comunicación directa. Por esta razón, los mandatos del protocolo ICE paralizantes lo utiliza como primer recurso y gira el uso cuando se trata de NATs simétricos, u otras situaciones en las que STUN no se puede utilizar.

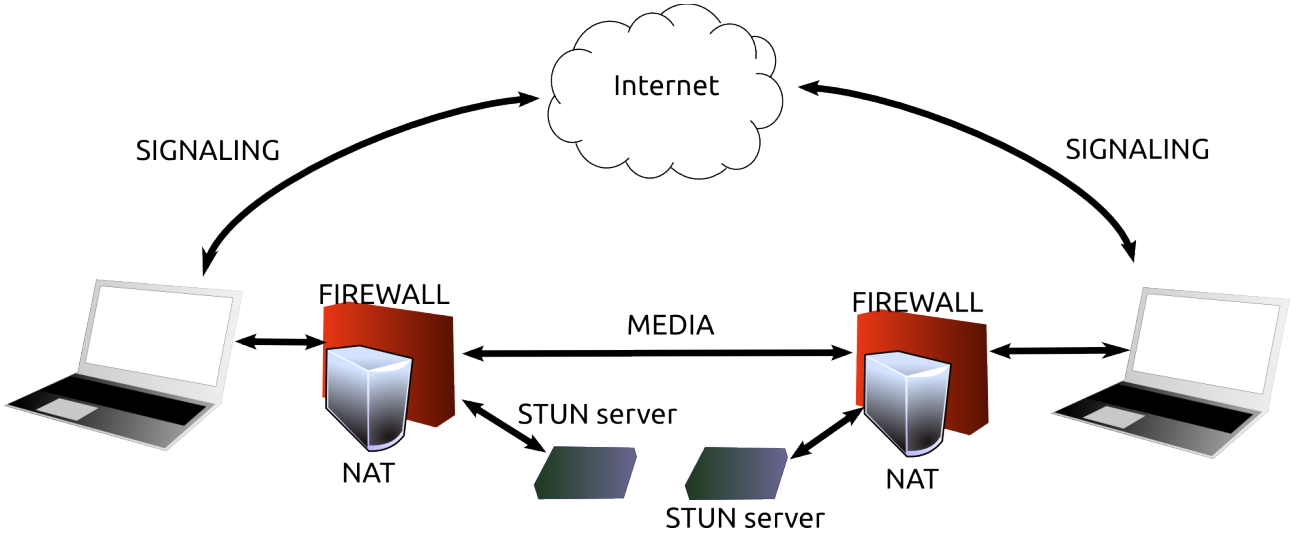

Figura 3.2.- Diagrama servidor STUN/TURN en WebRTC

3.7.5.- Interactive connectivity establishement (ICE)

El establecimiento de conectividad interactiva (**ICE**) es una técnica utilizada en las redes de computadoras que involucra traductores de direcciones de red (NAT), aplicaciones de voz sobre protocolo de internet (VoIP), comunicación entre pares (peer-to-peer), vídeo, mensajería instantánea y otros medios interactivos. En tales aplicaciones, NAT es un componente importante para facilitar las comunicaciones que involucran los hosts de la red privada de instalaciones, a menudo situados detrás de los cortafuegos.

ICE ha sido desarrollado por la Internet Engineering Task Fuerza MMUSIC grupo de trabajo y se publicó como el RFC 5245 y ha sido actualizado por el RFC 6336 que propone un estandar para las opciones del establecimiento de la conectividad interactiva.

Para conseguir la información sobre la red cuando ha atravesado los firewalls y NATs, WebRTC ejecuta el protocolo ICE que permite que se prueben distintas rutas para comunicar dos terminales entre sí, acordando una ruta común. De forma que, por ejemplo, si están en la misma red local, se comparte la información de forma local sin necesidad de utilizar otros servicios. Pueden existir 3 tipos de candidatos:

1. Host candidates: Son candidatos locales, como por ejemplo las tarjetas de red del equipo, conexiones LAN. Contiene direcciones IP privadas.

2. Reflexive candidates: Obtiene los candidatos realizando consultas a servidores STUN y contiene IPs públicas.

3. Relay candidates: Se obtienen realizando consultas a servidores TURN y contiene IPs públicas. Todos los datos se transmiten a través de él. Permite atravesar NATs simétricos.

Primero el protocolo ICE, mediante un servidor STUN, intenta conectar los pares directamente utilizando el protocolo UDP para conseguir la latencia más baja posible. Si no es posible realizar la conexión mediante UDP, ICE intenta hacerlo por HTTP, usando TCP. En caso de que la conexión vuelva a fallar (normalmente porque al menos uno de los pares está detrás de un NAT simétrico o un firewall) ICE hace uso de un servidor TURN para obtener las credenciales. Una vez se obtienen los candidatos (IP y puerto) se añaden al mensaje SDP y se comparten los pares. De esta manera se puede realizar una videollamada para recibir e enviar los datos.

Los candidatos se añaden en la conexión entre pares o en la RTCPeerConnection por medio de las funciones configuradas en el cliente en JavaScript para que WebRTC pueda obtener la configuración óptima para enviar y recibir los datos.

```
candidate:2 1 UDP 1686052863 46.26.90.45 32816 typ srflx raddr
192.168.0.157 rport 32816
candidate:2 2 UDP 1686052862 46.26.90.45 46026 typ srflx raddr
192.168.0.157 rport 46026
...
candidate:2 2 UDP 1686052862 46.26.90.45 48598 typ srflx raddr
192.168.0.157 rport 48598
```

Tabla 3.7.- Ejemplo de los candidatos (ICE) enviados y recibidos para configurar WebRTC

3.7.6.- Datagram transport layer security (DTLS)

La capa de transporte seguro de datagramas es un protocolo definido y publicado en el RFC 4347 que proporciona privacidad en las comunicaciones para protocolos de datagramas. Este protocolo permite a las aplicaciones cliente/servidor comunicarse de manera que se eviten las escuchas no deseadas (eavesdropping), accesos no permitidos, o modificación de mensajes. El protocolo DTLS está basado en el protocolo TLS y proporciona garantías de seguridad equivalentes. La semántica de los datagramas de los protocolos subyacentes no es modificada al utilizar DTLS.

3.7.7.- Session description protocol (SDP)

El protocolo de inicio de sesión (SDP), es un protocolo para inicialización de los flujos multimedia donde se describe los parámetros de cada uno de los participantes. Este protocolo está recogido y redactado en el RFC 4566 por la IETF, dejando obsoletos los anteriores RFC 2327 de abril de 1998 y RFC 3266 de junio de 2002.

Al iniciar teleconferencias multimedia, voz sobre IP llama, la transmisión de vídeo, u otras

sesiones, existe un requisito para transmitir detalles medios, direcciones de transporte, y otra descripción de la sesión metadatos a los participantes. SDP proporciona una representación estándar para dicha información, con independencia de cómo se transporta esa información. SDP es puramente un formato para la descripción de la sesión, no incorpora un protocolo de transporte, y se pretende utilizar diferentes protocolos de transporte apropiados como el Protocolo de Anuncio de Sesión, Session Initiation Protocol, Real Time Streaming Protocol, el correo electrónico utilizando las extensiones MIME, y el hipertexto Protocolo de transporte.

SDP está pensado para describir sesiones de comunicación multimedia cubriendo aspectos como anuncio de sesión, invitación a sesión y negociación de parámetros. SDP no se encarga de entregar los contenidos propiamente dichos sino de entablar una negociación entre las entidades que intervienen en la sesión como tipo de contenido, formato, y todos los demás parámetros asociados.

Una sesión se describe con una serie de atributos, cada uno en una línea. Los nombres de estos atributos son un carácter seguido por '=' y el valor respectivo. Existen parámetros opcionales, denotados con '=*'. Los valores pueden ser una cadena ASCII, o una secuencia específica de tipos separada por espacios. La sintaxis de SDP se puede ampliar y ocasionalmente se agregan nuevos atributos a la especificación. En la siguiente tabla se muestra un formato para el uso de SDP:

```
Descripción de la sesión
     v=  (Versión del protocolo)
     o=  (Origen e identificador de sesión)
     s=  (Nombre de sesión)
     i=  (Información de la sesión)
     u=  (URI de descripción)
     e=  (Correo electrónico)
     p=  (Número telefónico)
     c=* (Información de conexión)
     b=* (Cero o más líneas con información de ancho de banda)
     z=* (Ajustes de zona horaria)
     k=* (Clave de cifrado)
     a=* (Cero o más líneas de atributos de sesión)

Descripción de tiempo
     t=  (Tiempo durante el cual la sesión estará activa)
     r=* (Cero o más veces de repetición)

Descripción de medios, si está presente
     m=  (Nombre de medio y dirección de transporte)
     i=* (Título)
     c=* (Información de conexión)
     b=* (Cero o más líneas con información de ancho de banda)
     k=* (Clave de cifrado)
     a=* (Cero o más líneas de atributos de sesión)
```

Tabla 3.8.- Formato del Protocolo de descripción de sesión

Capítulo 4
Tecnologías Web

4.- Tecnologías Web

4.1.- HTML5

HTML, son las siglas de ***HyperText Markup Language*** (lenguaje de marcas de hipertexto), es el lenguaje de marcas más extendido porque es el fundamento del Word Wide Web para la elaborar páginas web. HTML es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, vídeos, entre otros. Es un estándar a cargo de la W3C.

HTML es considerado como una ampliación de SGML, en 1993 fue reconocido por la IETF. Tim Berners-Lee en 1991 describió 18 elementos en un diseño inicial, 13 de estos elementos existen en HTML4.

HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>,/). El HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir o hacer referencia a un tipo de programa llamado *script*, el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML. Consta de varios componentes vitales, entre ellos los *elementos* y sus *atributos*, *tipos de datos* y la *declaración de tipo de documento*.

HTML también sirve para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (como HTML 4.01 y anteriores).

El formato HTML tiene una estructura básica o esquema mínimo para definir un documento HTML:

- Un documento HTML comienza siempre con la etiqueta <HTML>, que indica que el documento en cuestión está construido con dicho lenguaje.
- La mayoría de las etiquetas son pareadas, es decir, <...> corresponde al principio de la acción y </...> indica el fin de dicha acción.
- Por tanto, una página web estará siempre contenida entre las etiquetas <HTML> y </HTML>.
- Por otra parte, todo documento HTML consta de dos partes: la cabecera (**head**) y el cuerpo del documento (**body**):

    1. La cabecera contiene básicamente información destinada al browser (o navegador), que queda oculta al usuario. Su etiqueta es <HEAD>.

2. El cuerpo es el documento que ve el usuario. Su etiqueta es <BODY>.

- La extensión que tiene que tener un archivo HTML es: ***.HTML** o ***.HTM**:

```
<HTML>
      <HEAD>
      </HEAD>
      <BODY>
      </BODY>
</HTML>
```

Código 4.1.- Esquema básico de un documento HTML5

HTML5 es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML donde se especifica dos variantes de sintaxis para HTML: una «clásica», HTML (text/html), conocida como *HTML5*, y una variante XHTML conocida como sintaxis *XHTML5* que deberá servirse con sintaxis XML (application/xhtml+xml). Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo. La versión definitiva de la quinta revisión del estándar se publicó en octubre de 2014.

HTML5 introduce soporte integrado para el contenido multimedia gracias a los elementos <audio> y <video>, ofreciendo la posibilidad de insertar contenido multimedia en documentos HTML, en este proyecto se utilizará las etiquetas HTML5 como son <video> y <canvas>,:

4.1.1.- Etiqueta <video>

- La etiqueta vídeo sirve tanto para reproducir vídeo como para capturar el vídeo de la cámara y mostrarlo en pantalla. Los formatos compatibles con cada navegador es distinto aunque se pueden configurar para que con un mismo código pueda ser ejecutado por cualquier navegador.

```
<video controls>
      <source src="algunvideo.webm" type="video/webm">
      <source src="algunvideo.mp4" type="video/mp4">
      Lo siento tu navegador no soporta la etiqueta HTML5 vídeo.
</video>
```

Código 4.2.- Ejemplo etiqueta <video> en HTML5

Formatos y códecs compatibles :

- **Ogg**: es un formato contenedor multimedia abierto y libre desarrollado y mantenido por la Fundación Xiph.Org. Ogg no está restringido por las patentes de software y está diseñado para proporcionar una difusión multimedia de flujo eficiente y de alta calidad.

-**Webm**: es un formato contenedor multimedia abierto y libre desarrollado por Google y orientado a usarse con HTML5. Tiene una licencia permisiva similar a la licencia BSD. Está

compuesto por el códec de vídeo VP8 (desarrollado originalmente por On2 Technologies) y el códec de audio Vorbis dentro de un contenedor multimedia Matroska.

- **Vorbis**: es un códec de audio digital general con pérdidas, libre desarrollado por la Fundación Xiph.Org, que utiliza el formato de archivo de audio o contenedor Ogg.

- **H.264**: es un códec de alta compresión de vídeo desarrollado por el ITU-T Video Coding Experts Group (VCEG) y el ISO/IEC Moving Picture Experts Group (MPEG). Es un formato muy extendido y en reproductores de vídeo de todo tipo. El códec H.264 se puede combinar con los códecs de audio ACC o MP3 dentro del contenedor MPEG-4 (conocido como MP4).

- **V8**: códec de vídeo publicado en RFC6386, fue a sus inicios desarrollado por On2 Technologies y adquirido por Google en el 2009, siendo este liberado el 19 de Mayo de 2010. Este códec es parte del proyecto WebM.

La etiqueta **<video>** es compatible en muchos navegadores aunque algunos códecs de vídeo no lo son, en esta tabla se muestran los formatos compatibles y las versiones de los navegadores que la soportan actualmente:

Característica\Navegador	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Soporte básico	3.0	3.5	9.0	10.50	3.1
<audio>: WAVE, PCM	Si	3.5	No soportado	No soportado	3.1
<audio>: WebM, Vorbis	Si	4.0	No soportado	10.60	3.1 (plugin)
<audio>: Ogg, Vorbis	Si	3.5	No soportado	10.50	3.1 (plugin)
<audio>: MP4, MP3	Si (no en Chromium)	Parcial	9.0	No soportado	3.1
<audio>: MP4, AAC	Si (no en Chromium)	Parcial	9.0	No soportado	3.1
<audio>: Ogg, Opus	27.0	15.0	?	?	?
<video>: WebM, VP8, Vorbis	Si	4.0	9.0 (plugin)	10.60	3.1 (plugin)
<video>: Ogg, Theora, Vorbis	Si	3.5	No soportado	10.50	3.1 (plugin)
<video>: MP4, H.264, MP3	Si (No en Chromium)	Parcialmente	9.0	No soportado	3.1
<video>: MP4, H.264, AAC	Si	Parcialmente	9.0	No soportado	3.1

Tabla 4.1.- Códecs de video y audio soportados por los navegadores

El captura de la webcam puede pasarse a la etiqueta <video> por medio del objeto MediaStream el cual puede se configurado por medio de JavaScript, pudiendo también grabar la captura de pantalla en un vídeo WebM o en formato AVI.

4.1.2.- Etiqueta <canvas>

- El elemento "canvas" puede definirse como un entorno para crear imágenes o rederizaciones dinámicas. El propio elemento es relativamente simple, lo único que hay que especificar al usarlo son sus dimensiones y por medio de su identificador "id" es fácil de referenciar para poder ser manipulado a través de JavaScript.

```
<canvas id="canvasLocalVideo"></canvas>
```

Código 4.3.- Ejemplo etiqueta `<canvas>` en HTML5

4.2.- JavaScript

JavaScript es el lenguaje interpretado orientado a objetos desarrollado por *Netscape* que se utiliza en millones de páginas web y aplicaciones de servidor en todo el mundo. JavaScript de Netscape es un gran conjunto de inscripto estándar de la edición de ECMA4-262 3 (ECMAScript) que presenta sólo leves diferencias respecto a la norma publicada.

ECMAScript es una especificación de lenguaje de programación publicada por ECMA International. El desarrollo empezó en 1996 y estuvo basado en el popular lenguaje JavaScript propuesto como estándar por Netscape Communications Corporation. Actualmente está aceptado como el estándar ISO 16262.

JavaScript se utiliza principalmente en su forma del lado del cliente (FrontEnd), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Actualmente se puede también programar con JavaScript en el lado del servidor (BackEnd), por ejemplo actualmente con el servidor Node.js.

JavaScript no es "Java interpretativo", es decir que JavaScript es un lenguaje de programación dinámico que soporta construcción de objetos basado en prototipos. La sintaxis básica es similar a Java y C++ con la intención de reducir el número de nuevos conceptos necesarios para aprender este lenguaje.

El lenguaje JavaScript tiene la singularidad de poder incluir en sus funciones otras funciones en el argumento de entrada creando de esta manera el elemento callBack o de vuelta. Este mecanismo sirve para ejecutar varias funciones de manera síncrona, una después de otra. Los callBacks no son asíncronos, es decir, dispara el callBack y cuando termina la ejecución de esa función de albahaca continua desde donde lo disparó. En el siguiente ejemplo se muestra un ejemplo de como declarar un callBack:

```
function ejemploCallBack(callBack1, callBack2, callBack3){
     //primera ejecución
     callBack1('1º callBack');

     //segunda ejecución
     callBack2('2ª callBack');

     //tercera ejecución
     callBack3('3ª callBack');
}
```

Código 4.4.-. Ejemplo de una función con callBacks. Lenguaje JavaScript

¿Qué implementaciones de JavaScript están disponibles actualmente en los navegadores?

• mozilla.org alberga dos implementaciones de JavaScript. La **primera** implementación de JavaScript fue creada por Brendan Eich en Netscape, y desde entonces ha sido actualizada (en JavaScript 1.5) para cumplir con ECMA-262 Edición 5. Este motor, cuyo nombre en código es SpiderMonkey, se implementa en C. El motor Rhino, creado principalmente por Norris Boyd (también en Netscape) es una implementación de Javascript en Java. Al igual que SpiderMonkey, Rhino cumple con ECMA-262 Edición 3.

• V8 de Google, **V8** es un motor de código abierto para JavaScript creado por Google, siendo su programador jefe Lars Bak. Está escrito en C++ y es usado en Google Chrome. También el "V8 JavaScript" está integrado en el navegador de internet del sistema operativo Android 2.2 “Froyo”.

• JavaScriptCore (SquirrelFish/Nitro) usado en algunos navegadores basados en WebKit tales como Apple Safari.

• Carakan en Opera.

La función de JavaScript a parte de poder realizar cálculos, definir objetos, funciones, o métodos es poder interactuar con el árbol DOM de HTML para realizar operaciones en él de manera dinámica y atender eventos o realizar peticiones por medio de su objeto `XMLHttpRequest`.

Para obtener JavaScript un elemento del DOM HTML se suele utilizar el identificador del objeto(etiqueta) HTML, también se puede obtener por referencias o por clases, en el siguiente ejemplo se muestra como obtener un objeto del árbol DOM HTML por medio de su identificador (getElementById) para poder manipularlo con JavaScript:

```
<!DOCTYPE html>
<html>
...
<div id="content">  // Etiqueta HTML
...
<script type="text/javascript">
var body = document.getElementById("content");
...
```

Código 4.5.- Ejemplo como obtener un elemento del DOM HTML con `getElementById()`. Lenguaje JavaScript

4.2.1.- Objeto `XMLHttpRequest`

`XMLHttpRequest` es un objeto JavaScript que fue diseñado por Microsoft y adoptado por Mozilla, Apple y Google. Actualmente es un estándar de la W3C. Proporciona una forma fácil de obtener información de una URL sin tener que recargar la pagina completa. Una pagina web puede actualizar sólo una parte de la pagina sin interrumpir lo que el usuario esta haciendo. `XMLHttpRequest` es ampliamente usado en la programación AJAX.

A pesar de su nombre, `XMLHttpRequest` puede ser usado para recibir cualquier tipo de dato, no solo XML, y admite otros formatos además de HTTP (incluyendo file y ftp).

En la siguiente tabla se muestra como declarar una instancia del objeto `XMLHttpRequest`:

```
var req = new XMLHttpRequest();
```

Código 4.6.- Ejemplo declarar una instancia del objeto `XMLHttpRequest`. Lenguaje JavaScript

4.2.2.- Objeto `JSON`

Notación de Objetos de JavaScript (JavaScript Object Notation (`JSON`)) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999. `JSON` es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que `JSON` sea un lenguaje ideal para el intercambio de datos.

`JSON` está constituido por dos estructuras:

- Una colección de pares de **nombre/valor**. En varios lenguajes esto es conocido como un *objeto*, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una **lista ordenada** de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

En la siguiente tabla se muestra un ejemplo de un objeto `JSON`:

```
{"usuarios":[
      {"nombre":"Juan", "apellido":"Marquez"},
      {"nombre":"Ana", "apellido":"Sanz"},
      {"nombre":"Pedro", "apellido":"Jimenez"}
]};
```

Código 4.7.- Ejemplo estructura de un objeto `JSON`. Lenguaje JavaScript

Los métodos utilizados en este proyecto para manipular los datos y los objetos `JSON` son:

- **`JSON.parse(...)`**: Este método transforma una cadena de texto en formato de objeto o lista del formato JSON en un objeto JSON.

- **`JSON.stringify(...)`**: Este método transforma un objeto de tipo `JSON` en una cadena de texto.

En resumen, para enviar la información con el objeto `XMLHttpRequest` se encapsula primero estos datos en un objeto `JSON` para poder manejarlos y poder enviarlos, el siguiente ejemplo se muestra como encapsular un objeto `JSON` y su posterior envío por el método POST a una URL:

```
function sendMessage() {
     try {
          var message = {  // Objeto JSON
               type : "type data",
               "message" : "message data"
          };
          var msgString = JSON.stringify(message);
          path = '/message';
          var xhr = new XMLHttpRequest(); // Objeto XMLHttpRequest
          xhr.open('post', path, true);
          xhr.send(msgString); // XMLHttpRequest envía los datos
     } catch (error) { // Bloque catch para capturar una excepción
          console.log(error);
     };
}
```

Código 4.8.- Ejemplo de objeto `JSON` y objeto `XMLHttpRequest`. Lenguaje JavaScript

4.3.- CSS

Hoja de estilo en cascada (**C**ascading **S**tyle **S**heets) es una tecnología que utiliza un lenguaje propio para definir y describir como los elementos deben ser renderizados en una pantalla, en una impresión en papel o en otro medio. Esta tecnología se suele utilizar para crear la presentación y renderización de un documento escrito en lenguaje de marcas HTML o XML. La tecnología CSS es usada por la mayoría de los sitios web para crear páginas web visualmente atractivas, interfaces de usuario para aplicaciones web y aplicaciones móviles. El World Wide Web Consortium (W3C) es el encargado de formular la especificación de las hojas de estilo que sirven de estándar para los navegadores y para los agentes de usuario.

Con la tecnología CSS se obtiene un control centralizado del estilo de una página web facilitando los cambios o modificaciones porque el mismo estilo puede ser aplicado a muchos elementos con una mismo selector optimizando de esta forma también el ancho de banda necesario para la carga del mismo.

Los selectores son el nexo de unión entre la hoja de estilo y las etiquetas del DOM HTML del documento al que se aplica. Una hoja de estilos se compone de una lista de reglas, cada regla o conjunto de reglas consiste en uno o mas selectores y un bloque de declaración. Cada bloque de estilo del selector se define entre llaves, dentro se declara la propiedad o propiedades del estilo y se asigna su valor o valores.

La siguiente tabla muestra como declarar una propiedad a un selector del DOM HTML:

```
selector {
     propiedad: valor;
}
```

Código 4.9.- Ejemplo de código CSS básico

El código de estilo CSS puede estar definido y cargado en varias partes del documento HTML:

- **Externo:** en un fichero separado con la extensión css (*.css) y referenciado en el documento con la etiqueta <*link*>.

```
<link href="/css/style.css" rel="stylesheet" type="text/css">
```

Código 4.10.- Declaración en el documento HTML un fichero externo CSS

- **Interno: e**n el mismo documento HTML con la etiqueta *<style>* para definir estilos generales solo dentro del documento que lo contiene.

```
<style>
selector{
     propiedad: valor;
}
</style>
```

Código 4.11.- Declaración 01 en el documento HTML la etiqueta `<style>` CSS

- **En linea:** en cada etiqueta particular con el atributo "style".

```
<div style="propiedad:valor;"></div>
```

Código 4.12.- Declaración 02 en el documento HTML la etiqueta `<style>` CSS

La prioridad con la que se aplica el estilo se realiza de abajo hacia arriba, es decir, tiene prioridad la renderización de la etiqueta particular con el atributo "style" que el estilo general definido dentro de este documento y la renderización del estilo general tiene prioridad a la carga del estilo desde un fichero externo.

4.3.1.- Responsive Design

El diseño web responsivo **(**Responsive Web Design), es una técnica de diseño y desarrollo cuyo objetivo es adaptar la interfaz gráfica de las páginas web al dispositivo donde se está renderizando.

Hoy día las páginas web se visualizan en multitud de tipos de dispositivos como en tablets, smartphones, libros electrónicos, portátiles, ordenadores, etc. y cada dispositivo tiene sus propias características concretas: tamaño de pantalla, resolución, potencia de CPU, capacidad de memoria, entre otras. Esta tecnología pretende que con un solo diseño web, se tenga una visualización adecuada en cualquier dispositivo.

En la siguiente tabla se muestra un ejemplo de código CSS para definir el tamaño de la fuente según la resolución de la pantalla del dispositivo utilizado para renderizar una web:

```
@media only screen and (max-width: 500px){
     body { font-size:200%;}
}
@media only screen and (max-width: 750px){
     body { font-size:160%;}
}
@media only screen and (max-width: 1000px) {
     body { font-size:120%;}
}
@media only screen and (max-width: 1250px) {
     body { font-size:100%;}
}
```

Código 4.13.- Ejemplo responsive design CSS

4.4.- Java

Java es un lenguaje de programación de propósito general, concurrente y orientado a objetos. Esta plataforma informática fue comercializada por primera vez en 1995 por Sun Microsystems. Java fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible cliente-servidor de web

La intención de esta plataforma es permitir a los desarrolladores de aplicaciones que escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como *WORA*, o "*write once, run anywhere*"), es decir que el código es ejecutado en una plataforma y no tiene que ser recompilado para correr en otra.

Java es uno de los lenguajes de programación más populares en uso y la base para prácticamente todos los tipos de aplicaciones de red, además del estándar global para desarrollar y distribuir aplicaciones móviles y embebidas, juegos, contenido basado en web (cliente-servidor) y software de empresa. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet, Java está en todas partes. Con más de 10 millones de desarrolladores en todo el mundo, Java permite desarrollar, implementar y utilizar de forma eficaz interesantes aplicaciones y servicios, siendo esta tecnología muy potente y versátil.

La sintaxis de Java deriva básicamente de C y C++, pero tiene menos utilidades de bajo nivel que estos lenguajes de programación. Las aplicaciones de Java son generalmente compiladas en bytecodes que puede ejecutarse en cualquier máquina virtual Java (JVM) sin importar la arquitectura de la computadora subyacente.

Java se compone de varios elementos para poder ejecutarse en una máquina, estos elementos son:

- Java Development Kit (JDK): es un entorno de desarrollo para crear aplicaciones, applets y componentes utilizando el lenguaje de programación Java. El JDK incluye herramientas útiles para desarrollar y probar programas escritos en el lenguaje de programación Java y se ejecuta en la plataforma Java. Puede instalarse en una computadora local o en una unidad de red.

- Java Runtime Environment (JRE): es el entorno de ejecución de Java, también conocido como tiempo de ejecución de Java, es parte del Java Development Kit (JDK), un conjunto de herramientas de programación para el desarrollo de aplicaciones con Java. El entorno de ejecución de Java proporciona los requisitos mínimos para ejecutar una aplicación Java. JRE está formado por Java Virtual Machine (JVM), clases del núcleo de la plataforma Java y bibliotecas de la plataforma Java de soporte. JRE es la parte de tiempo de ejecución del software de Java, que es todo lo que necesita para ejecutar una aplicación en un explorador web.

- Java Virtual Machine (JMV): es una máquina virtual de proceso nativo, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario o bytecodes, el cual es generado por el compilador del lenguaje Java.

La descarga de Java es gratuita y se puede obtener la última versión en java.com.

4.5.- J2EE/JavaEE

Java Platform Enterprise Edition (Java EE/J2EE) es el estándar en software empresarial impulsado por la comunidad. JavaEE, se desarrolla utilizando la Java Community Process, con las aportaciones de expertos de la industria, organizaciones comerciales y de código abierto, Java User Group, y una gran número de personas. Cada versión integra nuevas características que se alinean con las necesidades de la industria, mejora la portabilidad de las aplicaciones y aumenta la productividad del desarrollador.

JavaEE tiene varias especificaciones de API, tales como JDBC, RMI, e-mail, JMS, Servicios Web, XML, etc y define cómo coordinarlos. JavaEE también configura algunas especificaciones únicas para JavaEE para componentes. Estas incluyen servlets, websockets, Enterprise JavaBeans, portlets (siguiendo la especificación de Portlets Java), JavaServer Pages y varias tecnologías de servicios web. Ello permite al desarrollador crear una Aplicación Empresarial portable entre plataformas y escalable, a la vez que integrable con tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel.

Un **contenedor web,** en la plataforma JavaEE es la implementación que hace cumplimiento del contrato de componentes web de la arquitectura J2EE. Este contrato especifica un entorno de ejecución para componentes web que incluye seguridad, concurrencia, gestión del ciclo de vida, procesamiento de transacciones, despliegue y otros servicios. Un contenedor web suministra los mismos servicios que el contenedor de JSP así como también una vista de las API de la plataforma J2EE. Un contenedor web se suministra incluido en un servidor web o J2EE.

Ejemplos de contenedores web son:

- Sun Java System Application Server

- Sun Java System Web Server

- Tomcat para Java Web Services Development Pack

4.6.- WebSocket

WebSocket es una tecnología que proporciona un canal de comunicación bidireccional sobre un único socket TCP. El websocket está diseñado para ser implementado en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente-servidor. La API de WebSocket en el cliente (FrontEnd) está siendo normalizada por el W3C, mientras que el protocolo WebSocket ya fue normalizado por la IETF publicado en el RFC 6455.

Un websocket se encarga de enviar información desde un extremo al otro o al mismo servidor sobre el protocolo TCP. Los websockets son utilizados en muchas aplicaciones donde los usuarios tiene que mantener abierta una conexión bidireccional para poder enviar y recibir información en tiempo real. Los websockets se utilizan en aplicaciones como videojuegos en linea, chats, envío de información en tiempo real, etc.

En el lado del cliente, el objeto WebSocket está ya implementado en Mozilla Firefox, Google Chrome y Safari, así como la versión móvil de Safari en iOS, en Internet Explorer y Opera, permitiendo ser programado en lenguaje JavaScript aunque también se puede programar un cliente (FrontEnd) que use websockets en varios lenguajes. En el lado del servidor un websocket se puede implementar con múltiples lenguajes y plataformas como Java, JavaScript (Node.js), Phyton, .NET y en muchas otras.

El websocket debe de ser primero creado en el lado del servidor para que los clientes puedan conectase por este canal y mantener una comunicación bidireccional. Cuando los clientes se conecten a la aplicación esta comunicación puede ser multidireccional para todos (broadcast), entre pares, por grupos o como se quiera configurar. En la siguiente imagen se representa como se envía un mensaje por medio de un websocket:

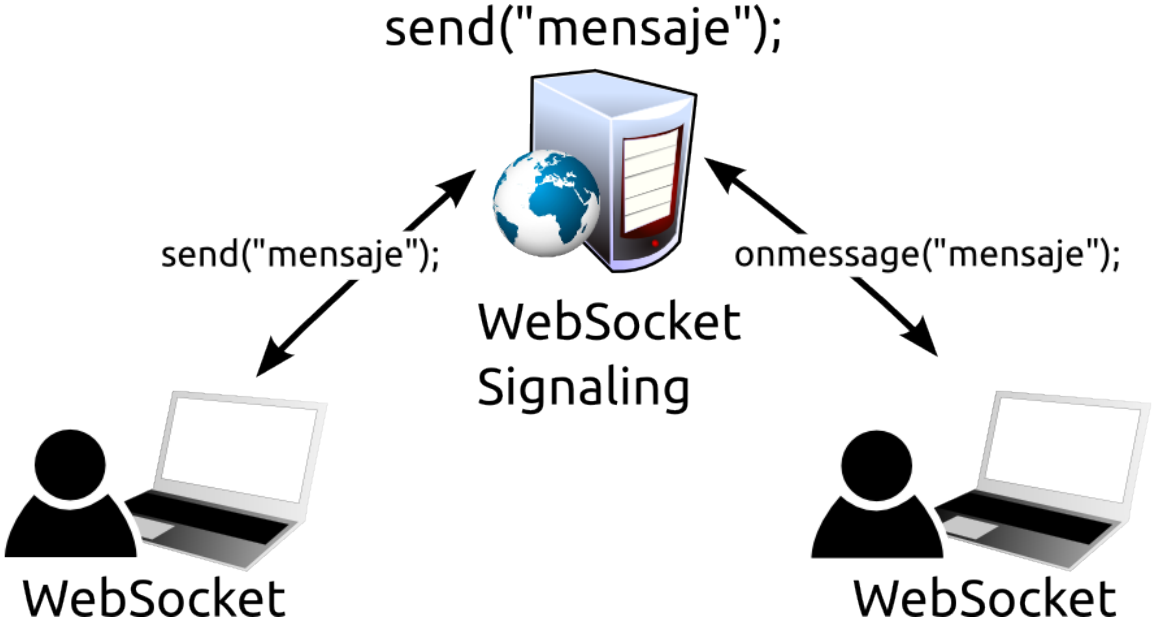

Figura 4.1.- Diagrama ejemplo de un Websocket para la comunicación entre usuarios

El objeto `WebSocket` en el lado del servidor o en el lado del cliente posee una interfaz con unos métodos que pueden ser implementados para cumplir la funcionalidad deseada. Esta interfaz es muy parecida en el lado del servidor que en el lado del cliente y poseen también métodos muy parecidos:

- **`onopen()`** : este método se ejecuta cuando el websocket se abre por un cliente.

- **`send()`** : este método envía un mensaje por el websocket.

- **`onmessage()`** : este método se ejecuta cuando se recibe un mensaje

- **`onclose():`** este método se ejecuta cuando se cierra el websocket.

- **`onerror()`** : este método se ejecuta cuando se produce un error en el websocket.

En la siguiente tabla se puede observar la estructura de los métodos comunes de un websocket en lenguaje Java:

```
import javax.websocket.server.ServerEndpoint;
import javax.websocket.OnClose;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;

@ServerEndpoint(value="/")
public class SignalingWebSocket  {
     @OnOpen
     public void onOpen(Session session) {
          SignalingWebSocket ws = this;
          ws.send("mensaje");
     }
     @OnMessage
     public void onMessage(String message) {
          ...
     }
     @OnClose
     public void onClose(Session session){
          ...
     }
}
```

Código 4.14.- Ejemplo de los métodos de un WebSocket ServerEndPoint. Lenguaje Java

El websocket que se muestra del código 4.1.4, utiliza la especificaciones de la API en JSR 356 utilizando marcadores POJO (Plain Old Java Object) con la etiqueta `@ServerEndpoint`.

En la siguiente tabla se puede observar la estructura de los métodos de un `WebSocket` en lenguaje JavaScript:

```
var location = "ws://ip_del_servidor:puerto";
var channel = new WebSocket(location);

channel.onopen = function (event) {
     channel.send(”mensaje”);
     ...
}
channel.onmessage = function (event) {
     ...
}
channel.onclose = function (event) {
     ...
}
channel.onerror = function (event) {
     ...
}
```

Código 4.15.- Ejemplo configuración de un WebSocket ClientEndPoint. Lenguaje JavaScript

Para saber que un websocket funciona correctamente, en un navegador web, se deben de observar las *trazas de red* que indican el estado del websocket, para ver estas trazas solo hay que abrir las herramientas de desarrolladores del navegador web y en la sección de red tiene que mostrar la siguiente información:

```
General
Request URL: ws://webrtc-jvrodrigo.rhcloud.com:8000/webrtc
Request Method: GET
Status Code:  101 Switching Protocols
Cabeceras de la petición GET (Petición del navegador al servidor)
Accept-Encoding: gzip, deflate, sdch
Accept-Language: es-ES,es;q=0.8
Cache-Control: no-cache
Connection: Upgrade
Cookie: JSESSIONID=B8BFB7D9617666CA5CB32458A975334D
Host: webrtc-jvrodrigo.rhcloud.com:8000
Origin: http://webrtc-jvrodrigo.rhcloud.com
Pragma: no-cache
Sec-WebSocket-Extensions: permessage-deflate;
client_max_window_bits
Sec-WebSocket-Key: ik5hi7yBaCstukxiWgpZgA==
Sec-WebSocket-Version: 13
Upgrade: websocket
User-Agent: Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/46.0.2490.86 Safari/537.36

Cabeceras de la respuesta
Connection: Upgrade
Sec-WebSocket-Accept: CuT5lJ0iW8ndGq8wHTl7+v8DvXA=
Upgrade: websocket
```

Tabla 4.2.- Trazas de ejemplo configuración de WebSocket ClientEndPoint

Las trazas de un websocket se puede visualizar con varios programas de lectura de la red o sniffers, en este caso he querido incluir un ejemplo de las trazas que crea un websocket al establecer una conexión con el programa WireShark, en la siguiente imagen se muestra una captura de pantalla del programa WireShark:

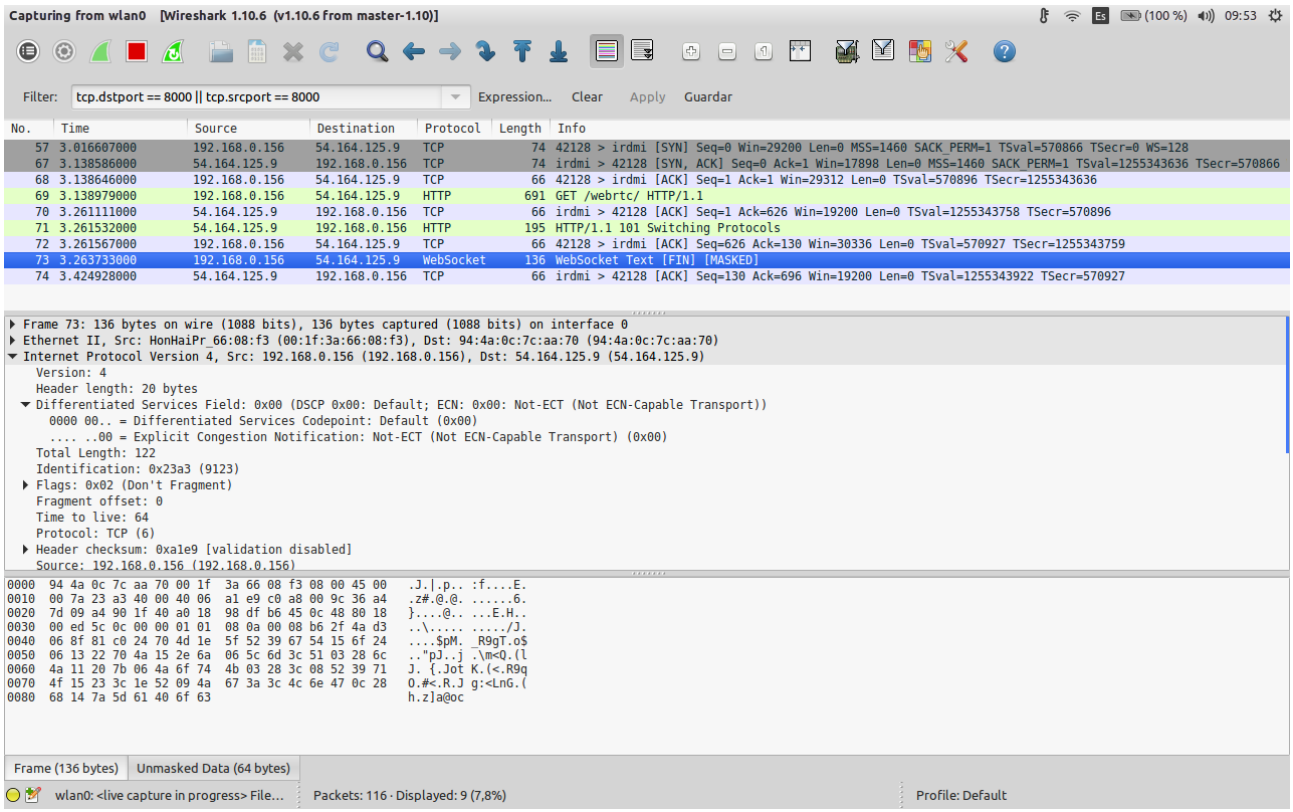

Figura 4.2.- Trazas de ejemplo capturadas de la actividad un WebSocket con WireShark

4.7.- Tomcat

Apache Tomcat es una aplicación de software de código abierto del tipo servidor web, que ejecuta tecnologías en lenguaje Java como Servlets, ServerPages y WebSockets. Todas estas especificaciones se desarrollan bajo la supervisión de la Java Community Process y la Apache Software Foundation(ASF) en un contexto abierto y participativo liberado bajo la versión de licencia Apache 2. Apache Tomcat está destinado a ser una colaboración de todos mejores desarrolladores del mundo.

Tomcat es un servidor web el cuál atenderá las peticiones del cliente (FrontEnd) para servirlas por el servidor (BackEnd) dentro de un modelo cliente-servidor.

La arquitectura de la aplicación desplegada dentro del servidor Tomcat es del tipo modelo-vista-controlador, para un mejor orden y estructura del código.

Los servlets reciben y controlan de las peticiones http realizadas por el usuario, en estos servlets se aplica la lógica de negocio definida para la aplicación utilizando los modelos necesarios para poder mostrar la información por medio de las ServerPages (JSP) o cargarlas por Simple template engine for Java (JTPL) y posteriormente servirlas al usuario (cliente) por medio del protocolo http.

En el siguiente diagrama, se muestra el papel que ejerce de servidor TomCat de contenedor de los componentes de la aplicación. También se puede apreciar como es el flujo de comunicación con el cliente atiendiendo las peticiones y sirviendo las respuestas:

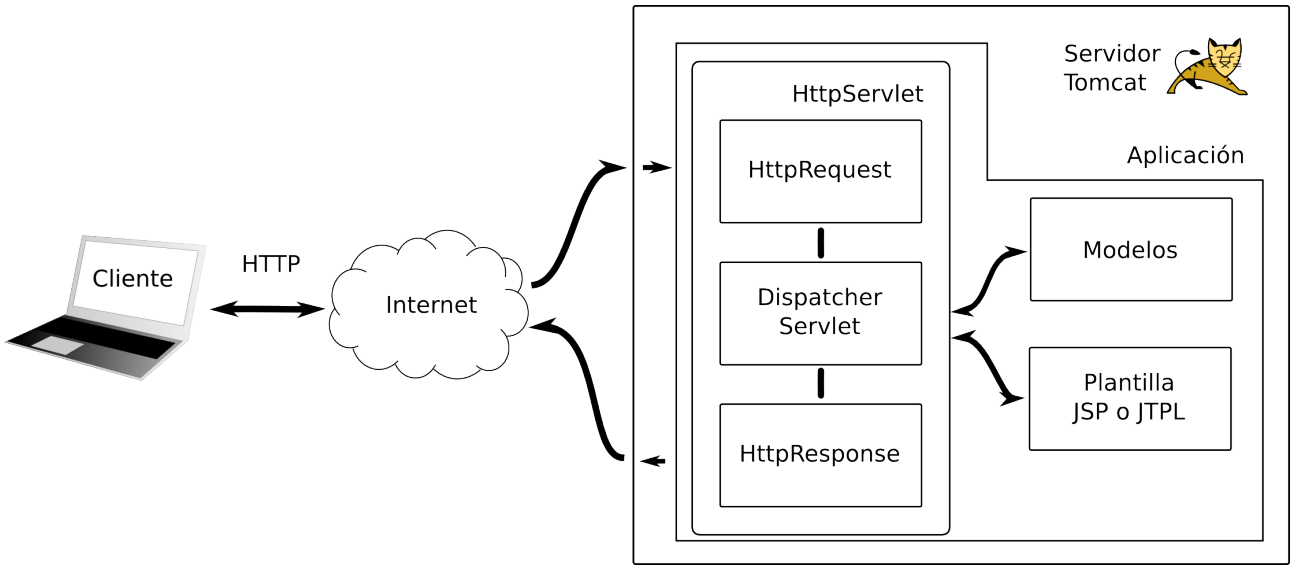

Figura 4.3.- Diagrama simple del servidor Apache Tomcat. Cliente-servidor

El servidor Tomcat tiene varios archivos de configuración principal para el arranque y poder definir el contexto de la localización de los archivos, los puertos para atender las peticiones, los tiempo máximos de ejecución, los formatos válidos, etc. Estos archivos principales son cuatro:

- **`context.xml:`** en este fichero se define el contexto del servidor para monitorizar los recursos y poder localizarlos, en el se configura la localización del archivo *`web.xml`*.

- **`server.xml:`** Este fichero *server.xml* se encuentra en TOMCAT-HOME/conf/server.xml y es el archivo de configuración principal de Tomcat, y el responsable de especificar la configuración inicial en el arranque de Tomcat, así como la definición de la manera, orden y los pasos para construir una aplicación. Los elementos del archivo server.xml pertenecen a cinco categorías básicas: *Elementos de alto nivel, Conectores, Contenedores, Componentes del Cluster y Configuración global.* Todos los elementos dentro de estas categorías tienen muchos atributos que se pueden utilizar para afinar su funcionalidad. Muy a menudo, es necesario hacer cambios importantes en la instalación de Tomcat, como especificar el número de puertos de la aplicación para abrir un socket por ejemplo, el tiempo de arranque o el despliegue de archivos *.war en el servidor.

- **`web.xml:`** El archivo *web.xml* contiene la información utilizada para desplegar y configurar los componentes de las aplicaciones web. Al configurar Tomcat, por primera vez, aquí es donde se puede definir asignaciones de servlets para componentes centrales como JSP. Dentro de Tomcat, este archivo funciona de la misma manera que se describe en la especificación Servlet. La única divergencia en el manejo de Tomcat de este archivo es que el usuario tiene la opción de utilizar TOMCAT-HOME/conf/web.xml para definir valores por defecto para todos los contextos. Si se utiliza este método, Tomcat utilizará TOMCAT-HOME/conf/web.xml como una configuración de base, que puede ser sobrescrita por archivos WEB-INF/web.xml de la aplicación específica.

- **`tomcat-users.xml:`** este archivo se encuentra en TOMCAT-HOME/conf/tomcat-users.xml y en este fichero se define los usuarios con sus nombres, roles y contraseñas.

4.8.- Jetty

Jetty es un servidor JAVA HTTP (web) y Java Servlet contenedor. Jetty se desarrolla como software libre y de código abierto como parte del proyecto de la Fundación Eclipse. El servidor web se utiliza en productos muchos productos y compañías tales como en Google App Engine, Eclipse y Streaming API de Twitter como en otras. Jetty soporta la última API Java Servlet (con JSP apoyo), así como los protocolos HTTP/2 y WebSocket.

El servidor Jetty también puede embeberse dentro de un servidor Tomcat, configurar su puerto y dirección Ip. Este servidor embebido sirve de soporte para los WebSockets abriendo el camino hasta ellos para poder atender las peticiones bidireccionales.

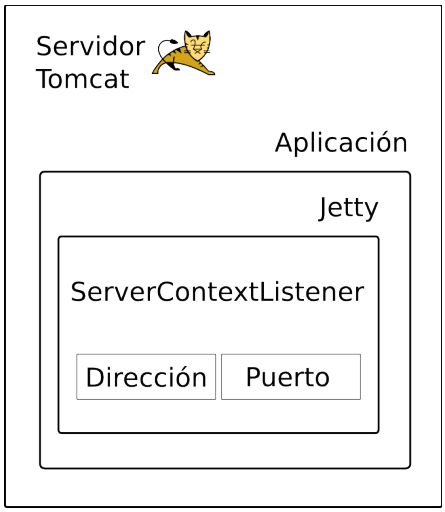

Figura 4.4.- Diagrama de la arquitectura del servidor Tomcat con Jetty

Capítulo 5
Gestión del proyecto

5.- Gestión del proyecto

5.1.- Estimación del proyecto

5.1.1.- Estimación por puntos de función

Los puntos de función es una técnica de medición de las funcionalidades de un software desde una perspectiva de los usuarios. Este proceso de medición se basa en la evaluación de los requerimientos funcionales. Este tipo de técnica de medición está recogida por el Grupo Internacional de Puntos de Función de Usuarios (IFPUG). Los puntos de función no miden directamente el esfuerzo, la productividad y el costo del software es solo una medida de tamaño funcional del software.

Los valores de los dominios de información y su complejidad se resumen en 5 grupos, Entradas Externas, Salidas Externas, Consultas Externas, Archivos Lógicos Internos, y Archivos de Interfaz Externos. Los dominios de información identificados en este proyecto son:

Entradas Externas *(Numero de entradas de usuario)*. Son los procesos en los que se introducen datos y que supone la actualización de cualquier archivo interno.

- Registrar usuario: complejidad baja

- Conexión Sala Principal: complejidad baja

- Eliminar usuario Sala Principal: complejidad baja

- Llamar: complejidad baja

- Notificar llamada: complejidad baja

- Aceptar llamada: complejidad baja

- Conexión Habitación: complejidad baja

- Desconectar usuario Habitación: complejidad baja

- Hacer videollamada: complejidad alta

- Colgar videollamada: complejidad baja

- Hacer Foto: complejidad baja

- Grabar videollamada: complejidad media

- Enviar archivo: complejidad media

- Recibir archivo: complejidad baja

- Cancelar enviar archivo: complejidad baja

- Cancelar recibir archivo: complejidad baja

- Enviar mensaje por Chat: complejidad baja

- Volver a la Sala Principal: complejidad baja

Salidas Externas *(Número de salidas de usuario).* Son los procesos en los que se envía datos al exterior de la aplicación.

- Mostrar datos del usuario: complejidad baja

- Listar usuarios de la Sala Principal: complejidad media

- Recibir mensaje por Chat: complejidad baja

- Mostrar foto previsualizada: complejidad baja

- Descargar archivo de vídeo: complejidad media

- Descargar archivo compartido: complejidad baja

- Notificar desconexión en una videollamada activa: complejidad baja

Consultas Externas *(Número de consultas de usuario).* Son los procesos que se componen de la combinación de una entrada y una salida, en la que la entrada no produce ningún cambio en ningún archivo y la salida no contiene información derivada.

- Rechazar llamada: complejidad baja

- Habilitar compartir audio: complejidad baja

- Habilitar compartir vídeo: complejidad baja

- Mover Menú: complejidad baja

- Activar pantalla completa: complejidad baja

- Desactivar pantalla completa: complejidad baja

- Activar sonido: complejidad baja

- Desactivar sonido: complejidad baja

- Mover Chat: complejidad baja

- Subir opacidad del Chat: complejidad baja

- Bajar opacidad del Chat: complejidad baja

- Mostrar Chat: complejidad baja

- Ocultar Chat: complejidad baja

Archivos de Interfaz Externos y Archivos Lógicos Internos *(Número de ficheros internos)*. Los archivos de interfaz son los ficheros que el sistema utiliza para formar las vistas para mostrar los datos al usuario. Los ficheros lógicos internos son los que el sistema utilizará para realizar el trabajo esperado. Estos ficheros son los grupos de datos relacionados entre sí internos al sistema.

- Interfaz de Inicio de la aplicación: complejidad baja

- Interfaz de la Sala Principal: complejidad media

- Interfaz de la Habitación: complejidad baja

- Interfaz Menú: complejidad baja

- Interfaz del Chat: complejidad media

- Fichero de configuración del contexto JavaEE (web.xml): complejidad baja

- Fichero de logs del sistema: complejidad baja

Número de ficheros Externos. Son los grupos de datos que se mantienen externamente.

- Manual de usuario: complejidad baja

- Manual de instalación y despliegue en OpenShift: complejidad baja

Clasificación de las Entradas Externas. Los "Archivos referenciados" son el número de Archivos Lógicos Internos mantenidos por la Entrada Externa. Los "Elementos de datos" (ED) es la cantidad de elementos que componen la Entrada Externa.

Archivos referenciados	1 a 4 ED	5 a 15 ED	15 o más ED
0 - 1	Baja	Baja	Media
2	Baja	Media	Alta
3 o más	Media	Alta	Alta

Tabla 5.1.- Clasificación de las Entradas Externas

Clasificación de las Salidas Externas y Consultas Externas. Los "Archivos referenciados " son el número de Archivos Lógicos Internos y Archivos de Interfaz Externos vinculados con la Salida Externa y a la Consulta Externa. Los "Elementos de datos" (ED) son la cantidad de elementos de datos de entrada y de salida que componen la Salida Externa o Consulta Externa.

Archivos referenciados	1 a 5 ED	6 a 19 ED	19 o más ED
0 - 1	Baja	Baja	Media
2	Baja	Media	Alta
3 o más	Media	Alta	Alta

Tabla 5.2.- Clasificación de las Salidas Externas y Consultas Externas

Clasificación de los Archivos Lógicos Internos y Archivos de Interfaz Externos. Los "Registros" representan un grupo o subgrupo de elementos reconocibles por el usuario. Los "Elementos de datos" es la cantidad de datos (campos únicos) que componen el archivo.

Archivos Lógicos Internos / Archivos de Interfaz Externos	1 a 19 ED	20 – 50 ED	51 o más ED
1 registro	Baja	Baja	Media
2 – 5 registros	Baja	Media	Alta
6 o más registros	Media	Alta	Alta

Tabla 5.3.- Clasificación de los Archivos Lógicos Internos y Archivos de Interfaz Externos

Asignación de valores numéricos: Los valores numéricos que se asignan a cada complejidad (Baja, Media, Alta), se muestran en la siguiente tabla, para cada tipo de transacción.

Clasificación	Salidas Externas	Consultas Externas	Entradas Externas	Archivo Lógico Interno	Archivo de Interfaz Externo
Baja	4	3	3	7	5
Media	5	4	4	10	7
Alta	7	6	6	15	10

Tabla 5.4.- Asignación de los valores numéricos

En resumen se obtiene:

Entradas/Salidas/Consultas/Archivos	Complejidad	Cantidad	Valor PF total
Entradas Externas	Baja → 3	15	45
Entradas Externas	Media → 4	2	8
Entradas Externas	Alta →6	1	6
Salidas Externas	Baja →4	5	20
Salidas Externas	Media →5	2	10
Salidas Externas	Alta →7	0	0
Consultas Externas	Baja →3	13	36
Consultas Externas	Media →4	0	0
Consultas Externas	Alta →6	0	0
Archivos de Interfaz Externos	Baja →5	3	15
Archivos de Interfaz Externos	Media →7	2	14
Archivos de Interfaz Externos	Alta →10	0	0
Archivos Lógicos Internos	Baja → 7	2	14
Archivos Lógicos Internos	Media →10	0	0
Archivos Lógicos Internos	Alta →15	0	0

Tabla 5.5.- Valoración de los puntos de función del proyecto

La suma total: 45 + 8 + 6 + 20 + 10 + 36 + 15 + 14 + 14 = **168**

El significado del valor asignado a cada característica es el siguiente:

0 → No presente o sin influencia
1 → Influencia incidental
2 → Influencia moderada
3 → Influencia media
4 → Influencia significativa
5 → Influencia fuerte

Característica	Pregunta
Comunicación de datos	¿Cuantas facilidades de comunicación hay disponibles para ayudar en el intercambio de información con la aplicación o el sistema?
Procesamiento distribuido de datos	¿Cómo se manejan los datos y las funciones de procesamiento distribuido?
Rendimiento	¿Existen requerimientos de velocidad o tiempo de respuesta?
Configuraciones fuertemente utilizadas	¿Se usan intensivamente las plataformas hardware donde se ejecuta el sistema?
Frecuencia de transacciones	¿Con qué frecuencia se ejecutan las transacciones?
Entrada de datos on-line	¿Qué porcentaje de la información se ingresa on-line?
Eficiencia del usuario final	¿Está la aplicación diseñada para maximizar la eficiencia del usuario final?
Actualizaciones on-line	¿Cuántos archivos lógicos internos se actualizan por una transacción on-line?
Procesamiento complejo	¿Hay procesamientos lógicos o matemáticos intensivos en la aplicación?
Reusabilidad	¿Cómo de compleja es la instalación y la conversión al nuevo sistema?
Facilidad de instalación	¿Cómo de efectivos y/o automatizados deben ser los procedimientos de arranque, parada, backup y restore?
Facilidad de operación	¿Cómo de efectivos y/o automatizados deben ser los procedimientos de arranque, parada, backup y restablecimiento de funciones?
Instalación en distintos lugares	¿La aplicación fue concebida para su instalación en múltiples sitios y organizaciones?
Facilidad de cambio	¿La aplicación fue concebida para facilitar los cambios sobre la misma?

Tabla 5.6.- Características y preguntas para el cálculo de los puntos de función ajustados

A continuación se resumen las características y se determina como se va a hacer la valoración. (Jose A. Giménez, 2002). En cada tabla se ha seleccionado, cambiando el fondo a gris, la característica y valoración propia de este sistema.

Comunicación de datos. Los datos e informaciones de control utilizados por la aplicación son enviados o recibidos a través de recursos de comunicación de datos. Terminales y estaciones de trabajo son algunos ejemplos. Todos los dispositivos de comunicación utilizan algún tipo de protocolo de comunicación.

La aplicación es puramente batch o funciona en una computadora aislada.	0
La aplicación es batch pero utiliza entrada de datos remota o impresión remota.	1
La aplicación es batch pero utiliza entrada de datos remota y utilización de periféricos de salida remotos.	2
La aplicación incluye entrada de datos on-line vía entrada de vídeo o un procesador front-end para alimentar procesos batch o sistemas de consultas.	3
La aplicación es más que una entrada on-line, y soporta apenas un protocolo de comunicación.	4
La aplicación es más que una entrada on-line, y soporta más de un protocolo de comunicación.	5

Tabla 5.7.- Valoración: Comunicación de datos

Procesamiento distribuido de datos. Datos o procesamiento distribuidos entre varias unidades de procesamiento (CPUs) son características generales que pueden influenciar en la complejidad de la aplicación.

La aplicación no contribuye en la transferencia de datos o funciones entre diferentes unidades de procesamiento.	0
La aplicación prepara datos para el usuario final en otra CPU.	1
La aplicación prepara datos para transferencia, los transfiere y entonces son procesados en otro equipamiento de la empresa (no por el usuario final).	2
Procesamiento distribuido y la transferencia de datos son on-line en apenas una dirección.	3
Procesamiento distribuido y la transferencia de datos son on-line en ambas direcciones.	4
Las funciones de procesamiento son dinámicamente ejecutadas en el equipamiento más adecuado.	5

Tabla 5.8.- Valoración: Procesamiento distribuido de datos

Rendimiento. Los objetivos de rendimiento del sistema, establecidos y aprobados por el usuario en términos de respuesta, influye o podría influenciar el proyecto, desarrollo, implementación o soporte de la aplicación.

Ningún requerimiento especial de rendimiento fue solicitado por el usuario.	0
Requerimientos de rendimiento y de diseño fueron establecidos y previstos, sin embargo, ninguna acción especial fue requerida.	1
El tiempo de respuesta y el volumen de datos son críticos durante horarios pico de procesamiento. Ninguna determinación especial para el uso del procesador fue establecida. El intervalo de tiempo límite para la disponibilidad de procesamiento es siempre el próximo día hábil.	2
El tiempo de respuesta y volumen de procesamiento son elementos críticos durante todo el tiempo de funcionamiento del programa. Ninguna determinación especial para la utilización del procesador fue establecida. El tiempo límite necesario para la comunicación con otros sistemas es un aspecto importante.	3
Los requerimientos de rendimiento establecidos requieren tareas de análisis de rendimiento en la fase de análisis y diseño de la aplicación.	4
Además de lo descrito en el ítem anterior, herramientas de análisis de rendimiento fueron usadas en las fases de diseño, desarrollo y/o implementación para atender los requerimientos de rendimiento establecidos por el usuario.	5

Tabla 5.9.- Valoración: Rendimiento

Configuraciones fuertemente utilizadas. También denominada ”Configuración del equipamiento”, esta característica representa la necesidad de realizar consideraciones especiales en el diseño de los sistemas para que la configuración del equipamiento no sea sobrecargada.

Ninguna restricción operacional explícita o implícita fue incluida.	0
Existen restricciones operacionales leves. No es necesario un esfuerzo especial para resolver estas restricciones.	1
Algunas consideraciones de ajuste de rendimiento y seguridad son necesarias.	2
Son necesarias especificaciones especiales de procesador para un módulo específico de la aplicación.	3
Restricciones operacionales requieren cuidados especiales en el procesador central o procesador dedicado.	4
Además de las características del ítem anterior, hay consideraciones especiales en la distribución del sistema y sus componentes.	5

Tabla 5.10.- Valoración: Configuraciones fuertemente utilizadas

Frecuencia de transacciones. El nivel de transacciones es alto y tiene influencia en el diseño, desarrollo, implementación y mantenimiento de la aplicación.

No están previstos periodos picos de volumen de transacción.	0
Están previstos picos de transacciones mensualmente, trimestralmente, anualmente o en un cierto periodo del año.	1
Se prevén picos semanales.	2
Se prevén picos diariamente.	3
Alto nivel de transacciones fue establecido por el usuario, el tiempo de respuesta necesario exige un nivel alto o suficiente para requerir análisis de rendimiento y diseño.	4
Además de lo descrito en el ítem anterior, es necesario utilizar herramientas de análisis de rendimiento en las fases de diseño, desarrollo y/o implementación.	5

Tabla 5.11.- Valoración: Frecuencia de transacciones

Entrada de datos on-line. Esta característica cuantifica la entrada de datos on-line proveída por la aplicación.

Todas las transacciones son procesadas en modo batch.	0
De 1 % al 7 % de las transacciones son entradas de datos on-line.	1
De 8 % al 15 % de las transacciones son entradas de datos on-line.	2
De 16 % al 23 % de las transacciones son entradas de datos on-line.	3
De 24 % al 30 % de las transacciones son entradas de datos on-line.	4
Más del 30 % de las transacciones son entradas de datos on-line.	5

Tabla 5.12.- Valoración: Entrada de datos on-line

Eficiencia del usuario final. Las funciones on-line del sistema hacen énfasis en la amigabilidad del sistema y su facilidad de uso, buscando aumentar la eficiencia del usuario final. El sistema posee:

1. Ayuda para la navegación (teclas de función,accesos directos y menús dinámicos).	9. Selección de datos vía movimiento del cursor en la pantalla.
2. Menús.	10. Utilización intensa de campos en vídeo reverso, intensificados, subrayados, coloridos y otros indicadores.
3. Documentación y ayuda on-line.	11. Impresión de la documentación de las transacciones on-line.
4. Movimiento automático del cursor.	12. Utilización del mouse.
5. Scrolling vertical y horizontal.	13. Menús pop-up.
6. Impresión remota (a través de transacciones on-line).	14. El menor número de pantallas posibles para ejecutar las funciones del negocio.
7. Teclas de función preestablecidas.	15. Soporte bilingüe (el soporte de dos idiomas, contar como cuatro ítems).
8. Ejecución de procesos batch a partir de transacciones on-line.	16. Soporte multilingüe (el soporte de más de dos idiomas, contar como seis ítems).

Tabla 5.13.- Parámetros para la valoración de la eficiencia del usuario final

Ningún de los ítems descritos.	0
De uno a tres de los ítems descritos.	1
De cuatro a cinco de los ítems descritos.	2
Más de cinco de los ítems descritos, no hay requerimientos específicos del usuario en cuanto a amigabilidad del sistema.	3
Más de cinco de los ítems descritos, y fueron descritos requerimientos en cuanto a amigabilidad del sistema suficientes para generar actividades específicas.	4
Más de cinco de los ítems descritos y fueron establecidos requerimientos en cuanto a la amigabilidad suficientes para utilizar herramientas especiales y procesos especiales para demostrar anticipadamente que los objetivos fueron alcanzados.	5

Tabla 5.14.- Valoración: Eficiencia del usuario final

Actualización on-line. La aplicación posibilita la actualización on-line de los archivos lógicos internos.

Ninguna.	0
Actualización on-line de uno a tres archivos lógicos internos.	1
Actualización on-line de más de tres archivos lógicos internos.	2
Actualización on-line de la mayoría de los archivos lógicos internos.	3
Además del ítem anterior, la protección contra pérdidas de datos es esencial y fue específicamente proyectado y codificado en el sistema.	4
Además del ítem anterior, altos volúmenes influyen en las consideraciones de costo en el proceso de recuperación. Procesos para automatizar la recuperación fueron incluidos minimizando la intervención del operador.	5

Tabla 5.15.- Valoración: Actualización on-line

Procesamiento complejo. El procesamiento complejo es una de las características de la aplicación, los siguientes componentes están presentes:

1. Procesamiento especial de auditoría y/o procesamiento especial de seguridad.	4. Gran cantidad de procesamiento de excepciones, resultando en transacciones incompletas que debe ser procesadas nuevamente. Por ejemplo, transacciones de datos incompletas interrumpidas por problemas de comunicación o con datos incompletos.
2. Procesamiento lógico extensivo.	
3. Procesamiento matemático extensivo.	5. Procesamiento complejo para manipular múltiples posibilidades de entrada/salida.

Tabla 5.16.- Parámetros para la valoración del procesamiento complejo

Ninguno de los ítems descritos.	0
Apenas uno de los ítems descritos.	1
Dos de los ítems descritos.	2
Tres de los ítems descritos.	3
Cuatro de los ítems descritos.	4
Todos los ítems descritos.	5

Tabla 5.17.- Valoración: Procesamiento complejo

Reusabilidad. La aplicación y su código serán o fueron proyectados, desarrollados y mantenidos para ser utilizados en otras aplicaciones.

No presenta código reutilizable.	0
Código reutilizado fue usado solamente dentro de la aplicación.	1
Menos del 10 % de la aplicación fue proyectada previendo la utilización posterior del código por otra aplicación.	2
10 % o más de la aplicación fue proyectada previendo la utilización posterior del código por otra aplicación.	3
La aplicación fue específicamente proyectada y/o documentada para tener su código fácilmente reutilizable por otra aplicación y la aplicación es configurada por el usuario a nivel de código fuente.	4
La aplicación fue específicamente proyectada y/o documentada para tener su código fácilmente reutilizable por otra aplicación y la aplicación es configurada para uso a través de parámetros que pueden ser alterados por el usuario.	5

Tabla 5.18.- Valoración: Reusabilidad

Facilidad de instalación. La facilidad de instalación, implementación y conversión de datos son características de la aplicación. Un plan de conversión e implementación y/o herramientas de conversión fueron proveídas y probadas durante la fase de prueba de la aplicación.

Ninguna consideración especial fue establecida por el usuario y ningún procedimiento especial fue necesario en la implementación.	0
Ninguna consideración especial fue establecida por el usuario, más procedimientos especiales son requeridos en la implementación.	1
Requerimientos de conversión e implementación fueron establecidos por el usuario y rutinas de conversión e implementación fueron proporcionados y probados. El impacto de conversión en el proyecto no es considerado importante.	2
Requerimientos de conversión e implementación fueron establecidos por el usuario y rutinas de conversión e implementación fueron proporcionados y probados. El impacto de conversión en el proyecto es considerado importante.	3
Además del ítem 2, conversión automática y herramientas de implementación fueron proporcionadas y probadas.	4
Además del ítem 3, conversión automática y herramientas de implementación fueron proveídas y probadas.	5

Tabla 5.19.- Valoración: Facilidad de instalación

Facilidad de operación. La facilidad de operación es una característica del sistema. Procedimientos de inicialización, respaldo y recuperación fueron proveídos y probados durante la fase de prueba del sistema. La aplicación minimiza la necesidad de actividades manuales, tales como montaje de cintas magnéticas, manoseo de papel e intervención del operador.

0 → Ninguna consideración especial de operación, además del proceso normal de respaldo establecido por el usuario. 5 → La aplicación fue diseñada para trabajar sin operador, ninguna intervención del operador es necesaria para operar el sistema, excepto ejecutar y cerrar la aplicación. La aplicación posee rutinas automáticas de recuperación en caso de error	1 – 4 → Verificar cuáles de las siguientes afirmaciones pueden ser identificadas en la aplicación. Cada ítem vale un punto, excepto se defina lo contrario: (a) fueron desarrolladas procedimientos de inicialización y respaldo, siendo necesaria la intervención del operador, (b) se establecieron procesos de inicialización, respaldo y recuperación sin ninguna intervención del operador (contar como 2 ítems), (c) la aplicación minimiza la necesidad de montaje de cintas magnéticas y (d) la aplicación minimiza la necesidad de manoseo de papel.

Tabla 5.20.- Valoración: Facilidad de operación

Instalación en distintos lugares. La aplicación fue específicamente proyectada, diseñada y mantenida para ser instalada en múltiples locales de una organización o para múltiples organizaciones.

Los requerimientos del usuario no consideran la necesidad de instalación de más de un lugar.	0
La necesidad de múltiples lugares fue considerada en el proyecto y la aplicación fue diseñada para operar apenas sobre el mismo ambiente de hardware y software.	1
La necesidad de múltiples lugares fue considerada en el proyecto y la aplicación fue diseñada para operar en ambientes similares de software y hardware.	2
La necesidad de múltiples lugares fue considerada en el proyecto y la aplicación está preparada para trabajar sobre diferentes ambientes de hardware y/o software.	3
Plan de mantenimiento y documentación fueron proporcionados y probados para soportar la aplicación en múltiples lugares, además los elementos 1 y 2 caracterizan a la aplicación.	4
Plan de documentación y mantenimiento fueron proveídos y probados para soportar la aplicación en múltiples lugares, además el elemento 3 caracteriza a la aplicación.	5

Tabla 5.21.- Valoración: Instalación en distintos lugares

Facilidad de cambios. La aplicación fue específicamente proyectada y diseñada con vistas a facilitar su mantenimiento, modificación, corrección de problemas y errores o cambios de entorno. Las siguientes características pueden ser atribuidas a la aplicación:

1. Están disponibles facilidades como consultas e informes flexibles para atender necesidades simples (contar 1 elemento).	4. Datos de control son almacenados en tablas que son mantenidas por el usuario a través de procesos on-line pero los cambios se hacen efectivos solamente al día siguiente.
2. Están disponibles facilidades como consultas e informes flexibles para atender necesidades de complejidad media (contar 2 elementos).	5. Datos de control son almacenados en tabla que son mantenidas por el usuario a través de procesos on-line pero los cambios se hacen efectivos inmediatamente (contar 2 elementos).
3. Están disponibles facilidades como consultas e informes flexibles para atender necesidades complejas (contar 3 elementos).	

Tabla 5.22.- Parámetros para la valoración de la facilidad de cambios

Ninguno de los ítems descritos.	0
Apenas uno de los ítems descritos.	1
Dos de los ítems descritos.	2
Tres de los ítems descritos.	3
Cuatro de los ítems descritos.	4
Todos los ítems descritos.	5

Tabla 5.23.- Valoración: Facilidad de cambios

A continuación se muestra los pesos de los factores que la aplicación ha obtenido por cada característica explicada anteriormente. Los factores de peso se recogen en las siguiente tabla por característica:

Características	Puntuación
Comunicación de datos	5
Procesamiento distribuido de datos	4
Rendimiento	1
Configuraciones fuertemente utilizadas	0
Frecuencia de transacciones	0
Entrada de datos on-line	5
Eficiencia del usuario final	3
Actualización on-line	3
Procesamiento complejo	2
Reusabilidad	4
Facilidad de instalación	0
Facilidad de operación	0
Instalación en distintos lugares	4
Facilidad de cambios	3
Total	34

Tabla 5.24.- Cálculo de los puntos de función ajustados

El total de la suma de los factores de peso obtenemos **34**.

El factor de ajuste se calcula mediante la fórmula:

Factor de ajuste = (Nivel de Influencia * 0.01) + 0.65

Obtenemos, **Factor de Ajuste = (34 * 0.01) + 65 = 0.99**

Calculado el factor de ajuste, se ajusta los puntos de función mediante la siguiente formula:

Puntos de Función(ajustados) = Puntos de función * Factor de Ajuste

PF(aj) = 168 * 0,99 = > PF(aj) = 166,32

El tamaño de la aplicación es de **166,32 PF**.

Según el Tribunal de Tasaciones de la Nación en la tasación del software recoge en el punto 2.5 Análisis y determinación de puntos de función la relación LOC/PF (lenguaje de programación / punto de función) el valor del lenguaje Java y JavaScript = 53. El código de esta aplicación está en un repositorio alojado actualmente e indica que la cantidad de lenguaje JavaScript es del 68.5%, Java 21.4% y CSS 10.1% (Datos tomados de https://github.com/jvrodrigo/webrtc)

1 punto de función = 54 líneas de código Java/JavaScript

Ahora calculamos el número de líneas del código estimadas tomando la equivalencia de LDC

LCD estimadas Java/JavaScript = PF * LDC(Java/Js) = 166,32 * 54 = 8981,28 líneas

Se estima que el proyecto tiene 9662 líneas de código en Java/JavaScript y esto equivale, aproximadamente a **8,981 KLDC.**

5.1.2.- Estimación mediante COCOMO

El Modelo Constructivo de Costes (Constructive Cost Model) fue desarrollado por B. W. Boehm a finales de los 70 y comienzos de los 80, plasmándolo detalladamente en su libro "Software Engineering Economics" (Prentice-Hall, 1981).

COCOMO es una jerarquía de modelos de estimación de costes software que incluye submodelos básico, intermedio y detallado. Este modelo trata de estimar, de una manera rápida y más o menos burda, la mayoría de proyectos pequeños y medianos.

Mediante la técnica COCOMO se va determinar los costes económicos y temporales de este proyecto pero primero tiene que ser definido, esta técnica de valoración tiene tres modelos los cuales son:

- Modelo orgánico: el tamaño de software es pequeño desde unas miles lineas de código a una decena de miles (software de tamaño medio). Un pequeño equipo de desarrolladores con poca experiencia diseñan y desarrollan el software en un entorno familiar.

- Modelo semilibre: el tamaño del software es intermedio entre el orgánico y el rígido. El equipo de desarrolladores está compuesto por desarrolladores con mucha y poca experiencia.

- Modelo rígido: el proyecto tiene fuertes restricciones relacionadas con aspectos funcionales y/o técnicos. El problema a resolver puede se único y es difícil basarse en la experiencia, puesto que puede no haberla.

Modelo	**Básico**		**Intermedio**	
	A_i	B_i	A_i	B_i
Orgánico	2.4	1.05	2.5	0.38
Semilibre	3	1.12	2.5	0.35
Rígido	3.6	1.2	2.5	0.32

Tabla 5.25.- Modelos de COCOMO

El sistema a desarrollar de este proyecto se clasifica en un sistema semilibre, puesto que se necesita un equipo de desarrollo con programadores con experiencia en lenguajes de programación como son Java y JavaScript en entornos web JavaEE y con conocimientos básicos de la tecnología WebRTC. Lo complicado de este software es saber programar sockets de conexión con JavaEE, JavaScript y sincronizar los datos necesarios para realizar la videollamada con WebRTC. El tamaño del proyecto no es muy grande, se sitúa entre el modelo orgánico y rígido puesto que la API de WebRTC ofrece las herramientas necesarias para realizar videollamadas con el menor código posible.

	Valor de los factores					
Factores	Muy bajo	Bajo	Medio	Alto	Muy alto	Extra
Fiabilidad requerida	0.75	0.88	**1.00**	1.15	1.4	
Tamaño de la base de datos		**0.94**	1.00	1.08	1.16	
Complejidad del software	0.70	**0.85**	1.00	1.15	1.30	1.65
Restricciones de tiempo de ejecución			**1.00**	1.11	1.30	1.66
Restricciones de memoria			**1.00**	1.06	1.21	1.56
Volatilidad del hardware		0.87	**1.00**	1.15	1.30	
Restricciones de tiempo de respuesta		0.87	1.00	**1.07**		
Capacidad de análisis	1.46	1.19	1.00	**0.86**	0.71	
Experiencia con el tipo de aplicación	1.29	**1.13**	1.00	0.91	0.82	
Exp. con el hardware	1.21	**1.10**	1.00	0.90		
Exp. con el lenguaje de programación	1.14	1.07	1.00	**0.95**		
Calidad de los programadores	1.42	1.17	1.00	**0.86**	0.70	
Técnicas modernas de programación	1.24	1.10	1.00	0.91	**0.82**	
Empleo de herramientas de software	1.24	1.10	1.00	0.91	**0.83**	
Restricciones a la duración del proyecto	1.23	1.08	1.00	**1.04**	1.10	

Tabla 5.26.- Valor de los factores COCOMO

Los valores de la tabla han sido seleccionados dadas las siguientes directrices:

- La *fiabilidad requerida* de esta aplicación es media porque el usuario tiene que poder utilizar la aplicación en distintos dispositivos y sistemas. Esta aplicación puede ser utilizada tanto en navegadores móviles como de escritorio compatibles con WebRTC.

- El tamaño de la base de dato se ha seleccionado el valor más bajo puesto que esta aplicación no utiliza base de datos en esta fase, aunque se puede añadir una base de datos

- La *complejidad del software* es baja porque solo se desarrollará una Sala Principal y la Habitación como interfaz de usuario. Lo que hace complejo a este software es la utilización de sockets, la configuración del servidor y conocer las herramientas que el API de WebRTC nos ofrece pero facilita mucho el desarrollo.

-Las *restricciones de tiempo de ejecución* es media al ser esta aplicación un sistema de videollamadas los usuarios demandan un sistema donde las conexiones sean rápidas.

- Las *restricciones de memoria y volatilidad del hardware* son medias puesto que la tecnología WebRTC requiere navegadores web compatibles. No se registran restricciones de memoria.

- Las *restricciones de tiempo de respuesta* son altas al situarse esta aplicación en un entorno web donde la mayoría de las transacciones se realizan on-line.

- La *capacidad de análisis* es alta debido a la sincronicidad que esta aplicación necesita para que los usuarios se conecten, envíen, reciban datos y realicen videollamadas.

- La *experiencia con este tipo de aplicación* es intuitiva porque todo el mundo sabe que es una videollamada y el resultado que se espera de ella. El usuario solo tiene que saber utilizar un navegador web compatible con WebRTC (firefox, chrome, opera,...).

- La *experiencia con el hardware* es básico la aplicación está instalada en un contexto web donde el usuario solo tiene que utilizar un ordenador para poder utilizarla.

- La *experiencia con el lenguaje de programación requiere* un experiencia media para programadores web con JavaEE y JavaScript para poder implementar los objetos y los sockets de conexión. La tecnología WebRTC no requiere tampoco gran experiencia pero si un conocimiento básico de qué puede hacer esta tecnología y cómo hacerlo.

- La *calidad de los programadores* se necesitan programadores de calidad y con experiencia en desarrollo web, capacidad de análisis y diseño de componentes y objetos. Como este sistema puede ser integrado ya sea el BackEnd o el FrontEnd en distintas plataformas y sistemas, la adaptación de este tipo de software requiere programadores de alta calidad para implementar e implantar la aplicación en situaciones de cambio.

- Las *técnicas modernas de programación* son muy altas puesto la aplicación está dentro de un entorno web con JavaEE y JavaScript en continua evolución y sobre todo porque la tecnología WebRTC es nueva y está en pleno desarrollo; el alcance que WebRTC tiene sobre otras tecnologías en desarrollo y cambio hace que esta aplicación esté utilizando técnicas modernas de programación.

- El *empleo de herramientas de software* es muy alto porque para realizar las videollamadas utiliza la API WebRTC que da todos los servicios necesarios para configurar la videollamada entre pares. También para realizar la grabación de la llamada se utiliza el fichero "record.js" con las funciones establecidas para este propósito.

- Las *restricciones a la duración del proyecto* son altas puesto que este proyecto debe ser entregado en esta convocatoria de TFG.

El cálculo el peso del factor de ajuste se realiza de la siguiente forma:

m(x) = $\prod m(x_i)$ = 1,00 * 0,94 * 0,85 * 1,00 * 1,00 * 1,00 * 1,07 * 0,86 * 1,13 * 1,10 * 1,00 * 0,86 * 0,82 * 0,83 * 1,04 = **0,556**

Con el peso del factor de coste calculado se calcula el esfuerzo ha realizar personas/mes.

- Esfuerzo nominal → $(MM) = a_i * (KI^{b_i}) = 3 * (8{,}98^{1{,}12}) = 35{,}058$

- Esfuerzo → $(E) = MM * \prod m(x_i) = 35{,}05 * 0{,}55 = 19{,}277$ *personas*

- Tiempo de desarrollo del proyecto → $(TDEV) = c * (E^d) = 2{,}5 * (19{,}27^{0{,}35}) = 7{,}04$ *meses*

Aproximadamente **211 días** para realizar el proyecto completo.

- Personas necesarias para realizar el proyecto:

$$(CosteH) = E / TDEV = 19{,}27 / 7{,}04 = 2{,}7 \quad \text{personas.}$$

5.1.3.- Presupuesto

El desarrollo del proyecto necesita equipos hardware y herramientas software con su coste asociado, este coste se tiene que incluir en el presupuesto al igual que el coste de los recursos humanos utilizados. Se ha tomado de referencia **1 año laboral = 217 días/laborales**, entonces se obtiene 1 año laboral = 1736 horas laborales/año

- Presupuesto Hardware

El hardware que se utiliza se calcula el gasto en función de las horas de uso y una estimación de la vida útil del dispositivo, también se incluye la conexión a Internet puesto que es un requisito no funcional y requiere un gasto.

Hardware	**Uso(horas)**	**Coste del hardware(€)**	**Coste Total(€)**
Ordenador personal (5 años)	1689 hrs	599€	116,55€
2 Cámaras web (5 años)	1689 hrs	27€	5,25€
Conexión a Internet (1 año)	1689 hrs	504€	490,35€
		Total	**1097,25€**

Tabla 5.27.- Desglose del presupuesto hardware

- Presupuesto Software

En el proyecto utilizar varias herramientas software con sus costes asociados. Este proyecto está desarrollado con herramientas software GNU libre y abierto, lo que no supondrá ningún gasto asociado al software que se utiliza. De todas formas se desglosa el presupuesto software en la siguiente tabla:

Software	**Uso(horas)**	**Coste del software(€)**	**Coste Total(€)**
SO Linux/Ubuntu 14.04 (4 años)	1700 hrs	0€	0€
IDE Eclipse (1 año)	1700 hrs	0€	0€
Apache Server Tomcat 7 (1 año)	1700 hrs	0€	0€
Jetty Server (1 año)	1700 hrs	0€	0€
WebRTC (1 año)	1700 hrs	0€	0€
		Total	**0€**

Tabla 5.28.- Desglose del presupuesto software

- Presupuesto de mano de obra

Habiendo calculado el esfuerzo en el apartado anterior con la técnica COCOMO en su vertiente intermedia, se puede realizar una estimación para obtener un presupuesto de mano de obra

o coste de personal.

El proyecto para ser terminado es necesario **7** meses de trabajo con el esfuerzo de 3 personas, las cuales van a desarrollar roles diferentes como son: Jefe de proyecto, Analista - Diseñador, Programador - QA.

Tarea (horas)	Función profesional (Rol)	Salario en horas	Horas totales	Coste total
Estudio de la tecnología y técnica	Jefe del proyecto	19.8€	21*8 = 168 hrs	3326,4
Documentación del estudio de la técnica y tecnología	Jefe del proyecto	19.8€	7*8 = 56 hrs	1108,8 €
Requisitos del sistema	Analista	17€	14*8 = 112 hrs	1904 €
Documentación de los requisitos del sistema	Analista	17€	5*8 = 40 hrs	680 €
Análisis de componentes	Analista	17€	7*8 = 56 hrs	952 €
Documentación del análisis de componentes	Analista	17€	7*8 = 56 hrs	952 €
Diseño de componentes	Analista	17€	7*8 = 56 hrs	952 €
Documentación del diseño	Analista	17€	5*8 = 40hrs	680 €
Implementación de componentes	Programador	11.3€	42*8 = 336 hrs	3796,8 €
Documentación de la implementación	Programador	11.3€	15*8 = 120 hrs	1356 €
Pruebas de componentes	Programador	11.3€	21*8 = 168 hrs	1898,4 €
Pruebas de aplicación	Tester QA	11.3€	10*8 = 80 hrs	904 €
Documentación de pruebas	Tester QA	11.3€	5*8 = 40 hrs	452 €
			Total:	**18962,4 €**

Tabla 5.29.- Desglose del presupuesto personal

Presupuesto Total

Presupuesto Hardware		**1097,25 €**
Presupuesto Software		**0 €**
Presupuesto Personal		**18962,4 €**
	Total	**20059,65 €**

Tabla 5.30.- Desglose del presupuesto total

5.1.4.- Planificación temporal

El proyecto se ha planificado según el **modelo en cascada**, denominado así por el enfoque metodológico que ordena las fases de desarrollo en etapas de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior.

La planificación del proyecto se divide en 3 Fases:

- 1º FASE.- Videollamada WebRTC y componentes → En esta fase se desarrollan los siguientes requisitos: realizar videollamadas entre pares con WebRTC, tomar fotos, grabar vídeo, enviar archivos, enviar/recibir mensajes por Chat.

- 2º FASE.- Página de Inicio y Sala Principal → En esta fase se desarrolla la Página de Inicio donde se registra el usuario y Sala Principal donde se lista los usuarios conectados, se hacen las llamadas, se notifican las llamadas, se aceptan o se rechazan las llamadas.

- 3º FASE.- Integración de la Fase 1 y Fase 2 → En esta fase se integran las dos fases anteriores para que los usuarios puedan realizar videollamadas de forma segura entre pares.

En el siguiente diagrama de Gantt se muestra la planificación temporal de proyecto divido en 3 fases:

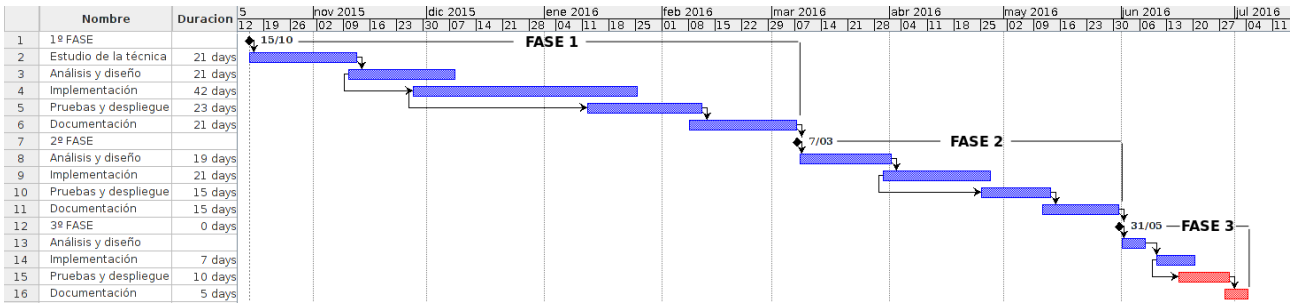

Figura 5.1.- Diagrama planificación temporal

Capítulo 6
Análisis del sistema

6.- Análisis del sistema

6.1.- Introducción

La aplicación objeto de este proyecto ha sido crear un sistema con software libre atendiendo a los estándares recogidos por W3C y ITEF para realizar llamadas VoIP, videoconferencias, chat, transmisión de archivos y grabación de sesiones en tiempo real.

El sistema de videollamadas se ha desarrollado con la tecnología WebRTC ya que ofrece las herramientas necesarias y un API en lenguaje JavaScript fácil de utilizar y programar para este propósito. Las tecnologías utilizadas en la Sala Principal donde los usuarios se conectan y realizan llamadas entre ellos, se ha optado por utilizar en el servidor un servlet y un websocket Java, y en el lado del cliente JavaScript, HTML5 y CSS para renderizar la interfaz y enviar/recibir datos. Para el sistema de Chat y la transmisión de archivos se ha optado por utilizar el websocket para enviar/recibir los datos desde el servidor y desde el lado del cliente.

Los lenguajes de programación y tecnologías que utiliza esta aplicación son:

- En el lado del **cliente** (FrontEnd): HTML5, JavaScript(AJAX, WebSocket), CSS, WebRTC.

- En el lado del **servidor**(BackEnd): Java, WebSocket, Tomcat, Jetty.

Al margen de todas estas tecnologías la aplicación tiene que reunir las siguientes características:

- Realizar videollamadas VoIP entre pares de clientes utilizando navegadores web, sin necesidad de instalar plugins ni programas externos o embebidos, para que la usabilidad sea sencilla, eficaz, e intuitiva para el usuario.

- La interfaz tiene que ser intuitiva y simple, similar a otras aplicaciones como HangOut o Skype.

- Dotar a la aplicación de seguridad bloqueando el posible acceso de usuarios no deseados a la videollamada entre pares.

- Tiene que tener varios sistemas de comunicación como un Chat y compartir archivos.

6.2.- Actores del sistema

Los actores del sistema son los roles que el usuario puede tener, también puede ser un actor del sistema algún componente software o hardware pero en este tipo de sistema no se ha detectado ningún usuario de este tipo. En esta aplicación se ha detectado el siguiente actor:

ACT-01	**Usuario**
Versión	1.0 (3/12/2015)
Descripción	Este actor representa al usuario que va a utilizar la aplicación para comunicarse con otros usuarios
Comentarios	La comunicación entre usuarios es bidireccional

Tabla 6.1.- ACT – 01: Usuario

6.3.- Casos de uso

6.3.1.- Introducción

Los casos de uso son descripciones o actividades que tienen que realizarse para llevar a cabo un proceso o una funcionalidad. Los personajes o entidades que participan en un caso de uso se denominan actores. Los casos de uso se centran en describir o alcanzar un objetivo y se representa gráficamente mediante diagramas de casos de uso.

Los diagramas de casos, dentro del contexto de la ingeniería del software, representan el comportamiento que el sistema debe de realizar evitando utilizar lenguaje técnico y prefiriendo utilizar un lenguaje adaptado al usuario final o del experto del campo a aplicar.

Los casos de uso se relacionan entre sí a través de varios tipos de relaciones:

Comunica (**<<communicates>>**) → Describe la participación del actor con un caso de uso. Se representa con una línea continua.

Usa (**<<uses>>**) → Describe el uso de un caso por otro caso de uso para realizar una tarea.

Incluye (**<<include>>**) → Describe la relación de dependencia entre dos casos de uso que denota la inclusión del comportamiento de un escenario en otro.

Extiende (**<<extends>>**) → Describe la relación de dependencia entre dos casos de uso, denota que un caso de uso es un especialización de otro.

Herencia → Un actor puede heredar los casos de uso que comunican con otro actor mediante el uso de la notación generalización.

En la siguiente figura se muestra una representación gráfica de las notaciones de casos de uso:

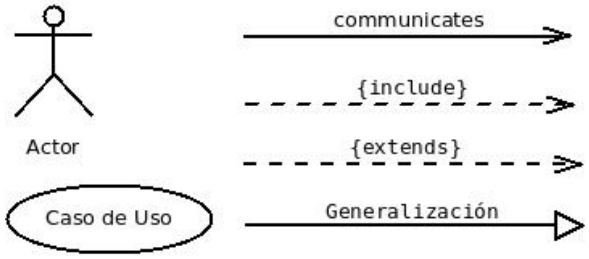

Figura 6.1.- Notación de casos de uso

6.3.2.- Diagrama casos de uso

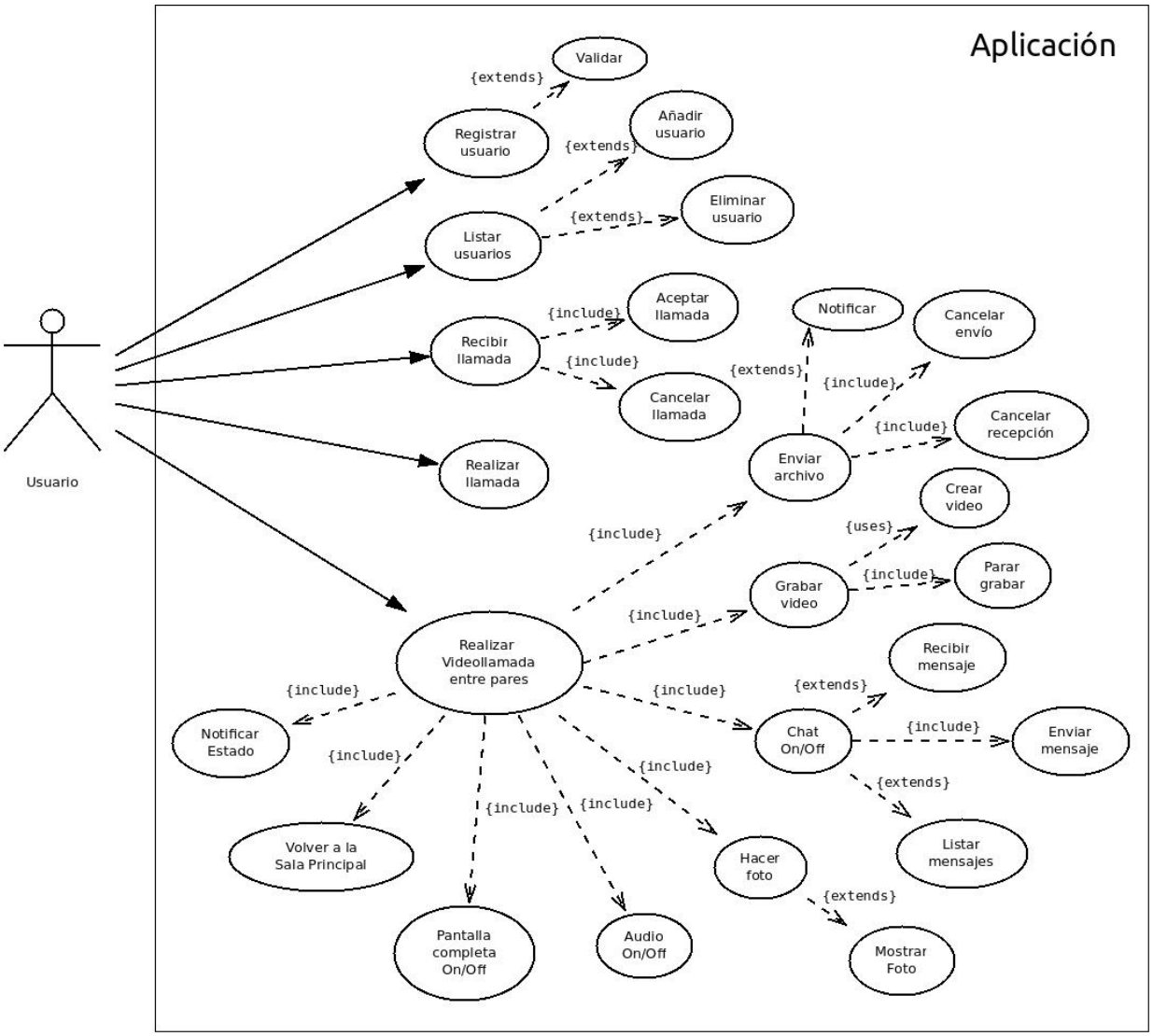

Figura 6.2.- Diagrama **c**asos de uso

6.3.3.- Especificación

Listado de los casos de uso de la aplicación:

CU – 01.- Registrar usuario

CU – 02.- Listar los usuarios (Sala Principal)

CU – 03.- Añadir usuario (Sala Principal)

CU – 04.- Eliminar usuario (Sala Principal)

CU – 05.- Realizar llamada (Sala Principal)

CU – 06.- Recibir llamadas (Sala Principal)

CU – 07.- Aceptar llamada (Sala Principal)

CU – 08.- Rechazar llamada (Sala Principal)

CU – 09.- Realizar videollamada entre pares (Habitación WebRTC)

CU – 10.- Enviar archivos

CU – 11.- Cancelar transferir archivos entre pares

CU – 12.- Grabar videollamada

CU – 13.- Parar grabar vídeo

CU – 14.- Chat On/Off

CU – 15.- Enviar mensaje (Chat)

CU – 16.- Listar mensajes (Chat)

CU – 17.- Hacer foto

CU – 18.- Audio On/Off

CU – 19.- Pantalla completa On/Off

CU – 20.- Volver a la Sala Principal

Ver Anexo A. Casos de Uso

6.4.- Requisitos

6.4.1.- Requisitos de negocio

Para realizar la aplicación de este proyecto he localizado 17 requisitos de negocio para satisfacer una funcionalidad atractiva para el usuario. Para ello he creado unos servicios para que la usabilidad del usuario sea máxima y abarque desde una primera fase los casos de uso más requeridos por este tipo de sistema.

La idea base de este proyecto es crear un sistema de videollamadas, este es un requisito de negocio principal y el objetivo final de esta aplicación, para realizar las videollamadas necesitamos un sistema que gestione y guíe al usuario realizar este objetivo de una manera simple y sencilla, por este motivo la sencillez del sistema es muy importante para poder ser utilizado por todas las personas que quieran acceder a él. La sencillez es otro requisito de negocio importante

En el momento que se crea una videollamada, el usuario puede realizar varias acciones como compartir archivos, enviar-recibir mensajes por chat, sonido on/off,... por el motivo de que quedarme solo en el sistema de videollamada estaba bien pero ofrecer al usuario varias herramientas para comunicarse con otro usuario es un valor añadido para la aplicación.

La aplicación tiene los siguientes requisitos de negocio:

RN – 01.- El sistema tiene ser sencillo, intuitivo y usable.

RN – 02.- El sistema tiene que mostrar en la página de inicio un formulario para introducir el nombre de usuario.

RN – 03.- El sistema verificar y validar que el nombre de usuario tiene más de 2 caracteres para poder acceder a la aplicación.

RN – 04.- El sistema debe de asignar al usuario un token único para acceder a la sala principal de la aplicación para registrar al usuario.

RN – 05.- El sistema debe mostrar un listado de los usuarios ya conectados a la aplicación, en la sala principal.

RN – 06.- El sistema debe mostrar los nuevos usuarios que se conectan a la aplicación a los usuarios ya conectados, en la sala principal.

RN – 07.- El sistema tiene que eliminar del listado e interfaz, a los usuarios que se desconecten de la sala principal.

RN – 08.- El sistema permitirá que los usuarios puedan realizar videollamadas entre pares.

RN – 09.- El sistema tiene que notificar al usuario que le está llamando otro usuario cuando están en la Sala Principal y le mostrará el nombre de usuario y dos botones para cancelar o aceptar la llamada.

RN – 10.- El sistema redirigirá a los usuarios a una habitación única y privada para realizar

videollamadas entre pares.

RN – 11.- El sistema notificará al usuario el estado de la videollamada.

RN – 12.- El sistema permitirá enviar y compartir archivos entre pares en tiempo real y cancelar su envío o recepción en el momento.

RN – 13.- El sistema permitirá mantener conversaciones por chat entre pares.

RN – 14.- El sistema permitirá hacer fotos de la videollamada.

RN – 15.- El sistema permitirá realizar grabación de vídeo de la sesión(videollamada) entre pares.

RN – 16.- El sistema permitirá activar y desactivar el sonido de la videollamada.

RN – 17.- El sistema permitirá volver al la sala principal desde la videollamada.

6.4.2.- Requisitos funcionales

Para realizar la aplicación de este proyecto me he basado en desarrollar los siguientes requisitos funcionales para cumplir el objetivo de crear una aplicación de calidad, atractiva y con funcionalidad. Los requisitos que he decidido establecer son:

RQF – 00: Mostrar pagina de inicio y bienvenida (Inicio)

RQF – 01: Registrar usuario (Inicio)

RQF – 02: Validar usuario (Inicio)

RQF – 03: Asignar un token al usuario (Inicio)

RQF – 04: Redirigir al usuario a la Sala Principal (Inicio)

RQF – 05: Establecer un socket de conexión (Sala Principal)

RQF – 06: Listar usuarios (Sala Principal)

RQF – 07: Añadir usuario (Sala Principal)

RQF – 08: Eliminar usuario (Sala Principal)

RQF – 09: Realizar llamada (Sala Principal)

RQF – 10: Notificar llamada

RQF – 11: Rechazar llamada

RQF – 12: Aceptar llamada

RQF – 13: Crear la Habitación para la videollamada entre pares (WebRTC)

RQF – 14: Redirigir a la Habitación (WebRTC)

RQF – 15: Securizar la Habitación (WebRTC)

RQF – 16: Establecer un socket de conexión (Habitación)

RQF – 17: Obtener los recursos de la máquina (getUserMedia())

RQF – 18: Sincronizar el par de usuarios (Habitación)

RQF – 19: Configurar la Conexión entre Pares. WebRTC (Habitación)

RQF – 20: Obtener y establecer SDP Offer/Answer (WebRTC)

RQF – 21: Enviar SDP Offer/Answer (WebRTC)

RQF – 22: Establecer la SDP Remota Offer/Answer (WebRTC)

RQF – 23: Obtener y establecer los ICECandidates (WebRTC)

RQF – 24: Enviar los ICECandidates (WebRTC)

RQF – 25: Recibir los ICECandidates Remotos (WebRTC)

RQF – 26: Añadir los ICECandidates Remotos (WebRTC)

RQF – 27: Establecer y mantener la conexión con WebRTC entre pares (Habitación)

RQF – 28: Cerrar la conexión con WebRTC entre pares (Habitación)

RQF – 29: Adaptar la vista para la videollamada (Habitación)

RQF – 30: Menú GUI (Habitación)

RQF – 31: Pantalla Completa On/Off (Habitación)

RQF – 32: Hacer Foto (Habitación)

RQF – 33: Grabar vídeo (Habitación)

RQF – 34: Crear vídeo (Habitación)

RQF – 35: Transferir archivos (Habitación)

RQF – 36: Notificar estado de la transferencia del archivo

RQF – 37: Cancelar envío del archivo

RQF – 38: Cancelar recepción del archivo

RQF – 39: Sonido On/Off (Habitación)

RQF – 40: Chat On/Off (Habitación)

RQF – 41: Chat (Habitación)

RQF – 42: Enviar mensaje (Chat)

RQF – 43: Recibir mensaje (Chat)

RQF – 44: Listar mensajes (Chat)

RQF – 45: Volver a la Sala Principal (Habitación)

Ver **Anexo B**. Requisitos funcionales

6.4.3.- Requisitos no funcionales

Los requisitos no funcionales son aquellos que describen las propiedades que la aplicación tiene que cumplir, en ningún momento expresan funcionalidad del sistema, sino características que tiene que tener el sistema. Esta aplicación tiene los siguiente requisitos no funcionales:

RQNF – 01.- Interfaz de usuario simple e intuitiva.

RQNF – 02.- Interfaz de usuario responsiva

RQNF – 03.- Conexión a Internet

RQNF – 04.- Ejecución correcta de la aplicación en varios navegadores

RQNF – 01	**Interfaz de usuario simple e intuitiva**
Versión	1.0(11/03/06)
Descripción	El sistema tiene que tener una interfaz de usuario simple e intuitiva. La interfaz tiene que guiar al usuario Los elementos que componen la interfaz como son el Chat o el Menú GUID deben de poder ser movidos por la pantalla para optimizar la usabilidad de la aplicación.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla 6.2.- RQNF – 01: Interfaz de usuario simple e intuitiva

RQNF – 02	**Interfaz de usuario responsiva**
Versión	1.0(11/03/06)
Descripción	El sistema tiene que tener una interfaz de usuario responsiva que se adapte a distintos tamaños de la pantalla y a distintos dispositivos.
Importancia	Alta
Prioridad	Media
Comentarios	Ninguno

Tabla 6.3.- RQNF – 02: Interfaz de usuario responsiva

RQNF – 03	**Conexión a Internet**
Versión	1.0(11/03/06)
Descripción	El sistema tiene que tener conexión a internet para poder ejecutarse.
Importancia	Alta
Prioridad	Alta
Comentarios	Al utilizar 4G, 3G o HSDPA para realizar videollamadas VoIP, algunos proveedores de red no añaden a su tarifa este servicio y tiene que ser contratado adicionalmente. En este caso al realizar una vídeo llamada la conexión se realizará con éxito, pero la red estará bloqueada para enviar datos por VoIP.

Tabla 6.4.- RQNF – 03: Conexión a Internet

RQNF – 04	**Ejecución correcta de la aplicación en varios navegadores**
Versión	1.0(11/03/06)
Descripción	El sistema tiene que ejecutarse de forma correcta en varios navegadores compatibles con WebRTC tanto navegadores web de escritorio como navegadores web móvil.
Importancia	Alta
Prioridad	Alta
Comentarios	Como mínimo tiene que ser compatible con el navegador web FireFox de escritorio y móvil.

Tabla 6.5.- RQNF – 04: Ejecución correcta de la aplicación en varios navegadores

6.4.4.- Requisitos de información

Los requisitos de información son los datos e información que el sistema debe almacenar y gestionar para dar soporte a diferentes procesos de negocio.

RI – 01: Fichero de logs del sistema

RI – 01	**Fichero de logs del sistema**
Versión	1.0(11/03/06)
Descripción	El sistema tiene que almacenar un fichero con los eventos de entrada/salida del servidor BackEnd.
Importancia	Media
Prioridad	Baja
Comentarios	El fichero tiene que contener las trazas de los servlets y de los sockets del sistema.

Tabla 6.6.- RI – 01: Fichero de logs del sistema

Capítulo 7
Diseño del software

7.- Diseño del software

7.1.- Introducción

En este capítulo se recoge como se ha diseñado el software, los componentes principales y la arquitectura física del sistema. Primero se explicará la arquitectura física y sus componentes, para poder entender la arquitectura lógica que compone el sistema. Por último se representa por medio de diagramas de secuencia, la lógica que el sistema tiene que tener para satisfacer sus funcionalidades.

7.2.- Arquitectura física

7.2.1.- Modelo cliente-servidor

El sistema está basado en una arquitectura cliente-servidor, la cual es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados **servidores**, y los demandantes, llamados **clientes**. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta a su petición.

En el siguiente diagrama se muestra la arquitectura simple del modelo cliente-servidor:

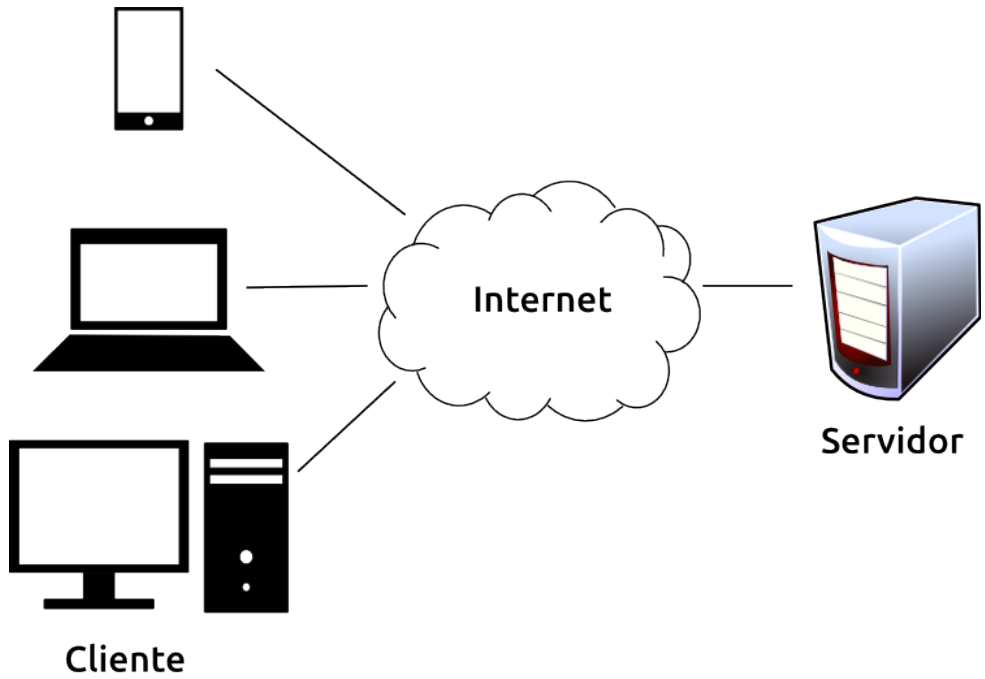

Figura 7.1.- Diagrama arquitectura cliente-servidor

7.2.2.- Arquitectura física Sala Principal

La arquitectura física de la Sala Principal es un modelo cliente-servidor compuesto por un servlet y un websocket para atender las peticiones de los clientes. El objetivo de Sala Principal es la interconexión entre múltiples usuarios para que puedan enviar eventos y consumir recursos entre ellos. En la siguiente imagen se muestra un diagrama de la arquitectura física de la Sala Principal:

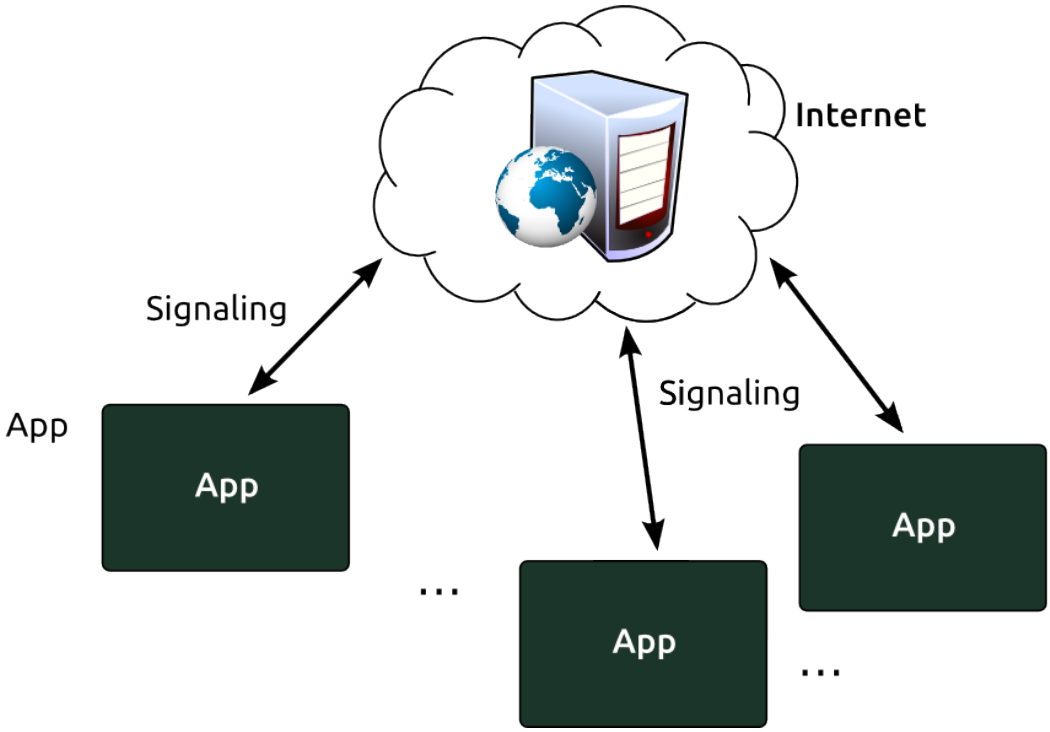

Figura 7.2.- Diagrama de la arquitectura física de la Sala Principal

En la figura 7.2 se puede observar que lo más importante de la Sala Principal es la conexión y la señalización entre el cliente-servidor para la comunicación cliente-cliente. Este proceso de señalización bidireccional lo realiza un socket, por este motivo es un requisito fundamental que el tanto como el cliente como el servidor tienen que soportar la ejecución de sockets de conexión.

7.2.3.- Arquitectura física Habitación WebRTC

La arquitectura de la Habitación WebRTC es una arquitectura cliente-servidor con un WebSocket bidireccional para que los usuarios puedan enviar y recibir datos entre pares. Para la implementación y diseño de la conexión entre pares con WebRTC consiste en realizar una secuencia ordenada para enviar y recibir los datos necesarios de los usuarios que van a participar en la videollamada, para que el navegador se configure y establezca la conexión de forma transparente para el usuario y también para el programador.

Para utilizar WebRTC lo necesario es que como se muestra en el capitulo anterior, que el navegador sea compatible con WebRTC, esto significa que esté incluida esta tecnología en las funcionalidades del navegador. Para saber si la API de WebRTC está disponible o está en el navegador es tan fácil como pulsar Ctrl+Shift+I (Firefox o Chrome) y ver si los método de la API está incluidos en la lista de funciones del Scope Local del objeto Window (ej. RTCIceCandidate, RTCSessionDescription).

WebRTC necesita para funcionar que se pasen unos parámetros entre los navegadores para configurar la sesión del navegador y establecer la conexión entre pares. Lo más importante para implementar el cliente (ClientEndPoint) es saber que se tiene que compartir y en que orden. En el caso del servidor hay que implementar los servlets para poder recibir peticiones y servirlas y el websocket para establecer una comunicación bidireccional entre los usuarios. En el siguiente diagrama se muestra un esquema de como sería la arquitectura de WebRTC en una conexión entre pares desde una perspectiva general:

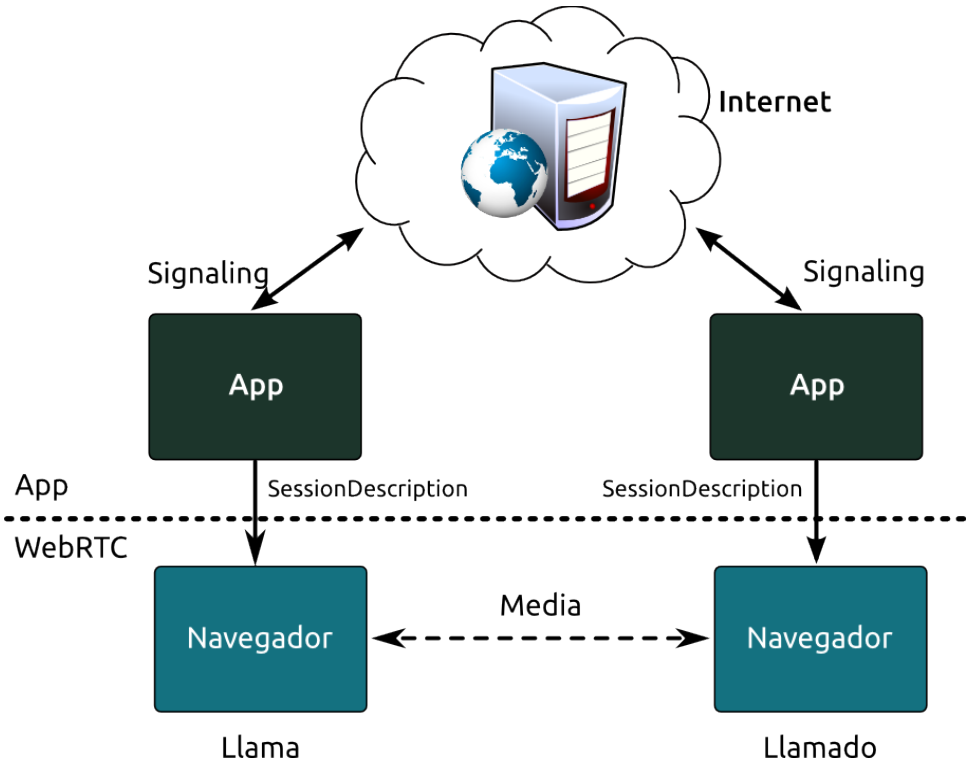

Figura 7.3.- Diagrama de la arquitectura física de la Habitación WebRTC

Como se puede distinguir en la anterior imagen, la parte del cliente (FrontEnd) de la aplicación a desarrollar se encuentra en una capa por encima de la capa del navegador donde se encuentra WebRTC. Lo importante es señalizar la descripción de sesión (SessionDescription) para que el navegador se configure y establezca la comunicación con la tecnología WebRTC.

Para realizar una secuencia ordenada para enviar la señal (signaling) nos tenemos que apoyar en la tecnología WebSocket alojada en el servidor(BackEnd) de peticiones. Esta señalización consiste básicamente en que un usuario envía unos datos al servidor y el servidor debe de gestionar las señalizaciones para que los datos los envíe al usuario del otro extremo y no se envíen a cualquier usuario que esté conectado a la aplicación.

El objetivo que el WebSocket tiene en este punto es señalizar los datos entre dos pares de usuarios según lo que la máquina demande, aquí es donde el programador tiene que sacar el ingenio y saber establecer unos parámetros y variables para que se sincronicen el par de usuarios y obtengan exactamente lo que demande la tecnología WebRTC para configurarse.

En el siguiente capítulo 8, se explica como implementar programando esta parte desde el cliente (FrontEnd) para sincronizar los usuarios y también como implementar el Socket y la Habitación del par de usuarios para señalizar la información.

En el punto *7.7.2.-Diagrama de secuencia de la Habitación WebRTC* se ilustra la secuencia lógica que se tiene que seguir para la configuración de la Habitación WebRTC entre pares, en el diagrama se muestra en que orden tienen que enviar y recibir los datos, que tipos de datos son necesarios para que el navegador de cada usuario se configure y pueda disfrutar esta tecnología.

7.3.- Arquitectura lógica

7.3.1.- Diagrama de componentes

En la siguiente figura se muestra el diagrama de componentes del proyecto, se puede apreciar dos entidades principales diferentes, el cliente(FrontEnd) y el servidor(BackEnd), el flujo de los datos y los componentes que forma cada entidad.

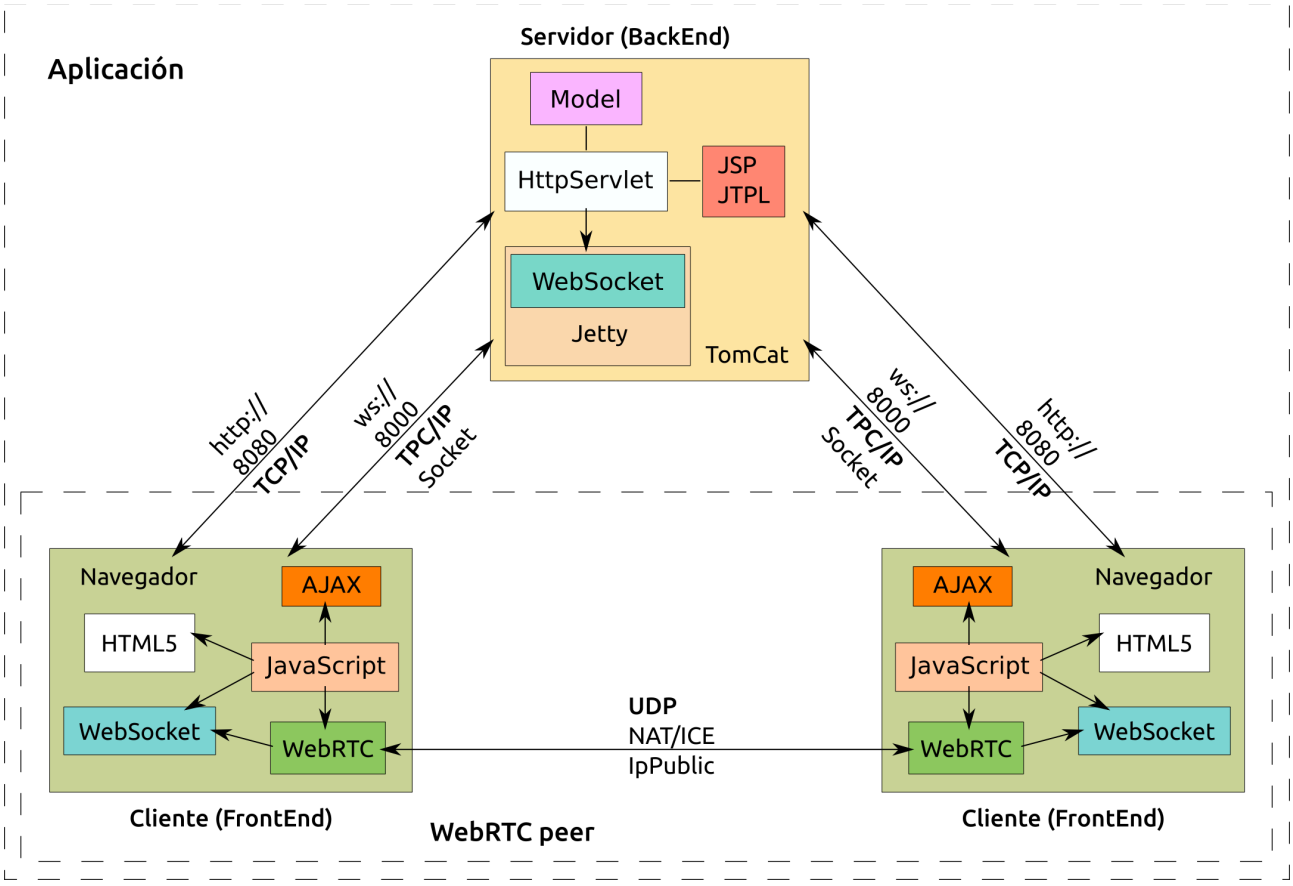

Figura 7.4.- Diagrama de componentes del proyecto

7.3.2.- Diagrama de clases del servidor (BackEnd)

El lenguaje de programación que se utiliza en este proyecto para programar la parte del servidor(BackEnd) es Java, el proyecto se realiza en una plataforma JavaEE utilizando servlets ywebsockets. En el siguiente diagrama UML se ilustra las clases que componen el proyecto y sus relaciones.

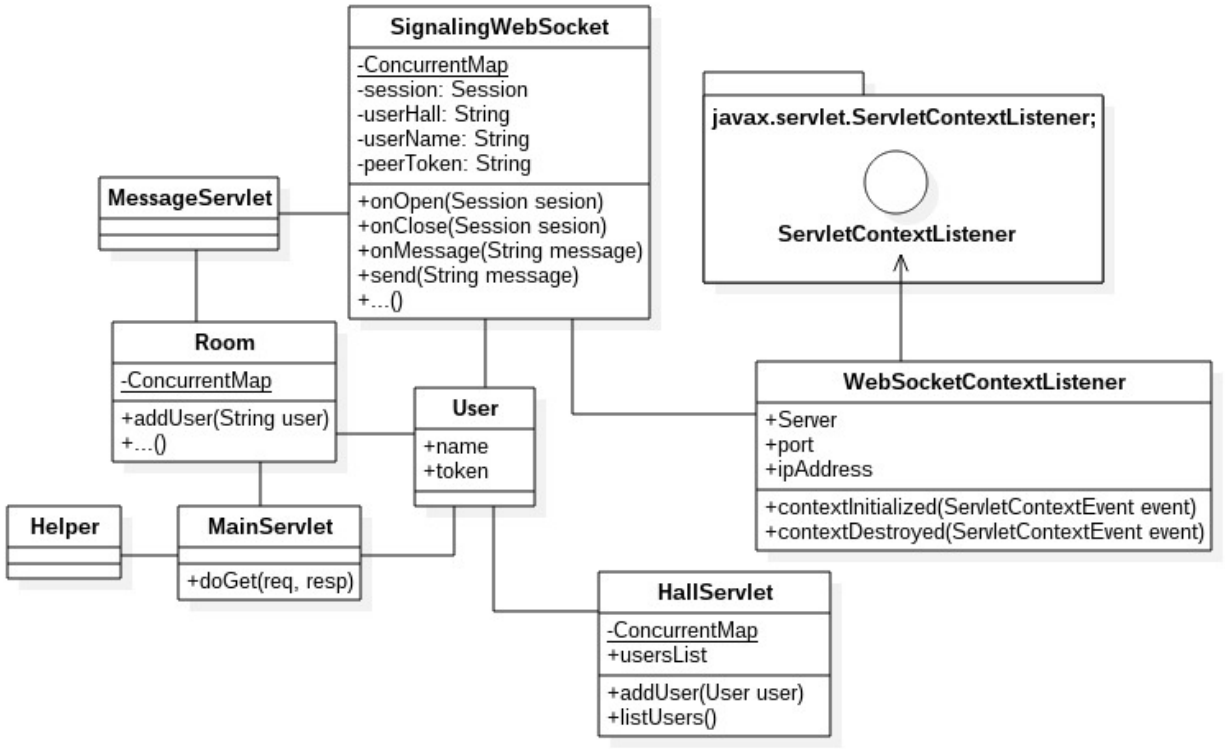

Figura 7.5.- Diagrama de clases del Servidor (BackEnd)

7.3.3.- Diagramas de secuencia

7.3.3.1.- Diagrama de secuencia de la Sala Principal

En el siguiente diagrama de secuencia se muestra desde una visión general la lógica de la Sala Principal cuando dos usuarios se conectan a la aplicación:

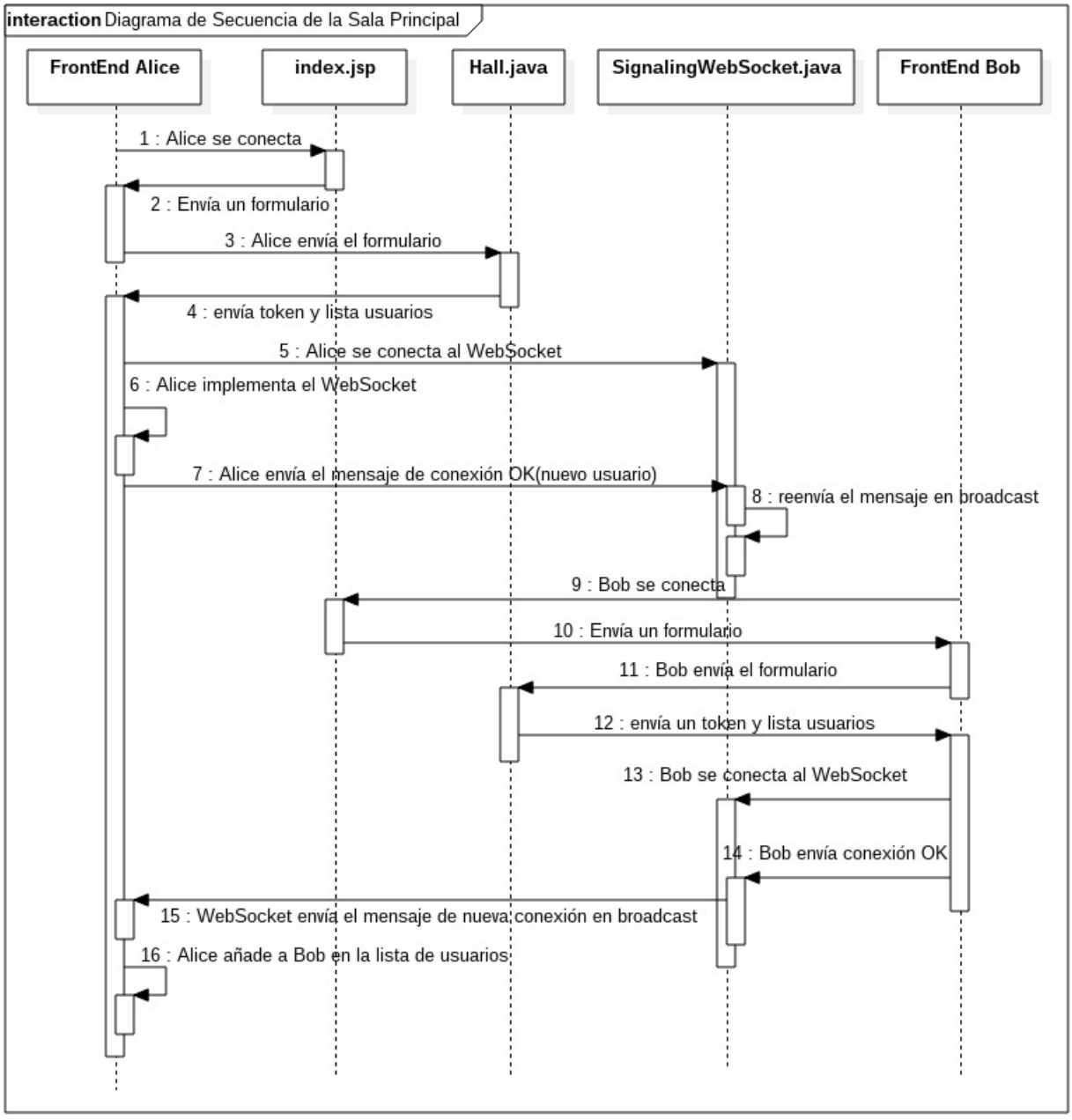

Figura 7.6.- Diagrama de secuencia Sala Principal

7.3.3.2.- Diagrama de secuencia de la Habitación WebRTC

En el siguiente diagrama de secuencia se ilustra la lógica de la conexión de la Habitación WebRTC cuando dos usuarios realizan una videollamada.

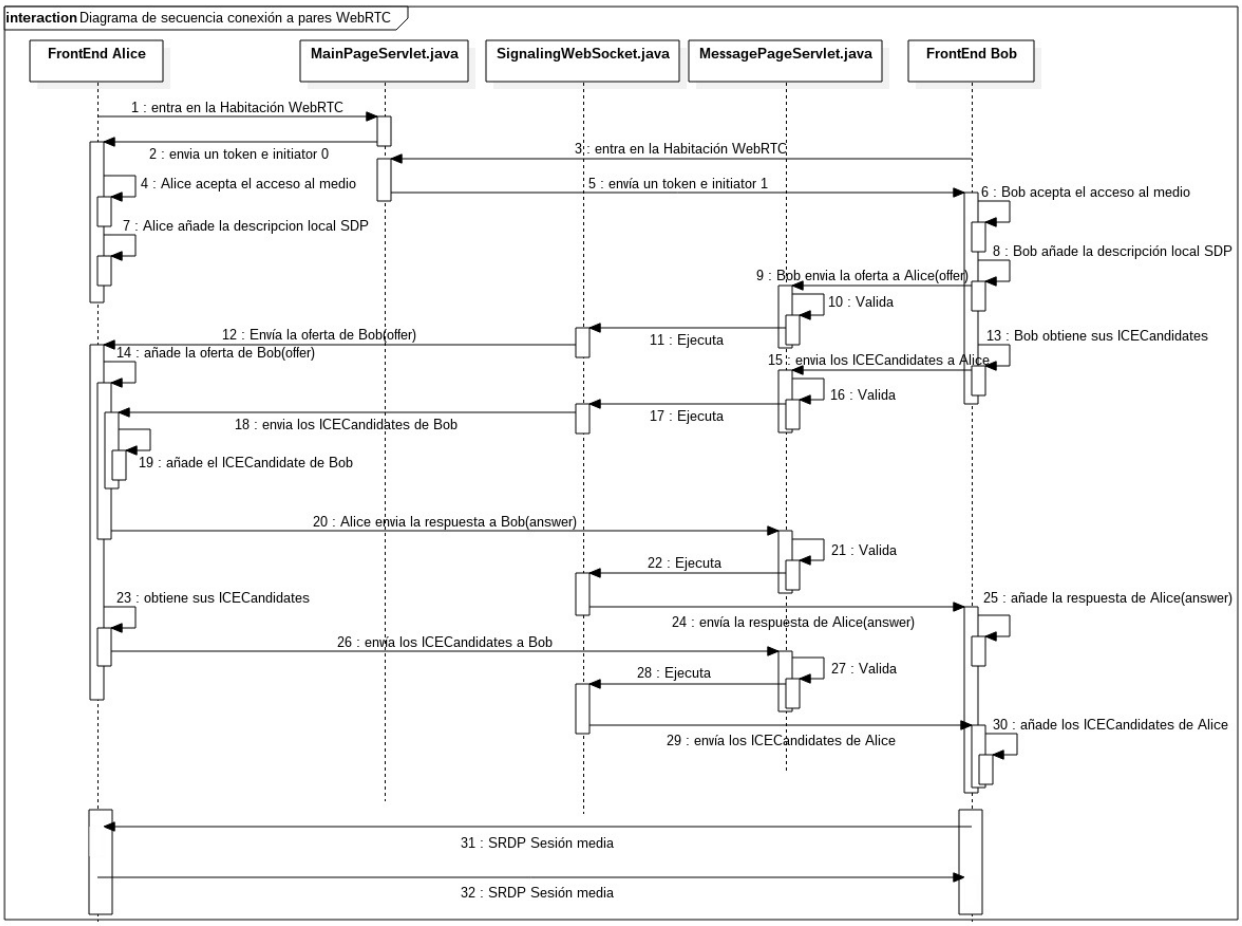

Figura 7.7.- Diagrama de secuencia conexión Habitación WebRTC

7.3.3.3.- Diagrama de secuencia compartir archivos (Habitación)

En el siguiente diagrama se muestra la secuencia que sigue el proceso compartir archivo desde el momento que el usuario pulsa el botón compartir archivo de la interfaz.

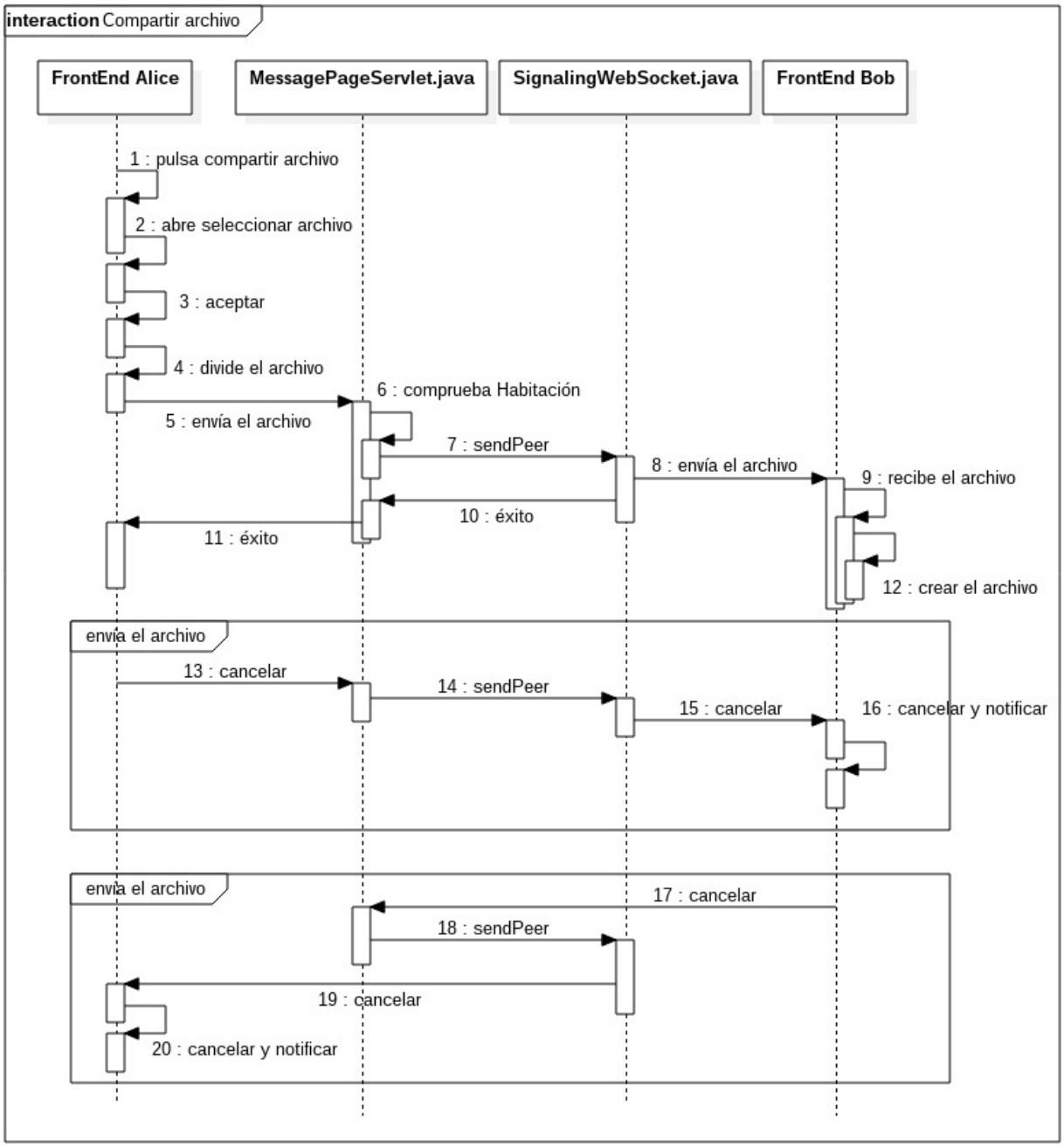

Figura 7.8.- Diagrama de secuencia compartir archivos

7.3.3.4.- Diagrama de secuencia hacer foto archivo (Habitación)

En el siguiente diagrama se muestra la secuencia del proceso hacer foto, como se puede ver el FrontEnd se encarga de ejecutar el proceso.

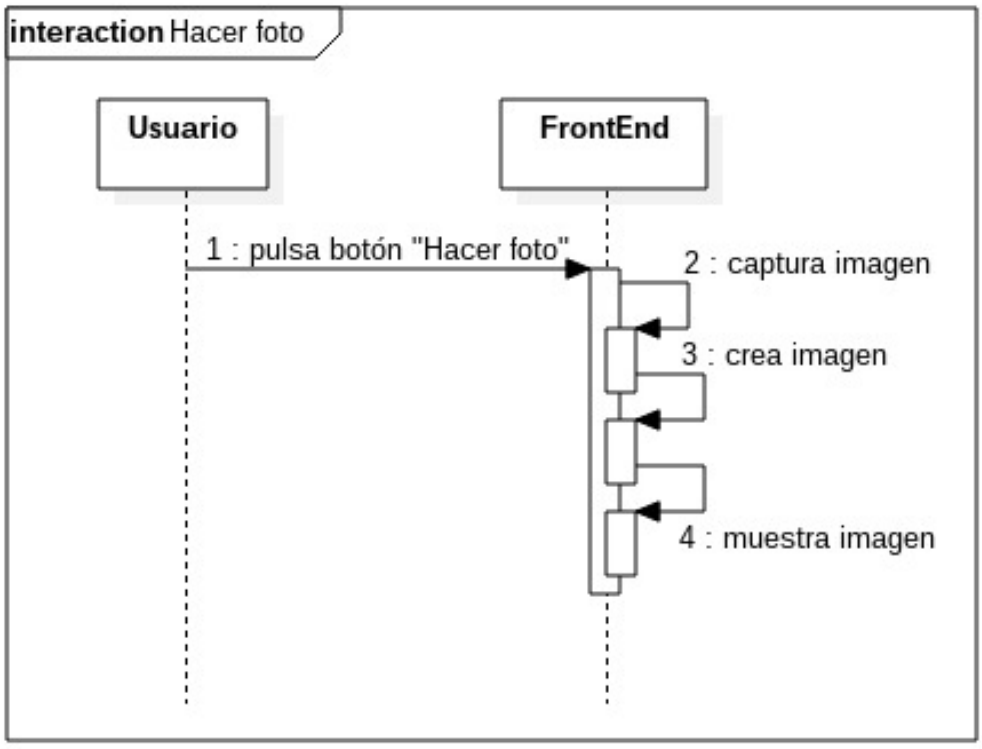

Figura 7.9.- Diagrama de secuencia hacer foto

7.3.3.5.- Diagrama de secuencia grabar vídeo (Habitación)

En el siguiente diagrama de secuencia se ilustra como grabar video. Al grabar un video sigue el mismo proceso estandar de pulsar el botón On para empezar la grabación y el botón Off para parar la grabación.

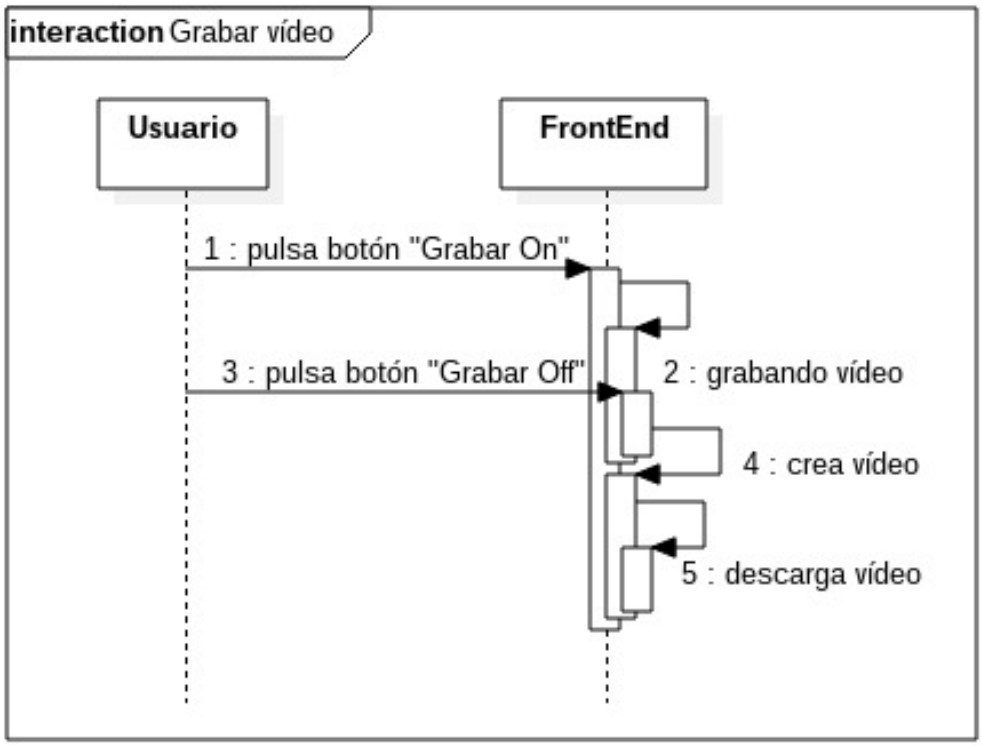

Figura 7.10.- Diagrama de secuencia grabar vídeo

7.3.3.6.- Diagrama de secuencia enviar/recibir mensaje por Chat (Habitación)

En el siguiente diagrama de secuencia se observa el proceso que un mensaje de texto por Chat sigue cuando se envía entre pares.

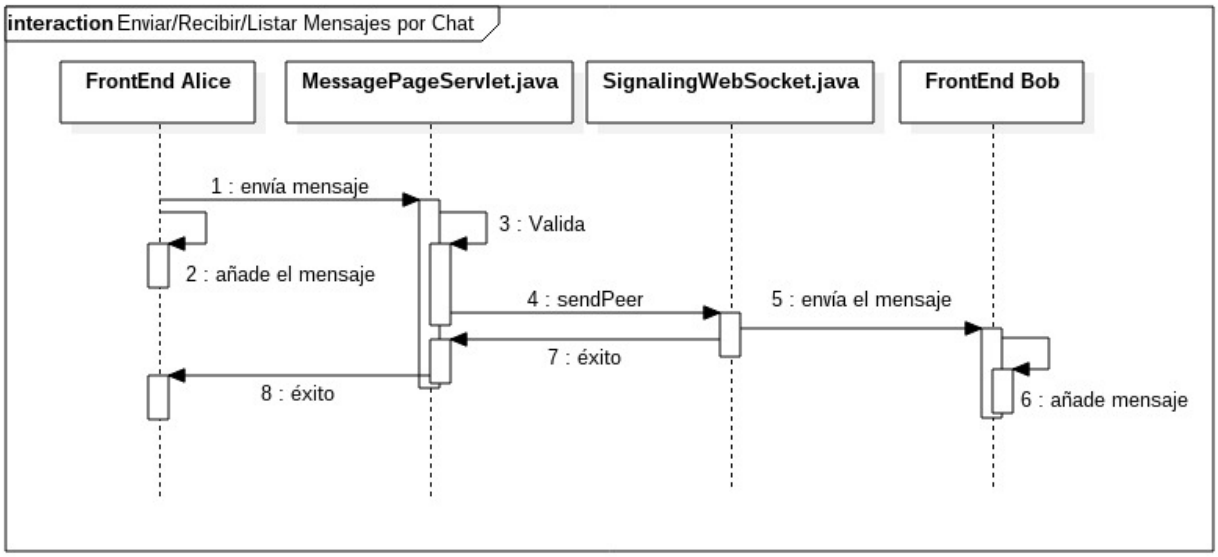

Figura 7.11.- Diagrama de enviar/recibir mensajes por Chat

7.4.- Modelo de datos

Este proyecto no tiene modelo de datos puesto que no tiene que guardar la persistencia de ninguna entidad, objeto o modelo de datos.

Este proyecto posee un diccionario de datos para definir los servicios y eventos (ver apartado *8.1.1.- Eventos FrontEnd (Cliente) y BackEnd(Servidor)*) que la aplicación envía y recibe.

La aplicación en la parte del servidor (BackEnd) guarda y elimina los datos necesarios en memoria dinámica por medio del objeto `ConcurrentHashMap<Object, Object>` el cual guada las variables en forma *clave=valor*, manteniendo la instancia del objeto o clase en memoria dinámica Los objetos que se guardan en memoria dinámica son:

-`SignalingWebSocket`

- `Room`

- `User`

El cliente (FrontEnd) no necesita persistencia de datos.

Como el proyecto está realizado con la tecnología JavaEE, se le puede añadir librerias o frameworks para mantener la persistencia de los datos si fuera necesario. En este caso se le podría añadir un conector JDBC y alguna tecnología Java para mantener la persistencia de los datos JPA como puede ser EclipseLink, Hibernate, ObjectDB, TopLink,...

7.5.- Diagrama de despliegue

La siguiente imagen muestra el diagrama de despliegue del proyecto. La aplicación es de tipo cliente-servidor, se compone de varios clientes para que puedan hacer videollamadas entre pares. Es un requisito fundamental la conexión a internet para que la aplicación funcione y que el navegador del dispositivo sea compatible con la tecnología WebRTC para realizar videollamadas.

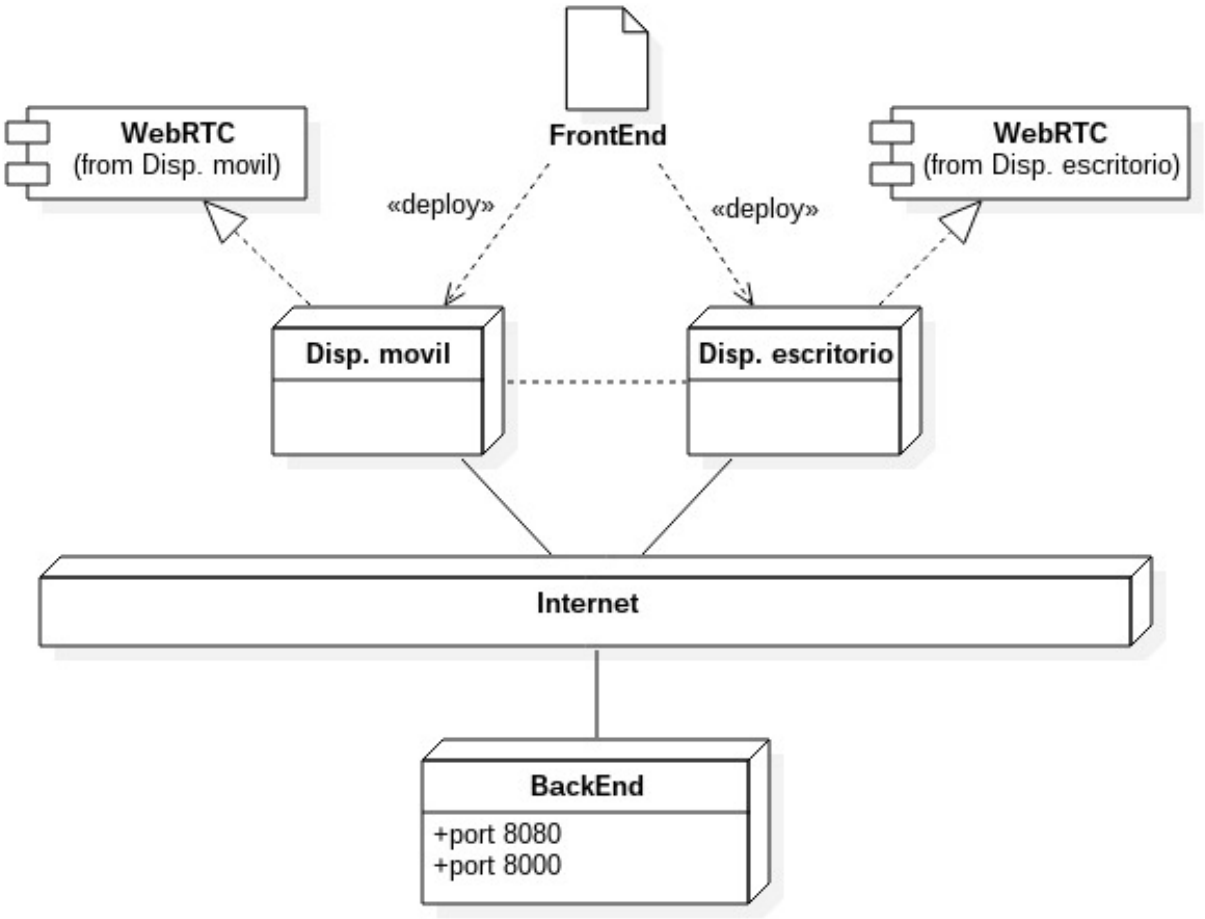

Figura 7.12.- Diagrama de despliegue del proyecto

La aplicación soporta tanto dispositivos móviles como dispositivos de escritorio. En el capítulo *3.4.- Navegadores que soportan actualmente la API WebRTC* se especifica que navegadores son compatibles con la tecnología WebRTC.

Capítulo 8
Implementación del proyecto

8.- Implementación del proyecto

8.1.- Introducción

En el siguiente capitulo se muestra y explica el diseño, el desarrollo e implementación de la aplicación objeto de este proyecto. Este siguiente capítulo se ha estructurado de la siguiente manera:

- **Creación de la Página de Bienvenida y Sala Principal:** La Página de Bienvenida de la aplicación contiene el punto de entrada del sistema y enlaza al usuario a la Sala Principal. En la Sala Principal se muestran los usuarios que están conectados a la aplicación, pueden realizar llamadas o videollamadas y pueden ser llamados.

- **Conexión entre pares con WebRTC:** Esta parte del capítulo es la videollamada en sí, aquí se hace uso de la tecnología WebRTC, se muestran los diagramas de secuencia para como realizar una videollamada entre pares con WebRTC y como se implementa en lenguaje JavaScript y Java.

Estos dos bloques se ha estructurado en otros dos bloques: La parte del diseño, modelado, análisis e implementación desde el lado del cliente y desde el lado del servidor para realizar la aplicación.

Para implementar el proyecto en la parte del servidor (**BackEnd**) son necesarias estas librerías JAVA adicionales:

- `commons-io-2.1.jar` : Biblioteca JAVA de utilidades para la entrada y salida de datos.

- `java-json.jar` : Biblioteca multipropósito JAVA para el procesado de datos JSON.

- `jtpl-2.0.jar`: Es una biblioteca JAVA para el procesado de archivos *.JTPL.

- `jetty-server-8.jar` : Biblioteca JAVA de Servlets Jetty.

- `jetty-websocket-8.jar` : Biblioteca JAVA de WebSocket Jetty.

- `jetty-io-8.jar` : Biblioteca JAVA de utilidades para la entrada y salida de datos del servidor Jetty.

- `jetty-http-8.jar` : Biblioteca JAVA para el protocolo HTTP.

- `jetty-util-8.jar` : Biblioteca JAVA de utilidades para el servidor Jetty.

- `jetty-continuation-8.jar` : Biblioteca JAVA de nuevas características del servidor Jetty.

Para implementar el proyecto en la parte del cliente (**FrontEnd**) solo es necesario que el navegador sea compatible con WebRTC y habilitar lJavaScript.

8.2.- Implementación del WebSocket (BackEnd)

La parte del desarrollo e implementación del WebSocket es una parte fundamental del proyecto para que los usuarios puedan comunicarse entre ellos y enviar y recibir información de forma transparente.

8.2.1.- `WebRTCServerServletContextListener.java`

Para implementar el WebSocket, se realiza en lenguaje JAVA y primero se tiene que crear el `listener` en el archivo ***web.xml*** del proyecto referenciado a la clase del listener del WebSocket, `WebRTCServerServletContextListener`:

```
<listener>
     <listener-class>
          org.webrtc.WebRTCServerServletContextListener
     </listener-class>
</listener>
```

Código 8.1.- Configuración del archivo *web.xml* para el listener del WebSocket

El siguiente paso es crear la clase JAVA `WebRTCServerServletContextListener` en el proyecto para implementarlo. La clase `WebRTCServerServletContextListener` básicamente es un servlet que se inicial al arrancar el proyecto y se mantiene a la escucha de eventos. En esta clase `WebRTCServerServletContextListener` hereda del objeto `ServletContextListener` y embebe un servidor Jetty. El servidor abre el puerto 8000 (se puede utilizar otros puertos como el 8081, 433) y la dirección Ip para incluirla en el objeto `InetAddress` y pasarle la configuración al objeto `server = new Server(bindAddr)`. Por último cuando el objeto `Server` está configurado lo inicia con la función `server.start()`.

En la siguiente tabla se muestra la implementación de clase JAVA `WebRTCServerServletContextListener`:

```
import org.eclipse.jetty.server.Server;
...
public class WebRTCServerServletContextListener implements
ServletContextListener {

     private Server server = null;

     /** Comienza el servidor Jetty embebido cuando la aplicación
      * WEB comienza. */
     @Override
```

```
	public void contextInitialized(ServletContextEvent event) {
		try {
			// 1) Crea un servidor Jetty con el puerto 8000
			int port = 8000; // WebSocket port support
			InetAddress address =
			InetAddress.getByName("ip_address"); //Dirección Ip
			InetSocketAddress bindAddr = new
			InetSocketAddress(address,port);
			server = new Server(bindAddr);
			// Empieza el servidor Jetty a ejecutarse.
			server.start();

		} catch (Throwable e) {
			e.printStackTrace();
		}
	}

	/** Para el servidor Jetty embebido cuando la aplicación WEB
	 * Application para. */
	@Override
	public void contextDestroyed(ServletContextEvent event) {
		if (server != null) {
			try {// stop the Jetty server.
				server.stop();
			} catch (Exception e) {
				e.printStackTrace();
			}
		}
	}
}
```

Código 8.2.- Implementación del WebSocket Listener `ServletContextListener`. Lenguaje Java

Cuando el `Server` inicia la clase SignalingWebSocket se pone a la escucha en el WebSocket. En este punto se puede hacer de dos formas la escucha de WebSocket: si se utiliza el servidor Jetty, existe un handler para atender las peticiones, pero como en este caso la aplicación se va a desplegar en la plataforma OpenShift se tiene que utilizar la API JSR 356. En el código adjunto con esta memoria posee el código fuente de la implementación del `WebRTCServerServletContextListener` con el *handler* para el servidor Jetty.

8.2.2.- SignalingWebSocket.java

La clase `SignalingWebSocket` se encarga de recibir, enviar mensajes y mantener los datos dinámicos en memoria para poder aplicar la lógica para atender las peticiones y enviar las respuestas a los usuarios. El WebSocket gestiona las peticiones de la Sala Principal y las Habitaciones WebRTC. En la siguiente tabla se muestra un ejemplo de los métodos básicos para la implementación de esta clase y la lógica de negocio que del método `onMessage(String)`:

```
import ...

@ServerEndpoint(value="/")
public class SignalingWebSocket {

     private static final ConcurrentMap<String,
     SignalingWebSocket> channels = new ConcurrentHashMap<String,
     SignalingWebSocket>();
     private Session session;
     private String hallToken;
     private String userName;
     private String peerToken;
     /**
      * Método público para abrir el socket ServerEndPoint
      * @param session
      */
     @OnOpen
     public void onOpen(Session session) {
          this.session = session;
     }
     /**
      * Método publico para cerrar las conexiones con el socket
      * @param session
      */
     @OnClose
     public void onClose(Session session){
          try{
          if (userHall != null) {
               channels.remove(hallToken, this);
               Hall.userList.remove(hallToken);
               deleteFromHall(hallToken);
          }
          if (peerToken != null) {
               channels.remove(peerToken, this);
               Room.disconnect(peerToken);
          }
          }catch(Exception e){
```

```
            e.printStackTrace();
        }
    }
    @OnMessage
    public void onMessage(String message) {
        System.out.println("Se recibe -> " + message);
        JSONObject jsonObject;
        try {
            jsonObject = new JSONObject(message);
            if (!jsonObject.isNull("type")) {

                if (jsonObject.get("type").equals("connect")){
                    ...
                    channels.put(hallToken, this);
                    addUserToHall(hallToken, userName);
                }
                if (jsonObject.get("type").equals("calling")){
                    ...
                    callingToUser(from, username, to);
                }
                if (jsonObject.get("type").equals("token")){
                    ...
                    channels.put(peerToken, this);
                }
            }
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }

    private static void addUserToHall(String userToken, String
    userName) {...
    }
    private static void deleteFromHall(String userToken) {
    ...
    }
    private static void callingToUser(String from, String
    userName, String to) {
    ...
    }
    /**
     *  Método público para enviar el mensaje desde el socket a
     * los clientes de la Sala Principal
     * @param message
     */
    public void sendMessageOut(String message) {
```

```
            if (session != null) {
                try {
                    session.getBasicRemote().sendText(message);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
        /**
         * Método para enviar la oferta, respuesta, icecandidates
         * etc, de WebRTC
         * @param token
         * @param message
         * @return
         */
        public static boolean sendPeer(String token, String message){
            boolean success = false;
            SignalingWebSocket ws = channels.get(token);
            if (ws != null) {
                success = ws.send(message);
            }
            return success;
        }
        /**
         * Método para enviar los datos de WebRTC para la
         * videoconexión
         *
         * @param message
         * @return
         */
        private boolean send(String message) {

            if (session != null) {
                try {
                    session.getBasicRemote().sendText(message);
                    return true;
                } catch (Exception e) {
                    e.printStackTrace();
                    return false;
                }
            }
            return false;
        }
}
```

Código 8.3.- Implementación de la clase `SignalingWebSocket`. Lenguaje Java

8.3.- Página de Bienvenida y Sala Principal

Para que los usuarios puedan hacer llamadas entre ellos, se ha creado una Sala Principal donde se muestra una lista de los usuarios conectados a la aplicación, siendo esta lista actualizada en el mismo momento que un usuario se conecta o se desconecta a esta Sala Principal de la aplicación.

Antes de crear la Sala Principal hay que crear la **Página de Bienvenida** para que los usuarios introduzcan su nombre o alias y puedan acceder a la sala principal. Cuando un usuario entra en la sala principal la aplicación le asigna un identificador (token) único para que la aplicación lo reconozca y pueda localizarlo para enviar/recibir información.

Los usuarios que se conecten a la **Sala Principal**, pueden ser de dos tipos: usuarios ya conectados dentro de la sala principal o usuarios nuevos que se acaban de conectar a la sala principal. Para que se puedan listar los usuarios ya conectados, el servlet que atiende las peticiones guarda un registro en memoria de los usuarios conectados, guardando el usuario con el identificador (token) anteriormente mencionado y su instancia del websocket. Al listar los usuarios conectados se identifican con su token único en el id HTML para poder ser diferenciados.

Cuando un nuevo usuario que se conecta a la aplicación abre un websocket entre él y la aplicación para tener comunicación bidireccional con el servidor y poder de esta manera comunicarse directamente con otros usuarios.

Para listar nuevos usuarios se realizar por medio de JavaScript y el WebSocket. Cuando un usuario nuevo se conecta, envía una señal en *broadcast* a todos los usuarios conectados informando de que un nuevo usuario se ha conectado a la aplicación, los demás usuarios tienen en implementado en el FrontEnd un tipo de evento "`newuser`" que crea una caja con el nuevo usuario en la lista de usuarios.

En el caso de que un usuario se desconecte de la aplicación, se tiene que eliminar de la lista de todos los usuarios que están en la sala principal, para realizar esto se utiliza la señal del websocket onclose() para enviar la señal en broadcast a los demás usuarios del evento "`deleteuser`" gestionando así la lista de usuarios de la interfaz.

La sala principal tiene el objetivo de mostrar los usuarios pero lo más importante es que se puedan realizar llamadas entre ellos de manera eficaz. La llamada es simplemente un evento de tipo "`calling`" que elimina de la lista al usuario que está llamando para que no pueda ser llamado y se tiene que notificar al usuario llamado de quien le está llamando.

8.3.1.- Eventos Cliente (FrontEnd) y Servidor (BackEnd)

Los eventos y señales del Cliente FrontEnd por medio del WebSocket de BackEnd de la Sala Principal que está a la escucha para la recepción y emisión de eventos son:

- **connect**: Informa a la aplicación que un nuevo usuario se a conectado. La estructura de datos que se envía tiene la siguiente información:

```
{"type":"connect",
"username":"Manuel",
"token":"XXXXXXXXXX..."}
```

Tabla 8.1.- Diccionario de datos: Evento tipo → *connect* (Sala Principal)

- **newuser** :Informa de un nuevo usuario conectado a la Sala Principal. La estructura del tipo de datos que se envía a los usuarios tiene esta información:

```
{"type":"newuser",
"username":"User Name or Alias",
"usertoken":"XXXXXXXXXX..."}
```

Tabla 8.2.- Diccionario de datos: Evento tipo → *newuser* (Sala Principal)

- **calling**: Evento que se envía cuando un usuario llama a otro para realizar una videollamada. La estructura de datos de este evento tiene la siguiente información:

```
{"type":"calling",
"from":"XXXXXXXXXX...",
"username":"Jose Vicente",
"to":"YYYYYYYYY..."}
```

Tabla 8.3.- Diccionario de datos: Evento tipo → *calling* (Sala Principal)

- **onclose**: Evento que envía por defecto el WebSocket JavaScript y Java cuando se desconecta la conexión. Este evento se envía por TCP/IP de forma automática al salir de la sesión y no es necesario implementar ninguna estructura de datos.

- **deleteuser**: Evento para indicar que el usuario ha salido de la Sala Principal o se ha desconectado. Este evento elimina la instancia de este usuario en el servidor y en la lista de usuarios de la Sala Principal y actualiza la lista de usuarios de los demás usuarios en broadcast.

```
{"type":"deleteuser",
"usertoken":"XXXXXXXXXX..."}
```

Tabla 8.4.- Diccionario de datos: Evento tipo → *deleteuser* (Sala Principal)

8.3.2.- La Sala Principal (FrontEnd)

En el siguiente código se muestra las variables y los métodos principales para implementar la Sala Principal (FrontEnd), en el CD-ROM adjunto con este proyecto se puede ver la implementación completa, el archivo se llama ***hall.jtpl***.

```
var host = "http://{server_name}";
var port = 8080;
var wsPort = 8000;
var myToken = document.getElementById("myToken");
var myName = document.getElementById("myName");

function initialize() {
     openChannel();
};
function openChannel() {
     console.log("Abriendo el canal.");
     var location = "ws://"+ window.location.host  +":" + wsPort +
     "/webrtc/";
     channel = new WebSocket(location);
     channel.onopen = onChannelOpened;
     channel.onmessage = onChannelMessage;
     channel.onclose = onChannelClosed;
     channel.onerror = onChannelError;
}
function onChannelOpened() {
     console.log('Conectado a la app');
     channel.send('{"type":"connect", "username":"' +
myName.innerHTML
          + '","token":"' + myToken.innerHTML + '"}');
}
function onChannelMessage(message) {
     processSignalingMessage(message.data);
}
function onChannelError() {
...
```

```
}
function onChannelClosed() {
...
}
var userList = document.getElementById("userList");
function processSignalingMessage(message) {
     if (message) {
          var msg = JSON.parse(message);
          if (msg.type == "newuser") {
               ...
          }
          if (msg.type == "deleteuser") {
               ...
          }
          if (msg.type == "calling") {
          ...
          }
     }
}
function calling(toUser) {
...
}
function acceptCall() {
...
}
function hangUp() {
...
}
initialize();
```

Código 8.4.- Implementación del la Sala Principal (FronEnd). Lenguaje JavaScript

8.3.3.- La Sala Principal (BackEnd)

En el siguiente código se muestra las variables y los métodos principales para implementar la Sala Principal (BackEnd), en el CD-ROM adjunto con este proyecto se puede ver la implementación completa, el archivo se llama ***Hall.java***.

```
public class Hall extends HttpServlet{

     private String message;
     private String userName;
     private User user;
     public static ConcurrentMap<String, User> userList = new
     ConcurrentHashMap<String, User>();

public void doPost(HttpServletRequest req,
HttpServletResponse resp) throws IOException {
     ...
     user.setName(req.getParameter("userName"));
     user.setToken(Helper.generate_random(16));
     userList.put(user.getToken(), user);
     String userListLi = "";
     for (Entry<String, User> user : userList.entrySet()) {
     if( !this.user.getToken().equals(user.getValue().getToken()))
     {
          ...
          //Carga la lista de usuarios
     }

     }
     String userListResp = ...
     Map<String, String> template_values = new HashMap<String,
String>();
     ...
     // Carga los datos y redirige al usuario
     // a la página "hall.jtpl".
}
```

Código 8.5.- Implementación del la Sala Principal (BackEnd). Lenguaje Java

8.4.- Habitación WebRTC Videollamada

8.4.1.- Eventos Cliente (FrontEnd)

En esta parte se describen los eventos necesarios para realizar la videollamada, estos eventos son enviados y recibidos por el WebSocket cliente(FrontEnd), el WebSocket del servidor(BackEnd) solo gestiona a que usuairo tiene que enviar este evento. Para enviar los eventos WebRTC se ha implementado un servlet llamado `MessagePageServlet.java` como se menciona anteriormente, para gestionar el envío de datos y el websocket para .

Eventos:

- **offer**: Evento para recibir una oferta(*offer*) de la descripción de sesión de otro usuario para añadirla a la descripción remota. La estructura de datos que se envía tiene la siguiente información:

```
{"type":"offer",
"msg":"sdp:..."}
```

Tabla 8.5.- Diccionario de datos: Evento tipo → *offer* (Habitación WebRTC)

- **answer**: Evento para recibir la respuesta(*answer*), es la descripción de sesión de otro usuario. La estructura de datos que se envía tiene la siguiente información:

```
{"type":"answer",
"msg":"sdp:..."}
```

Tabla 8.6.- Diccionario de datos: Evento tipo → *answer* (Habitación WebRTC)

- **candidate**: Evento para recibir los candidatos(*answer*). La estructura de datos que se envía tiene la siguiente información:

```
{"type":"candidate",
"msg":"candidate:..."}
```

Tabla 8.7.- Diccionario de datos: Evento tipo → *candidate* (Habitación WebRTC)

8.4.2.- Habitación WebRTC videollamada (FrontEnd)

En este punto se explica los pasos para implementar la conexión entre pares con WebRTC en lenguaje de programación **JavaScript**(FronEnd).

Pongamos el caso que Alice ya ha llamado al Bob y la aplicación redirige a Alice a la habitación para realizar la conexión entre pares. En este punto Alice se reconecta al servidor pasando un argumento del token de la habitación para que otro usuario que no sea ni Alice ni Bob pueda entrar en la habitación. Cuando Alice y Bob se conectan a la aplicación el servidor asigna un token único para Alice y otro para Bob de nuevo para realizar la conexión y poder señalizar la oferta/respuesta (offer/answer) y los iceCandidates de cada usuario.

Este token puede almacenar en una lista en memoria como veremos más adelante en la implementación del BackEnd (ej. ConcurrentMap<String, Object>) o en base de datos. También en la conexión entre pares se declara una variable llamada "initiator" para saber si el usuario ha permitido el uso de la cámara o micrófono para realizar la conexión, esta variable puede tener valor de 0 o 1.

0.- Declaramos la variables globales para hacer uso de la tecnología WebRTC y la API que el navegador ofrece, variables globales referenciadas al objeto DOM HTML5 y variables globales necesarias para realizar el algoritmo para que la conexión entre pares sea un éxito.

Objeto Navigator

El objeto *"navigator"* contiene la información y los recursos que son accesibles por medio del navegador web como: el nombre del navegador, el lenguaje del navegador, el nombre de agente de usuario, la plataforma de compilación del navegador o la cookies válidas del mismo, entre otras muchas más características. En este caso se utilizar el objeto *"navigator"* para poder acceder a los recursos de la máquina como son el micrófono y la webcam o cámara conectada del dispositivo, es decir el navegador puede acceder al medio de la máquina utilizando la función getUserMedia. El parámetro navigator.getUserMedia debe de ser adaptado cada navegador por este motivo debe de declararse con la expresión "||" para usar el prefijo firefox (moz), chrome (webkit), Iexplorer(ms) o Opera.

Objeto Window

El objeto *"window"* representa la ventana abierta del navegador que contiene el documento DOM, este objeto apunta al documento DOM HTML y con el se puede acceder a las propiedades de la ventana del navegador también. El objeto *"window"* es padre del objeto *"navigator"*. Con el objeto *"window"* se declara la RTCPeerConnection para la conexión entre pares, RTCICECandidate para los candidatos usado por el protocolo ICE de WebRTC y la RTCSessionDescription para obtener la descripción de la sesión de la máquina local.

La variable *"peerConnectionConfig"* es la dirección de los servidores STUN que se utilizará para establecer la conexión con WebRTC para obtener los candidatos ICE. En la siguiente tabla se muestra un ejemplo en JavaScript:

```
navigator.getUserMedia = navigator.getUserMedia
          || navigator.mozGetUserMedia ||
navigator.webkitGetUserMedia;
window.RTCPeerConnection = window.RTCPeerConnection
          || window.mozRTCPeerConnection
          || window.webkitRTCPeerConnection;
window.RTCIceCandidate = window.RTCIceCandidate
          || window.mozRTCIceCandidate ||
window.webkitRTCIceCandidate;
window.RTCSessionDescription = window.RTCSessionDescription
          || window.mozRTCSessionDescription
          || window.webkitRTCSessionDescription;
var peerConnectionConfig = {'iceServers' : [ {'urls' :
'stun:stun.services.mozilla.com'   }, {'urls' :
'stun:stun.l.google.com:19302'}]};
```

Código 8.6.- Variables globales utilizadas para realizar conexión entre pares. Lenguaje JavaScript

En la siguiente tabla se muestra las variables que se van a utilizar para referenciar el vídeo local, el video remoto, el flujo de datos local y remoto, el canal de comunicación y más variables necesarias para realizar el algoritmo del comunicación con WebRTC:

```
var localVideo;
var remoteVideo;
var localStream;
var remoteStream;
var channel;
var channelReady = false;
var pc;
var initiator =  {initiator}; // 0 or 1 (1 Send offer)
var started = false; // Booleano
```

Código 8.7.- Variables globales para instanciar el streaming de vídeo. Lenguaje JavaScript

La variables variables localVideo y remoteVideo están referenciados al objeto del DOM que tienen la etiqueta <video>. La variable localStream está referenciada al evento que se crea cuando se accede los recursos de la máquina (micrófono y webcam) y remoteStream están referenciadas al evento que window.RTCPeerConnection creará cuando la conexión entre pares sea un éxito.

1.- Primero iniciamos el algoritmo al cargar, donde se declaran algunas variables, los elementos del DOM HTML5 que se van a utilizar en la conexión como por ejemplo las etiquetas <video></video>, <canvas></canvas>, y las funciones a ejecutar:

```
function initialize() {
     localVideo = document.getElementById("localVideo");
     remoteVideo = document.getElementById("remoteVideo");
     openChannel();
     getUserMedia();
}
```

Código 8.8.- Ejemplo para iniciar el algoritmo para crear la conexión entre pares. Lenguaje JavaScript

2.- Luego se abre un **WebSocket** desde el cliente (ClientEndPoint) con el servidor (ServerEndPoint) y se implementa los métodos de la interfaz (onopen, onclose, onmessage, error):

```
function openChannel() {

     console.log('Abriendo el canal');
     var location = 'ws://{server_name}:8000/webrtc/';

     channel = new WebSocket(location);

     channel.onopen = function() {
          channel.send(
               '{"type":"token", "value":"' + token + '"}'
          );
     };

     channel.onmessage = function (message) {
          console.log(
               'ServerEndPoint -> ClientEndPoint: ' + message.data
          );
     };

     channel.onclose = function() {
          console.log('Canal cerrado');
     };vídeo

     channel.error = function() {
          console.log('Error del canal');
     };
}
```

Código 8.9.- Constructor e interfaz del WebSocket en el cliente(FrontEnd). Lenguaje JavaScript.

Cuando el canal (channel) se abre, se envía el token que el servidor le envió previamente al entrar en la web para crear una habitación en el servidor y poder protegerla de esta manera para que otro usuario no pueda acceder a la conexión entre pares.

3.- Posteriormente se ejecuta el código de la función getUserMedia para obtener la los permisos de usuario para utilizar el audio y el vídeo de su pc. Si el usuario acepta la petición de utilizar la cámara y el micrófono se sigue con la función "success", si ocurre algún error se ejecuta la función "error" o se captura con una excepción:

```
function getUserMedia() {
     try {
          navigator.getUserMedia({
               audio : true,
               video : true
          }, success, error);
     } catch (e) {
          alert("getUserMedia() error: " + e.message);
     };
}

function success(stream) {

     console.log('El usuario ha permitido el acceso al hardware');
     var url = window.URL.createObjectURL(stream);
     localVideo.src = url;
     localStream = stream;
     // La llamada crea la PeerConnection.
     console.log("Initiator: " + initiator);
     if (initiator)
          maybeStart();
}
```

Código 8.10.- Función `getUserMedia` y `success` cliente (FrontEnd). Lenguaje JavaScript

En este punto todavía no se a producido la conexión entre pares, porque el valor de la variable "initiator" es 0, y se producirá la conexión entre pares cuando el usuario remoto se añada a la conexión, es decir cuando Bob acepte la conexión con Alice.

Cuando Bob se conecte a la aplicación se realizará lo mismo que hasta este punto pero el servidor le habrá asignado el valor de 1 a la variable initiator la función "maybeStart()" se ejecuta y se crea la conexión entre pares:

```
function maybeStart() {
     if (localStream) {
          console.log("Creando PeerConnection");
```

```
            createPeerConnection();
            pc.addStream(localStream);
            // El llamado inicia la oferta "offer" al par
            if (initiator)
                 doCall();
      }
}

function createPeerConnection() {
      try {
            pc = new window.RTCPeerConnection(peerConnectionConfig);
      } catch (e) {
            console.log("Error: " + e.message);
            return;
      }
      pc.onconnecting = onSessionConnecting;
      pc.onopen = onSessionOpened;
      pc.onaddstream = onRemoteStreamAdded;
      pc.onicecandidate = onIceCandidate;
      pc.onremovestream = onRemoteStreamRemoved;
}

function onSessionConnecting(message) {
      console.log("Conectando la sesion");
}

function onSessionOpened(message) {
      console.log("Sesión abierta");
}

function onRemoteStreamAdded(event) {

      console.log("Stream remoto añadido");
      var url = window.URL.createObjectURL(event.stream);
      remoteVideo.src = url;
      waitForRemoteVideo();
}

function onRemoteStreamRemoved(event) {
      console.log("Stream remoto eliminado");
}
```

Código 8.11.- Objeto `RTCPeerConnection` y sus métodos, cliente (FrontEnd). Lenguaje JavaScript

En la PeerConnection Bob es el primero que envía la oferta (offer) a Alice porque ejecuta la función doCall(), Bob realiza la llamada y crea la oferta. La función pc.createOffer(gotDescription,

success, error) es de la forma callBack, la cuál ejecuta primero la función gotDescription() donde pc.setLocalDescription() toma la descripción de la máquina y si al tomarla tiene éxito se envía el mensaje por el Servlet (MessagePageServlet) con la descripción local. Si todo este proceso tiene éxito en la función inicial pc.createOffer(...) entonces se ejecuta posteriormente la función "success()" y si no se ejecutará la función "error()".

```
function doCall() {
     console.log("Usuario: " + initiator + " - Envía la oferta al
par");
     pc.createOffer(gotDescription, success, error);
}

function gotDescription(description) {

     console.log('Toma la descripción local');
     pc.setLocalDescription(description, function()
{sendMessage(description);}, error);
}
```

Código 8.12.- Función para crear la oferta(*offer*) y enviarla desde el cliente (FrontEnd). Lenguaje JavaScript

En este momento la oferta de Bob se le envía a Alice la cual tiene que añadirla a la descripción remota de su máquina. En la siguiente tabla se muestra como se implementa la función *processSignalingMessage* que se ejecuta cuando se recibe un mensaje del websocket desde el servidor al cliente, en esta función se incluye la función *setRemoteDescription* para capturar la oferta enviada de Bob a Alice. El mensaje que se recibe por el objeto `JSON` es de tipo 'offer' y el mensaje de la oferta se añade a la RemoteDescription de Alice, entonces si la función tiene éxito se ejecuta el callBack y crea la respuesta con la función createAnswer.

La answer se añade a la descripción local de Alice al crear el objeto RTCSessionDescription(answer) y si al añadirla tiene éxito se ejecuta el callBack y envía la answer a Bob con la función sendMessage(answer). En el siguiente código se muestra como se procesan los eventos recibidos:

```
function processSignalingMessage(message) {
     var msg = null;
     if (message) {
        try{
          msg = JSON.parse(message);
        }catch (e) {
          console.log("Error JSON");
        }
        if (msg.type === 'offer') {
          pc.setRemoteDescription(new RTCSessionDescription(msg),
          function() {
               pc.createAnswer(
```

```
                function(answer) {
                     pc.setLocalDescription(new
                     RTCSessionDescription(answer),
// envía la answer al servidor para ser entregada a quien llama
                     function() {
                          sendMessage(answer);
                     }, errorHandler);
                }, errorHandler);
          }, errorHandler);
        }else if (msg.type === 'answer' && started) {
          pc.setRemoteDescription(new RTCSessionDescription(msg));
          console.log("Añadido answer a la descripción remota");

        }else if (msg.type === 'candidate' && started) {
          var candidate = new window.RTCIceCandidate(msg.ice);
          if (pc.addIceCandidate(candidate)) {
               console.log("Candidato añadido correctamente "
               + candidate.candidate);
          };
        }
     }
}
```

Código 8.13.- Función `processSignalingMessage`, cliente (FrontEnd). Lenguaje JavaScript

Seguidamente cuando se obtiene la descripción remota se activa el Protocolo ICE para obtener y enviar los ICECandidates para realizar la conexión WebRTC. En la siguiente tabla se muestra como implementar la función onIceCandidate(Event):

```
function onIceCandidate(event) {

     if (!pc || !event || !event.candidate) {
          console.log("End of candidates");
          return;
     } else {
          sendMessage({
               type : 'candidate',       // tipo de dato
               'ice' : event.candidate, // ICECandidate
          });
     }
}
```

Código 8.14.- Método `onIceCandidate` implementado para enviar los candidatos y sus IPs (privadas y públicas). Lenguaje JavaScript

Todos los eventos WebRTC del cliente se envían por medio de un servlet "MessagePageServlet.java" para gestionarlos, añadir seguridad y la conexión entre el par de usuarios WebRTC. Este servlet como veremos más adelante envía la información por medio del WebSocket Java si la información es correcta. En la siguiente tabla se muestra como se implementa el servicio en el FrontEnd para enviar la información WebRTC.

```
function sendMessage(message) {
     try {
          var msgString = JSON.stringify(message);
          var path = '/webrtc/message?r={room_key}' + '&u={me}';
          var xhr = new XMLHttpRequest(); // Objeto XMLHttpRequest
          xhr.open('post', path, true); // Método 'POST'
          xhr.send(msgString); // XMLHttpRequest envía los datos
     } catch (error) {          // Bloque catch para
          console.log(error);  // capturar posibles excepciones
     };
}
```

Código 8.15.- Función `sendMessage` con objeto `XMLHttpRequest` a un Servlet Java. Lenguaje JavaScript

8.4.3.- Habitación WebRTC videollamada (BackEnd)

Para gestionar la transferencia de los datos necesarios para la conexión entre pares con WebRTC, se ha desarrollado en el BackEnd en lenguaje Java con la tecnología JavaEE.

Las clases, servlets y objetos necesarios que se han implementado para el desarrollo de la Habitación de la videollamada entre pares con WebRTC son:

MainPageServlet.java → Servlet http para atender las peticiones para configurar la Habitación WebRTC. Este servlet es el punto de entrada para realizar la conexión entre pares WebRTC.

Room.java → Objeto que guarda la instancia de la Habitación WebRTC.

MessajePageServlet.java → Servlet http para atender los mensajes entre pares.

SignalingWebSocket.java → WebSocket para la conexión entre pares, solo se utiliza la función sendPeer(); para enviar los datos entre par de usuarios.

Helper.java → Clase de ayuda para realizar cálculos y algoritmos generales.

- MainPageServlet.java

En esta parte se explica como implementar la clase MainPageServlet.java con ejemplos de código del proyecto.

El servlet MainPageServlet.java se extiende del servlet TomCat HttpServlet, tiene un punto de entrada para atender las peticiones GET en el contexto o path "webrtc/":

```
public class MainPageServlet extends HttpServlet {

     public String token;
     public String roomKey;
     public static final String PATH = "webrtc/";
     private static final String INDEX =
     "http://webrtc-jvrodrigo.rhcloud.com/webrtc/index.jsp";

     public void doGet(HttpServletRequest req, HttpServletResponse
     resp) throws IOException {
          // Obtiene la petición
          String query = req.getQueryString();
          ...
          if(query==null) {
               resp.setStatus(HttpServletResponse.SC_BAD_REQUEST);
               resp.sendRedirect(INDEX);
               return;
          }
```

```
        ...
        // Obtiene params: parámetros de la petición
        Map<String, String> params =
        Helper.get_query_map(query);
        if(params.get("r")==null || params.get("r").equals("")
        || params.get("userName")==null ||
        params.get("userName").equals("")) {
        logger.info("Sin habitacion (room key) o sin nombre de
        usuario (userName)");
        resp.setStatus(HttpServletResponse.SC_BAD_REQUEST);
        resp.sendRedirect(INDEX);
             return;
        }else {
...
```

Código 8.16.- Punto de entrada `MainPageServlet.java`. Lenguaje Java

El servlet MainPageServlet.java espera dos argumentos en la petición GET:

- r: Token de la Habitación

- userName: Nombre de usuario.

Si alguno de estos argumentos no se encuentra en la petición, la respuesta del servidor es código 400 HttpServletResponse.SC_BAD_REQUEST y el servidor redirecciona a la página de inicio. En el caso de que todos estos parámetros de entrada "userName" y "r(room)", se ejecuta el siguiente código, que lo que basicamente hace es:

1.- Comprobar si la habitación está vacia → Crea una nueva habitación WebRTC, añade el usuario, carga las variables y el presentador, y lo redirige a la página *webrtc/index.jtpl*.

2.- Comprobar si la habitación tiene un usuario → Si tiene un usuario añade al otro usuario a la habitación WebRTC, carga las variables y el presentador, y lo redirige a la página *webrtc/index.jtpl*.

3.- Si la habitación si está creada y tiene más de un usuario → Redirige al usuario a la página Habitación Ocupada *webrtc/full.jtpl*.

```
    ...
    }else {
         String room_key = Helper.sanitize(params.get("r"));
         userName = Helper.sanitize(params.get("userName"));
         Room room = Room.get_by_key_name(room_key);
         if(room==null) {
              // Asigna el token al usuario, crea una nueva
              // Habitación y añade al usuario. Initiator=0
         } else if(room!=null && room.get_occupancy() == 1) {
              // Asigna el token al usuario y añade otro usuario
```

```
                // a la Habitación. Initiator=1
            } else {
                // Habitación completa, redirige al usuario a la
                // página "webrtc/full.jtpl"
                return;
            }
            // Carga las variables para redirigir al usuario a la
            // página "webrtc/index.jtpl"
        }
}...
}
```

Código 8.17.- Implementación de la lógica de la Habitación WebRTC. Lenguaje Java

8.4.4.- Compartir archivos (FrontEnd)

Compartir archivos en tiempo real aporta a la comunicación un recurso muy importante para poner pensamientos, ideas en común entre los usuarios participan en este proceso. Compartir archivos en tiempo real en una misma plataforma de comunicación es una solución muy práctica y eficiente no tener que utilizar otras aplicaciones para el mismo fin ahorrando tiempo, esfuerzo y favoreciendo la experiencia del usuario al utilizar esta única plataforma.

Para compartir ficheros en tiempo real entre dos pares, a nivel técnico se utilizará el motor JavaScript para manipular el fichero, archivo, imagen, sonido,etc. y enviarlo por http al servlet MessageServlet.java (BackEnd) y que servidor Java envíe la información por el websocket (SignalingWebSocket.java) al par final.

En la siguiente imagen se muestra un esquema del la arquitectura de como compartir un archivo en tiempo real:

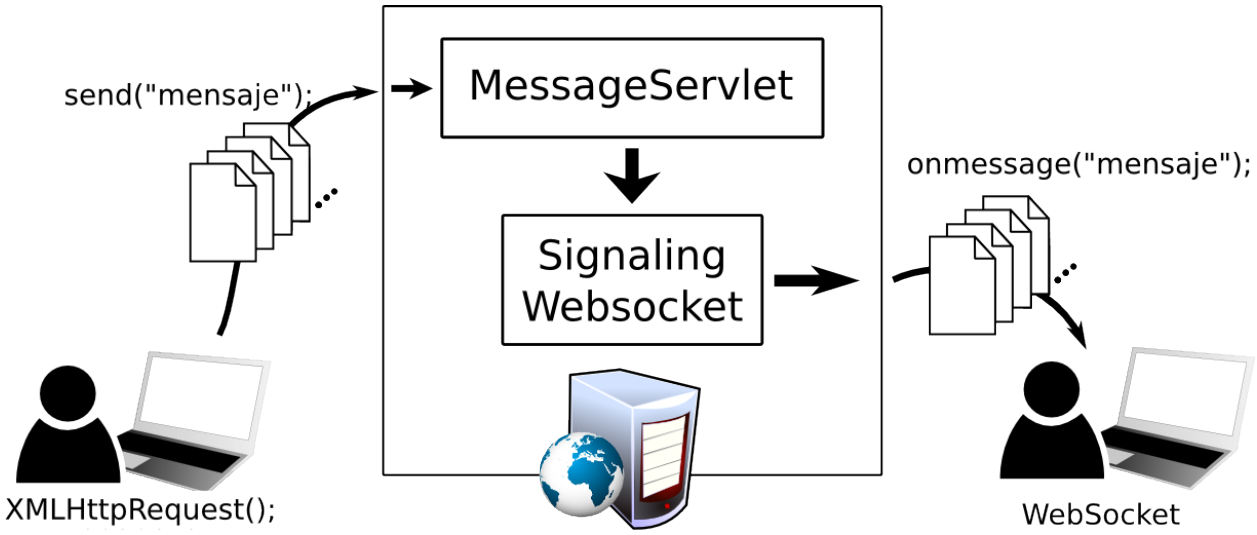

Figura 8.1.- Esquema compartir archivos en tiempo real

Los archivos que se envíen pueden tener diferentes tamaños por lo que se debe limitar el tamaño del archivos a enviar en el lado del usuario que envía el archivo.

Para compartir un archivo no se envía en su totalidad en una único envío, sino que el archivo se tiene que trocear en pedazos (chunk) para enviarlos uno por uno a otro punto y para reconstruirlo en el otro punto para poder leerlo.

Parte técnica

Para compartir archivos se crea un objeto HTML *button* y otro objeto HTML *input* para que el usuario pulse el botón y abra el explorador de archivos, seleccione el archivo para posteriormente manipular y enviar los datos del archivo al otro par:

```
<button id="sendFileOn" title="Compartir archivos"></button>
<input id="sendFile" type="file" value="">
```

Código 8.18.- Objetos HTML `input` y `button` para compartir archivos

Después se debe de poner un *listener* para capturar el evento del botón cuando el usuario lo pulsa y lo gestione el input para abrir el explorador de archivos para que el usuario pueda elegir un archivo. En el siguiente código en JavaScript se muestra como crear esta funcionalidad.

```
// Función para abrir el explorador de archivos del S.O.
document.querySelector("#sendFileOn").addEventListener("click",
function() {
     var clickEvent = document.createEvent('MouseEvents');
     clickEvent.initMouseEvent('click', true, true, window, 0, 0,
0, 0, 0, false, false, false, false, 0, null);

document.querySelector("#sendFile").dispatchEvent(clickEvent);
});

document.getElementById('sendFile').onchange = function() {
    var file = this.files[0];
    fileName = file.name;
    var reader = new window.FileReader()
    reader.onload = onReadAsDataURL;
};
```

Código 8.19.- Ejemplo para obtener y leer un archivo. Lenguaje JavaScript

Cuando el usuario elige un archivo y pulsa abrir en el explorador de archivos, este evento crea un objeto FileReader para que cargue el archivo en memoria y pueda ser manipulado.

El objeto FileReader al cargar (“onload”) el archivo ejecuta la función onReadAsDataURL para manipular y enviar el archivo. En el siguiente código se explica como trocear y enviar el archivo al otro par cuando se ejecuta la función onReadAsDataURL.

```
var fileName;
var chunk = 0;
var maxChunks;
function onReadAsDataURL(event, text) {
     var data = {}; // objeto data para transmitirlo por el canal
          // de mensajes (MessageServlet.java)
     if (event){ // la primera invocación
          chunk = 0;
          text = event.target.result;
          data.maxChunks = Math.ceil(text.length / chunkLength);
          // Trocea el archivo en chunks de 1000 bytes y calcula
```

```
            // el máximo
            maxChunks = Math.ceil(text.length / chunkLength);
            data.fileName = fileName;
            // Pasa el nombre del archivo en el primer evento
        }
        chunk = (100 / maxChunks) + chunk; // Porcentaje completado
        if (text.length > chunkLength) {
        // trocea en un chunk usando el tamaño predefinido para el
        // chunk
            data.message = text.slice(0, chunkLength);
            data.type = "file";
        } else { // ultimo chunk
            data.message = text;
            data.last = true;
            data.type = "file";
        }

        sendMessage(data);

        var remainingDataURL = text.slice(data.message.length);
        if (remainingDataURL.length) setTimeout(function () {
            onReadAsDataURL(null, remainingDataURL);
            // continue transmitting
            }, 500);
        }else{ // Se cancela el envío del archivo
            ...
        }
}
```

Código 8.20.- Función `onReadAsDataURL` para enviar un archivo en tiempo real. Lenguaje JavaScript

En la función onReadAsDataURL(event,text) se puede distinguir 3 bloques principales, el primer bloque

Los eventos posibles al enviar y recibir un archivo pueden ser:

- Se recibe el primer trozo (chunk) del archivo, con el nombre y los primeros datos del archivo. Esto significa que se va ha enviar un nuevo archivo.

- Se recibe el último trozo (chunk) del archivo.

- El usuario que envía el archivo cancela el envío.

- El usuario que recibe el archivo cancela la recepción

En la siguiente tabla de código se muestra como se declaran estos mensajes en Lenguaje JavaScript para recibir y/o enviar un fichero:

```
if(msg.type === 'file'){

      if(msg.maxChunks){ // Evento de envío->recepción
      // Si recibe el atributo maxChunks significa que se va ha
      empezar a recibir un nuevo archivo, también se recibe en este
      primer mensaje el nombre del fichero: "fileName" y el primer
      chunk de datos del archivo :"message"
      }
      if (msg.last) { // Evento de envío->recepción
      // Se recibe El ultimo chunk
      }
      if(msg.cancel){ // Evento de envío->recepción
      // Si el usuario que envía el archivo cancela el envío
      }
      if(msg.cancelRecieveFile){ // Evento de recepción->envío
      // Si el usuario que recibe el archivo cancela la recepción
      }
}
```

Código 8.21.- Eventos al recibir y/o enviar un archivo compartido. Lenguaje JavaScript

Al final de recibir el último trozo del archivo, el archivo se reconstruye y se muestra en la interfaz del Chat un link con los datos del fichero en base64, para que el usuario quien recibe el archivo pueda abrirlo y leerlo o ejecutarlo.

Capítulo 9
Pruebas

9.- Pruebas

En este capítulo se plasman las pruebas realizadas a la aplicación objeto de este proyecto. Se han realizado pruebas QA del software recogiendo el resultado en un documento en forma de tablas donde se describe el funcionamiento de la aplicación. Estas pruebas describen la funcionalidad y calidad del software para evaluar si es válido cada caso de uso recogido. Existen dos tipos de pruebas QA para observar la calidad del software:

- Pruebas de caja blanca (white-box testing): son pruebas realizadas sobre las funciones internas de la aplicación como puede ser la evaluación de una función, módulo, clase, objeto,...

- Pruebas de caja negra (black-box testing): son pruebas que evaluan los requisitos funcionales y no funcionales que tiene que cumplir la aplicación.

9.1.- Resultados

Durante el desarrollo del proyecto se han ido realizado pruebas funcionales y de integración del software mientras se iba programando cada requisito propio de la aplicación, aunque en este documento se recogen las pruebas posteriores al desarrollo para así tener una visión global del proyecto y evaluar la calidad del software de una manera más correcta.

Los datos de las prueba se han recogido en tablas según el propósito de la prueba, los prerrequisitos que tiene que cumplir dicha prueba, los datos de entrada, los pasos para realizar la prueba y la evaluación de los resultados.

Estas pruebas se ha realizado en dos sistemas operativos: Ubuntu 14.04 y Windows 8.1, en los navegadores web y móvil FireFox y Chrome.

Hay que destacar que al realizar las pruebas en el navegador Chrome web y móvil se obtuvieron los mismos resultados de éxito que en FireFox, pero desde la implantación de CORS (origen seguro) de forma restrictiva en el navegador Chrome, la función getUserMedia no puede ser invocada y el servicio WebRTC no puede ejecutarse. Para ejecutar el software en el navegador Chrome tiene que cumplir las siguientes normas recogidas en esta web https://goo.gl/Y0ZkNV

Chrome "Origen seguro": son los orígenes que coincidan con al menos uno de los siguientes patrones (esquema , host, puerto):

- (https , * , *)
- (wss , * , *)
- (* , localhost , *)
- (Chrome - extensión, * , -)
- (* , 127/8 , *)
- (* , :: 1/128 , *)
- (archivo, *, -)

Las pruebas que se han ejecutado son de tipo caja negra y los resultados obtenidos han sido:

CP – 01	**Registrar usuario (Página de Bienvenida)**
Propósito	El usuario tiene que ser registrado en la aplicación. El propósito de esta prueba es asignar al usuario un token único para crear una conexión bidireccional entre los usuarios y la aplicación.
Prerrequisito	- La aplicación tiene que mostrar una Página de Bienvenida con un formulario para que el usuario introduzca su nombre o alias y poder acceder a la aplicación. La aplicación tiene validar el nombre de usuario para que este dato tenga más de 2 caracteres.
Datos de entrada	- Nombre o alias del usuario
Pasos	0.- Se accede a la aplicación por medio de un navegador web. 1.- Se muestra la Página de Bienvenida con un formulario con un campo para introducir el nombre o alias y un botón de envío. 2.- Se escribe el nombre de usuario o alias: 2.1.- Si el nombre de usuario no tiene más de 2 caracteres se muestra un mensaje de información y el botón de envío se deshabilita. 3.- Se pulsa el botón ENTRAR. 4.- La aplicación redirige al usuario a la Sala Principal.
Resultado esperado	- La validación de los datos del usuario. - Redirección a la Sala Principal. - Asignación de token.
Resultado obtenido	Correcto

Tabla 9.1.- CP – 01: Registrar usuario (Página de Bienvenida)

CP – 02	**Listar usuarios (Sala Principal)**
Propósito	El sistema tiene que mostrar una lista de los usuarios conectados a la aplicación.
Prerrequisito	- Los usuarios han tenido que ser registrados y añadidos a la aplicación previamente desde el formulario de la Página de Bienvenida.
Datos de entrada	- Token de usuario asignado por el sistema.
Pasos	1.- El usuario entra en la Sala Principal. 2.- Se lista los usuarios conectados en la aplicación.
Resultado esperado	- Listar los usuarios conectados.
Resultado obtenido	Correcto

Tabla 9.2.- CP – 02: Listar usuarios (Sala Principal)

CP – 03	**Añadir usuario a la lista de usuarios (Sala Principal)**
Propósito	El sistema tiene que mostrar una lista de los usuarios conectados a la aplicación y añadir de forma dinámica a los nuevos usuarios que se conecten al listado de todos los demás usuarios conectados.
Prerrequisito	- Los usuarios tienen tener conexión con la Sala Principal.
Datos de entrada	- Nuevo usuario.
Pasos	1.- El usuario entra en la Sala Principal. 2.- Se lista los usuarios conectados en la aplicación.
Resultado esperado	- Añadir el nuevo usuario a la lista los usuarios conectados.
Resultado obtenido	Correcto

Tabla 9.3.- CP – 03: Añadir usuario a la lista de usuarios (Sala Principal)

CP – 04	**Eliminar usuario a la lista de usuarios (Sala Principal)**
Propósito	El sistema tiene que mostrar una lista de los usuarios conectados a la aplicación y eliminar de forma dinámica a los usuarios que se desconecten o cierren la aplicación.
Prerrequisito	- Los usuarios tienen tener conexión con la Sala Principal.
Datos de entrada	- Salir o cerrar aplicación
Pasos	1.- El usuario A entra en la Sala Principal. 2.- Se lista los usuarios conectados en la aplicación. 3.- El usuario A sale o cierra la aplicación. 4.- A los usuarios conectados a la Sala Principal se elimina de la lista el usuario A.
Resultado esperado	- Eliminar el usuario a la lista los usuarios conectados.
Resultado obtenido	Correcto

Tabla 9.4.- CP – 04: Eliminar usuario de la lista de usuarios (Sala Principal)

CP – 05	**Realizar y notificar llamada (Sala Principal)**
Propósito	El sistema tiene que poder realizar llamadas entre los usuarios.
Prerrequisito	- Los usuarios tienen tener conexión con la Sala Principal. - Tiene que haber más de 1 usuario conectado en la Sala Principal.
Datos de entrada	- Hacer click con el ratón sobre un usuario de la lista.
Pasos	1.- Se listan los usuario conectados a la Sala Principal. 2.- El usuario A pulsa sobre cualquier usuario de la lista o usuario B. 3.- Al usuario A se redirecciona a la Habitación WebRTC. 4.- Al usuario B se notifica que el usuario A le está llamando mostrando su nombre y dos botones para aceptar o rechazar la llamada.
Resultado esperado	- Redireccionar al usuario A que llama a la Habitación WebRTC - Notificar al usuario B llamado que el usuario A le está llamando.
Resultado obtenido	Correcto

Tabla 9.5.- CP – 05: Realizar y notificar llamada (Sala Principal)

CP – 06	**Aceptar o rechazar llamada (Sala Principal)**
Propósito	El sistema tiene que dar la posibilidad de aceptar o rechazar la llamada al usuario que está siendo llamado.
Prerrequisito	- Los usuarios tienen tener conexión con la Sala Principal. - El usuario ha sido llamado por otro usuario conectado a la Sala Principal.
Datos de entrada	- Salir o cerrar aplicación
Pasos	1.- El usuario A ha sido llamado por el usuario B 2.- Al usuario A se le notifica que el usuario B le está llamando mostrándole el nombre del usuario y dos botones para aceptar o cancelar la llamada. 3.- El usuario A acepta o rechaza la llamada. 4.- Si el usuario rechaza la llamada desaparece la notificación de llamada. 5.- Si el usuario A acepta la llamada es redireccionado a la Habitación WebRTC.
Resultado esperado	- Notificar y mostrar dos botones para aceptar o rechazar la llamada. - Si el usuario rechaza la llamada, se oculta la notificación. - Si el usuario acepta la llamada, es redirigido a la Habitación WebRTC.
Resultado obtenido	Correcto

Tabla 9.6.- CP – 06: Añadir usuario a la lista de usuarios

CP – 07	**Realizar videollamada entre pares (Habitación WebRTC)**
Propósito	El sistema tiene que realizar videollamadas entre pares
Prerrequisito	- El usuario A llama al usuario B en la Sala Principal. - El usuario A es redirigido a la Habitación WebRTC. - El usuario B acepta la llamada y es redirigido a la Habitación WebRTC. - El usuario A y B comparten los recursos de la máquina.
Datos de entrada	- Token formado por la concatenación de los tokens de usuarios en la Sala Principal para crear una nueva Habitación WebRTC, asignado por el sistema. - Se asignan nuevos tokens a los dos usuarios para la videollamada.
Pasos	1.- Los usuarios A y B han sido redirigidos a la Habitación WebRTC. 2.- Los usuarios A y B comparten los recursos de la máquina.
Resultado esperado	- Se realiza la videollamada - Se adapta la interfaz para la videollamada.
Resultado obtenido	Correcto

Tabla 9.7.- CP – 07: Realizar videollamada entre pares (Habitación WebRTC)

CP – 08	**Compartir archivos (Habitación WebRTC)**
Propósito	El sistema tiene que poder compartir archivos entre pares.
Prerrequisito	- La videollamada entre el par de usuarios tiene que estar activa.
Datos de entrada	- Fichero para compartir.
Pasos	1.- El usuairo A pulsa el botón compartir archivo. 2.- Se abre el explorador de archivos y el usuario A elige un archivo. 3.- El sistema empieza a compartir el archivo con el usuairo B. 4.- El usuario A y B es notificado de que se está compartiendo un archivo y muestra una barra de progreso y un botón para cancelar la transferencia del archivo. 5.- Cuando termina la transferencia, los usuarios son notificados y la interfaz vuelve a su estado original. 6.- Al usuario B que recibe el archivo, se le muestra en la interfaz un link para poder descargar/construir el archivo en su máquina.
Resultado esperado	- Se envía el archivo. - Hay un cambio en la interfaz para notificar la transferencia y un botón para cancelar la operación.
Resultado obtenido	Correcto

Tabla 9.8.- CP – 08: Compartir archivos (Habitación WebRTC)

CP – 09	**Cancelar la transferencia al compartir un archivo (Habitación WebRTC)**
Propósito	El sistema tiene que poder compartir archivos entre pares y poder cancelar su transferencia.
Prerrequisito	- La videollamada entre el par de usuarios tiene que estar activa. - Tiene que estar activa la operación de compartir archivos.
Datos de entrada	- Fichero para compartir.
Pasos	1.- El proceso de compartir un archivo está activo. 2.- El usuario A o B pulsa el botón cancelar la transferencia. 3.- El proceso se detiene y no se comparte el archivo. 4.- La interfaz vuelve a su estado original.
Resultado esperado	- Se cancela la transferencia del archivo compartido. - Hay un cambio en la interfaz para notificar la cancelación de la operación.
Resultado obtenido	Correcto

Tabla 9.9.- CP – 09: Cancelar la transferencia al compartir un archivo (Habitación WebRTC)

CP – 10	**Pantalla completa On/Off (Habitación WebRTC)**
Propósito	El sistema tiene que adaptar la interfaz a selección del usuario en pantalla completa activada o desactivada
Prerrequisito	- Ninguno
Datos de entrada	- Ninguno
Pasos	1.- Se pulsa sobre el botón pantalla completa On 2.- La interfaz se adapta a pantalla completa On 3.- Se pulsa otra vez sobre el botón pantalla completa off. 4.- La interfaz se adapta a pantalla completa Off
Resultado esperado	- La interfaz se adapta a Pantalla Completa On/Off
Resultado obtenido	Correcto

Tabla 9.10.- CP – 10: Pantalla completa On/Off (Habitación WebRTC)

CP – 11	**Hacer foto de la videollamada (Habitación WebRTC)**
Propósito	El sistema tiene que poder hacer fotos de la videollamada.
Prerrequisito	- La videollamada entre el par de usuarios tiene que estar activa.
Datos de entrada	- La videollamada.
Pasos	1.- Se pulsa sobre el botón hacer foto. 2.- El sistema toma la imagen de la videollamada, hace la foto y la muestra en la interfaz.
Resultado esperado	- Creación de una foto de la imagen de la videollamada
Resultado obtenido	No correcto Nota: Actualmente esta funcionalidad está desarrollada en el caso de que no está la videollamada activa; en este caso se realiza la foto del usuario que la hace de forma local.

Tabla 9.11.- CP – 11: Hacer foto de la videollamada (Habitación WebRTC)

CP – 12	**Grabar videollamada (Habitación WebRTC)**
Propósito	El sistema tiene que poder grabar las videollamadas en un archivo de vídeo y audio.
Prerrequisito	- La videollamada entre el par de usuarios tiene que estar activa.
Datos de entrada	- La videollamada.
Pasos	1.- Se pulsa sobre el botón grabar videollamada. 2.- La interfaz del botón cambia a estado “grabando”. 3.- Se vuelve a pulsar el botón de grabar videollamada. 4.- Se crea un archivo de vídeo/audio y se descarga automáticamente en el ordenador.
Resultado esperado	- Creación de un vídeo con la imágenes y audio de la videollamada.
Resultado obtenido	No correcto Nota: Actualmente esta funcionalidad está desarrollada en el caso de que no está la videollamada activa; en este caso se graba el vídeo y audio de la máquina local y se crea el vídeo.

Tabla 9.12.- CP – 12: Grabar videollamada (Habitación WebRTC)

CP – 13	**Audio On/Off (Habitación WebRTC)**
Propósito	El sistema tiene que poder activar o desactivar el audio de la videollamada.
Prerrequisito	- La videollamada entre el par de usuarios tiene que estar activa.
Datos de entrada	- El audio de la videollamada
Pasos	1.- Se pulsa sobre el botón de audio On. 2.- Se activa el audio. 3.- Se vuelve a pulsar el botón de audio Off. 4.- Se desactiva el audio.
Resultado esperado	- Se activa y desactiva el audio de la videollamada
Resultado obtenido	Correcto

Tabla 9.13.- CP – 13: Audio On/Off (Habitación WebRTC)

CP – 14	**Chat On/Off (Habitación WebRTC)**
Propósito	El sistema tiene que poder ocultar y mostrar la interfaz del Chat.
Prerrequisito	- Ninguno
Datos de entrada	- Ninguno
Pasos	1.- Se pulsa sobre el botón de Chat On. 2.- Se muestra el Chat. 3.- Se vuelve a pulsar el botón de Chat Off. 4.- Se oculta el chat.
Resultado esperado	- Se oculta y muestra la interfaz del Chat.
Resultado obtenido	Correcto

Tabla 9.14.- CP – 14: Chat On/Off (Habitación WebRTC)

CP – 15	**Enviar, recibir y listar mensajes por Chat (Habitación WebRTC)**
Propósito	El sistema tiene que poder enviar, recibir y listar los mensajes enviados por Chat, en la interfaz del Chat se tiene que mostrar la conversación entre los dos usuarios.
Prerrequisito	- La videollamada entre el par de usuarios tiene que estar activa.
Datos de entrada	- Mensajes de texto.
Pasos	1.- El usuario A escribe un mensaje de texto en el Chat y pulsa el botón enviar o *enter* para el usuario B. Se muestra el mensaje enviado en la conversación del Chat. 2.- El usuario B recibe el mensaje y lo muestra en la conversación del Chat.
Resultado esperado	- Se envía, recibe y lista los mensajes del Chat.
Resultado obtenido	Correcto

Tabla 9.15.- CP – 15: Enviar, recibir y listar mensajes por Chat (Habitación WebRTC)

CP – 16	**Volver a la Sala Principal (Habitación WebRTC)**
Propósito	El sistema tiene que poder reenviar al usuario a la Sala Principal sin tener que pasar por la Página de Bienvenida.
Prerrequisito	- La videollamada entre el par de usuarios tiene que estar activa.
Datos de entrada	- Ninguno.
Pasos	1.- El usuario pulsa el botón “Volver a la Sala Principal” 2.- El usuario es redirigido a la Sala Principal con su nombre de usuario.
Resultado esperado	- El usuario es redirigido a la Sala Principal con su nombre de usuario.
Resultado obtenido	Correcto

Tabla 9.16.- CP – 15: Volver a la Sala Principal (Habitación WebRTC)

Capítulo 10
Manual de instalación y uso

10.- Manual de instalación y uso

10.1.- Manual de instalación y despliegue de la aplicación en el servidor web público

En este capitulo se explica como desplegar la aplicación en un servidor web público para que se pueda disfrutar de la aplicación en todo el mundo con acceso a internet y un navegador web. En este caso la aplicación se ha desplegado en un servidor Tomcat7 (Jboss) alojado en la plataforma web Openshift.com con un simple archivo war.

10.1.1.- Crear el archivo WAR de la aplicación

Para crear el archivo *<nombre_de_la_aplicacion>.war* con el IDE Eclipse se selecciona la carpeta raíz del proyecto en el explorador del IDE, se pulsa en *File* en la barra principal para que muestre un menú desplegable y pulsar sobre la opción *Export*.

El IDE Eclipse abrirá una ventana con todos los tipos de formatos posibles de exportación, se selecciona la opción *Web/WAR file* y se pulsa siguiente. Posteriormente se selecciona la carpeta de destino para guardar el archivo *.war y se pulsa sobre el botón *Finish*.

10.1.2.- OpenShift

OpenShift es una plataforma de Red Hat, es una plataforma como servicio (**PaaS**) que permite a los desarrolladores realizar aplicaciones rápidamente, desplegarlas, y escalarlas en un entorno de nube. OpenShift permite configurar los servidores según las necesidades que requiera la aplicación, se puede abrir varios puertos como el 8000 o 8443 para utilizar websockets, desplegar archivos *.war solo con depositarlos en la carpeta raíz, se pueden añadir clases, servlets, etc. OpenShift tiene varios planes de pago aunque se puede optar por un plan gratuito, en este caso se ha optado por un plan gratuito con ciertas limitaciones, pero para el desarrollo de la aplicación es suficiente.

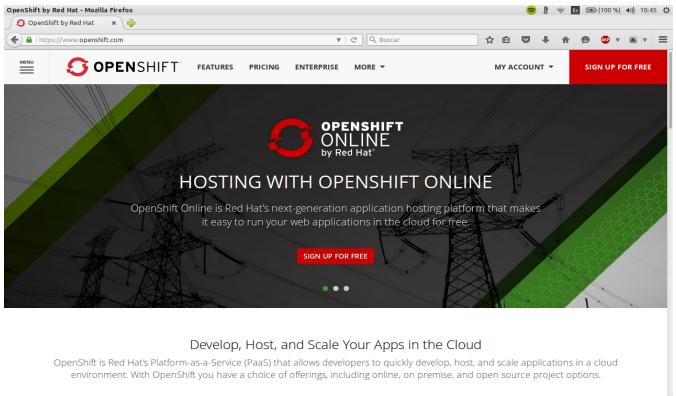

Figura 10.1.- Captura de pantalla de la página de inicio de OpenShift.com

Para crear una cuenta en la plataforma OpenShift es tan sencillo como registrase en su panel web por medio de un formulario sencillo y seguir los pasos del registro.

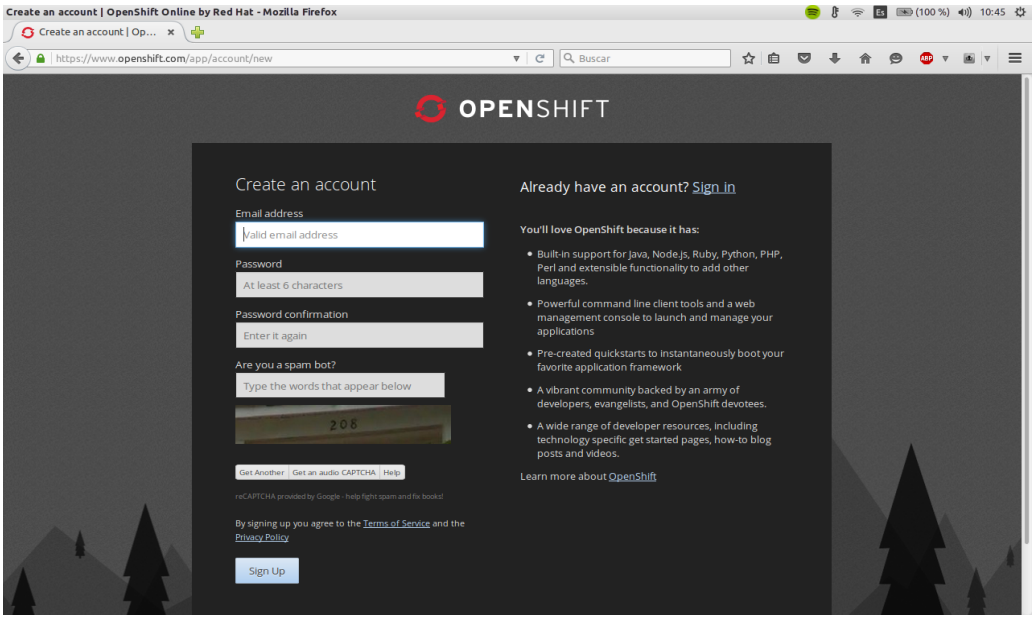

Figura 10.2.- Captura de pantalla del formulario de registro de OpenShift

Cuando se haya realizado el registro en la plataforma y obtengamos nuestras credenciales, se entra con ellas y accedemos al panel de control de la plataforma.

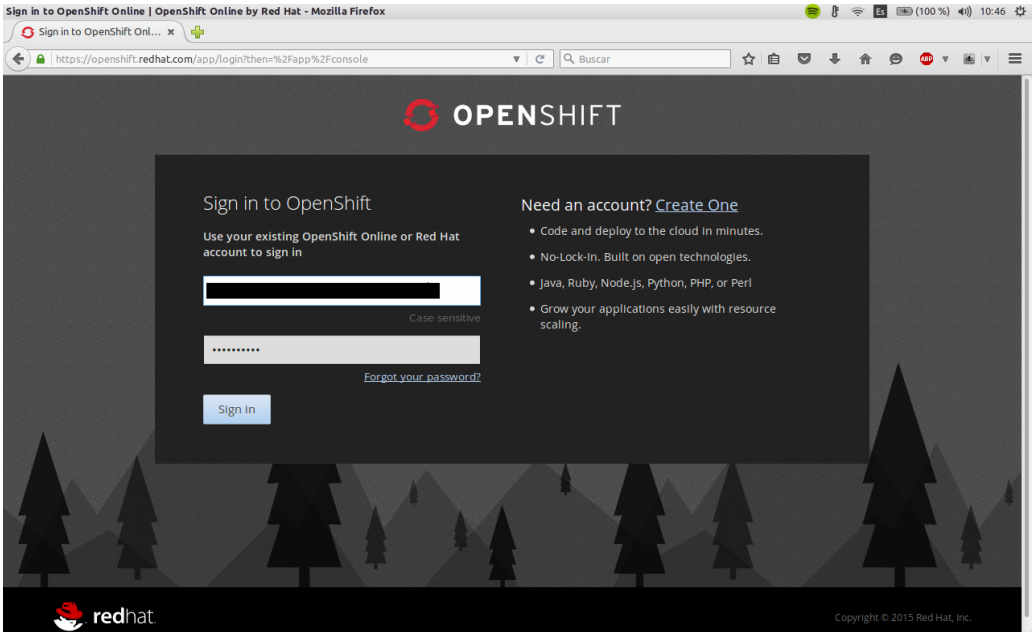

Figura 10.3.- Captura de pantalla del formulario de acceso de OpenShift

El panel de control es muy sencillo y solamente tenemos que añadir una aplicación para empezar a utilizar . Cuando añadimos una aplicación en OpenShift no muestra una lista de servidores a elegir para crearlo y así poder ejecutar nuestra aplicación. En este caso se ha optado por la elección de un servidor Tomcat7(JBoss) que la plataforma nos ofrece.

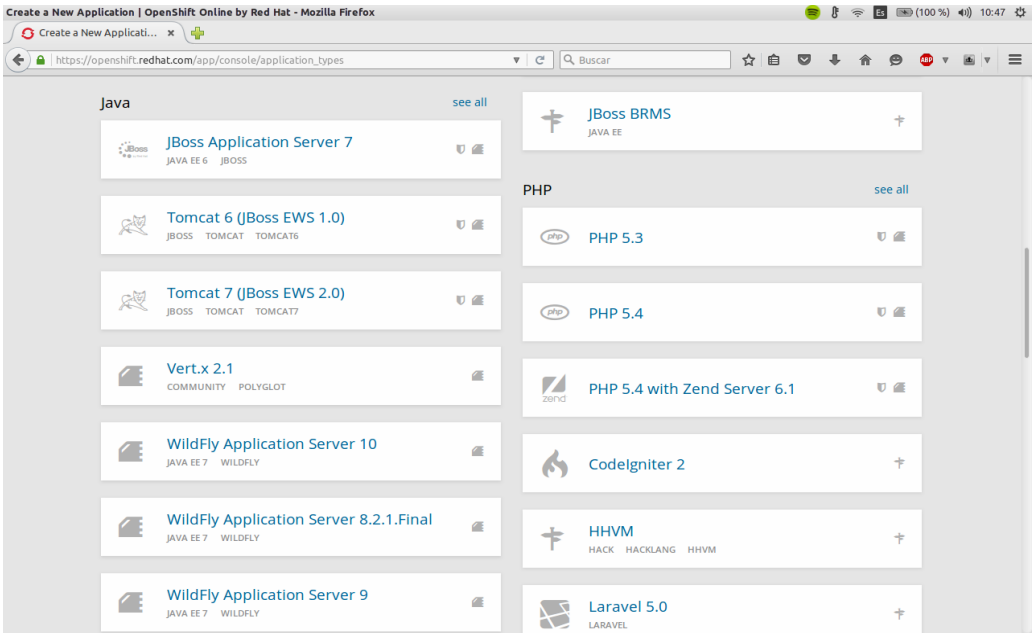

Figura 10.4.- Captura de pantalla del listado de servidores operativos en OpenShift

Cuando se elige un servidor y las herramientas que queramos integrar en él nos pedirá que creemos un nombre de dominio para poder ejecutar nuestra aplicación por medio de un navegador web con http. En este caso la aplicación motivo de este trabajo está alojada en un subdominio gratuito en la dirección http://webrtc-jvrodrigo.rhcloud.com/webrtc/.

Herramienta Git

Para realizar las subidas o cometidas en el servidor del código de la aplicación para su despliegue, se utilizará la herramienta *"git"* consola linux. Primero hay que instalar la herramienta *"git"* con una simple linea de comandos en la consola:

```
sudo apt-get update
sudo apt-get install git
```

Código 10.1.- Comandos por consola Ubuntu para instalar Git

Cuando la herramienta *"git"* esté instalada, se puede comprobar escribiendo git en la consola de comandos para que muestre las ordenes validas de este software. En este proyecto se ha utilizado las ordenes:

```
git clone
git add --all
git commit -a -m 'publicando en OpenShift'
git push
```

Código 10.2.- Comandos básicos de Git

OpenShift permite el despliegue de aplicaciones de integración continua por medio de Jenkins subiendo los ficheros del código de la aplicación con la herramienta *git*. En este punto existen varias maneras para subir los ficheros de nuestra aplicación al servidor, se puede subir mediante un repositorio ya creado en "GitHub" y de esta manera poder realizar cambios desde el repositorio, para esto es necesario que el proyecto contenga el archivo "pom.xml" para que OpenShift localice las dependencias de la aplicación y poder compilarlo en el servidor. Si no se elige la opción de utilizar un repositorio de *"git"* se puede desplegar la aplicación de dos formas más.

OpenShift crea un repositorio en el servidor de la aplicación creada el cual puede se clonado en nuestro disco duro y de esta manera poder realizar cambios en la aplicación de manera local. Para clonar el repositorio en nuestro disco OpenShift ofrece una dirección por "ssh" para introducir por consola o terminal con el parámetro *"git clone"*, al hacer este paso la consola pedirá unas credenciales para poder acceder También se puede optar por solo subir solo el fichero *.war donde la aplicación está comprimida para que el servidor Java pueda desplegarlo. Si se opta por subir solo el fichero *.war hay que clonar el repositorio, dejar solo la carpeta *"src"* y configurar el archivo *"server.xml"*.

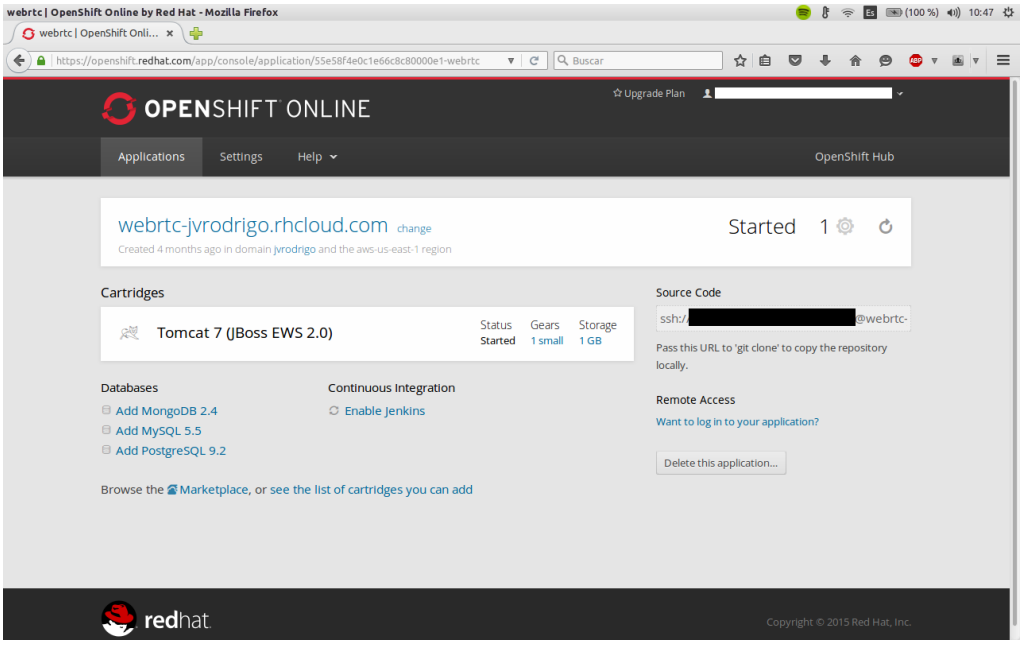

Figura 10.5.- Captura de pantalla de la aplicación en OpenShift

En este caso se clona la aplicación en el disco local para subir la aplicación en un fichero *.war, y configuraremos el fichero "server.xml" para que el servidor despliegue automáticamente ficheros *.war. Se clona la aplicación con el comando:

```
git clone <url>
```

Código 10.3.- Comando Git clone

Y luego en la carpeta ".openshift/config" se modifica el archivo *"server.xml"* y se cambia el valor de la variable *unpackWARs* de la etiqueta <*Host*> y se pone a valor "true" quedando de esta manera:

```
<Host name="localhost"  appBase="webapps"
      unpackWARs="true" autoDeploy="true">
```

Código 10.4.- Configuración del archivo *server.xml*

La estructura de la aplicación en el disco tiene la siguiente estructura después de clonara en el disco. El archivo *.war de la aplicación se alojará dentro de la carpeta *"<nombre_de_la_aplicacion>/webapps/*.war"*.

```
.
|-- <nombre_de_la_aplicacion>
|    -- .git
|    | |-- ...
|    ...
|    -- .openshift
|    |    |-- action_hooks
|    |    |    |-- README.md
|    |    |-- config
|    |    |    |...
|    |    |    |-- catalina.policy
|    |    |    |-- catalina.properties
|    |    |    |-- context.xml
|    |    |    |-- server.xml
|    |    |    |-- web.xml
|    |    |...
|    |-- README.md
|    |-- webapps
|         |-- <nombre_de_la_aplicacion>.war
```

Figura 10.6.- Diagrama de la estructura de los directorios de la aplicación de OpenShift

Cuando el archivo *<nombre_de_la_aplicacion>.war* este copiado a la carpeta local donde se ha clonado la aplicación ya solo queda subir el archivo por medio de la consola linux con la herramienta *"git"* conectándonos por ssh a nuestra plataforma OpenShift. La plataforma OpenShift ofrece una url ssh para la conexión segura a nuestra aplicación, esta se encuentra en el panel de administración de la aplicación.

Herramienta Client Tool

La herramienta Client Tool sirve para crear y administrar aplicaciones con OpenShift online, en este caso esta herramienta servirá para comprobar si el puerto 8000 del servidor está abierto y así poder utilizar websockets en nuestro servidor. Para instalar Client Tool por consola linux se realiza escribiendo este comando:

```
sudo apt-get install rhc
```

Código 10.5.- Comando de instalación de la herramienta Client Tool

Para ver los puertos abiertos de la aplicación se escribe el siguiente comando por consola:

```
rhc port-forward -a <nombre_de_la_aplicacion>
```

Código 10.6.- Comando para ver los puertos abiertos en OpenShift

Pedirá una contraseña para poder acceder a la aplicación, se introduce la contraseña de la cuenta de OpenShift y mostrará los puerto abiertos.

```
Password: **********
Checking available ports ... done
Forwarding ports ...

To connect to a service running on OpenShift, use the Local
address

Service Local               OpenShift
------- ------------- ---- -----------------
java    127.0.0.1:8000  =>  127.5.78.129:8000
java    127.0.0.1:8080  =>  127.5.78.129:8080

Press CTRL-C to terminate port forwarding
```

Código 10.7.- Vista consola linux, puertos abiertos en OpenShift

Acceso al servidor OpenShift (SSH)

Para tener acceso al servidor de la aplicación en OpenShift y así poder ver los los directorios, logs del servidor, los archivos de configuración, etc. se puede acceder por consola linux o putty (windows) escribiendo el comando ssh seguido de la clave que OpenShift asigna a la aplicación:

```
ssh XXXXXXXXXXXXXXXXXXXXX@webrtc-jvrodrigo.rhcloud.com
```

Código 10.8.- Comando de acceso a la aplicación en OpenShift por consola linux

La estructura de los directorios del servidor es muy grande y se tendrá atención donde la aplicación esta desplegada y donde se encuentra el archivo log del servidor para observar las trazas.

```
.
|-- app-deployments
...
|-- app-root
|   |--...
|   |-- logs
|   |--...
|   |-- runtime
|      |...
|       -- dependencies
|      |   |-- jbossews
|      |       |-- webapps
|      |           |-- <nombre_de_la_aplicacion>
|      |               ...
|      |...
|-- git
|   |-...
|-- jbossews
    |--...
    |-- logs
    |--...
    |-- webapps -> /var/lib/openshift/55eXXXXXXXXX000e1/app-
root/runtime/dependencies/jbossews/webapps
    |--...
```

Código 10.9.- Estructura de los directorios de la aplicación en el servidor OpenShift

Al log de la aplicación para ver las trazas, puede acceder por el comando *"tail"* y para eliminar el archivo se puede utilizar el comando *">"*

```
tail -f ~/app-root/logs/jbossews.log
> ~/app-root/logs/jbossews.log
```

Código 10.10.- Comando para ver las trazas de la aplicación en el servidor OpenShift por consola Ubuntu

10.2.- Manual de usuario

10.2.1.- Introducción

El siguiente documento muestra el manual de uso de la aplicación. Esta guía se compone de tres partes fundamentales: Página de Inicio, Sala Principal y Habitación WebRTC.

10.2.2.- Pagina de Inicio

La Página de Inicio da la bienvenida al usuario para comenzar a utilizar la aplicación. La Página de Inicio está compuesta por:

1.- Nombre de usuario: Campo de texto para introducir el nombre de usuario o alias. El nombre de usuario debe que tener más de 2 caracteres para que se active el botón "ENTRAR".

2.- Botón "ENTRAR": Botón para acceder a la Sala Principal. El botón ENTRAR se habilita si se cumple la condición: nombre de usuario con más de 2 caracteres.

3.- Mensaje de notificación: El mensaje de notificación se oculta si se cumple la condición: : nombre de usuario con más de 2 caracteres. En el caso contrario se muestra el mensaje: El nombre de usuario debe de tener más de 2 caracteres.

(nota: no es necesario registrarse en la aplicación para disfrutarla, la aplicación no exige email ni contraseña, solo hay que introducir cualquier nombre de usuario o alias que tenga más de 2 carácteres para acceder a la aplicación). Puede existir el mismo nombre de usuario en una misma sesión, el nombre de usuario no es significativo.

En la siguiente captura de pantalla se ilustra la Página de Inicio y los elementos que la componen:

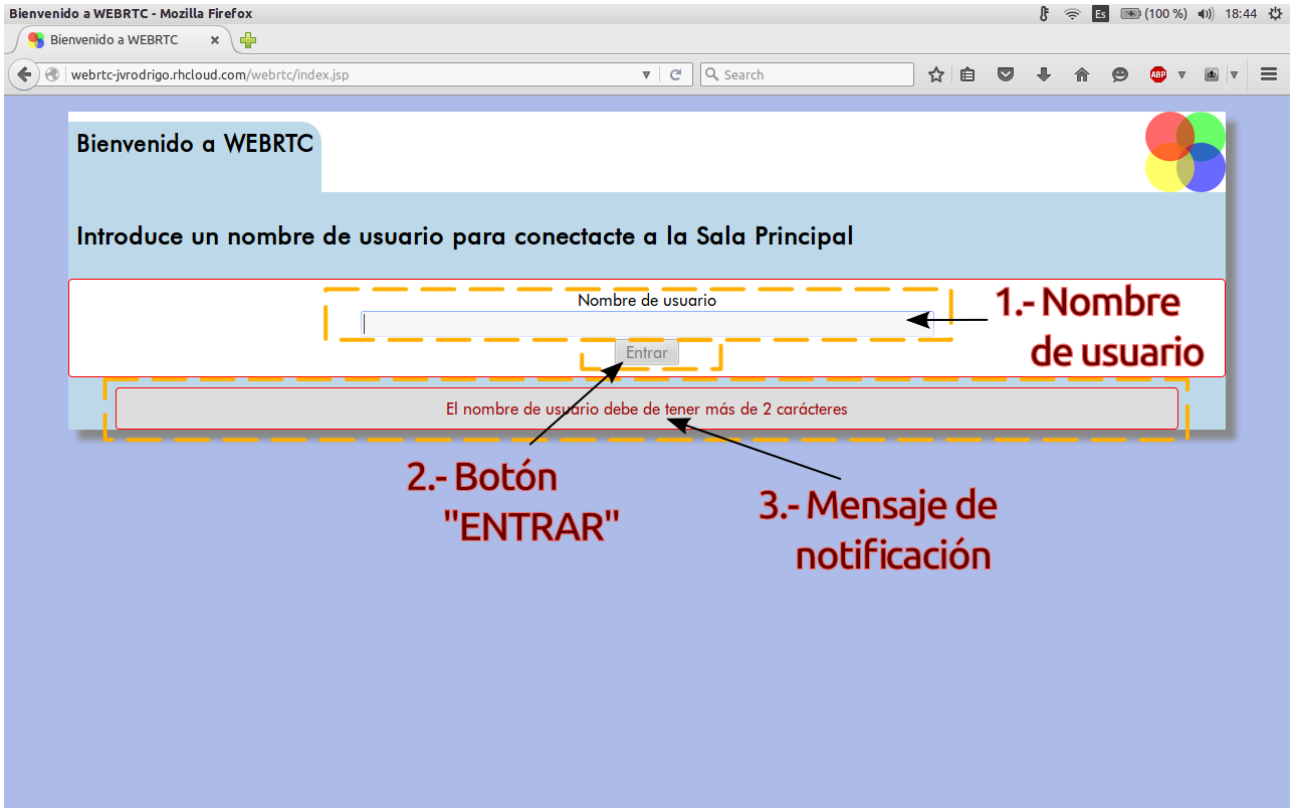

Figura 10.6.- Manual de usuario: Página de Inicio

10.2.3.- Sala Principal

La Sala Principal es la zona donde se lista los usuarios conectados a la aplicación para realizar llamadas entre pares. La Sala Principal está compuesta por los siguientes elementos:

1.- Tu nombre de usuario: Nombre de usuario entrado en la Página de Inicio.

2.- Lista de usuarios: Lista de usuarios conectados a la aplicación.

3.- Usuario disponible: El ítem de usuario conectado a la aplicación.

4.- Área de notificación de llamada: Es el área de notificación de llamada, este área se muestra cuando el usuario ha recibido una llamada.

5.- Nombre de usuario llamada entrante: Nombre de usuario que te está llamando.

6.- Aceptar llamada: Botón para aceptar la llamada entrante.

7.- Rechazar llamada: Botón para rechazar la llamada entrante.

8.- Volver atrás: Botón para volver a la Página de Inicio.

En la siguiente captura de pantalla se ilustra la Sala Principal y los elementos que la componen:

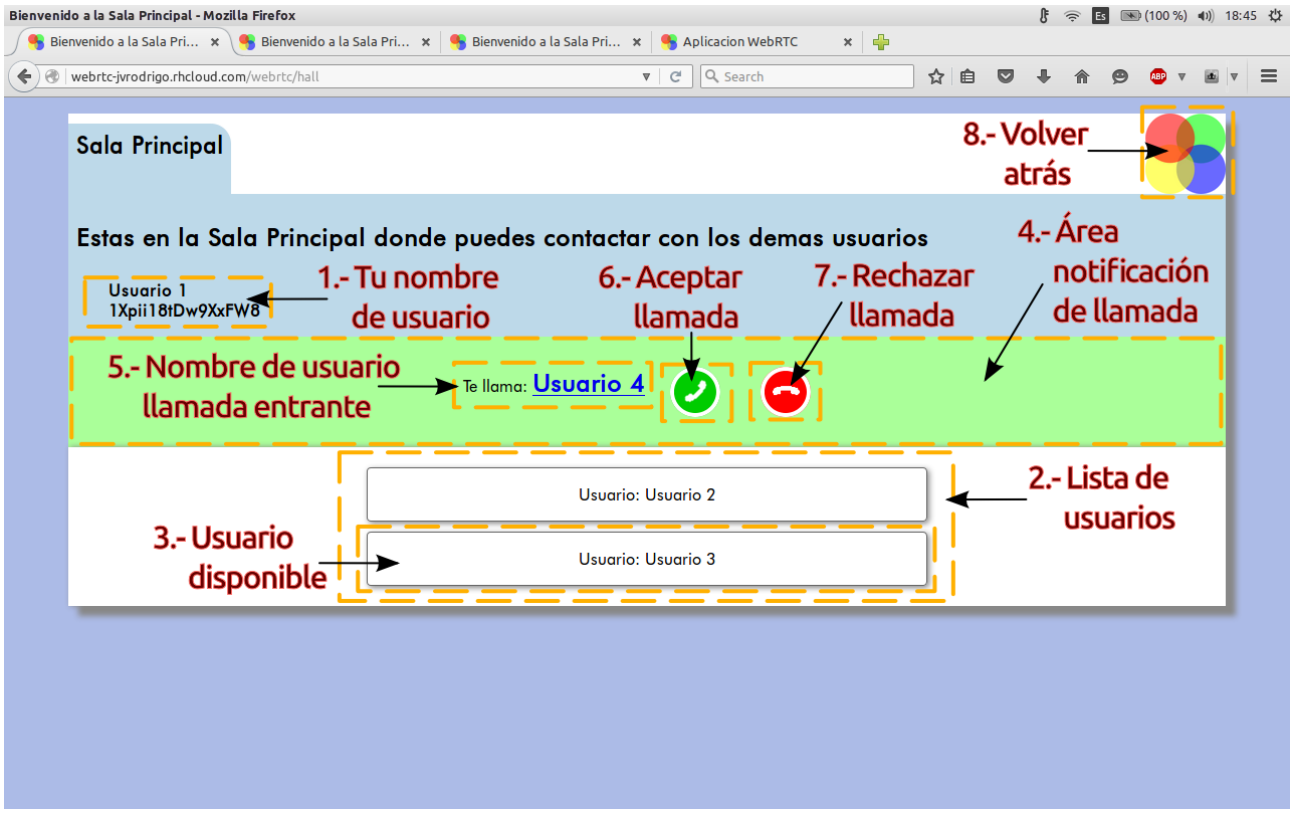

Figura 10.7.- Manual de usuario: Sala Principal

La llamada entre pares puede ser satisfecha de dos formas:

- Llamar a un usuario: Para llamar a un usuario hay que pulsar o hacer click sobre el ítem de usuario (3.- Usuario disponible) en la lista de usuarios (4.- Lista de usuarios).

- Recibir llamada: Al recibir una llamada, el área de notificación (4.- Área de notificación) se mostrará, notificando al usuario que está siendo llamada. Para aceptar una llamada hay que pulsar o hacer click sobre el botón Aceptar llamada (5.- Aceptar llamada) de color verde. Para cancelar la llamada hay que pulsar o hacer click sobre el botón Rechazar llamada (6.- Rechazar llamada).

10.2.4.- Habitación WebRTC

La Habitación WebRTC es la zona donde se realiza la videollamada entre pares. Es necesario tener un dispositivo de vídeo conectado al ordenador para utilizar la Habitación WebRTC. La Habitación WebRTC está compuesta por los siguiente elementos:

1.- Zona Videollamada: Zona activa donde se visualiza la vídeo cam propia y la vídeo llamada entre pares cuando está activa.

2.- Menú GUI: Menú de opciones y funcionalidades. El Menú GUI puede ser desplazado por toda la pantalla.

3.- Pantalla completa On/Off: Botón pantalla completa activa o desactivada.

4.- Hacer foto: Botón para hacer foto.

5.- Grabar vídeo: Botón para grabar la vídeo llamada.

6.- Compartir archivo: Botón para compartir archivos.

7.- Sonido On/Off: Botón activar/desactivar audio.

8.- Chat On/Off: Botón mostrar/ocultar Chat.

9.- Chat: Zona de Chat. El Chat puede ser desplazado por toda la pantalla.

10.- Conversación: Zona donde se muestra los mensajes enviados y recibidos. En esta zona también se muestra los links de descarga de los archivos compartidos.

11.- Mensaje a enviar: Campo de texto para escribir mensajes para enviar por Chat.

12.- Botón Enviar: Botón para enviar un mensaje. También se puede enviar mensajes por Chat pulsando ENTER.

13.- Opacidad Chat: Botones para subir y bajar la opacidad del Chat.

14.- Cerrar Chat: Botón para ocultar el Chat.

15.- Zona espejo foto: Zona donde se muestra la foto al pulsa el botón hacer foto (3.- Hacer foto).

16.- Zona espejo vídeo: Zona donde se muestra el vídeo propio al realizar la videollamada.

17.- Volver a la Sala Principal: Botón para volver a la Sala Principal.

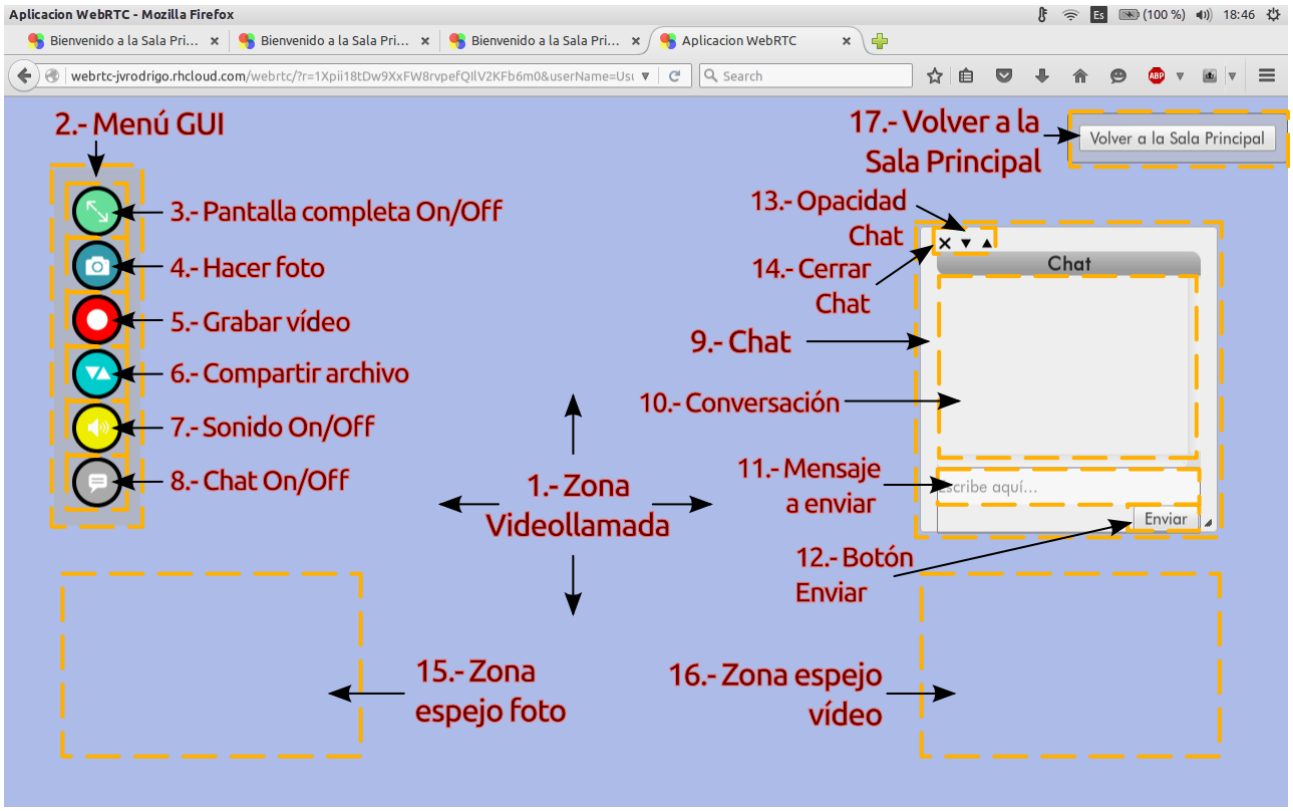

Figura 10.8.- Manual de usuario: Habitación WebRTC

Al entrar en la Habitación WebRTC el navegador web pedirá compartir los recursos de la máquina. Para compartir los recursos de vídeo y audio del sistema solamente hay que pulsar el botón "*Compartir los dispositivos seleccionados*" o "*Compartir*" (Firefox). En las siguiente figuras se muestra una captura de pantalla para permitir los recursos de la máquina en el navegador versión escritorio y móvil

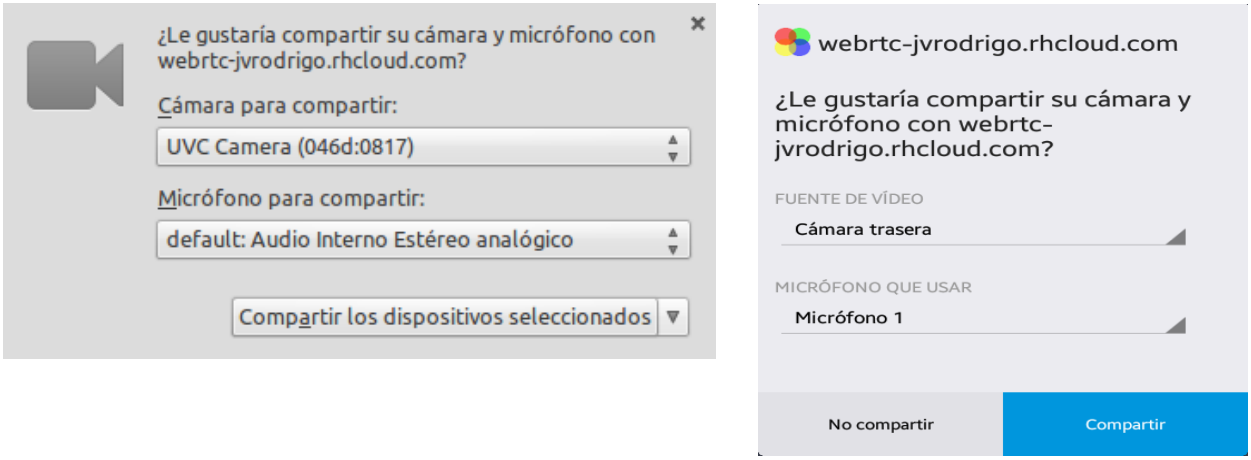

Figura 10.9.- Manual de usuario: Compartir dispositivos. Versión navegador de escritorio y móvil (Firefox)

Si la habitación donde se intenta acceder ya está ocupada por dos usuarios se muestra la página Habitación Ocupada. En la siguiente figura se muestra una captura de pantalla de la Habitación Ocupada.

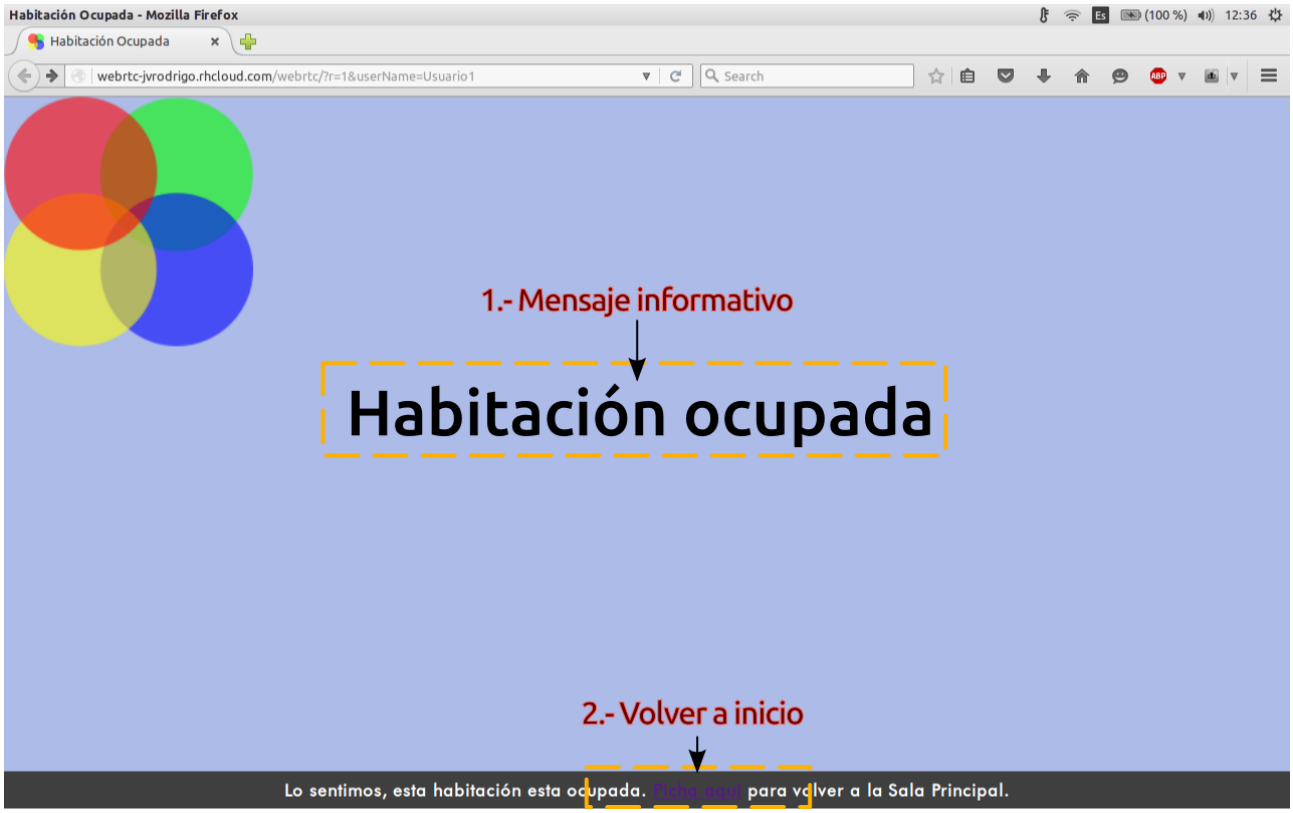

Figura 10.10.- Manual de usuario: Habitación Ocupada

La página Habitación Ocupada tiene varios elementos:

1.- Mensaje informativo: Mensaje informativo de la situación de la Habitación WebRTC.

2.- Volvel a inicio: Botón para volver a la Página de Incio de la aplicación.

Capítulo 11
Conclusiones y trabajos futuros

11.- Conclusiones y trabajos futuros

11.1.- Conclusiones

La tecnología **WebRTC** permite la comunicación en tiempo entre pares y entre múltiples usuarios, compartir recursos e integrarse fácilmente en cualquier sistema con una arquitectura cliente-servidor. Brinda al programador las herramientas necesarias para acceder los recursos de la máquina y realizar aplicaciones de conexión punto a punto y/o en broadcast entre usuarios. Desde el punto de vista del usuario esta tecnología es transparente y muy usable puesto que no se tienen que instalar plugins ni programas adicionales, solo necesita un navegador o una aplicación compatible con WebRTC.

La implantación de esta tecnología en arquitecturas PaaS es muy sencilla y su implementación en distintos lenguajes de programación es irrelevante porque el proceso de señalización es el mismo para configurar los servicios de WebRTC. WebRTC se puede integrar en plataformas Node.js, Java, Python,.NET,... en cualquier tecnología que permita la integración de Sockets.

A la aplicación desarrollada en este proyecto se le pueden añadir varias mejoras como:

- Reducción de ruido y acoplamiento del canal.

- Añadir compartir escritorio remoto en tiempo real entre pares de usuarios.

- Añadir comunicación múltiple entre múltiples usuarios en tiempo real, videollamadas entre múltiples usuarios.

- Añadir una pizarra interactiva para compartir entre pares o múltiples usuarios en tiempo real.

- Añadir persistencia de datos para integrar múltiples servicios adicionales.

11.2.- Trabajos futuros

El código fuente de **WebRTC** puede ser descargado desde: https://webrtc.org/native-code/. Con el código fuente se pueden desarrollar aplicaciones Android e iOS, el código se puede clonar con la herramienta *git*, se puede modificar y compilar. Un trabajo futuro sería desarrollar **la aplicación nativa en Android e iOS** apuntando a nuestros servicios para integrarlo con la versión web de este proyecto.

Otro trabajo futuro que se podría desarrollar es la integración de la tecnología *OpenCV* para el procesado de imágenes y vídeos con Java y/o con JavaScript (librerías similares). OpenCV permite dotar de visión artificial a nuestra aplicación.

Posibles trabajos futuros con WebRTC:

- **Sistema de videovigilancia:** Con WebRTC y OpenCV (o una librería similar) se puede desarrollar un sistema de videovigilancia con software libre de licencia pública GNU (GPL), que realice las funciones que se quieran programar en el sistema como activar la alarma cuando detecte movimiento o algún sonido, grabar vídeos, hacer fotos, etc. WebRTC se puede aplicar en este tipo de negocio o en negocios similares.

- **Sistemas interactivos de realidad aumentada**: La realidad aumentada junto WebRTC tienen un gran potencial para desarrollar aplicaciones software que den soluciones a muchos campos y profesiones como en medicina (estudio de proteínas, virus, anatomía,...), en arqueología, en ingenierías, sistemas simulados, etc. Como WebRTC está integrado en navegadores móviles puede acceder a los recursos del dispositivo y hacer uso de ellos como el geolocalizador, acelerómetro, giroscopio, magnetómetro, etc.

- **SmartCity**: La tecnología WebRTC junto OpenCV (o una librería similar) y una infraestructura de cámaras o dispositivos móviles que poseen más recursos instalados para este propósito, puede dotar a una ciudad de inteligencia artificial para ofrecer servicios e información relevante a sus ciudadanos y/o a todo el mundo en tiempo real. Por ejemplo puede informar de los aparcamientos libres, donde hay un atasco, o gestionar el encendido y apagado del alumbrado público de la ciudad. Ni decir tiene que esto plantea un debate sobre la privacidad del individuo. Creo que la tecnología WebRTC y los servicios que las SmarCities nos puede proporcionar son muy diversos; en el futuro serán parte de nuestras ciudades y de nuestras vidas.

Bibliografía y referencias

Bibliografía y referencias

1) ***WebRTC 1.0**: Real-time Communication Between BrowsersEditor's Draft*. W3C Working Draft 31 May 2016. URL: https://www.w3.org/TR/webrtc/. **© 2011-2016 W3C® (MIT, ERCIM, Keio, Beihang)**.

2) *Official **WebRTC** Web Home:* http://www.webrtc.org/. **Google**.

3) ***SIP** APIs for Voice and Video Communications on the **Web***. 30 Jun 2011. URL: http://arxiv.org/pdf/1106.6333.pdf. **Carol Davids, Alan Johnston, Kundan Singh, Henry Sinnreich, Wilhelm Wimmreuter**.

4) *Taking on **WebRTC** in an enterprise.* 4, s.l.: IEEE Communications Society, 11 de 4 de 2013, Communications Magazine, IEEE, Vol. 51, págs. 48-54. 0163-6804. **A. Johnston, J. Yoakum, y K. Singh**.

5) ***WebRTC**: APIs and RTCWeb protocols of the HTML5 real-time web.* 2. St. Louis: Digital Codex LLC, 2013. 978-0-9859788-4-6. **Johnston, B. Alan. y Burnett Daniel C.**

6) ***WebRTC** API Guide*. Mozilla Firefox. URL: https://developer.mozilla.org/en-US/docs/Web/Guide/API/WebRTC.

7) *Getting Started with **WebRTC***: http://www.html5rocks.com/en/tutorials/webrtc/basics/. **Sam Dutton**.

8) ***Real**-Time Transport Protocol (**RTP**) Parameters*. Last Updated 2016-03-18. URL: http://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml. **IANA**.

9) *Customizable, Ubiquitous **Real-Time Communication** over the Web.* http://lists.w3.org/Archives/Public/public-webrtc/2012Oct/att-0076/realtime-media.html. **Aboba, Bernard y Kaufman, Matthew.**

10) ***WebRTC**: Transforming Enterprise Communications.* Ottawa: Macadamian, 2013. **Michaelides, Doug y otros**.

11) *Interactive Connectivity Establishment (**ICE**).* IETF Journal November 2006. URL: http://www.internetsociety.org/articles/interactive-connectivity-establishment. **Jonathan Rosenberg**.

12) ***RFC5245** - Interactive Connectivity Establishment (**ICE**): A Protocol for Network Address Translator (**NAT**) Traversal for **Offer/Answer** Protocols*. Internet Engineering Task Force (IETF). URL: https://tools.ietf.org/html/rfc5245. **J. Rosenberg**.

13) ***RFC6336** - IANA Registry for Interactive Connectivity Establishment (ICE) Options*. Internet Engineering Task Force (IETF), Ericsson, University of

Glasgow, July 2011. URL:https://tools.ietf.org/html/rfc6336. M. Westerlund, C. Perkins.

14) ***RFC2663*** - *IP Network Address Translator (**NAT**)*: https://tools.ietf.org/html/rfc2663. August 1999. **P. Srisuresh, M. Holdrege**.

15) ***RFC5389*** - *Session Traversal Utilities for NAT (**STUN**).* Cisco October 2008. URL: https://tools.ietf.org/html/rfc5389. **J. Rosenberg, R. Mahy, P. Matthews, D. Wing**.

16) ***RFC7064*** - *URI Scheme for the Session Traversal Utilities for NAT (**STUN**) Protocol*.- S. November 2013. URL: https://tools.ietf.org/html/rfc7064. **Nandakumar, G. Salgueiro, P. Jones, M. Petit-Huguenin**.

17) ***RFC7065*** - *Traversal Using Relays around NAT (**TURN**) Uniform Resource Identifiers*. November 2013. URL: https://tools.ietf.org/html/rfc7065. **M. Petit-Huguenin, S. Nandakumar, G. Salgueiro, P. Jones**.

18) *Multiparty Multimedia Session Control (**mmusic**)*. IETF. URL: https://datatracker.ietf.org/wg/mmusic/charter/. **MMUSIC Working Group**.

19) *RTP Payload Format for the **iSAC** Codec.* Cisco Systems y Google. URL: https://tools.ietf.org/id/draft-ietf-avt-rtp-isac-02.html. **T. le Grand, P. Huart,, T. Shabestary, P. Jones**.

20) ***RFC3951*** - *Internet Low Bit Rate Codec (**iLBC**)*. Global IP Sound, December 2004. *URL:* http://www.ietf.org/rfc/rfc3951.txt. **A. Duric, Telio, H. Astrom, R. Hagen, W. Kleijn, J. Linden**.

21) ***RFC4566*** - *Session Description Protocol(**SDP**)*: https://tools.ietf.org/html/rfc4566. University of Glasgow, July 2006. **M. Handley, UCL, V. Jacobson, Packet Design, C. Perkins**.

22) *Media Capture and Streams.* 3 September 2013. W3C Working Draft. URL: http://www.w3.org/TR/mediacapture-streams/. **Daniel Burnett, Adam Bergkvist, Cullen Jennings, Anant Narayanan**.

23) *Web technology for developers - Web APIs – **RTCPeerConnection***: https://developer.mozilla.org/es/docs/Web/API/RTCPeerConnection. **Mozilla Foundation**.

24) *HTML Living Standard: **WebSocket***. Last Updated 12 June 2016. URL: https://html.spec.whatwg.org/multipage/comms.html#network. **WHATWG**.

25) ***RFC5389*** - *The **WebSocket** Protocol*. Internet Engineering Task Force (IETF). URL: https://tools.ietf.org/html/rfc6455. **I. Fette y A. Melnikov**.

26) *Try Out Low-Latency Connections with **WebSockets***. Openshift December 18, 2012. URL: https://blog.openshift.com/paas-websockets/. **Marek Jelen.**

27) ***RESTful** Web Services*. O'Reilly Media. May 2007. ISBN 978-0-596-52926-0. **Leonard Richardson y Sam Ruby**.

28) *Wide Web Consortium:* http://www.w3c.es/. **W3C**.

29) *Oracle Corporation - **JAVA***. URL: https://www.java.com. **Oracle Corporation**.

30) *Port Binding and Routing Requests to your **OpenShift Application***. Openshift. URL: https://developers.openshift.com/managing-your-applications/port-binding-routing.html .

31) *How to build Java WebSocket Applications Using the JSR 356 API.*OpenShift, July 30, 2013. URL: https://blog.openshift.com/how-to-build-java-websocket-applications-using-the-jsr-356-api/. **Shekhar Gulati.**

32) *Media Capture and Streams.* W3C Editor's Draft 13 May 2016. URL: https://w3c.github.io/mediacapture-main/#navigatorusermedia-interface-extensions.. **Daniel C. Burnett, Adam Bergkvist, Cullen Jennings, Anant Narayanan, Bernard Aboba**.

33) *Capturing Audio & Video in **HTML5***. Octubre 29th, 2013. URL: http://www.html5rocks.com/es/tutorials/getusermedia/intro/. **Eric Bidelman**.

34) ***RecordRTC**: WebRTC audio/video recording*. URL: https://www.webrtc-experiment.com/RecordRTC/. **Muaz Khan**.

35) *A **WebRTC**-based video confenrecing **application***: https://github.com/dzlab/jWebRTC. **Dzlab**.

36) *Function Point Languages Table*. URL: http://www.qsm.com/resources/function-point-languages-table. **Quantitative Software Management**.

Anexo A
Casos de uso

Anexo A.- Casos de uso

CU – 01: Registrar usuario

CU – 02: Listar los usuarios (Sala Principal)

CU – 03: Añadir usuario (Sala Principal)

CU – 04: Eliminar usuario (Sala Principal)

CU – 05: Realizar llamada (Sala Principal)

CU – 06: Recibir llamadas (Sala Principal)

CU – 07: Aceptar llamada (Sala Principal)

CU – 08: - Rechazar llamada (Sala Principal)

CU – 09: Realizar videollamada entre pares (Habitación WebRTC)

CU – 10: Enviar archivos

CU – 11: Cancelar transferir archivos entre pares

CU – 12: Grabar videollamada

CU – 13: Parar grabar vídeo

CU – 14: Chat On/Off

CU – 15: Enviar mensaje (Chat)

CU – 16: Listar mensajes (Chat)

CU – 17: Hacer foto

CU – 18: Audio On/Off

CU – 19: Pantalla completa On/Off

CU – 20: Volver a la Sala Principal

CU – 01	**Registrar usuario (Inicio)**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 0: Mostrar pagina de inicio y bienvenida RQF – 1: Registrar usuario RQF – 2: Validar usuario RQF – 3: Asignar un token al usuario
Precondición	El sistema tiene que mostrar un campo de texto vacío para que el usuario introduzca su nombre. El nombre de usuario tiene que tener más de 2 caracteres, si el usuario introduce un nombre de menos de 3 caracteres se muestra un mensaje de información y no permitirá entrar a la Sala Principal. Si el nombre tiene más de dos caracteres y se pulsa el botón de entrar el sistema tiene que redireccionar al usuario a la Sala Principal de la aplicación.
Descripción	El sistema tiene que registrar el usuario para crear posteriormente mostrar un campo de texto vacío para que el usuario introduzca su nombre para entrar en la aplicación
Secuencia normal	1.- El usuario entra en el inicio de la aplicación 2.- Se muestra un formulario sencillo con un campo de texto para introducir el nombre y un botón para aceptar. 3.- El usuario introduce su nombre o alias en el campo de texto y pulsa el botón aceptar 4.- El sistema asigna un token al usuario y lo registra redirigiéndolo a la Sala Principal
Postcondición	El usuario registrado tiene que ser redirigido a la Sala Principal.
Excepción	Si el websocket no tiene conexión se muestra un mensaje de error de conexión del websocket. Si ha ocurrido algún fallo o excepción el sistema tiene que mostrarlo al usuario.
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.1.- CU – 01: Registrar usuario

CU – 02	**Listar usuarios (Sala Principal)**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 4: Redirigir al usuario a la Sala Principal RQF – 5: Establecer un socket de conexión (Sala Principal) RQF – 6: Listar usuarios (Sala Principal)
Precondición	El usuario tiene que haber habilitado JavaScript para crear un socket de conexión con el servidor.
Descripción	El sistema tiene que listar los usuarios que está conectados en la aplicación.
Secuencia normal	1.- El usuario entra en la Sala Principal de la aplicación 2.- Se muestra un listado de los usuarios conectados en la aplicación
Postcondición	Ninguna
Excepción	Si no se crea la conexión del socket con la Sala Principal, el sistema tiene que notificarlo.
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.2.- CU – 02: Listar los usuarios (Sala Principal)

CU – 03	**Añadir usuario (Sala Principal)**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 5: Establecer un socket de conexión (Sala Principal) RQF – 6: Listar usuarios (Sala Principal) RQF – 7: Añadir usuario (Sala Principal)
Precondición	El nombre de usuario tiene que tener más de 2 caracteres y un token de usuario único.
Descripción	El sistema tiene que añadir a los usuarios que entren de nuevo a la Sala Principal a la lista de los usuarios conectados.
Secuencia normal	1.- El usuario entra en la Sala Principal de la aplicación 2.- Se añaden los nuevos usuarios al listado de los usuarios conectados en la aplicación de la Sala Principal.
Postcondición	El usuario tiene que poder recibir llamadas y realizar llamadas a los usuarios que están conectados en la aplicación. La lista tiene que poder refrescarse en el cliente de forma transparente al añadir nuevos usuarios.
Excepción	Ninguna excepción registrada
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.3.- CU – 03: Añadir usuario (Sala Principal)

CU – 04	**Eliminar usuario (Sala Principal)**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 5: Establecer un socket de conexión (Sala Principal) RQF – 6: Listar usuarios (Sala Principal) RQF – 8: Eliminar usuario (Sala Principal)
Precondición	El usuario tiene que estar conectado a la aplicación y estar en la Sala Principal.
Descripción	El sistema tiene que eliminar a los usuarios que salgan de la Sala Principal y borrarlos de la lista de usuarios conectados.
Secuencia normal	1.- El sale de la Sala Principal de la aplicación. 2.- Se elimina al usuario a la lista de la Sala Principal.
Postcondición	Eliminar al usuario de la lista de usuarios conectados en la Sala Principal y refrescar el listado de todos los usuarios conectados.
Excepción	Ninguna excepción registrada
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.4.- CU – 04: Eliminar usuario (Sala Principal)

CU – 05	**Realizar llamada (Sala Principal)**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 5: Establecer un socket de conexión (Sala Principal) RQF – 6: Listar usuarios (Sala Principal) RQF – 9: Realizar llamada
Precondición	El usuario A tiene que estar conectado a la aplicación y estar en la Sala Principal. El usuario B tiene que estar conectado a la aplicación y estar en la Sala Principal.
Descripción	El sistema tiene que permitir que los usuarios pueda realizar llamadas y notificar al otro usuario que está siendo llamado.
Secuencia normal	1.- Se conectan el usuario A y B a la Sala Principal de la aplicación. 2.- El usuario B llama al usuario A. 2.- El usuario B pulsa sobre el botón del usuario A para realizar la llamada.
Postcondición	El usuario B se redirige a la habitación para realizar la videollamada entre pares. Al usuario A se le notifica de que el usuario B le está llamando.
Excepción	Ninguna excepción registrada
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.5.- CU – 05: Realizar llamada (Sala Principal)

CU – 06	**Recibir llamada (Sala Principal)**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 9: Realizar llamada RQF – 10: Notificar llamada
Precondición	El usuario tiene que estar conectado a la aplicación y estar en la Sala Principal.
Descripción	El sistema tiene que permitir que el usuario pueda recibir llamadas y que se notifique en la interfaz que está siendo llamado.
Secuencia normal	1.- El usuario A está en la Sala Principal. 2.- El usuario B llama al usuario A. 3.- La interfaz muestra al usuario A que el usuario B le está llamando.
Postcondición	El usuario B, el que llama tiene que se redirigido a la habitación de comunicación entre pares.
Excepción	Ninguna excepción registrada
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.6.- CU – 06: Recibir llamadas (Sala Principal)

CU – 07	**Aceptar llamada (Sala Principal)**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 9: Realizar llamada RQF – 10: Notificar llamada RQF – 12: Aceptar llamada
Precondición	El usuario A tiene que estar conectado a la aplicación y estar en la Sala Principal. El usuario B realiza una llamada al usuario A.
Descripción	El sistema tiene que permitir que el usuario pueda recibir llamadas y que se notifique en la interfaz que está siendo llamado. El sistema tiene que mostrar un botón para aceptar la llamada.
Secuencia normal	1.- El usuario A está siendo llamado por el usuario B. 2.- El sistema muestra al usuario A una notificación de que está siendo llamado por el usuario B. 3.- El sistema muestra un botón para aceptar la llamada del usuario B.
Postcondición	El usuario A, tiene que ser redirigido a la habitación para realizar la llamada entre pares con el usuario B.
Excepción	Ninguna excepción registrada
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.7.- CU – 07: Aceptar llamada (Sala Principal)

CU – 08	**Rechazar llamada (Sala Principal)**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 9: Realizar llamada RQF – 10: Notificar llamada RQF – 11: Rechazar llamada
Precondición	El usuario A tiene que estar conectado a la aplicación y estar en la Sala Principal. El usuario B realiza una llamada al usuario A.
Descripción	El sistema tiene que permitir que el usuario pueda recibir llamadas y notificar que está siendo llamado. El sistema tiene que mostrar un botón para rechazar la llamada.
Secuencia normal	1.- El usuario A está siendo llamado por el usuario B. 2.- El sistema muestra al usuario A una notificación de que está siendo llamado por el usuario B. 3.- El sistema muestra un botón para rechazar la llamada del usuario B.
Postcondición	El sistema tiene que eliminar la notificación de la llamada de la interfaz del usuario llamado.
Excepción	Ninguna excepción registrada
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.8.- CU – 08 - Rechazar llamada (Sala Principal)

CU – 09	**Realizar videollamada entre pares (Habitación WebRTC)**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos: RQF – 9: Realizar llamada RQF – 12: Aceptar llamada RQF – 13: Crear la Habitación para la videollamada entre pares (WebRTC) RQF – 14: Redirigir a la Habitación (WebRTC) RQF – 15: Securizar la Habitación RQF – 16: Establecer un socket de conexión (Habitación) RQF – 18: Sincronizar el par de usuarios RQF – 19: Configurar la Conexión entre Pares. WebRTC RQF – 20: Obtener y establecer SDP (Descripción de la Sesión) Offer/Answer RQF – 21: Enviar SDP (Descripción de la Sesión) Offer/Answer RQF – 22: Establecer la SDP Remota (Descripción de la Sesión) Offer/Answer RQF – 23: Obtener y establecer los ICECandidates RQF – 24: Enviar los ICECandidates RQF – 25: Recibir los ICECandidates Remotos RQF – 26: Añadir los ICECandidates Remotos RQF – 27: Establecer y mantener la conexión con WebRTC entre pares RQF – 28: Cerrar la conexión con WebRTC entre pares RQF – 29: Adaptar la vista para la videollamada
Precondición	El usuario A tiene que estar conectado a la aplicación y estar en la Sala Principal. El usuario B realiza una llamada al usuario A. El usuario B se redirecciona a la habitación de la videollamada. El usuario A acepta la llamada, se redirige a la habitación de la videollamada.
Descripción	El sistema tiene que redireccionar a los usuarios que están realizando la videollamada entre pares a la habitación privada para su fin.
Secuencia normal	1.- Dos usuarios realizan una llamada entre ellos(pares) y la aceptan. 2.- Los usuarios dos usuario son redirigidos a la habitación de la videollamada WebRTC. 3.- Los usuarios aceptan compartir los recursos de su máquina, como es el audio y vídeo (micrófono y cámara).
Postcondición	El sistema adaptará la interfaz para que los usuarios que realizan la llamada entre pares puedan visualizarse entre ellos.
Excepción	Al utilizar 4G, 3G o HSDPA para realizar videollamadas VoIP, algunos proveedores de red no añaden a su tarifa este servicio y tiene que ser contratado adicionalmente. En este caso al realizar una vídeo llamada la conexión se realizará con éxito, pero la red estará bloqueada para enviar datos por VoIP.
Frecuencia	1 vez/día

Importancia	Alta
Prioridad	Alta
Comentarios	Este caso de uso es el más importante de la aplicación. El propósito de este es realizar la videollamada entre dos usuarios utilizando la tecnología WebRTC.

Tabla A.9.- CU – 09: Realizar videollamada entre pares (Habitación WebRTC)

CU – 10	**Enviar archivo**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 13: Crear la Habitación para la videollamada entre pares (WebRTC) RQF – 30: Menú GUI RQF – 35: Transferir archivos
Precondición	La conexión entre pares tiene que estar activa.
Descripción	El usuario A decide compartir un archivo con el usuario B. Al pulsar el botón de compartir archivo, se abrirá el explorador de archivos del sistema operativo para poder seleccionar el archivo a compartir. Al pulsar aceptar, el archivo tiene que enviarse de forma automática al usuario B.
Secuencia normal	1.- El usuario A quiere enviar un archivo al usuario B. 2.- El usuario A pulsa el botón “Compartir archivo”, lo selecciona y acepta. 3.- El usuario A envía automáticamente el archivo al usuario B. 4.- El usuario B recibe el archivo.
Postcondición	El sistema notificará a los usuarios que comparten el archivo el estado de la transferencia constantemente.
Excepción	El archivo puede configurarse para aceptar un tamaño máximo de MB.
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.10.- CU – 10: Enviar archivos

CU – 11	**Cancelar envío o recepción del archivo**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 13: Crear la Habitación para la videollamada entre pares (WebRTC) RQF – 30: Menú GUI RQF – 37: Cancelar envío del archivo RQF – 38: Cancelar recepción del archivo
Precondición	La conexión entre pares tiene que estar activa. Un usuario ha compartido un archivo con otro usuario.
Descripción	Cuando se está realizando la transferencia de un archivo entre pares, el sistema tiene que mostrar un botón para poder cancelar la transferencia cuando se está realizando este proceso.
Secuencia normal	1.- El usuario A comparte un archivo con el usuario B. 2.- El usuario A o B cancela la transferencia 3.- Se muestra mensaje de transferencia cancelada
Postcondición	El mensaje de la transferencia tiene que desaparecer a los 15 segundos de la interfaz del usuario.
Excepción	Ninguna
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.11.- CU – 11: Cancelar transferir archivos entre pares

CU – 12	**Grabar videollamada**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos: RQF – 33: Grabar vídeo RQF – 34: Parar grabar vídeo RQF – 35: Crear vídeo
Precondición	La conexión entre pares tiene que estar activa.
Descripción	El usuario pulsa sobre el botón de grabar videollamada. La videollamada empieza a grabar y el botón de grabar se muestra en estado activo.
Secuencia normal	1.- El usuario pulsa el botón grabar. 2.- El empieza a grabar, el botón cambia a estado activo.
Postcondición	Ninguna
Excepción	Ninguna
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.12.- CU – 12: Grabar videollamada

CU – 13	**Parar grabar vídeo**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos: RQF – 33: Grabar vídeo
Precondición	La conexión entre pares tiene que estar activa. La grabación ha sido activada anteriormente
Descripción	Cuando el usuario está grabando, al volver a pulsar el botón de Grabar Vídeo On/Off deja de grabar y se crea el vídeo para poder ser descargado.
Secuencia normal	1.- El usuario está grabando un vídeo 2.- El usuario pulsa el botón de Grabar Vídeo On/Off 3.- Se crea el vídeo y se muestra el link de descarga
Postcondición	Crear el vídeo y mostrar el link de descarga o descargar directamente
Excepción	Ninguna
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.13.- CU – 13: Parar grabar vídeo

CU – 14	**Chat On/Off**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos: RQF – 41: Chat On/Off RQF – 42: Crear Chat
Precondición	Ninguna
Descripción	En el GUI del Menú se mostrará un botón para ocultar/mostrar la interfaz del Chat. El GUI del Chat tiene que contener una cruz para cerrarlo y dos flechas para poder cambiar el alpha del la interfaz. Los usuarios tienen que poder mantener conversaciones por chat entre pares mientras la videollamada está activa. En el GUI del Chat tiene que diferenciar entre el mensaje enviado y el mensaje recibido por el usuario listándolo en la interfaz del Chat.
Secuencia normal	1.- La interfaz del Chat se muestra en la GUI de la videollamada. 2.- El usuario sobre el botón de Chat On/Off. 3.- La GUI del Chat se oculta. 4.- El usuario vuelve a pulsar el botón Chat On/Off. 5.- La GUI del Chat se muestra.
Postcondición	Ninguna
Excepción	Ninguna
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.14.- CU – 14: Chat On/Off

CU – 15	**Enviar mensaje (Chat)**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 41: Chat On/Off RQF – 42: Crear Chat RQF – 43: Enviar mensaje (Chat)
Precondición	La conexión entre pares tiene que estar activa. El GUI del chat tiene que estar activo en la interfaz del usuario
Descripción	Cuando el usuario quiere enviar el mensaje al otro usuario, rellena el campo de texto de Chat y pulsa “ENTER” o “Enviar” y se envía el mensaje.
Secuencia normal	1.- El usuario A escribe un mensaje en la GUI del Chat. 2.- El usuario A pulsa ”ENTER” o “ENVIAR” el mensaje al usuario B.
Postcondición	Ninguna
Excepción	Ninguna
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.15.- CU – 15: Enviar mensaje (Chat)

CU – 16	**Listar mensajes (Chat)**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 41: Chat On/Off RQF – 42: Crear Chat RQF – 43: Enviar mensaje (Chat) RQF – 44: Recibir mensaje (Chat) RQF – 45: Listar mensajes (Chat)
Precondición	La conexión entre pares tiene que estar activa. El GUI del chat tiene que estar activo en la interfaz del usuario
Descripción	Cuando el usuario quiere enviar el mensaje al otro usuario, rellena el campo de texto de Chat y pulsa “ENTER” o “ENVIAR” y se envía el mensaje. Inmediatamente el mensaje se añade a la GUI del Chat como enviado y el usuario que lo recibe lo muestra en la GUI del Chat como recibido.
Secuencia normal	1.- El usuario A escribe un mensaje en la GUI del Chat. 2.- El usuario A pulsa ”ENTER” o “ENVIAR” el mensaje al usuario B. 3.- El usuario A añade su mensaje a la GUI de la conversación del Chat. 4.- El usuario B recibe el mensaje y la GUI del chat lo muestra.
Postcondición	Ninguna
Excepción	Ninguna
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.16.- CU – 16: Listar mensajes (Chat)

CU – 17	**Hacer foto**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 30: Menú GUI RQF – 32: Hacer Foto
Precondición	La conexión entre pares tiene que estar activa. El GUI del chat tiene que estar activo en la interfaz del usuario
Descripción	Cuando el usuario pulse el botón hacer foto, se tiene que hacer una foto de la interfaz de la videollamada y mostrar una imagen pequeña en la posición inferior izquierda de la vista principal.
Secuencia normal	1.- El usuario A pulsa hacer foto 2.- Se hacer la foto y se muestra en la interfaz de la pantalla.
Postcondición	Ninguna
Excepción	Ninguna
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.17.- CU – 17: Hacer foto

CU – 18	**Sonido On/Off**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 30: Menú GUI RQF – 40: Sonido On/Off
Precondición	Los recursos de la máquina tienen que estar accesibles para el navegador
Descripción	Cuando el usuario pulse el botón Sonido On/Off el sonido se desactivará y se mostrará el botón en el estado desactivado, si el usuario vuelve a pulsar el botón del Sonido On/Off el sonido se activará y el botón Sonido On/Off pasará al estado activo.
Secuencia normal	1.- El usuario pulsa el botón Sonido On/Off 2.- Si el sonido está activado se desactivará, si el sonido está desactivado se activará.
Postcondición	Ninguna
Excepción	Ninguna
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.18.- CU – 18: Sonido On/Off

CU – 19	**Pantalla completa On/Off**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 30: Menú GUI RQF – 31: Pantalla Completa On/Off
Precondición	Ninguna
Descripción	Cuando el usuario pulse el botón Pantalla completa On/Off, la interfaz del usuario se establecerá a modo pantalla completa, si el usuario vuelve a pulsar el botón de Pantalla completa On/Off se desactivará el modo pantalla completa.
Secuencia normal	1.- El usuario pulsa el botón Pantalla completa On/Off 2.- La interfaz cambia a modo pantalla completa. 3.- El usuario vuelve a pulsar el botón Pantalla completa On/Off 4.- La interfaz desactiva el modo pantalla completa y se actualiza la interfaz del usuario.
Postcondición	Ninguna
Excepción	Ninguna
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.19.- CU – 19: Pantalla completa On/Off

CU – 20	**Volver a la Sala Principal**
Versión	1.0(11/03/16)
Dependencias	Depende de los siguientes requisitos RQF – 4: Redirigir al usuario a la Sala Principal RQF – 14: Redirigir a la Habitación (WebRTC)
Precondición	Tiene que haber sido registrado anteriormente en la Sala Principal y tiene que haber entrado en una Habitación.
Descripción	Cuando el usuario pulse el botón volver a la Sala Principal, será redirigido a la Sala Principal con el nombre que se registró al entrar en la aplicación.
Secuencia normal	1.- El usuario pulsa el botón Volver a la Sala Principal 2.- El usuario se redirige a la Sala Principal con el mismo nombre con el que se registró en la aplicación.
Postcondición	Ninguna
Excepción	Ninguna
Frecuencia	1 vez/día
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla A.20.- CU – 20: Volver a la Sala Principal

Anexo B
Requisitos Funcionales

Anexo B.- Requisitos Funcionales

RQF – 00	**Mostrar pagina de inicio y bienvenida**
Versión	1.0(11/03/16)
Dependencias	Ninguna
Descripción	El sistema debe mostrar una página de inicio y bienvenida de la aplicación.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.0.- RQF – 00: Mostrar página de inicio y bienvenida

RQF – 01	**Registrar usuario**
Versión	1.0(11/03/16)
Dependencias	Ninguna
Descripción	El sistema tiene que mostrar en la página de inicio un formulario sencillo para introducir el nombre de usuario.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.1.- RQF – 01: Registrar usuario

RQF – 02	**Validar usuario**
Versión	1.0(11/03/16)
Dependencias	RQF – 1: Registrar usuario
Descripción	El sistema tiene que verificar y validar que el nombre de usuario tiene más de 2 caracteres para poder acceder a la aplicación.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.2.- RQF – 02: Validar usuario

RQF – 03	**Asignar un token al usuario**
Versión	1.0(11/03/16)
Dependencias	RQF – 1: Registrar usuario RQF – 2: Validar usuario RQF – 15: Securizar la Habitación WebRTC
Descripción	El sistema debe de asignar a los usuarios tokens únicos para que puedan realizar acciones en el sistema de forma segura.
Importancia	Alta
Prioridad	Alta
Comentarios	Este requisito funcional se aplica en varios casos: - Cuando se añade un usuario a la Sala Principal. - Cuando un usuario crea una Habitación para realizar una videollamada.

Tabla B.3.- RQF – 03: Asignar un token al usuario

RQF – 04	**Redirigir al usuario a la Sala Principal**
Versión	1.0(11/03/16)
Dependencias	RQF – 2: Validar usuario RQF – 3: Asignar un token al usuario
Descripción	El sistema tiene que redirigir al usuario a la Sala Principal cuando se valide al usuario y se le asigne un token único.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.4.- RQF – 4: Redirigir al usuario a la Sala Principal

RQF – 05	**Establecer un socket de conexión (Sala Principal)**
Versión	1.0(11/03/16)
Dependencias	RQF – 2: Validar usuario RQF – 3: Asignar un token al usuario RQF – 4: Redirigir al usuario a la Sala Principal
Descripción	El sistema tiene que poder crear un socket de conexión con la Sala Principal para poder lanzar eventos a todos los usuarios conectados.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.5.- RQF – 05: Establecer un socket de conexión (Sala Principal)

RQF – 06	**Listar usuarios (Sala Principal)**
Versión	1.0(11/03/16)
Dependencias	RQF – 5: Establecer un socket de conexión (Sala Principal)
Descripción	El sistema tiene que listar los usuarios conectados a la aplicación en la Sala Principal.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.6.- RQF – 06: Listar usuarios (Sala Principal)

RQF – 07	**Añadir usuario (Sala Principal)**
Versión	1.0(11/03/16)
Dependencias	RQF – 5: Establecer un socket de conexión (Sala Principal) RQF – 6: Listar usuarios (Sala Principal)
Descripción	El sistema debe añadir un listado de los usuarios ya conectados a la aplicación en la sala principal. El listado tiene que actualizarse
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.7.- RQF – 07: Añadir usuario (Sala Principal)

RQF – 08	**Eliminar usuario (Sala Principal)**
Versión	1.0(11/03/16)
Dependencias	RQF – 4: Establecer un socket de conexión (Sala Principal) RQF – 7: Listar usuarios (Sala Principal)
Descripción	El sistema tiene que eliminar el usuario de la Sala Principal cuando: - El usuario salga de la Sala Principal - Se interrumpa la conexión - Cuando cierre el navegador
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.8.- RQF – 08: Eliminar usuario (Sala Principal)

RQF – 09	**Realizar llamada**
Versión	1.0(11/03/16)
Dependencias	RQF – 4: Establecer un socket de conexión (Sala Principal) RQF – 7: Listar usuarios (Sala Principal)
Descripción	El sistema permitirá que los usuarios puedan enviarse señales de llamada entre pares por medio del listado de la Sala Principal.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.9.- RQF – 09: Realizar llamada

RQF – 10	**Notificar llamada**
Versión	1.0(11/03/16)
Dependencias	RQF – 4: Establecer un socket de conexión (Sala Principal) RQF – 9: Realizar llamada
Descripción	El sistema permitirá que los usuarios puedan enviarse señales de llamada entre pares y los usuarios llamados tienen que ser notificados que son llamados por otro usuario .
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.10.- RQF – 10: Notificar llamada

RQF – 11	**Rechazar llamada**
Versión	1.0(11/03/16)
Dependencias	RQF – 4: Establecer un socket de conexión (Sala Principal) RQF – 10: Notificar llamada
Descripción	El usuario que esta siendo llamado tiene que ser notificado y se le tiene que brindar la posibilidad de elegir rechazar la llamada de otro usuario. La interfaz tiene que refrescarse para que no se mantenga la llamada rechazada.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.11.- RQF – 11: Rechazar llamada

RQF – 12	**Aceptar llamada**
Versión	1.0(11/03/16)
Dependencias	RQF – 4: Establecer un socket de conexión (Sala Principal) RQF – 10: Notificar llamada
Descripción	El usuario que esta siendo llamado tiene que ser notificado y se le tiene que brindar la posibilidad de elegir aceptar la llamada de otro usuario. El usuario tiene que ser redirigido a la videollamada directamente para mejorar la usabilidad.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.12.- RQF – 12: Aceptar llamada

RQF – 13	**Crear la Habitación para la videollamada entre pares (WebRTC)**
Versión	1.0(11/03/16)
Dependencias	RQF – 3: Asignar un token al usuario
Descripción	El sistema tiene que crear una habitación con capacidad máxima de 2 usuarios para realizar la videollamada.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.13.- RQF – 13: Crear la Habitación para la videollamada entre pares (WebRTC)

RQF – 14	**Redirigir a la Habitación (WebRTC)**
Versión	1.0(11/03/16)
Dependencias	RQF – 3: Asignar un token al usuario RQF – 13: Crear la Habitación para la videollamada entre pares
Descripción	El usuario cuando llama o cuando es llamado y acepta la llamada, tiene que ser redirigido a la Habitación para realizar la videollamada entre pares.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.14.- RQF – 14: Redirigir a la Habitación (WebRTC)

RQF – 15	**Securizar la Habitación**
Versión	1.0(11/03/16)
Dependencias	RQF – 3: Asignar un token al usuario RQF – 13: Crear la Habitación para la videollamada entre pares
Descripción	La Habitación donde se realiza la videollamada tiene que ser segura y no puede haber más de dos personas en ella.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.15.- RQF – 15: Securizar la Habitación

RQF – 16	**Establecer un socket de conexión (Habitación)**
Versión	1.0(11/03/16)
Dependencias	RQF – 3: Asignar un token al usuario RQF – 13: Crear la Habitación para la videollamada entre pares
Descripción	Los usuarios cuando están en una Habitación tiene que crearse una conexión bidireccional entre el cliente-servidor para poder enviar y recibir eventos. openChannel()
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.16.- RQF – 16: Establecer un socket de conexión (Habitación)

RQF – 17	**Obtener los recursos de la máquina (audio y vídeo)** ***getUserMedia()***
Versión	1.0(11/03/16)
Dependencias	RQF – 16: Establecer un socket de conexión (Habitación)
Descripción	El sistema tiene obtener los recursos del ordenador como son el audio y el vídeo, cuando el usuario esté en la Habitación para realizar la videollamada.
Importancia	Alta
Prioridad	Alta
Comentarios	El usuario tiene que aceptar compartir los recursos por medio del navegador web.

Tabla B.17.- RQF – 17: Obtener los recursos de la máquina (audio y vídeo) *getUserMedia()*

RQF – 18	**Sincronizar el par de usuarios**
Versión	1.0(11/03/16)
Dependencias	RQF – 16: Establecer un socket de conexión (Habitación)
Descripción	El sistema tiene que sincronizar al par de usuarios para realizar el correcto envío y recepción de datos para configurar la conexión con WebRTC.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.18.- RQF – 18: Sincronizar el par de usuarios

RQF – 19	**Configurar la Conexión entre Pares. WebRTC**
Versión	1.0(11/03/16)
Dependencias	RQF – 16: Establecer un socket de conexión (Habitación)
Descripción	El sistema tiene auto-configurar los parámetros necesarios para utilizar la tecnología WebRTC. Los parámetros necesarios son: iceServers y STUN/TURN server.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.19.- RQF – 19: Configurar la Conexión entre Pares. WebRTC

RQF – 20	**Obtener y establecer SDP (Descripción de la Sesión) Offer/Answer**
Versión	1.0(11/03/16)
Dependencias	RQF – 16: Establecer un socket de conexión (Habitación) RQF – 17: Obtener los recursos de la máquina (audio y vídeo)
Descripción	El sistema tiene obtener la Descripción de Sesión para establecer la configuración se va a utilizar para realizar la videollamada entre pares.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.20.- RQF – 20: Obtener y establecer SDP (Descripción de la Sesión) Offer/Answer

RQF – 21	**Enviar SDP (Descripción de la Sesión) Offer/Answer**
Versión	1.0(11/03/16)
Dependencias	RQF – 16: Establecer un socket de conexión (Habitación) RQF – 20: Obtener SDP (Descripción de la Sesión)
Descripción	El sistema tiene que enviar la Descripción de la Sesión del usuario para que el otro usuario lo reciba.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.21.- RQF – 21: Enviar SDP (Descripción de la Sesión) Offer/Answer

RQF – 22	**Establecer la SDP Remota (Descripción de la Sesión) Offer/Answer**
Versión	1.0(11/03/16)
Dependencias	RQF – 16: Establecer un socket de conexión (Habitación)
Descripción	El sistema tiene que establecer la Descripción de la Sesión Remota para poder configurar los parámetros que WebRTC necesita.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.22.- RQF – 22: Establecer la SDP Remota (Descripción de la Sesión) Offer/Answer

RQF – 23	**Obtener y establecer los ICECandidates**
Versión	1.0(11/03/16)
Dependencias	
Descripción	El sistema tiene obtener los ICECandidates válidos para configurar los parámetros necesarios para establecer la conexión por medio de STUN server y elegir los de menor latencia.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.23.- RQF – 23: Obtener y establecer los ICECandidates

RQF – 24	**Enviar los ICECandidates**
Versión	1.0(11/03/16)
Dependencias	RQF – 16: Establecer un socket de conexión (Habitación) RQF – 23: Obtener y establecer los ICECandidates
Descripción	El sistema tiene enviar los ICECandidates al otro usuario para que este último los añada a su lista de ICECandidates válidos y pueda configurar la conexión con WebRTC.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.24.- RQF – 24: Enviar los ICECandidates

RQF – 25	**Recibir los ICECandidates Remotos**
Versión	1.0(11/03/16)
Dependencias	RQF – 16: Establecer un socket de conexión (Habitación) RQF – 18: Sincronizar el par de usuarios RQF – 24: Enviar los ICECandidates
Descripción	El sistema tiene que enviar y recibir los ICECandidates válidos entre pares para configurar los parámetros necesarios para establecer la conexión por medio de STUN server y elegir los candidates de menor latencia.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.25.- RQF – 25: Recibir los ICECandidates Remotos

RQF – 26	**Añadir los ICECandidates Remotos**
Versión	1.0(11/03/16)
Dependencias	RQF – 16: Establecer un socket de conexión (Habitación) RQF – 18: Sincronizar el par de usuarios RQF – 25: Recibir los ICECandidates Remotos
Descripción	El sistema tiene que añadir los ICECandidates válidos para configurar los parámetros necesarios para establecer la conexión por medio de STUN server y elegir los de menor latencia.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.26.- RQF – 26: Añadir los ICECandidates Remotos

RQF – 27	**Establecer y mantener la conexión con WebRTC entre pares**
Versión	1.0(11/03/16)
Dependencias	RQF – 18 … RQF – 26
Descripción	El sistema tiene que establecer y mantener la conexión entre el par de usuarios de manera estable. Si la conexión se interrumpe se debe de dar la posibilidad al usuario de poder restablecer la videollamada.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.27.- RQF – 27: Establecer y mantener la conexión con WebRTC entre pares

RQF – 28	**Cerrar la conexión con WebRTC entre pares**
Versión	1.0(11/03/16)
Dependencias	RQF – 27: Establecer y mantener la conexión con WebRTC entre pares
Descripción	El sistema tiene que cerrar la conexión de forma segura para la aplicación y notificar de los posibles mensajes de desconexión a los usuarios afectados. El sistema tiene que limpiar de la memoria a los usuarios que se desconecten de la Habitación
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguno

Tabla B.28.- RQF – 28: Cerrar la conexión con WebRTC entre pares

RQF – 29	**Adaptar la vista para la videollamada**
Versión	1.0(11/03/16)
Dependencias	RQF – 27: Establecer y mantener la conexión con WebRTC entre pares
Descripción	El sistema tiene que adaptar la vista del usuario cuando se realice una videollamada.
Importancia	Alta
Prioridad	Alta
Comentarios	El usuario remoto tiene que visualizarse en el máximo de ancho y largo de la pantalla y el usuario local en una minipantalla.

Tabla B.29.- RQF – 29: Adaptar la vista para la videollamada

RQF – 30	**Menú GUI**
Versión	1.0(11/03/16)
Dependencias	Se localiza en la Habitación WebRTC
Descripción	El sistema tiene que poseer un menú de acciones: - Pantalla completa - Hacer foto - Grabar - Transferir archivos - Sonido On/Off - Chat On/Off
Importancia	Alta
Prioridad	Alta
Comentarios	El menú tiene que ser simple e intuitivo, los iconos tiene que poseer una descripción al ubicar el ratón sobre ellos. Cuando se elija una acción en el menú tiene que resaltar para visualizar que está seleccionada. El menú tiene que poder moverse (drag-drop) por la pantalla.

Tabla B.30.- RQF – 30: Menú GUI

RQF – 31	**Pantalla Completa On/Off**
Versión	1.0(11/03/16)
Dependencias	RQF – 30: Menú GUI
Descripción	El sistema tiene que poder poner la vista en pantalla completa. Para ello tiene que existir un botón booleano para poder activar o desactivar la pantalla completa.
Importancia	Alta
Prioridad	Alta
Comentarios	Tiene que poseer dos estados: Activado – Desactivado. Con su respectivo efecto visual.

Tabla B.31.- RQF – 31: Pantalla Completa On/Off

RQF – 32	**Hacer Foto**
Versión	1.0(11/03/16)
Dependencias	RQF – 27: Establecer y mantener la conexión con WebRTC entre pares RQF – 30: Menú GUI
Descripción	El sistema tiene que poder hacer fotos en la videollamada.
Importancia	Alta
Prioridad	Alta
Comentarios	La foto realizada tiene previsualizarse en la vista del usuario.

Tabla B.32.- RQF – 32: Hacer Foto

RQF – 33	**Grabar vídeo**
Versión	1.0(11/03/16)
Dependencias	RQF – 27: Establecer y mantener la conexión con WebRTC entre pares RQF – 30: Menú GUI
Descripción	El sistema tiene que poder grabar la sesión de la videollamada.
Importancia	Alta
Prioridad	Alta
Comentarios	La foto realizada tiene previsualizarse en la vista del usuario.

Tabla B.33.- RQF – 33: Grabar vídeo

RQF – 34	**Parar grabar vídeo**
Versión	1.0(11/03/16)
Dependencias	RQF – 27: Establecer y mantener la conexión con WebRTC entre pares RQF – 30: Menú GUI RQF – 33: Grabar vídeo
Descripción	El sistema tiene que poder grabar la sesión de la videollamada.
Importancia	Alta
Prioridad	Alta
Comentarios	La foto realizada tiene previsualizarse en la vista del usuario.

Tabla B.34.- RQF – 34: Parar grabar vídeo

RQF – 35	**Crear vídeo**
Versión	1.0(11/03/16)
Dependencias	RQF – 27: Establecer y mantener la conexión con WebRTC entre pares RQF – 30: Menú GUI RQF – 33: Grabar vídeo RQF – 34: Parar grabar vídeo
Descripción	El sistema tiene que poder grabar la sesión de la videollamada.
Importancia	Alta
Prioridad	Alta
Comentarios	La foto realizada tiene previsualizarse en la vista del usuario.

Tabla B.35.- RQF – 35: Crear vídeo

RQF – 36	**Transferir archivos**
Versión	1.0(11/03/16)
Dependencias	RQF – 16: Establecer un socket de conexión (Habitación) RQF – 27: Establecer y mantener la conexión con WebRTC entre pares RQF – 30: Menú GUI
Descripción	El sistema tiene que enviar archivos entre pares en tiempo real.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguna

Tabla B.36.- RQF – 36: Transferir archivos

RQF – 37	**Notificar estado de la transferencia del archivo**
Versión	1.0(11/03/16)
Dependencias	RQF – 36: Transferir archivos
Descripción	El sistema tiene que notificar el estado de la transferencia del archivo en tiempo real.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguna

Tabla B.37.- RQF – 37: Notificar estado de la transferencia del archivo

RQF – 38	**Cancelar envío del archivo**
Versión	1.0(11/03/16)
Dependencias	RQF – 36: Transferir archivos
Descripción	El sistema tiene poder cancelar la transferencia del archivo enviado.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguna

Tabla B.38.- RQF – 38: Cancelar envío del archivo

RQF – 39	**Cancelar recepción del archivo**
Versión	1.0(11/03/16)
Dependencias	RQF – 35: Transferir archivos
Descripción	El sistema tiene poder cancelar la recepción del archivo transferido.
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguna

Tabla B.39.- RQF – 39: Cancelar recepción del archivo

RQF – 40	**Sonido On/Off**
Versión	1.0(11/03/16)
Dependencias	RQF – 17: Obtener los recursos de la máquina (audio y vídeo) getUserMedia() RQF – 27: Establecer y mantener la conexión con WebRTC entre pares RQF – 30: Menú GUI
Descripción	El sistema tiene que poder configurar el sonido en dos estados: - Encendido - Apagado
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguna

Tabla B.40.- RQF – 40: Sonido On/Off

RQF – 41	**Chat On/Off**
Versión	1.0(11/03/16)
Dependencias	RQF – 30: Menú GUI
Descripción	El sistema tiene que poder configurar la interfaz del chat en dos estados: - Encendido - Apagado
Importancia	Alta
Prioridad	Alta
Comentarios	Ninguna

Tabla B.41.- RQF – 41: Chat On/Off

RQF – 42	**Crear Chat**
Versión	1.0(11/03/16)
Dependencias	RQF – 16: Establecer un socket de conexión (Habitación)
Descripción	El sistema tiene que tener un chat para comunicarse los usuarios dentro de la Habitación.
Importancia	Alta
Prioridad	Alta
Comentarios	La GUI del chat tiene que poder moverse por la pantalla y tiene que poder incrementar o disminuir su opacidad.

Tabla B.42.- RQF – 42: Crear Chat

RQF – 43	**Enviar mensaje (Chat)**
Versión	1.0(11/03/16)
Dependencias	RQF – 42: Crear Chat
Descripción	El sistema tiene que tener un chat para comunicarse los usuarios dentro de la Habitación y poder enviar mensajes .
Importancia	Alta
Prioridad	Alta
Comentarios	El chat tiene que poseer una lista de mensajes donde se tienen que añadir los mensajes enviados.

Tabla B.43.- RQF – 43: Enviar mensaje (Chat)

RQF – 44	**Recibir mensaje (Chat)**
Versión	1.0(11/03/16)
Dependencias	RQF – 42: Crear Chat RQF – 43: Enviar mensaje (Chat)
Descripción	El sistema tiene que tener un Chat para comunicarse el otro usuario dentro de la Habitación y poder recibir mensajes.
Importancia	Alta
Prioridad	Alta
Comentarios	El chat tiene que poseer una lista de mensajes donde se tienen que añadir los mensajes recibidos.

Tabla B.44.- RQF – 44: Recibir mensaje (Chat)

RQF – 45	**Listar mensajes (Chat)**
Versión	1.0(11/03/16)
Dependencias	RQF – 42: Crear Chat RQF – 43: Enviar mensaje (Chat) RQF – 44: Recibir mensaje (Chat)
Descripción	El sistema tiene que tener un chat para comunicarse los usuarios dentro de la Habitación y poder listar mensajes todos los mensajes enviados y recibidos.
Importancia	Alta
Prioridad	Alta
Comentarios	El chat tiene que poseer una lista de mensajes donde se tienen que añadir los mensajes recibidos diferenciando entre el usuario que lo envía y el que lo recibe con un sistema de colores.

Tabla B.45.- RQF – 45: Listar mensajes (Chat)

RQF – 46	**Volver a la Sala Principal**
Versión	1.0(11/03/16)
Dependencias	RQF – 14: Redirigir a la Habitación WebRTC
Descripción	El sistema tiene que ofrecer al usuario la posibilidad de volver al la Sala Principal desde la Habitación WebRTC sin tener que volver a introducir el nombre de usuario.
Importancia	Alta
Prioridad	Alta
Comentarios	El chat tiene que poseer una lista de mensajes donde se tienen que añadir los mensajes recibidos.

Tabla B.46.- RQF – 46: Volver a la Sala Principal

Anexo C
Diagrama de dependencias

Anexo C.- Diagrama de dependencias

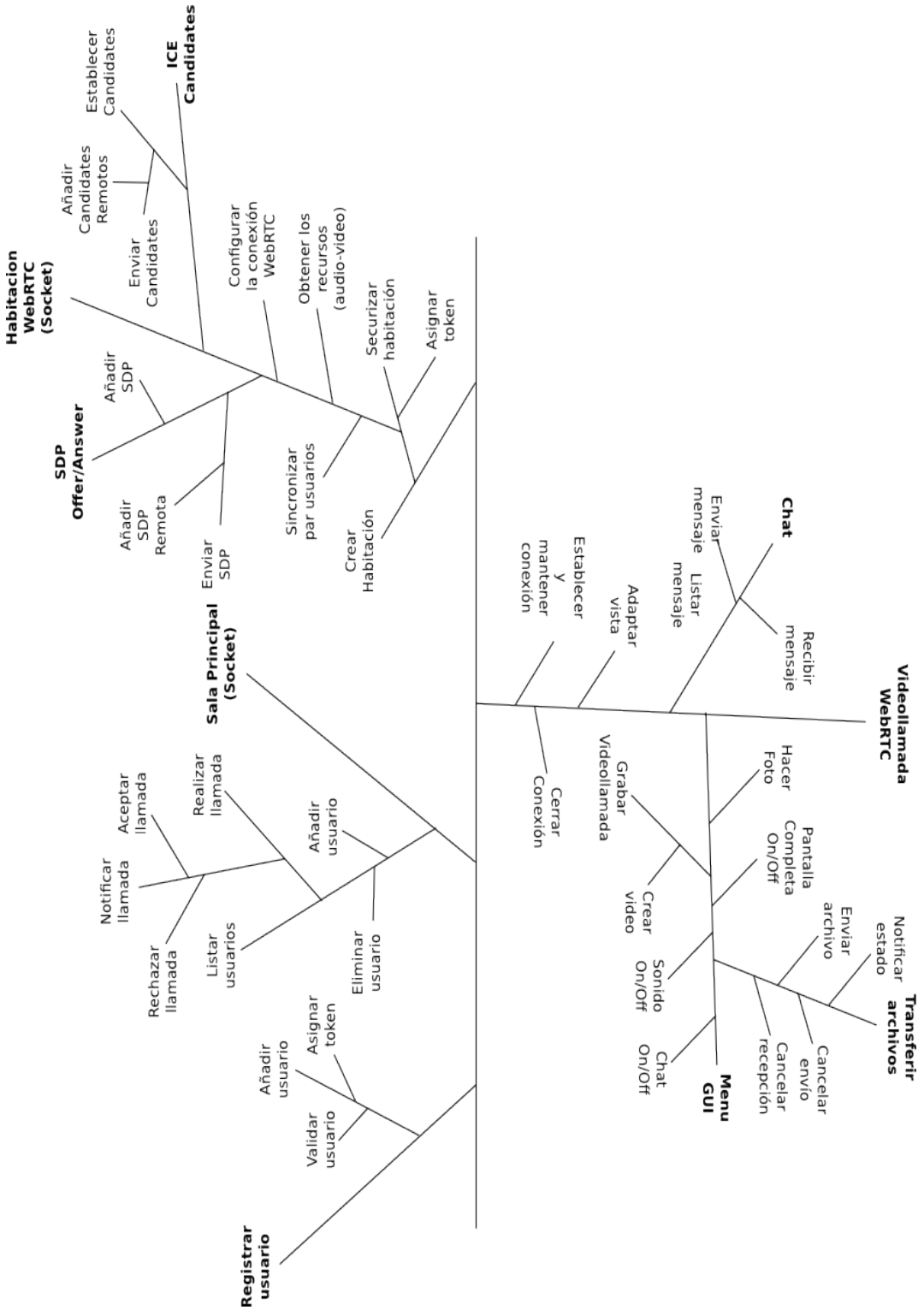

Figura C.1.- Diagrama de dependencias

Anexo D
Glosario de términos

Anexo D.- Glosario de términos

IP.- Una dirección IP es una etiqueta que identifica una interfaz en red de un dispositivo

Protocolo IP.- Es un protocolo de comunicación de datos digitales localizado en la capa de red según el modelo internacional OSI.

OSI.- Open System Iterconnection (OSI), es un modelo de referencia para los protocolos de la red de arquitectura de capas, creado en el año 1980 por la Organizacion Internacional de Normalización (ISO).

VoIP.- Voz sobre protocolo de internet es un conjunto de recursos que hacen posible que la señal de audio y de vídeo viaje a través de Internet empleando el protocolo IP.

SIP.- Session Inition Protocol (SIP), es un protocolo desarrollado por el grupo de trabajo MMUSIC de la IETF con la intención de ser el estándar para la iniciación, modificación y finalización de sesiones interactivas de usuario donde intervienen elementos multimedia como el vídeo, voz, mensajería instantánea, juegos en línea y realidad virtual.

Códec.- Un códec es un programa o dispositivo hardware capaz de codificar o decodificar una señal o flujo de datos digitales. Su uso está extendido para la codificación de señales de audio y vídeo dentro de un formato contenedor.

W3C.- World Wide Web Consortium (W3C). El Consorcio World Wide Web es una comunidad internacional donde las organizaciones Miembro, personal a tiempo completo y el público en general trabajan conjuntamente para desarrollar estándares Web. Liderado por el inventor de la Web Tim Berners-Lee y el Director Ejecutivo (CEO) Jeffrey Jaffe, la misión del W3C es guiar la Web hacia su máximo potencial.

IETF.- Internet Engineering Task Force (IETF). El Grupo de Trabajo de Ingeniería de Internet es una organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, como transporte, encaminamiento, seguridad. Fue creada en EE. UU. en 1986. El IETF es mundialmente conocido por ser la entidad que regula las propuestas y los estándares de Internet, conocidos como RFC.

API.- Application Programming Interface (API). La interfaz de programación de aplicaciones es el conjunto de funciones, métodos, procedimientos y servicios que ofrece una capa de abstracción software para ser utilizada por otro software,

WHATWG.- Web Hypertext Application Technology Working Group (WHATWG). El Grupo de Trabajo de la aplicación web hipertextual, es una comunidad de personas interesadas en la evolución de HTML y las tecnologías conexas. Fue fundado por integrantes de Apple, la Fundación Mozilla y Opera Software.

SGML.- Standard Generalized Markup Language (SGML), es un sistema para definir lenguajes para dar formato a documentos con marcas. Se utiliza para representar información estructural, presentacional y semántica junto con el contenido.

ANEXO D – GLOSARIO DE TERMINOS

DOM.- El Modelo de Objetos del Documento W3C (DOM) es una interfaz de la plataforma y lenguaje neutro que permite a los programas y scripts acceder y actualizar dinámicamente el contenido, la estructura y el estilo de un documento.

MIME.- Multipurpose Internet Mail Extensions (MIME), la extensiones multipropósito de correo de internet, son una serie de convenciones o especificaciones dirigidas al intercambio a través de Internet de todo tipo de archivos (texto, audio, vídeo, etc.) de forma transparente para el usuario.

XML.- eXtensible Markup Language (XML), lenguaje de marcas extensible es un lenguaje de marcas desarrollado por el World Wide Web Consortium, utilizado para almacenar datos en forma legible. Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos.

Node.js.- Es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Está escrito en C++ y en JavaScript. Posee una Licencia MIT.

Token.- Cadena de texto alfanumérica que suele utilizarse para asignar un valor único a una variable, objeto o cualquier elemento relacionado con la programación informática.

Agente de usuario (user-agent).- es una variable (cadena de texto) dentro del contexto web, la cual identifica al dispositivo de un usuario frente a un servidor.

JSON.- JavaScript Object Notation (JSON), es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

AJAX.- Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones. (Wikipedia).

- Sala Principal: Interfaz y servlet de la aplicación donde se muestra el listado de usuario conectados y pueden realizar y recibir llamadas.

- Habitación: Espacio y servlet donde se realiza la videollamada entre pares. El máximo de usuario por habitación son dos.

- Menú GUI: Menú de la aplicación localizado en la interfaz de la Habitación, da funcionalidades adicionales a la videollamada, como hacer fotos, grabar llamada, compartir archivos, muestra o oculta elementos de la interfaz, activa o desactiva elementos y funcionalidades de la aplicación y de la interfaz.

- Chat: Interfaz para enviar y recibir mensajes de texto con otro usuario.