

FACULTAD DE INFORMÁTICA



UNIVERSIDADE DA CORUÑA

TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO DE FIN DE GRADO DE INGENIERÍA INFORMÁTICA

Sistema de atención técnica remota basado en la tecnología WebRTC

Autor:

DIEGO OTERO
GONZÁLEZ

Director:

DIEGO FERNÁNDEZ
IGLESIAS

20 de Noviembre de 2017

Resumen:

Este proyecto se enmarca en el campo de la Informática orientada a las comunicaciones cuyo objetivo es simplificar la atención técnica de clientes mediante el uso de videoconferencias a través del navegador.

En estos momentos se emplean cada vez más elementos software en todos los sectores económicos por lo que es necesario dar soporte técnico en cualquier situación posible. Es por eso que se propone cambiar el tradicional soporte técnico de llamada telefónica a uno que permita videoconferencias y de esta manera garantizar una mejor ayuda.

El soporte técnico supone en muchas ocasiones un gran problema de entendimiento entre ambas partes provocando retrasos a la hora de encontrar la solución y en muchos casos implica desplazar a un técnico al foco del problema. Con la aplicación propuesta se tratan de evitar toda serie de posibles problemas que puedan ocurrir y simplificar el acceso para el cliente.

La aplicación permite al cliente solicitar el servicio técnico de una manera sencilla, mediante un mensaje y el acceso a través de un navegador. Además, permite a los técnicos responsables del servicio organizar sus tareas mediante un calendario. La aplicación no requiere la instalación de ningún software adicional en el ordenador, sino tan solo de un navegador y una conexión a Internet. Todo esto se desarrolla empleando una tecnología llamada WebRTC. Esta tecnología es usada en la gran mayoría de navegadores presentes en el mercado. Cabe destacar que aunque es relativamente nueva la gran mayoría de aplicaciones desarrolladas son de pago y solo existen una serie de librerías de código libre que la implementen.

Palabras clave:

WebRTC, atención técnica, videoconferencia, navegador, código libre.

*Dedicado a mi madre
y a mis amigos
que siempre me apoyaron
y animaron a seguir adelante.*

Agradecimientos

Me gustaría comenzar esta sección agradeciendo a mi madre el trabajo y esfuerzo que han supuesto estos años y que gracias a eso me ha permitido realizar la carrera.

A mi tutor, Diego Fernández Iglesias, por su paciencia y ayuda a lo largo del proyecto.

A Hugo Lorenzo García, Daniel Ferreiro Presedo, David Rivas González, Pablo Otero Aira, Aida Vidal Balea, Yolanda Moral Fernández, Alejandro Sánchez Sánchez por su ayuda a lo largo de la carrera.

A Jose Lorenzo Souto e Iván Collazo Mato por su amistad a lo largo de los años de instituto y carrera.

A mis compañeros de TTGT que me han aguantado y conseguido que todo la carga de trabajo y el estrés de la carrera se hiciera más llevadero.

A todos los compañeros que he conocido a lo largo de estos años y que me han ayudado y que me han hecho pasar toda esta época de una manera amena y divertida.

A todos los profesores que me han atendido tanto en tutorías como fuera de estas y que han sido capaces de resolver todas mis dudas.

Índice

1. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Estructura del documento	2
1.3.1. Apartados	3
1.3.2. Apéndices	4
2. Estado del arte	5
2.1. Skype[1]	5
2.2. Hangouts[2]	5
2.3. Mikogo[3]	6
2.4. Cisco WebEx[4]	6
2.5. GoToMeeting[5]	6
2.6. TeamViewer[6]	7
2.7. Facetime[7]	7
2.8. Microsoft Lync[8]	7
3. Fundamentos tecnológicos	8
3.1. WebRTC	8
3.1.1. Arquitectura WebRTC[9]	8
3.2. Licode[10]	10
3.3. EasyRTC[11]	12
3.4. Otras implementaciones	12
3.5. Node.js[12]	13
3.5.1. Módulos Node	13
3.6. JavaScript[13]	15
3.7. Sublime Text[14]	16
3.8. Bases de datos	17
3.8.1. MongoDB	17
3.8.2. MariaDB	18
3.8.3. Elección de la base de datos	18
4. Metodología	19
4.1. SCRUM[15][16]	19
4.2. Roles	20
4.3. Terminología	20

4.4.	Sprints	21
4.5.	Reuniones	22
4.6.	Adaptación de Scrum para este proyecto	22
4.7.	Iteraciones	24
4.7.1.	Sprint 1: Estudio de herramientas	25
4.7.2.	Sprint 2: Estudio de las tecnologías	25
4.7.3.	Sprint 3: Realización de mockups	26
4.7.4.	Sprint 4: Definir una estructura y funcionalidad básica	26
4.7.5.	Sprint 5: Realizar la autenticación frente a la aplicación	27
4.7.6.	Sprint 6: División de personal de soporte	27
4.7.7.	Sprint 7: Administración de usuarios	28
4.7.8.	Sprint 8: Calendario de eventos	28
4.7.9.	Sprint 9: Incorporar Licode a la aplicación	29
4.7.10.	Sprint 10: Decoración de la aplicación	30
4.7.11.	Sprint 11: Gestión de tokens	30
4.7.12.	Sprint 12: Realización de la videoconferencia	31
5.	Desarrollo	32
5.1.	Preparación	32
5.1.1.	Entorno de Trabajo	32
5.2.	Análisis	34
5.2.1.	Mockups de la Interfaz Gráfica	35
5.2.2.	Funciones Genéricas	39
5.2.3.	Funciones Específicas	45
5.3.	Diseño	49
5.3.1.	Arquitectura de la aplicación	50
5.3.2.	Servicios a implementar	51
5.3.3.	Base de datos	52
5.3.4.	Capas de la aplicación	53
5.4.	Implementación	54
5.4.1.	Estructura de la aplicación	54
5.4.2.	Relación de recursos en la videoconferencia	55
5.4.3.	MongoDB	56
5.4.4.	Problemas encontrados	57
5.5.	Pruebas	58
5.5.1.	Pruebas unitarias	59
5.5.2.	Pruebas de integración	60
5.5.3.	Pruebas de sistema	60

5.5.4. Pruebas de aceptación y validación	61
6. Planificación	62
6.1. Estimación de Costes	62
6.1.1. Diagrama de Gantt	62
7. Conclusiones y trabajos futuros	64
7.1. Conclusiones	64
7.2. Trabajos futuros	65
8. Bibliografía	66
I. Glosario de términos	69
II. Manual de instalación	71
II.1. Requisitos	71
II.1.1. Hardware	71
II.1.2. Software	72
II.1.3. Requisitos del servidor	72
II.1.4. Requisitos globales de la aplicación	72
II.2. Instalación	73
III. Manual de usuario	74
III.1. Descripción de la aplicación	74
III.2. Funcionalidades del usuario	74
III.2.1. Acceder a la aplicación	74
III.2.2. Cerrar Sesión	75
III.2.3. Recuperar contraseña	76
III.2.4. Crear eventos en el calendario	77
III.2.5. Listar salas	78
III.2.6. Crear salas	79
III.2.7. Modificar salas	81
III.2.8. Eliminar salas	81
III.2.9. Listar usuarios	82
III.2.10 Crear usuario	83
III.2.11 Modificar usuarios	84
III.2.12 Eliminar usuarios	84
III.2.13 Ver calendario de otros usuarios	85
III.2.14 Crear salas administrador	86

III.2.15Videoconferencia	87
------------------------------------	----

Índice de figuras

1.	WebRTC en los navegadores[17]	2
2.	Arquitectura de WebRTC[9]	10
3.	Desarrollo SCRUM[18]	19
4.	Esquema de los flujos de datos en git.	33
5.	Mockups: Pantalla de Login	35
6.	Mockups: Pantalla de recuperación de contraseña	36
7.	Mockups: Pantalla de lista de usuarios	36
8.	Mockups: Pantalla de lista de salas	37
9.	Mockups: Pantalla de creación de usuarios	37
10.	Mockups: Pantalla de creación de salas	38
11.	Mockups: Pantalla de videoconferencia	38
12.	Caso de uso: Iniciar sesión y Cerrar sesión	41
13.	Caso de uso: Recuperar contraseña y Cambio de contraseña	42
14.	Caso de uso: Gestión de salas	45
15.	Caso de uso: Gestión de usuarios	47
16.	Caso de uso: Modificar los calendarios de todos los empleados	48
17.	Caso de uso: Conexión del cliente a la videoconferencia	49
18.	Arquitectura de la Aplicación	50
19.	Capas de la aplicación	54
20.	Diagrama de Gantt Parte 1	62
21.	Diagrama de Gantt Parte 2	63
22.	Diagrama de Gantt Parte 3	63
23.	Diagrama de Gantt Parte 4	63
24.	Pantalla de Login	75
25.	Pantalla de Recuperación de la Contraseña	76
26.	Pantalla de Modificación del Calendario	77
27.	Pantalla de Lista de Salas del Soporte	78
28.	Pantalla de Crear Salas Soporte	79
29.	Advertencia Sala Creada Correctamente	80
30.	Pantalla de Lista de Salas Administrador	80
31.	Pantalla de Modificación Sala Soporte	81
32.	Pantalla de Lista de Usuarios	82
33.	Pantalla de Creación de Usuarios	83
34.	Pantalla de Modificación de Usuarios	84
35.	Pantalla de Calendario	85

36.	Pantalla de Creación de Salas Administrador	86
37.	Acceso a la sala del soporte	87
38.	Solicitud de compartición de pantalla	88
39.	Acceso a la sala del cliente	88
40.	Pantalla de videoconferencia del soporte	89
41.	Pantalla de videoconferencia del cliente	89

1. Introducción

A lo largo de esta memoria se presentarán los detalles del problema a resolver y sus características más visibles e inmediatas, los usuarios involucrados y los requerimientos técnicos y de uso.

1.1. Motivación

Con frecuencia surge la necesidad de dar soporte técnico a un cliente que no siempre dispone del conocimiento tecnológico adecuado, ya sea por su nivel de trabajo o la dependencia de un software en este. Es en ese momento en el que dar soporte técnico se convierte en una tarea dura y complicada, ya que no solo se necesita encontrar la solución sino que también indicar y guiar al usuario con poco conocimiento técnico a través de las distintas pruebas y comprobaciones que se realizan para encontrarla.

Para dar solución a este incidente surgen una serie de herramientas que simplifican esta tarea, pero todas estas presentan el mismo defecto y es que la gran mayoría de las herramientas actuales requieren la instalación de software más o menos complejo por parte del cliente. Por otro lado, muchas de estas soluciones no son multiplataforma. Esto provoca que aumente nuestro ya complicado problema por lo que tenemos que encontrar la manera de que el cliente pueda acceder a un soporte técnico de manera sencilla y que este le permita resolver sus problemas sin tener que derrochar demasiado tiempo.

Por estos motivos se propone el uso de WebRTC, ya que sólo exige tener instalado un navegador web actualizado. Además, se trata de una solución multiplataforma. De esta manera no se le complica el acceso ni al cliente ni al técnico, lo que permite una comunicación cliente-técnico sencilla y se obtiene así un servicio técnico eficiente y eficaz.



Figura 1: WebRTC en los navegadores[17]

1.2. Objetivos

Este proyecto tiene como objetivo principal el desarrollar un sistema para soporte técnico utilizando la tecnología WebRTC. Esta tecnología presenta la ventaja de permitir llevar a cabo el intercambio de contenidos multimedia directamente entre navegadores, sin necesidad de instalar ningún software adicional.

Para alcanzar este objetivo global será necesario cumplir los siguientes objetivos específicos:

- Se desarrollará una aplicación web que permita la compartición de escritorio de forma remota y segura del cliente al técnico.
- Se empleará únicamente el propio navegador web.
- Además, la aplicación web permitirá administrar las salas, la gestión de usuarios, el control de acceso y la gestión de eventos.

1.3. Estructura del documento

El presente documento se compone de ocho apartados que describen las tecnologías y metodologías empleadas en el proyecto, además de explicar el desarrollo de este. Cuenta también con tres apéndices que complementan su utilidad, siendo estos un glosario, un manual de instalación y un manual del usuario.

1.3.1. Apartados

En esta parte se describirá brevemente la estructura y el contenido de los ocho apartados:

- **Introducción**

Se analiza la motivación y los objetivos a cumplir en el proyecto. Además, se describe la estructura de la memoria.

- **Estado del arte**

Se muestran aplicaciones ya existentes que emplean videoconferencias. Se analizan también sus principales características.

- **Fundamentos tecnológicos**

En él se exponen las tecnologías empleadas para la realización del proyecto.

- **Metodología**

Se describe la metodología de desarrollo Scrum, empleada en la organización del desarrollo del proyecto. Se emplea una versión de Scrum adaptada para este proyecto y se describen los *Sprints* o fases en las que se dividió el desarrollo.

- **Desarrollo**

Se explica todo lo referente a la aplicación. Sigue las fases del paradigma de ingeniería de software:

- **Preparación:** Presenta el entorno de desarrollo y la arquitectura de la aplicación.
- **Análisis:** Se realiza un análisis más en profundidad de los requisitos.
- **Diseño:** Incluye el diseño conceptual de la aplicación.
- **Implementación:** Explica las decisiones y el trabajo relacionado con el software empleado.

- **Planificación**

Se realiza la estimación de costes de la realización del proyecto. Se incluye también un diagrama de Gantt.

- **Conclusiones y trabajos futuros**

Se exponen las conclusiones del proyecto y también las líneas futuras de desarrollo y posibles funcionalidades a añadir. Estas funcionalidades se han decidido no añadir o bien por limitaciones de la tecnología o por falta de tiempo para incluirlas.

- **Bibliografía**

En este apartado se recogen todas las referencias a información empleada en el proyecto y memoria.

1.3.2. Apéndices

En esta parte se describirá brevemente la estructura y el contenido de los apéndices.

- **Glosario de términos**

En él se definen términos que no se han explicado en la propia memoria, es decir que no son propios de la aplicación sino que se hace referencia a ellos.

- **Manual de instalación**

En él se describe el proceso de instalación y arranque de la aplicación, además de indicar sus requisitos.

- **Manual de usuario**

En este apartado se describen las funcionalidades del usuario en la aplicación. Se explican todas sus posibles interacciones.

2. Estado del arte

Actualmente no existen aplicaciones enfocadas directamente a la impartición de soporte técnico en línea por videoconferencia mediante el uso del navegador, aunque sí existen varias soluciones que optan por instalar software en los equipos que los utilizan. La mayor parte de estas aplicaciones ofrecen una prueba de manera gratuita (suelen ser pocas funcionalidades) y venden el producto completo (todas las funcionalidades). La iniciativa de código abierto en este ámbito parece tener aún poco recorrido. A continuación se explicarán una serie de aplicaciones que son muy utilizadas en la actualidad.

2.1. Skype[1]

Es el software para videoconferencia por excelencia y el más popular, debido a su versión gratuita. Este requiere la instalación de un paquete software. La comunicación se basa en un protocolo propietario de Microsoft y el flujo de datos durante la conferencia es de tipo P2P. Actualmente Skype permite las videoconferencias de manera gratuita y dispone de planes de pago para su servicio de llamadas (el precio varía en función del país al que se quiera llamar).

2.2. Hangouts[2]

Es una herramienta que pertenece a Google. Permite realizar videoconferencias y agrupa los servicios de Google tales como: Google Talk, Google +, Messenger y Google Hangouts. No necesita la instalación de ningún software pero sí un *plugin* para el navegador debido a que se accede vía web. La comunicación se basa en un protocolo propietario, creado tras el abandono de XMPP en el que se basaba Google Talk.

2.3. Mikogo[3]

Mikogo es una aplicación que permite la compartición de escritorio orientado a reuniones online y conferencias web. Mikogo ofrece su software como descargas nativas para Windows, Mac OS X, Linux, iOS y Android. El software es multiplataforma. De esta manera se permite a un organizador acoger la reunión en un Windows y sus asistentes pueden unirse desde un Mac, Linux, además de teléfonos y tabletas. La interfaz del software es multilingüe. Mikogo dispone de una versión gratuita para un organizador y un participante por sesión con características estándar. Para adquirir el producto completo se dispone de dos paquetes de compra: uno profesional (1 persona y 25 participantes por sesión con características premium por 15€/mes) y otro para empresas (su precio se determina en función de los organizadores y asistentes elegidos y las características seleccionadas).

2.4. Cisco WebEx[4]

Cisco WebEx es una compañía que ofrece la colaboración bajo demanda, de reunión en línea, conferencias web y aplicaciones de videoconferencia. Sus productos incluyen centro de reuniones, centro de formación, centro de eventos, centro de soporte, centro de ventas, MeetMeNow, PCNow, WebEx AIM Pro Business Edition, WebEx WebOffice, y WebEx Connect. Todos los productos de WebEx son parte de la cartera de colaboración de Cisco. Cisco WebEx dispone de una versión gratuita para 2 personas. Para incluir a más personas y más funcionalidades se dispone de una serie de planes con distintos precios.

2.5. GoToMeeting[5]

GoToMeeting permite reuniones en línea, compartición de escritorio y videoconferencias a través de Internet en tiempo real. Dispone de una versión gratuita que dura 14 días y distintos planes de pago cuyo precio varía en función de los participantes por sesión. Cuanto mayor es el precio más características incluye el plan.

2.6. TeamViewer[6]

TeamViewer es un software cuya principal función es conectarse remotamente a otro equipo. Incluye además: compartir y controlar escritorios, reuniones en línea, videoconferencias, transferencia de archivos entre ordenadores y acceso a un equipo remoto mediante un navegador web. Aunque no sean sus principales actividades también incluye funciones de trabajo en equipo y presentación. Dispone de una versión gratuita para pruebas muy simples. El programa completo se vende en distintas licencias con diversas limitaciones.

2.7. Facetime[7]

Es una aplicación de videollamada incluida como parte del conjunto estándar de aplicaciones en los sistemas operativos de Apple. Permite también la realización de videoconferencias y llamadas VoIP. Se requiere iOS o Mac OS X 10.6.6 o superiores. Emplea un protocolo basado en un conjunto de estándares abiertos, por lo que es totalmente gratuito.

2.8. Microsoft Lync[8]

Microsoft Lync (conocido actualmente como Skype Empresarial) es un servicio que permite mensajería instantánea, orientado a entornos empresariales. Emplea un protocolo basado en SIP/SIMPLE. Ofrece entre otras características: conversación instantánea, compartición de archivos, llamadas VoIP y videoconferencia múltiple entre usuarios del mismo servicio, además de sincronización con el resto de herramientas de Microsoft Office. Estas herramientas permiten, por ejemplo, comprobar si otras personas están trabajando en un mismo documento o sincronizar los contactos con Outlook.

3. Fundamentos tecnológicos

A continuación se van a explicar todas tecnologías empleadas en la realización de la aplicación. Se explicara tanto su estructura como su funcionamiento.

3.1. WebRTC

WebRTC (Web Real-Time Communication) es un framework de código abierto que permite comunicaciones en tiempo real desde el navegador sin la necesidad de instalar ningún componente adicional. Incluye los bloques fundamentales para las comunicaciones de alta calidad a través de web como son los componentes de red, audio y vídeo utilizados en aplicaciones de conversación por voz y vídeo. Estos componentes, cuando están implementados en un navegador, pueden ser accedidos a través del interfaz JavaScript, permitiendo a los desarrolladores implementar su propia aplicación RTC fácilmente. El esfuerzo de WebRTC está siendo estandarizado a nivel interfaz por el W3C y a nivel protocolo por el IETF.

Por otra parte, está formado por un conjunto de interfaces ofrecidas por JavaScript que están integradas en los principales navegadores (Chrome, Firefox y Opera) pese a ser un proyecto reciente, en continuo desarrollo y con cierta falta de estabilidad. Estas interfaces permiten obtener imagen y sonido empleando la cámara y micrófono o la propia pantalla del ordenador, además de codificar la emisión. La ventaja de estar estandarizado e incluido en los principales navegadores hace que no sea necesario ningún software adicional para emplearlo.

3.1.1. Arquitectura WebRTC[9]

WebRTC dispone de una implementación e interfaz estandarizadas, sin coste alguno. Establece que debe ser abierta para su extensión, pero cerrada para su modificación. La arquitectura se compone de:

- **Transport:** Un framework que se encarga de establecer los protocolos de comunicación. Está compuesto de:

- **Stack RTP (SRTP):** Una implementación del protocolo estándar RTP.
- **STUN, TURN y ICE:** Protocolos que permiten establecer la comunicación entre dos equipos a través de diferentes redes.
- **Session Management:** Consiste en un modelo abstracto que deja el trabajo de definir un protocolo de gestión de sesiones al desarrollador web.
- **VoiceEngine:** Un framework para la transmisión de sonido, abarcando desde la tarjeta de sonido hasta la emisión por red. Incluye los siguientes codecs:
 - **ISAC:** Orientado a banda ancha.
 - **ILBC:** Orientado a bitrates reducidos.
 - **Opus:** Soporta bitrates constantes, variables, de distintos tamaños de frame y con una frecuencia de muestra que puede cubrir el rango del oído humano. Definido en el RFC 6176 de la IETF.
 - **NetEQ for Voice:** No es más que un buffer de eliminación de jitter y defectos de sonido por pérdida de paquetes, orientado a reducir lo máximo posible la latencia mientras se conserva la calidad de la voz.
 - **Echo Canceled:** Consiste en un algoritmo que procesa la señal de entrada del micrófono, eliminando en gran medida el eco que se produce por la retroalimentación. Anula la reproducción retrasada en los altavoces del equipo.
 - **Noise Reduction:** Un componente que trata de reducir distintos ruidos de fondo típicos de VoIP.
- **VideoEngine:** Un framework análogo al VoiceEngine, pero orientado a la captura y transmisión de vídeo. Incluye el códec VP8. Está compuesto de:
 - **Video Jitter Buffer:** Consiste en un buffer equivalente a NetEQ for Voice, pero orientado a la corrección de vídeo.
 - **Image Enhancements:** Reduce el ruido procedente de la cámara.

A continuación se muestra una imagen donde se refleja la estructura WebRTC descrita anteriormente.

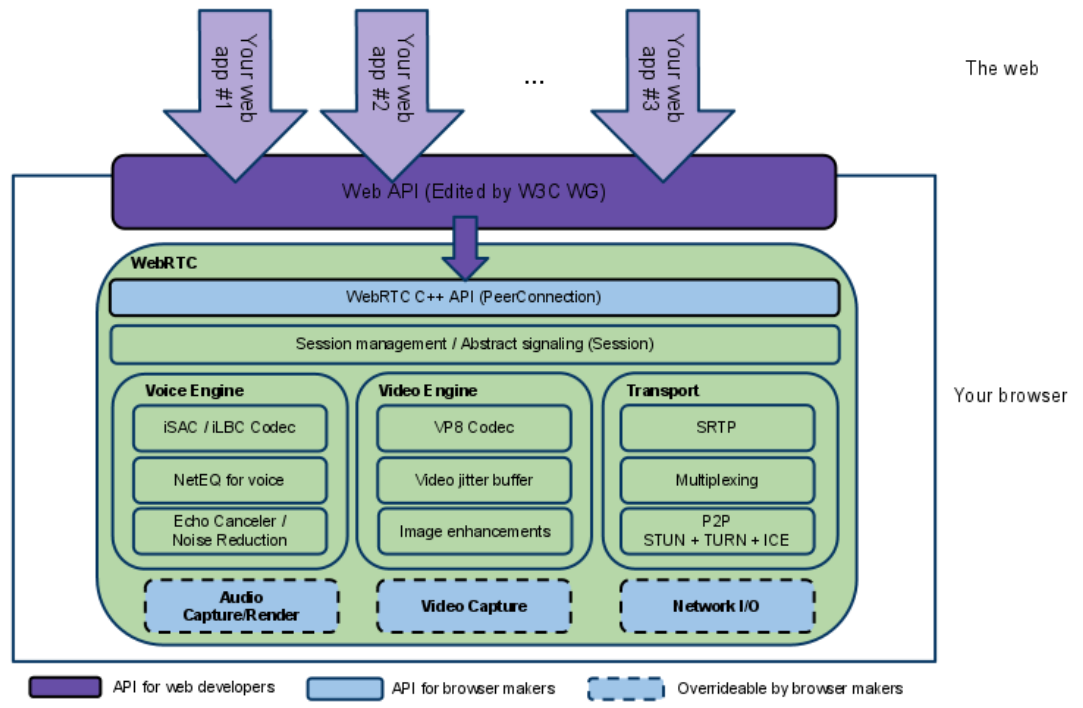


Figura 2: Arquitectura de WebRTC[9]

3.2. Licode[10]

Licode es un proyecto que emplea la tecnología WebRTC. La arquitectura del framework Licode se divide en los siguientes módulos:

- **Erizo:** Proporciona la unidad de control multipunto (Multipoint Control Unit, MCU). Está implementado en C++ y es compatible con los formatos y estándares empleados en WebRTC.
- **Erizo Client:** Es la implementación que se usa en el navegador del proyecto en lenguaje JavaScript. Se comunica con Erizo Controller y con la interfaz WebRTC del cliente mediante una capa abstracta.

- **Erizo Controller:** Maneja las salas, tokens de invitación y mecanismos de seguridad.
- **Erizo API:** Funciona como conexión entre Node.js y Erizo. Permite configurar y administrar todas las características de la MCU de Erizo desde la aplicación Node.js.
- **Nuve:** Consiste en una implementación de Licode como un servicio de videoconferencia utilizable por aplicaciones de terceros. Permite de este modo usar una interfaz que facilita la integración de la funcionalidad de videoconferencia y registro de servicios. Además, permite la escalabilidad del servicio en la nube. Licode introduce conceptos como salas, usuarios o flujos. Cada uno de estos conceptos tiene una clase JavaScript asociada, cuyos objetos abstraen y simplifican el manejo de la aplicación.
- **Erizo.Stream:** Representa un flujo de comunicación que un usuario transmite al resto de participantes. Licode da la posibilidad de transmitir audio, vídeo y datos. Todos los medios son opcionales y el de datos puede ser utilizado para enviar cualquier información (archivos o mensajes de texto).
- **Erizo.Room:** Las salas de Licode se comportan como una sala física. Los participantes de cada sala registran su flujo para compartirlo con el resto de asistentes en la misma sala. También será cada uno notificado del registro o eliminación de flujos de los otros usuarios en la sala, pudiendo inscribirse individualmente a cada uno para su recepción o ignorarlos.
- **Erizo.Event:** Los eventos son objetos que indican cambios de estado en los clientes. De este modo es sencillo añadir manejadores a los distintos eventos que pueden suceder por parte de los clientes.
- **Usuario:** Se corresponde con el usuario de la aplicación final. Cada usuario puede participar en las salas como una unidad para transmitir y/o recibir un flujo multimedia. El usuario recibe un token de un único uso que va vinculado a la sala a la que puede acceder. Este token es subministrado desde el servidor (Nuve), garantizando control de acceso a las salas. Los usuarios en Licode tienen un rol el de “presenter” (con derecho a registrar y enviar flujos).

3.3. EasyRTC[11]

EasyRTC es un stack WebRTC completo con licencia libre, desarrollado por Priologic Software. Se divide en dos partes: EasyRTC Client y EasyRTC Server.

- **EasyRTC Client:** Es el equivalente a Erizo Client en Licode. Se trata de una librería en Javascript que se ejecuta en el navegador, que nos permite conectarnos a EasyRTC Server.
- **EasyRTC Server** Está escrito en Node.js, y es equivalente al resto de Licode. Se incorpora a la aplicación como una librería más, y con unas pocas líneas de código, se despliega el servidor EasyRTC, a diferencia de Licode que se trataba de un servicio aparte.

Cabe destacar, que a pesar de gestionar salas, al igual que Licode, aquí el token de acceso no corresponde a la sala, sino a otro interlocutor, y mediante este token el servidor nos conecta a dicho participante, si éste acepta la comunicación, como si se tratase de una llamada telefónica convencional.

3.4. Otras implementaciones

A la hora de la implementación de la pila, se busca aquel framework que esté bastante desarrollado y sea estable pese a que el estándar de WebRTC no esté del todo definido. El principal problema es que estos proyectos WebRTC suelen cambiar continuamente. Debido a esto, se eligió en un primer momento Licode como la pila. Existían otras alternativas como Kurento[19], pero no está tan desarrollado como Licode. Muchos de los competidores son de código cerrado, con cobro por licencia de uso, como OpenTOK o las soluciones de Ecosmob, y ofrecen una funcionalidad similar a la de Licode por lo que son descartados.

Durante la mayor parte del proyecto, la aplicación, fue desarrollada y orientada a la integración del framework de Licode para la videoconferencia. La parte servidora de Licode fue integrada con la base de datos de MongoDB y probando las funciones con esta. Una vez se comprobó que la parte servidora funcionaba se comenzó la parte cliente, siendo aquí donde se encontraron los problemas.

Como se ha referido anteriormente, la idea inicial era utilizar Licode y finalmente se migró a EasyRTC. Esto fue debido, a un problema al implementar SSL con Licode (5.4.4).

3.5. Node.js[12]

Node.js es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, que lo hace liviano y eficiente. El ecosistema de paquetes de Node.js, npm, es el más grande de librerías de código abierto en el mundo.

Al contrario que la mayoría del código JavaScript, no se ejecuta en un navegador, sino en el servidor. Node.js implementa algunas especificaciones de CommonJS.5 y incluye un entorno REPL para depuración interactiva.

3.5.1. Módulos Node

Node.js es solo un motor básico que necesita de distintos módulos para emplear funcionalidades específicas. En este caso los módulos utilizados en el proyecto son:

- **Express[20]**: Es el framework de desarrollo web más común en Node.js. Se caracteriza por su sencillez a la hora de crear una aplicación web desde cero. Además, se considera robusto y estable para entornos de producción.
- **Npm[21]**: Es el administrador de paquetes para JavaScript.
- **Body-Parser[22]**: Establece en el sistema el formato JSON como estructura de ficheros.
- **Jade[23]**: Es un motor de plantillas HTML para Node.js. La principal ventaja frente a utilizar ficheros HTML estáticos es el procesado en el momento de renderizar la plantilla: se permite utilizar condicionales o iteraciones y modificar el resultado en base a ello. Se trata de un lenguaje más simple que HTML que tiene en cuenta el tabulado del código para su estructura.

- **Cookie-Parse**[24]: Permite emplear el objeto `req.cookies` para el almacenamiento de las cookies del navegador. Opcionalmente puede habilitar el soporte de cookies firmadas pasando una cadena secreta, que asigna `req.secret` para que pueda ser utilizado por otro middleware.
- **Crypto**[25]: El módulo `Crypto` proporciona funcionalidad criptográfica, que incluye un conjunto de envolturas para las funciones `hash`, `HMAC`, `cifrado`, `descifrado`, `firma` y `verificación de OpenSSL`.
- **Easyrtc**[26]: un paquete de `WebRTC` de código abierto que incorpora una API de cliente e instalación de un servidor de `EasyRTC` además del código fuente de la aplicación en funcionamiento. Funciona sobre `HTML5` y `JavaScript` bajo una licencia `BSD 2`.
- **ErrorHandler**[27]: `Error-handler` es un módulo `Node.js` para manejar errores de servidor de una manera elegante.
- **Express-Flash**[28]: `Flash` es una extensión de `connect-flash` con la capacidad de definir un mensaje flash y renderizarlo sin redirigir la solicitud.
- **Express-Session**[29]: Crea un middleware de sesión con las configuración que deseemos.
- **Method-Override**[30]: Permite usar comandos `HTTP` como `PUT` o `DELETE` en lugares donde el cliente no lo admite.
- **Moment**[31]: Una biblioteca de fechas de `JavaScript` para analizar, validar, manipular y dar formato a estas.
- **Mongodb**[32]: El controlador oficial de `MongoDB` para `Node.js`. Proporciona una API de alto nivel en la parte superior de `mongodb-core` que está destinada a los usuarios finales.
- **Mongoose**[33]: Es una herramienta de modelado de objetos de `MongoDB` diseñada para trabajar en un entorno asíncrono.
- **Morgan**[34]: Interfaz de registro de peticiones `HTTP` para `Node.js`.
- **Nodemailer**[35]: Permite enviar correos electrónicos desde `Node.js`.
- **Sprintf-js**[36]: Es una implementación completa de código abierto `JavaScript` para el navegador y `Node.js`.

- **Stylus**[37]: Es un nuevo lenguaje que proporciona una forma eficiente, dinámica y expresiva de generar CSS. Soporta sintaxis de indentación y el CSS convencional.
- **Fs**[38]: Permite a Node.js trabajar con el sistema de archivos del ordenador.
- **Https**[39]: Un contenedor que elige HTTP o HTTPS para las solicitudes.

3.6. JavaScript[13]

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (Server-Side JavaScript o SSJS). Su uso en aplicaciones externas a la web como, por ejemplo, en documentos PDF o aplicaciones de escritorio (mayoritariamente widgets), es también significativo. Desde el 2012, todos los navegadores modernos soportan completamente ECMAScript 5.1, una versión de JavaScript.

JavaScript se diseñó con una sintaxis similar a C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo, Java y JavaScript tienen semánticas y propósitos diferentes. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM). Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con ayuda de otras tecnologías como AJAX. JavaScript se interpreta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

3.7. Sublime Text[14]

Sublime Text es un editor de texto y editor de código fuente. Está escrito en C++ y Python para los plugins. Desarrollado originalmente como una extensión de vi, con el tiempo fue creando una identidad propia. Por este motivo aún conserva un modo de edición tipo vi llamado Vintage mode. Se puede descargar y evaluar de forma gratuita. Sin embargo, no es software libre o de código abierto y se debe obtener una licencia para su uso continuado, aunque la versión de evaluación es plenamente funcional y no tiene fecha de caducidad. Actualmente se encuentra en la versión número 3.

■ Principales características

- **Minimapa:** Consiste en una previsualización de la estructura del código. Es muy útil para desplazarse por el archivo cuando se conoce bien la estructura de este.
- **Soporte nativo para infinidad de lenguajes:** Soporta de forma nativa 43 lenguajes de programación y texto plano.
- **Soporte de Snippets y Plugins:** Añade soporte a los snippets, que son similares a las macros o los bundles y a una gran cantidad de plugins.
- **Pestañas:** Se pueden abrir varios documentos y organizarlos en pestañas.
- **Resaltado de paréntesis e indentación:** Cuando el usuario coloca el cursor en un paréntesis, corchete o llave, resalta ésta y el paréntesis, corchete o llave de cierre o apertura correspondiente.
- **Coloreado y envoltura de sintaxis:** Si se escribe en un lenguaje de programación o marcado, resalta las expresiones propias de la sintaxis de ese lenguaje para facilitar su lectura.

3.8. Bases de datos

En el momento de guardar y gestionar datos en nuestra aplicación nos encontramos con dos posibles tipos de bases de datos:

- **Base de datos relacional** La base de datos relacional (BDR) es un tipo de base de datos (BD) que cumple con el modelo relacional. En el modelo relacional, la base de datos tiene una estructura rígida, que requiere un tipo concreto para cada valor y exige la inclusión de ciertos campos para realizar una inserción.
- **NoSQL** NoSQL es un término que hace referencia a un nuevo enfoque en los Sistemas de Gestión de Bases de Datos (SGBD). NoSQL permiten la inclusión de datos en un formato que queda a elección del cliente en el momento de la inserción. No se restringe la aceptación de las nuevas entradas según los campos que realmente se hayan definido. Ni siquiera es necesario definir con antelación la estructura o qué campos existen. De este modo, se podría considerar que cada inserción es un objeto independiente, que puede tener o no campos en común con el resto. Un gestor NoSQL muy conocido en el desarrollo web, especialmente cuando se trabaja con JavaScript, es MongoDB.

A continuación se presentarán las herramientas probadas para realizar la implementación de estas dos posibles alternativas:

3.8.1. MongoDB

MongoDB es un sistema de base de datos NoSQL orientado a documentos. En lugar de guardar los datos en tablas, como se hace en las bases de datos relacionales, este guarda estructuras de datos en documentos similares a JSON con un esquema dinámico (MongoDB utiliza una especificación llamada BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

3.8.2. MariaDB

MariaDB es un sistema de gestión de bases de datos relacional derivado de MySQL con licencia GPL. Tiene una alta compatibilidad con MySQL, ya que posee las mismas órdenes, interfaces, APIs y bibliotecas, siendo su objetivo poder cambiar un servidor por otro directamente.

3.8.3. Elección de la base de datos

En el momento de elegir una topología u otra se decidió optar por una topología NoSQL debido a principalmente a carecer de una estructura definida de la base de datos (su estructura y tipos de datos). Se escoge la herramienta de MongoDB por su compatibilidad con el framework de Licode.

4. Metodología

En este capítulo se explica la metodología empleada para el desarrollo del proyecto y se detallan las iteraciones o fases en las que se ha dividido.

4.1. SCRUM[15][16]

Para la realización de este proyecto se utilizará la metodología ágil de software Scrum. Esta metodología se basa en un proceso iterativo e incremental que permite acortar significativamente los ciclos de desarrollo y asegura que al final de cada ciclo se dispondrá de un software funcional. Scrum es una metodología adecuada para este proyecto debido a que se va a trabajar con requisitos que podrían variar durante el desarrollo, se utilizarán tecnologías experimentales que están cambiando de forma continua y se quieren obtener productos funcionales desde las etapas más tempranas del desarrollo del proyecto.

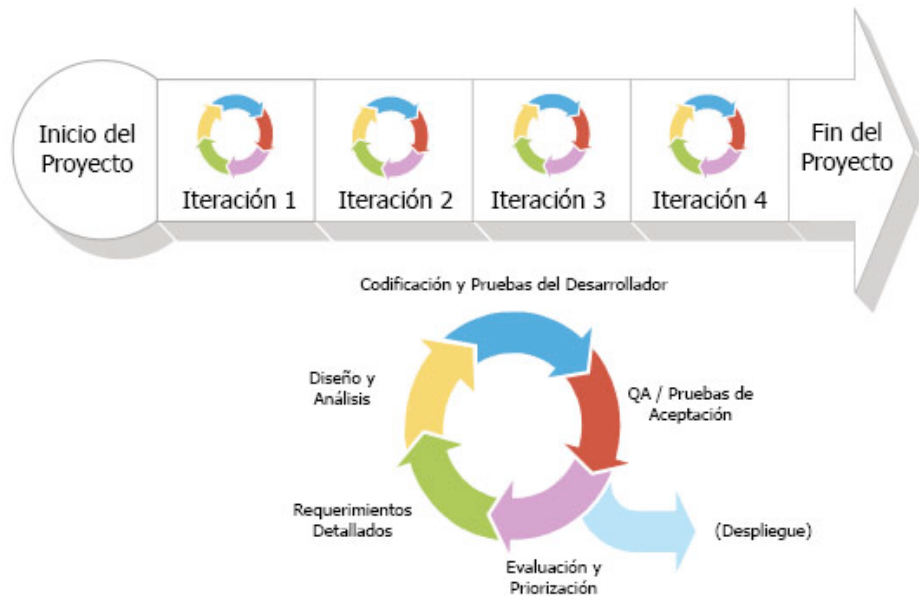


Figura 3: Desarrollo SCRUM[18]

4.2. Roles

Hay tres roles principales en el modelo Scrum:

- **Product Owner (cliente):** Representa el cliente, aunque el rol puede ser tomado por un encargado de marketing, el líder de proyecto u otra persona con conocimientos sobre el ámbito y requisitos del producto final. También ha de controlar la financiación del proyecto.
- **Scrum Master:** Es el encargado de eliminar los obstáculos que dificulten la llegada a las metas definidas. En cierto modo es como un líder de equipo, pero este no organiza al resto de trabajadores del proyecto. El Scrum Master debe ser un miembro del equipo de trabajo.
- **Equipos de trabajo:** Constituido por todos los trabajadores responsables del resultado final. Conocedores del proyecto Scrum, el equipo ha de ser multidisciplinar, cubriendo todos los campos necesarios para realizar el proyecto. Se debe gestionar, planificar y organizar para alcanzar los objetivos de cada sprint. El tamaño recomendado es de entre 3 y 9 individuos, con habilidad para realizar los trabajos requeridos, tanto de análisis, como de diseño, implementación, pruebas, documentación, etc.

Opcionalmente, se pueden añadir los llamados roles auxiliares”, que aparecen de forma esporádica durante la ejecución de proyecto. Estos roles no necesitan tener conocimiento exhaustivo del modelo de trabajo empleado.

4.3. Terminología

Existen una serie de términos propios para definir elementos relacionados con Scrum. Los más relevantes son:

- **Equipo:** Formado por el Product Owner, el Scrum Master y el equipo de trabajo.
- **Backlog del producto:** Un documento público el cual está disponible para todos los involucrados en el proyecto, pero solo es editable por el Product Owner. Está compuesto por una lista exhaustiva de todos

los objetivos del proyecto, priorizados por el ROI (Retorno de Inversión). Incluye también estimaciones y objetivos desde el punto de vista económico.

- **Sprint:** El nombre de cada iteración de Scrum.
- **Tareas:** Un trabajo del cual se obtiene un resultado que ayuda en el avance del proyecto. Se trata de una división del trabajo cuya duración máxima es de 16 horas. En caso de una duración mayor esta se subdivide en otras de menor alcance.
- **Backlog del Sprint:** Es la lista de tareas que se han de realizar en el Sprint. En Scrum las tareas no son asignadas a ninguna persona o grupo, sino que los miembros del grupo de trabajo las toman para sí cuando lo ven adecuado y tienen disponibilidad.

4.4. Sprints

Usando la metodología Scrum el objetivo del proyecto se alcanza mediante iteraciones breves, los Sprints. La duración y objetivos de cada Sprint son definidos con antelación en los distintos encuentros de planificación.

Tras comenzar el trabajo, se realizan reuniones diarias con el equipo para decidir y planificar el trabajo del día siguiente. En estas reuniones diarias la gerencia debe estar presente para comprobar los avances diarios del proyecto y ser informada de los problemas que se han encontrado y su posible solución.

La duración de los Sprints puede variar entre 15 y 60 días. Si su duración fuera mayor perdería la flexibilidad y una rápida respuesta que permite esta metodología. Cuando se finaliza el plazo se realiza una reunión retrospectiva, en la que se analizan los objetivos cumplidos y si el avance es el adecuado. Cada iteración produce un incremento del trabajo el cual se debe mostrar al cliente.

4.5. Reuniones

Existen diversas reuniones en la metodología Scrum:

- **Planificación del Sprint:** Se determinan los trabajos y resultados esperados en cada iteración. Se realiza al principio de cada Sprint y no superará las 8 horas de duración.
- **Diaria:** Una reunión diaria de 15 minutos que debe ser en el mismo lugar y hora siempre. Se debe realizar sin la falta de miembros del equipo y de manera puntual. En estas reuniones son los roles los que tiene la palabra, se define lo hecho el día anterior a la reunión, existencia de obstáculos, posibles soluciones y lo que se espera alcanzar el día siguiente.
- **Revisión del Sprint:** Se realizan al terminar la iteración, con un límite de duración de 4 horas. En esta reunión se revisa el trabajo realizado hasta el momento y el planificado pero no terminado. El trabajo terminado se presenta a aquellos interesados mientras que los no finalizados no se enseñan.
- **Retrospectiva:** Se realiza al final de cada Sprint y es convocada por el Scrum Master. Todos los integrantes del equipo dan su impresión sobre el transcurso de cada iteración con la finalidad de identificar los aspectos que han ido bien y aquellos que se pueden mejorar de cara a la próxima iteración.

Existen otros encuentros que pueden realizarse en función de las características del proyecto como, por ejemplo, las llamadas "Scrum de Scrum", en las que los representantes de cada grupo se coordinan diariamente.

4.6. Adaptación de Scrum para este proyecto

Después de presentar la metodología de Scrum se pueden ver puntos que no son del todo realizables en este proyecto de Fin de Grado. Un proyecto de este tipo requiere una planificación y unas fases más estrictas que las que propone Scrum. El proyecto debe constar como mínimo de:

- **Realización de anteproyecto:** Es inevitable su realización, en la que se deben exponer ya unos objetivos finales definidos en su cierta medida. La existencia de un documento de requisitos no concuerda con la metodología de Scrum, donde los objetivos no se establecen de forma tan rígida y se pueden modificar en cada Sprint.
- **Realización del proyecto:** Desde el punto de vista de análisis, diseño e implementación del producto o resultado.
- **Revisión y entrega del proyecto:** En este caso el tribunal valorará el proyecto en su finalización. En un proyecto Scrum una parte importante es la evaluación por parte del cliente o destinatario en cada Sprint. En este caso el Proyecto de Fin de Grado no se puede considerar como destinatario al tribunal encargado de valorarlo.

Se ha decidido modificar el modelo Scrum en cierta medida, conservando su filosofía e ideas. Estos cambios realizados en el modelo son:

- **Duración de los Sprints:** Se establece entre una semana natural (7 días) o dos semanas naturales (14 días), dándose el caso incluso de periodos mayores como máximo de 1 mes (30 días). Estos Sprints son breves pero al no tener reuniones diarias para resolver problemas se necesita esta brevedad para encauzar rápidamente cualquier problema u obstáculo.
- **Cambios en los roles:** Para este proyecto se reparten de la siguiente forma:
 - **Producto Owner:** Se asigna al director del proyecto, Diego Fernández Iglesias, por haber propuesto la aplicación de emplear la tecnología WebRTC para soporte técnico y haber observado la falta de una aplicación gratuita que no empleara la instalación de software.
 - **Scrum Manager:** Se asignará al director del proyecto Diego Fernández Iglesias, ya que se encarga principalmente de supervisar la realización e implementación de la aplicación desarrollada.
 - **Equipo de trabajo:** Será el alumno, Diego Otero González, ya que es requisito que el trabajo sea realizado solo por él, siendo el director una guía y en ocasiones ayuda para ello.

- **Planificación de las reuniones y Sprints:** Se realizará de la siguiente forma:
 - **Planificación de Sprint:** En las reuniones entre el director y el alumno se establece qué trabajo se realizará en ese sprint. Dependiendo de la carga de trabajo la duración de este puede variar.
 - **Reunión diaria:** Dadas las limitaciones de horarios y tiempo tanto del alumno como del director, es necesario eliminar las reuniones diarias y alargar la duración de las iteraciones.
 - **Retrospectiva:** En cada reunión se analiza y evalúa el trabajo realizado y los obstáculos encontrados durante la iteración anterior.

4.7. Iteraciones

En esta sección se indican todas las iteraciones en el desarrollo del proyecto, con las fechas de inicio y fin de cada una. No se indican las características importantes del trabajo ni se profundiza en los detalles, sino simplemente se muestra un registro y calendario de evolución del proyecto. Cabe destacar que existen espacios entre los Sprints, ya que coincidieron con algún evento ajeno al proyecto (exámenes, fechas de entrega de prácticas...), por lo que esos días no se avanzó en el mismo.

4.7.1. Sprint 1: Estudio de herramientas

Fecha de inicio: 02/02/2017

Fecha de fin: 16/02/2017

Sprint log:

- Realizar un estudio de las herramientas que emplean WebRTC del mercado y con funcionalidades parecidas a la nuestra.

En este primer Sprint se buscan en el mercado aplicaciones que empleen videoconferencia para de esta manera observar sus ventajas y defectos. También se analiza si existe ya alguna aplicación que realice lo propuesto. Si ese fuese el caso tendríamos que cambiar la funcionalidad. Los datos obtenidos en este Sprint se reflejan en la sección de Estado del arte (1.3.2) descrita al principio de la memoria.

4.7.2. Sprint 2: Estudio de las tecnologías

Fecha de inicio: 16/02/2017

Fecha de fin: 06/03/2017

Sprint log:

- En este Sprint se busca realizar una toma de contacto con las tecnologías que se van a emplear en el proyecto.

Para la realización del Sprint se consultaron un serie de guías, tutoriales y libros entre ellos "JavaScript: The Good Parts" [\[40\]](#). La idea era probar a hacer un ejemplo básico tanto de Node.js como de MongoDB. En ese momento se decidió emplear un sistema operativo Ubuntu 14.04 debido a que más adelante emplearíamos Licode y es la recomendación de los propios desarrolladores de Licode como distribución Linux.

Se instaló Node.js junto a express y MongoDB. En un primer intento se probó MongoDB desde el propio terminal para tratar de ver su funcionamiento. Más adelante se creó un ejemplo en Node.js en el que se incorporó una conexión con MongoDB empleando mongoose para ello. Con la ayuda de mongoose se pudo operar con la base de datos de manera más sencilla.

4.7.3. Sprint 3: Realización de mockups

Fecha de inicio: 06/03/2017

Fecha de fin: 16/03/2017

Sprint log:

- Realizar una serie de mockups para hacer una captura de requisitos inicial.

Durante la realización de este Sprint se empleó la herramienta Balsamiq Mockups[41] para el diseño de los mockups, que se mostrarán más adelante en la memoria (5.2.1).

4.7.4. Sprint 4: Definir una estructura y funcionalidad básica

Fecha de inicio: 16/03/2017

Fecha de fin: 30/03/2017

Sprint log:

- Realizar la estructura básica del proyecto y probar una funcionalidad básica.

Para la realización de este Sprint se empleó Sublime Text, que permitió crear una estructura básica de la aplicación. Se creó un diseño jerárquico basado en los ejemplos de express. Se probó también a crear y arrancar un ejemplo de aplicación.

4.7.5. Sprint 5: Realizar la autenticación frente a la aplicación

Fecha de inicio: 30/03/2017

Fecha de fin: 06/04/2017

Sprint log:

- Realizar el caso de autenticación para la aplicación, guardar usuarios en la base de datos y comprobar su correcto inicio y cierre de sesión.

En este Sprint se incorporó a la aplicación MongoDB como se había probado con anterioridad en un ejemplo. Se necesita que cuando arranque la aplicación también arranque la conexión con nuestra base de datos. Con la ayuda de MongoDB se creó el esquema de definición de los usuarios que se emplea para la creación de estos.

Para el caso de autenticación se diseñó una vista de login y se comprobó que permitía tanto iniciar como cerrar sesión, validando los datos contra la base de datos. Se añadió también la posibilidad de guardar las cookies y mantener al usuario siempre logueado. Cuando se estaba empleando MongoDB se encontró que se producían errores algunas veces al tratar los datos. Se optó por buscar y descargar una versión más estable.

4.7.6. Sprint 6: División de personal de soporte

Fecha de inicio: 06/04/2017

Fecha de fin: 13/04/2017

Sprint log:

- Realizar la división entre personal de soporte y administrador.

Para este Sprint se diseñaron pantallas de inicio distintas para el soporte y administrador. Se separó la funcionalidad de estos y se incorporó un campo rol para diferenciarlos en la base de datos.

4.7.7. Sprint 7: Administración de usuarios

Fecha de inicio: 13/04/2017

Fecha de fin: 27/04/2017

Sprint log:

- Realizar la creación y modificación de usuarios por parte del administrador.
- Realizar la lista de usuarios y su ordenación.
- Realizar el caso de recuperación de la contraseña.

Para este Sprint se crearon una serie de vistas y se incluyó para el administrador el caso de uso de crear y modificar usuarios. Para ello se diseñó una lista de usuarios, la cual se puede ordenar por todos sus campos y en a la que se añadió una serie de iconos que permite la modificación y eliminación individual de estos.

Se diseñó la recuperación de contraseñas mediante el envío de una URL al correo solicitante. Se añadieron también tokens de tiempo para la URL y el cambio de la contraseña, se estableció un hora como tiempo máximo para cambiarla después de solicitar el cambio. Pasado ese tiempo se necesita volver a solicitar la recuperación.

4.7.8. Sprint 8: Calendario de eventos

Fecha de inicio: 27/04/2017

Fecha de fin: 4/04/2017

Sprint log:

- Incluir el calendario de eventos.
- Permitir al administrador visualizar el calendario de cada empleado.

Para este Sprint se empleó un framework llamado Dhtmlx Scheduler[42]. Surgieron una serie de errores a la hora de utilizar este framework ya que cada vez que se realizaba la inserción/modificación/eliminación de algún evento del calendario el framework asignaba manualmente las cabeceras de la vista,

lo cual solo se podía hacer una vez como máximo. Para resolver este error se modificó parte del código que en su mayor parte fue cambiando el valor retornado de las funciones.

Para que el administrador pueda ver el calendario de cada empleado se optó por la colocación de un botón y un campo por el que buscar. De esta manera el administrador puede buscar el calendario por el email del empleado siempre que sea correcto. Los eventos creados en los calendarios se almacenan en la base de datos.

4.7.9. Sprint 9: Incorporar Licode a la aplicación

Fecha de inicio: 1/05/2017

Fecha de fin: 8/05/2017

Sprint log:

- Incluir el framework de Licode.
- Implementar el caso de uso para que el soporte y la administración pueda crear, modificar y eliminar salas.

Para la realización de este Sprint se incluyó Licode en la aplicación. Para su instalación se siguió la guía de la página oficial. Se decidió crear un script de arranque personalizado para el framework. Se añadieron los casos de uso de una manera similar al de usuarios, se crea una lista de salas que se pueden ordenar por todos sus campos y unos iconos para modificar y eliminar cada una individualmente. En este caso el soporte solo puede eliminar las suyas propias mientras que el administrador tiene acceso a todas las que se han creado independientemente del empleado que sea. Todo esto se realizó siguiendo la documentación de la parte servidor descrita en la página de Licode. Se añade a la base de datos una nueva colección en la que se guardaban las salas creadas.

4.7.10. Sprint 10: Decoración de la aplicación

Fecha de inicio: 8/05/2017

Fecha de fin: 15/05/2017

Sprint log:

- Realizar todo el CSS de la aplicación.

Para este Sprint se empleó para la realización CSS tanto Bootstrap[43] como Font Awesome[44]. Se diseñaron las vistas de una manera simple e intuitiva para facilitar el uso por parte de los empleados.

4.7.11. Sprint 11: Gestión de tokens

Fecha de inicio: 13/06/2017

Fecha de fin: 27/06/2017

Sprint log:

- Realizar la creación de tokens para el soporte y cliente.
- Realización de la sala de videoconferencia.

Para este Sprint se realizó un diseño para la sala que se compondría de: un chat, la webcam del empleado, la webcam del cliente y una vista del escritorio del cliente. También se asignaron los tokens para los dos participantes: uno para el cliente y otro para el empleado. Los tokens creados se almacenan en la base de datos.

4.7.12. Sprint 12: Realización de la videoconferencia**Fecha de inicio:** 3/07/2017**Fecha de fin:** 31/07/2017**Sprint log:**

- Añadir la funcionalidad a la sala de videoconferencia.
- Terminar la realización de la memoria.

En este sprint se presentó un problema con Licode que impedía la compartición de la pantalla. Se trató de buscar varias soluciones incluso posteando en el foro los errores que se presentaban sin llegar a obtener una solución. Como última esperanza nos pusimos en contacto con uno de los desarrolladores de Licode, pero tampoco llegamos a una solución. Fue en ese momento que tomamos la decisión de cambiar de framework.

Cambiar de framework y tener que adaptar la aplicación a este, supuso aumentar la duración del sprint a un mes de duración.

5. Desarrollo

A la hora de desarrollar las fases de nuestro proyecto se emplean aquellas del paradigma de la Ingeniería del Software, referentes al modelo Cascada, de esta manera se hace mas legible este apartado. El trabajo se organiza en Preparación, Análisis, Diseño e Implementación.

5.1. Preparación

Tras el estudio de las tecnologías existentes y aquellas necesarias para el desarrollo del proyecto se establece una estructura y entorno que apenas sufrió cambios.

5.1.1. Entorno de Trabajo

Se utilizó un sistema operativo basado en Linux llamado Ubuntu con la versión 14.04, con un navegador web compatible con WebRTC, como Firefox o Chrome. Se necesitó también un sistema de gestión de base de datos, en este caso MongoDB. El proyecto se realizó sobre la plataforma Node.js y se empleó el framework EasyRTC como implementación de la pila WebRTC.

Se empleara Git para realizar un control sobre las versiones. En este caso se ha empleado un servidor externo para guardar el proyecto llamado GitHub[45].

Git es una herramienta de control de versiones descentralizada. No es necesario un servidor git para crear y mantener un repositorio: siempre existe uno local que se puede sincronizar con uno o varios repositorios remotos.

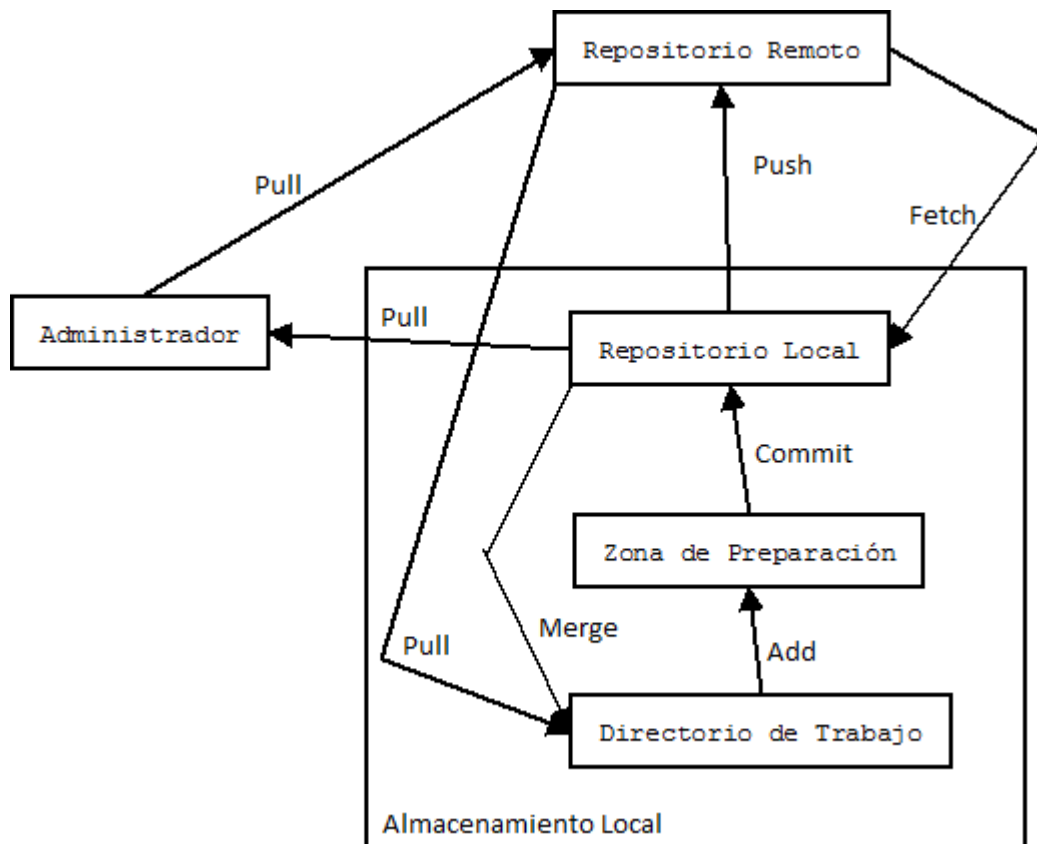


Figura 4: Esquema de los flujos de datos en git.

Las principales acciones con git son:

- **Add**

Indica que debemos añadir los cambios de uno o varios ficheros desde la copia de trabajo. De este modo pasan a la “zona de preparación”: un paso intermedio situado entre la copia de trabajo y el repositorio local. En esta fase los cambios pueden aún deshacerse sin que quede registro. Es donde se va preparando el commit.

- **Commit**

Envía los cambios de la “zona de preparación” al repositorio local, donde se conforma el registro de cambios definitivo. Cada commit lleva asociado un mensaje que debería ser descriptivo de los cambios que contiene. El autor va identificado mediante nombre y email.

- **Pull**

Conjunción de fetch y de merge, realizados en ese orden. Se sincronizan el repositorio local y la copia de trabajo con el repositorio remoto.

- **Push**

Envía a un servidor remoto los commits del repositorio local para que los incluya. Para ejecutar esta orden se necesitan permisos de escritura en el repositorio remoto.

- **Merge**

Actualiza el conjunto de trabajo con los commits en el repositorio local que todavía no se hayan aplicado. Este comando es necesario para aplicar los cambios recibidos mediante un fetch a los ficheros con los que trabaja el usuario.

- **Fetch**

Trae cambios del repositorio remoto al local. Sin embargo, los ficheros de trabajo no se actualizan ni modifican.

5.2. Análisis

Una vez establecido el entorno de trabajo se ha buscado en este apartado cumplir los objetivos establecidos al principio de la memoria.

Los usuarios están separados en los siguientes roles:

- **Clientes (Client):** Acceden mediante el navegador al soporte técnico.
- **Soporte (Support):** Pueden crear y modificar las salas, puede modificar su calendario de trabajo.
- **Administradores (Admin):** Pueden crear y modificar los usuarios de la base de datos, puede crear y modificar salas, puede modificar los calendarios de todos los empleados.

En este proyecto, por parte del cliente, nos hemos centrado en su simplicidad y facilidad de uso. De esta manera solo necesita acceder a un enlace que le llegara al correo para comenzar con el servicio técnico.

5.2.1. Mockups de la Interfaz Gráfica

A continuación se van a mostrar una serie de mockups realizados para tener una idea general de los requisitos.

- Mockup 1: Login

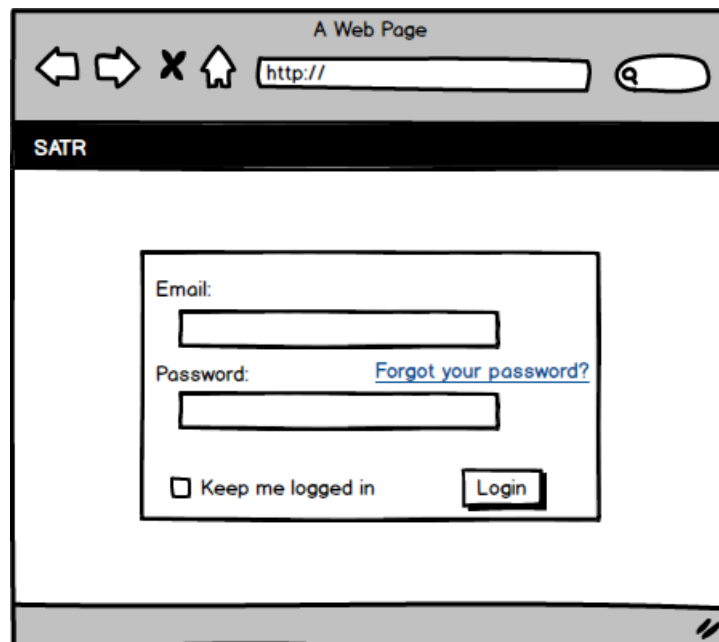


Figura 5: Mockups: Pantalla de Login

- Mockup 2: Recuperar contraseña

A Web Page

← → × ↗ http://

SATR

Your Email:

Reset Password

Login

Figura 6: Mockups: Pantalla de recuperación de contraseña

- Mockup 3: Lista de usuarios

A Web Page

← → × ↗ http://

SATR Logout

New User

Username	Email	Rol	Options
Giacomo Guilizzoni	ejemplo1@gmail.com	Support	x
Marco Botton	ejemplo2@gmail.com	Admin	x
Mariah Maclachlan	ejemplo3@gmail.com	Support	x
Guido Jack	ejemplo4@gmail.com	Support	x

Figura 7: Mockups: Pantalla de lista de usuarios

- Mockup 4: Lista de salas

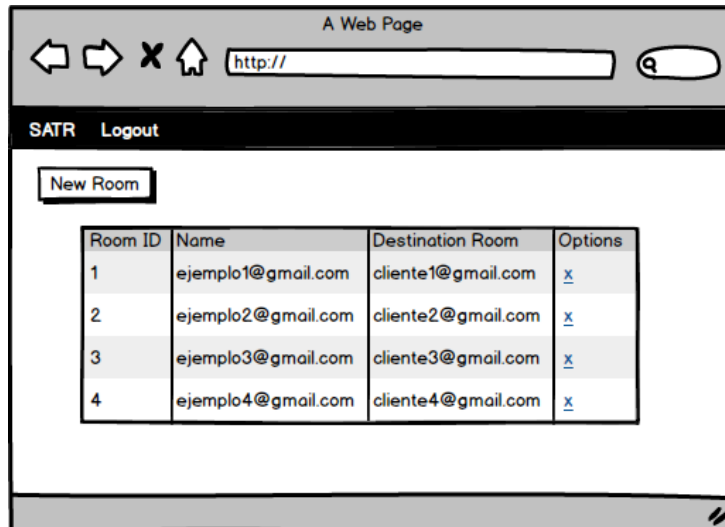


Figura 8: Mockups: Pantalla de lista de salas

- Mockup 5: Crear usuario

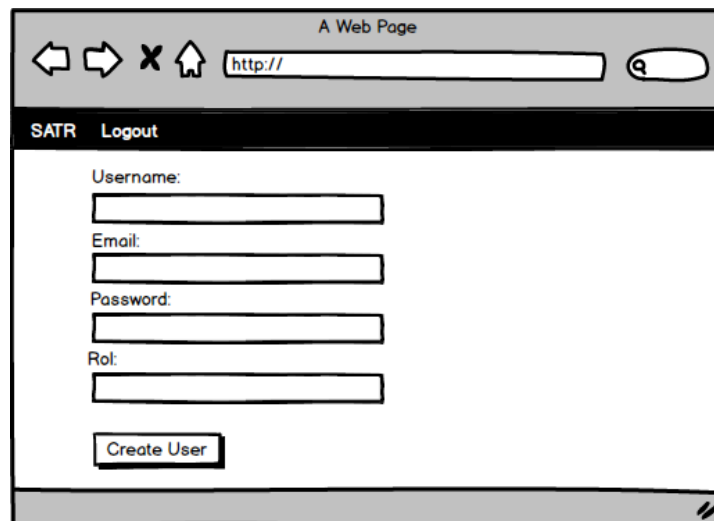


Figura 9: Mockups: Pantalla de creación de usuarios

- Mockup 6: Crear sala

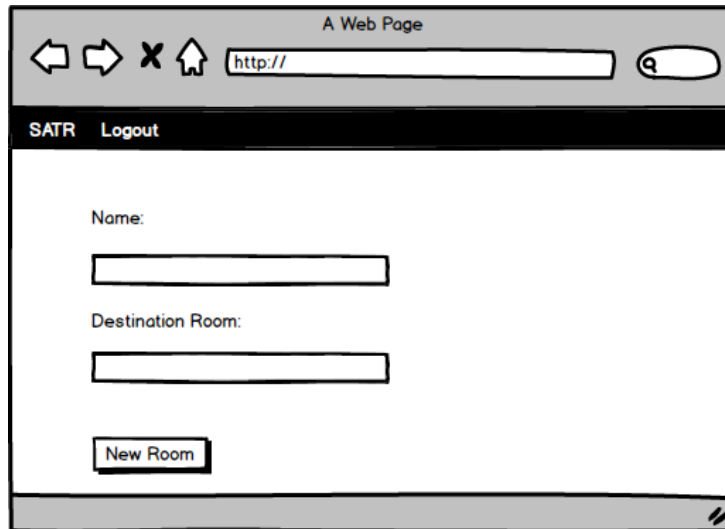


Figura 10: Mockups: Pantalla de creación de salas

- Mockup 7: Videoconferencia

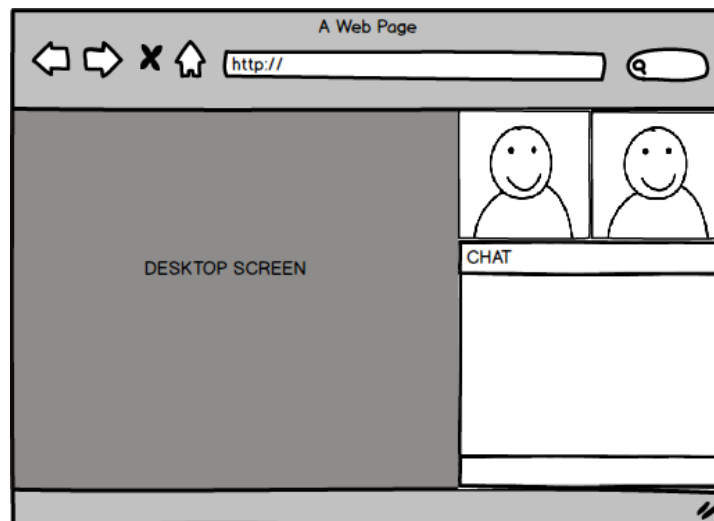


Figura 11: Mockups: Pantalla de videoconferencia

5.2.2. Funciones Genéricas

A continuación se detallarán los requisitos del sistema mediante las historias de usuario (casos de usos), por ser un medio más adecuado para describir su funcionamiento. En los casos de uso se diferencia a los agentes según distintos roles. Los roles de la aplicación corresponden con los tipos de usuario que se esperan (Clientes, Soporte, Administradores). Para facilitar la lectura del apartado se agrupan los casos de uso según su función y ámbito. En cada grupo se añade un esquema simplificado con los casos de uso que incluye y una definición de los roles utilizados.

Atendiendo a los distintos tipos de usuario que se esperan, los posibles agentes (se considera agente a toda entidad que se relaciona con el sistema) son:

- **Usuario sin autenticar:** Cualquier persona o programa que acceda a la página y no tenga una sesión iniciada.
- **Usuario autenticado:** Cualquier usuario con una sesión válida y activa. Son el soporte y administradores.
- **Administrador.** Rol encargado de gestionar usuarios y salas.
- **Cliente:** Persona que solicita el servicio técnico.
- **Soporte:** Los empleados de la empresa se encargan de gestionar sus salas y atender a los clientes.

Estas son las funcionalidades compartidas por el soporte y administradores ya que el cliente solo tiene una funcionalidad.

A continuación se comenzarán a detallar los distintos casos de usos definidos para la aplicación.

1. Caso de uso: Iniciar sesión

Roles involucrados: Usuario sin autenticar y Usuario autenticado.

Descripción: Para asignar un rol y dar permiso al usuario es necesario que se identifique con su email y contraseña.

Actores: Usuario sin autenticar.

Precondiciones: El usuario no tiene que estar autenticado en el sistema.

Escenario Base: El usuario accede a la aplicación, introduce el email y contraseña y pulsa el botón "Login".

Escenario 1: El email y contraseña introducidos son correctos.

Postcondiciones 1: Se crea una sesión y el usuario pasa a tener el rol que le corresponde.

Escenario 2: El email y la contraseña introducidos no son correctos.

Postcondiciones 2: Se muestra un mensaje de error manteniendo el navegador en la pantalla de login.

2. Caso de uso: Cerrar sesión

Roles involucrados: Usuario sin autenticar y Usuario autenticado.

Descripción: El usuario debe poder cerrar su sesión en la aplicación.

Actores: Usuario autenticado.

Precondiciones: El usuario tiene que estar autenticado en el sistema.

Escenario Base: El usuario pulsa el botón de "logout" en la barra de navegación.

Escenario 1: El usuario es redirigido a la pantalla de Login.

Postcondiciones 1: Se elimina la sesión asociada, el usuario pasa a tener el rol de usuario sin autenticar.

A continuación se muestra el diagrama de casos de uso para Iniciar sesión y Cerrar sesión.

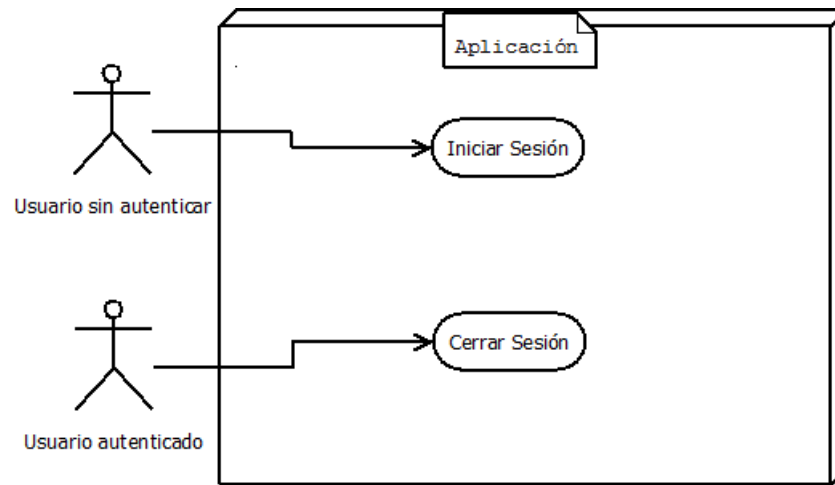


Figura 12: Caso de uso: Iniciar sesión y Cerrar sesión

3. Caso de uso: Recuperar contraseña

Roles involucrados: Usuario sin autenticar.

Descripción: El usuario puede solicitar un cambio de contraseña si este la ha olvidado.

Actores: Usuario no autenticado.

Escenario Base: El usuario pulsa en el botón para recuperar contraseñas en la ventana login. El usuario introduce su correo electrónico y pulsa el botón.

Escenario 1: El email no es válido o no existe ningún usuario con ese correo registrado en la aplicación.

Postcondiciones 1: Se indica al usuario que el email no es correcto y se le mantiene en la página para que pruebe de nuevo.

Escenario 2: El email pertenece a un usuario de la aplicación.

Postcondición 2: Se inicia el caso de uso “Cambio de contraseña” para ese usuario.

4. Caso de uso: Cambio de contraseña

Roles involucrados: Usuario sin autenticar.

Descripción: Cualquier usuario puede cambiar su contraseña mediante su email cuando se le ha olvidado.

Actores: Usuario no autenticado.

Escenario Base: La aplicación envía al correo del usuario un enlace para establecer una nueva contraseña. El usuario pincha en el enlace recibido por email. El usuario introduce su nueva contraseña en los dos campos.

Escenario 1: No coinciden ambos campos.

Postcondiciones 1: Se indica al usuario la modificación necesaria y se mantiene en el formulario para que lo corrija.

Escenario 2: La nueva contraseña es válida.

Postcondición 2: Se actualiza la contraseña en la base de datos y se redirige al usuario a la página de login. El enlace del correo caduca y no puede volver a utilizarse.

A continuación se muestra el diagrama de casos de uso para Recuperar contraseña y Cambiar de contraseña.

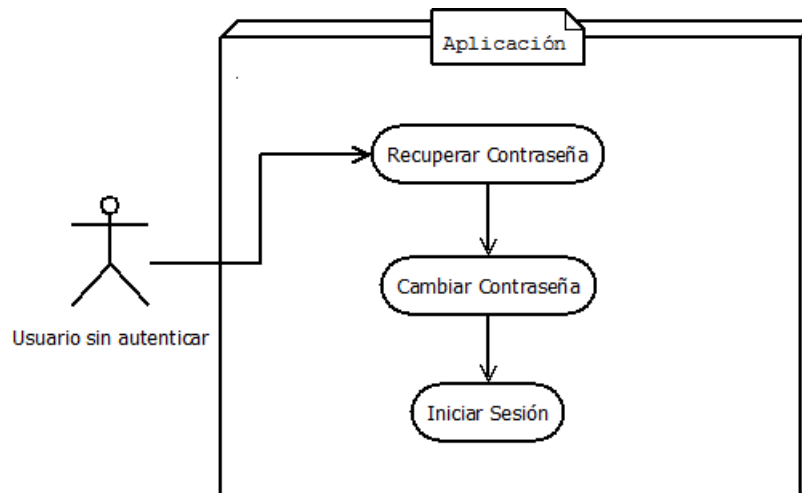


Figura 13: Caso de uso: Recuperar contraseña y Cambio de contraseña

5. Gestión de salas

■ Caso de uso: Listar salas

Roles involucrados: Administrador y soporte.

Descripción: Tanto administrador como soporte pueden ver la lista de salas, solo que el administrador puede ver todas las creadas mientras que el soporte solo las creadas por el mismo.

Actores: Soporte y Administrador.

Escenario Base: Una vez el usuario está autenticado pincha en la barra de navegación Rooms”.

Postcondiciones 1: El usuario es dirigido a la lista de salas donde puede pinchar en distintas opciones.

■ Caso de uso: Crear salas

Roles involucrados: Administrador y soporte. **Descripción:** Tanto administrador como soporte pueden crear las salas, solo que el administrador tiene diferente manera de crear salas.

Actores: Soporte y Administrador.

Escenario Base: El usuario pulsa el botón ”New Room”. A continuación en la nueva página el usuario rellena los campos.

Escenario 1: Algún dato es inválido o está sin cubrir siendo obligatorio.

Postcondiciones 1: Se indica al usuario que debe volver a introducir los datos y se mantiene en el formulario para que corrija el error.

Escenario 2: Los datos son válidos.

Postcondiciones 2: Se crea la nueva sala y mantiene al usuario en la misma página por si desea crear más.

- **Caso de uso: Modificar salas**

Roles involucrados: Administrador y soporte.

Descripción: Tanto administrador como soporte pueden modificar las salas.

Actores: Soporte y Administrador.

Escenario Base: Al pulsar el icono del lápiz al lado de una sala en la lista, se redirige al usuario a una nueva página donde puede modificar los datos de esa sala.

Escenario 1: El usuario cambia los datos de la sala.

Postcondiciones 1: Los datos son corregidos y se reflejan en la lista, de mantiene al usuario en la misma página por si quiere hacer otro cambio.

- **Caso de uso: Eliminar salas**

Roles involucrados: Administrador y soporte.

Descripción: Tanto administrador como soporte pueden eliminar las salas.

Actores: Soporte y Administrador.

Escenario Base: El usuario se encuentra en la lista y pulsa el icono de la X al lado de una sala en la lista.

Escenario 1: La sala se elimina.

Postcondiciones 1: La sala desaparece de la lista y se mantiene al usuario en la misma página.

- **Caso de uso: Conectarse a una sala**

Roles involucrados: Administrador y soporte.

Descripción: Tanto administrador como soporte pueden conectarse a las salas que crean.

Actores: Soporte y Administrador.

Escenario Base: El usuario se encuentra en la lista y pulsa el icono del mundo al lado de una sala en la lista.

Escenario 1: El usuario se conecta a la sala.

Postcondiciones 1: Comienza la videoconferencia.

A continuación se muestra el caso de uso de gestión de salas:

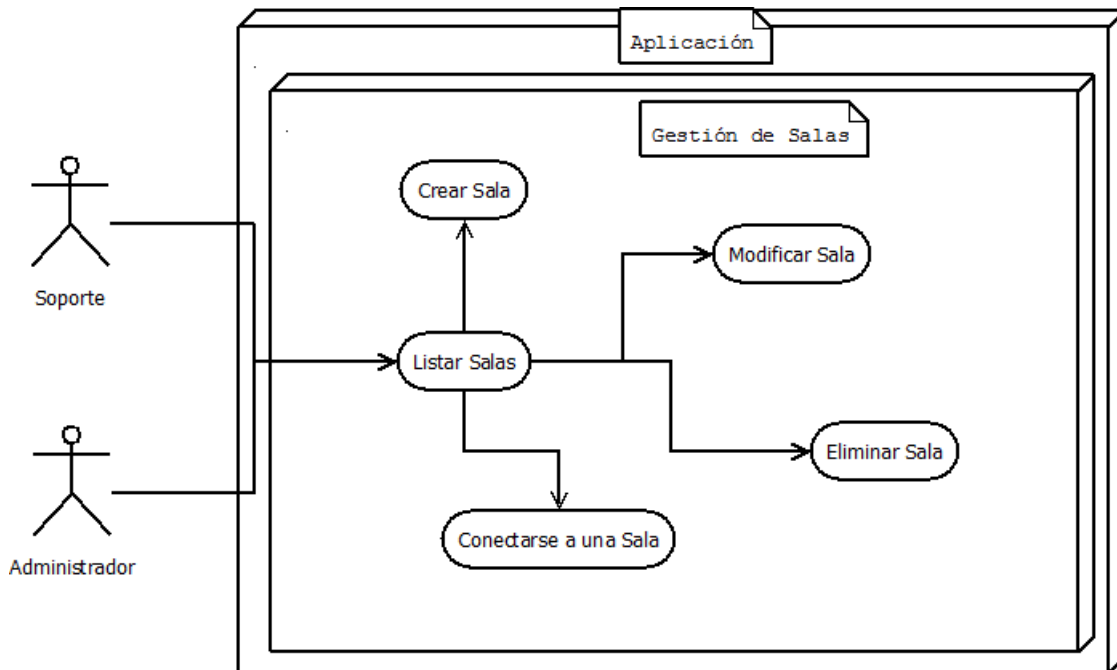


Figura 14: Caso de uso: Gestión de salas

5.2.3. Funciones Específicas

Estas son las funcionalidades específicas de cada usuario según su rol.

1. Gestión de usuarios

■ Caso de uso: Listar usuarios

Roles involucrados: Administrador.

Descripción: Solo el administrador puede ver la lista de usuarios.

Actores: Administrador.

Escenario Base: Una vez el usuario está autenticado pincha en la barra de navegación "Users".

Postcondiciones 1: El usuario es dirigido a la lista de usuarios donde puede pinchar en distintas opciones.

- **Caso de uso: Crear usuarios**

Roles involucrados: Administrador.

Descripción: Solo el administrador puede crear a los usuarios.

Actores: Administrador.

Escenario Base: El administrador pulsa el botón "New User", a continuación en la nueva página el administrador rellena todos los campos.

Escenario 1: Algún dato es inválido o está sin cubrir siendo obligatorio.

Postcondiciones 1: Se indica al administrador que debe volver a introducir los datos y se mantiene en el formulario para que corrija el error.

Escenario 2: Los datos son válidos.

Postcondiciones 2: Se crea el nuevo usuario y se mantiene al administrador en la misma página por si desea crear más.

- **Caso de uso: Modificar usuarios**

Roles involucrados: Administrador.

Descripción: Solo el administrador puede modificar a los usuarios.

Actores: Administrador.

Escenario Base: Al pulsar el icono del lápiz al lado de un usuario en la lista, se redirige al administrador a una nueva página donde puede modificar los datos de ese usuario.

Escenario 1: El administrador cambia los datos del usuario.

Postcondiciones 1: Los datos son corregidos y se reflejan en la lista, se mantiene al administrador en la misma página por si quiere hacer otro cambio.

- **Caso de uso: Eliminar usuarios**

Roles involucrados: Administrador.

Descripción: Solo el administrador puede eliminar a los usuarios.

Actores: Administrador.

Escenario Base: El administrador se encuentra en la lista y pulsa el icono de la X al lado de un usuario en la lista.

Escenario 1: El usuario se elimina.

Postcondiciones 1: El usuario desaparece de la lista y se mantiene al administrador en la misma página.

A continuación se muestra el caso de uso de gestión de usuarios:

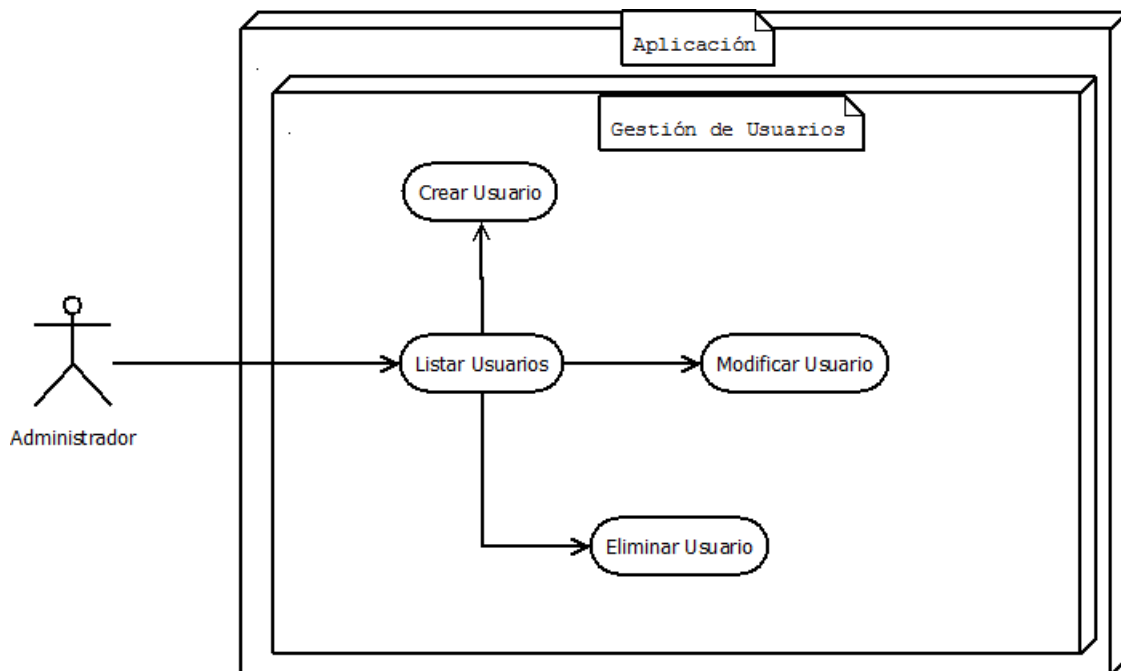


Figura 15: Caso de uso: Gestión de usuarios

2. Caso de uso: Modificar los calendarios de todos los empleados

Roles involucrados: Administrador.

Descripción: Solo el administrador modificar los calendarios que no sean suyos.

Actores: Administrador.

Escenario Base: El administrador en su pantalla de inicio entonces introduce un email de un empleado y pulsa el icono de la lupa.

Escenario 1: El email no es correcto.

Postcondiciones 1: El administrador se mantiene en su pantalla de inicio sin ningún cambio.

Escenario 1: Se carga el calendario del empleado.

Postcondiciones 1: El administrador puede modificar el calendario.

A continuación se muestra el caso de uso de modificar los calendarios de todos los empleados de todos los empleados:

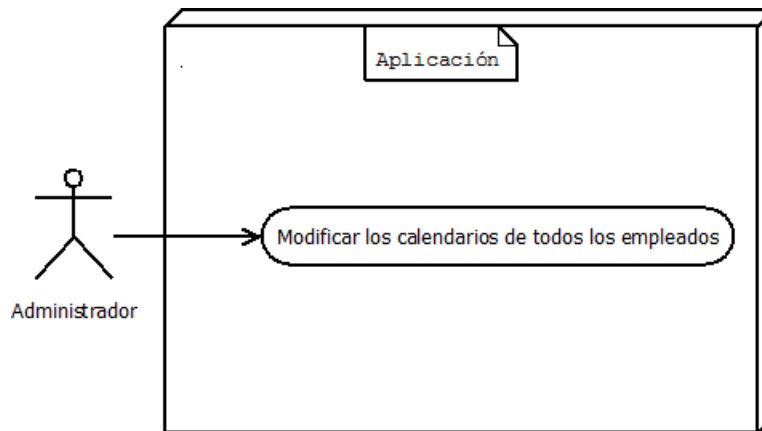


Figura 16: Caso de uso: Modificar los calendarios de todos los empleados

3. Caso de uso: Conexión del cliente a la videoconferencia

Roles involucrados: Cliente.

Descripción: El cliente se conecta a la videoconferencia mediante la URL que le llega al correo.

Actores: Cliente.

Escenario Base: El cliente pulsa la url del correo y le lleva a la pantalla de videoconferencia.

Escenario 1: El cliente comienza la videoconferencia.

Postcondiciones 1: El cliente recibe la atención técnica..

A continuación se muestra el caso de uso de conexión del cliente a la videoconferencia:

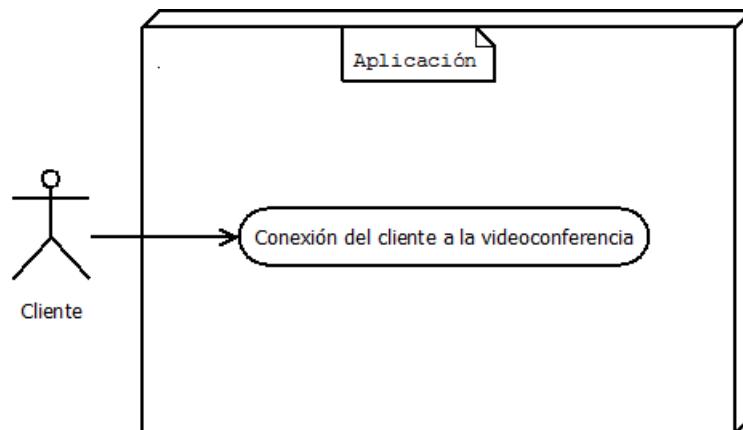


Figura 17: Caso de uso: Conexión del cliente a la videoconferencia

5.3. Diseño

Una vez analizados los objetivos y casos de uso que debe cumplir la aplicación, se presenta el diseño que da sustento a este comportamiento. Debido al uso de la metodología Scrum, esta estructura ha ido evolucionando con las características del programa, partiendo de un conjunto mínimo de datos imprescindibles.

5.3.1. Arquitectura de la aplicación

La plataforma a desarrollar es una aplicación web, cuyo principal lenguaje de programación es JavaScript. En esta aplicación se va a emplear tanto por parte del lado cliente como por parte del servidor. Esto implica que se pueden programar en JavaScript los procedimientos que generan las páginas HTML que recibe el navegador, el acceso a datos, la lógica de negocio... En nuestro caso como intérprete de este lenguaje empleamos Node.js y EasyRTC como el entorno elegido para el desarrollo de la plataforma de videoconferencia. Como base de datos se emplea MongoDB. A continuación se muestra una imagen donde se detalla la arquitectura de la aplicación.

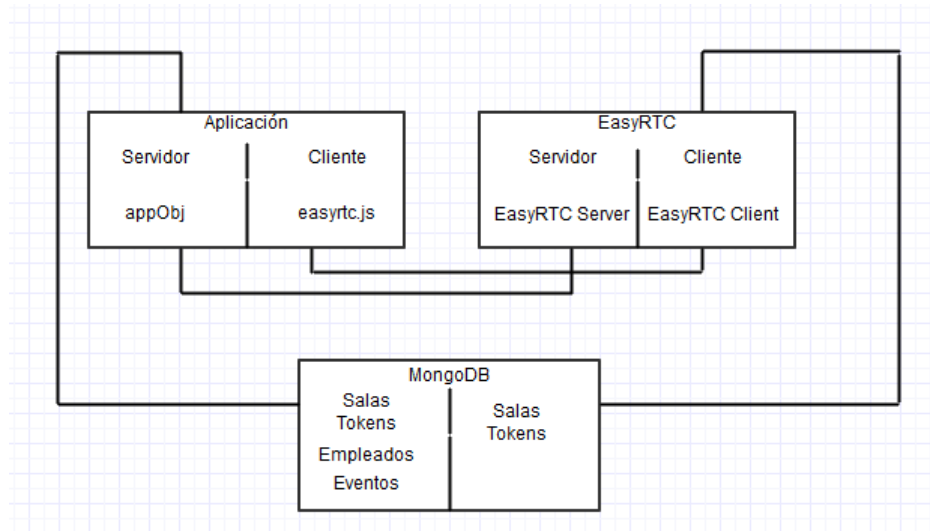


Figura 18: Arquitectura de la Aplicación

5.3.2. Servicios a implementar

En este apartado se mencionaran los servicios que se establecieron para incluir en la aplicación.

- **Autenticación**

Para el caso de la Autenticación se estableció que en la aplicación solo podían iniciar sesión el personal de soporte y administración. Estos se identifican mediante un email que debe ser único para cada cliente y una contraseña.

- **Recuperación de contraseña:** Para el caso de la Recuperación de contraseña se estableció que se realizaría a través del correo del empleado. Una vez que solicita la recuperación se le envía al correo un mensaje con una url donde puede cambiar la contraseña. Para esto se crean unos tokens para ese empleado con validez de 1 hora. Si en esa hora no a cambiado de contraseña el token expira y tendría que volver a solicitar la recuperación. De este modo también expira en 1 hora la url enviada al correo.

- **Creación/Modificación de empleados:** Para el caso de la Creación/Modificación de empleados se estableció que solo los administradores deberían tener esta tarea.

- **Creación/Modificación de salas:** Para el caso de Creación/Modificación de salas se estableció que tanto soporte como administración deberían tener esta tarea, pero con pequeñas diferencias: el soporte solo puede crear salas para su uso inmediato y solo bajo su nombre mientras que el administrador puede crearlas bajo el nombre de cualquier empleado y crearlas en el horario que desee. Además el administrador puede ver en todo momento todas las salas creadas mientras que el soporte solo puede ver las creadas por el empleado.

- **Calendario de eventos:** Para el caso del Calendario de eventos se determinó que cada empleado debería tener un calendario de eventos, pero que el administrador puede ver el de cualquier empleado y modificarlo.

- **Acceso del cliente a la sala desde una url:** En este caso se determinó que, para mantener la aplicación lo más sencilla posible para el cliente, solo necesite solicitar la atención técnica para que le llegue un mensaje al correo y acceda mediante una url (incluida en el mensaje) a la sala y puede iniciar la atención.
- **Atención Técnica:** Para el caso de la Atención Técnica se estableció que se compondría de una pantalla en la que se debería incluir un chat, la vista de la webcam del empleado, la vista de la webcam del cliente y la vista del escritorio del cliente.

5.3.3. Base de datos

A continuación se definen los objetos primarios a almacenar:

- **Users:** Se guarda los datos como su nombre, contraseña, email y el rol.
- **Rooms:** Se guardan los datos de las salas tales como un identificador, su nombre, y el destinatario de esta.
- **Tokens:** Se guardan los datos como un email, el identificador del token y el identificador de la sala.
- **Events:** Se guardan los datos como el nombre del empleado, información del evento y una fecha de inicio y fin.

Es por tanto un esquema sencillo, con pocas entidades. Aún así la elección del software que se va a utilizar para administrar la base de datos es decisiva. En este caso se hacen notar las ventajas de un modelo de desarrollo ágil, dado el carácter de este proyecto.

5.3.4. Capas de la aplicación

En cuanto al diseño de la aplicación se ha tratado de mantener un modelo de desarrollo software en el que el objetivo primordial es la separación de las partes que componen un sistema software para ello se ha realizado una arquitectura basada en capas.

La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sobrevenga algún cambio, solo afectará al nivel requerido sin tener que revisar entre el código fuente de otros módulos.

A continuación explicaremos las capas de la aplicación:

- **Capa de presentación:** la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso. En este caso se compone de las vistas empleadas por la aplicación, que hacen uso de los archivos de diseño gráfico (css) y de los archivos javascript (js).
- **Capa de negocio:** es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. En este caso se compone de las funciones internas de aplicación (Routing) para gestionar la funcionalidad esta. La aplicación se inicia sobre app y este llama a Routing permitiendo que la aplicación este lista para funcionar.
- **Capa de datos:** es donde residen los datos y es la encargada de acceder a los mismos. En este caso se compone de las funciones de la aplicación que se encargan de interactuar con la base de datos de MongoDB (Database). Estas se encargan de realizar las búsquedas, modificaciones, inserciones y borrados en la base de datos.

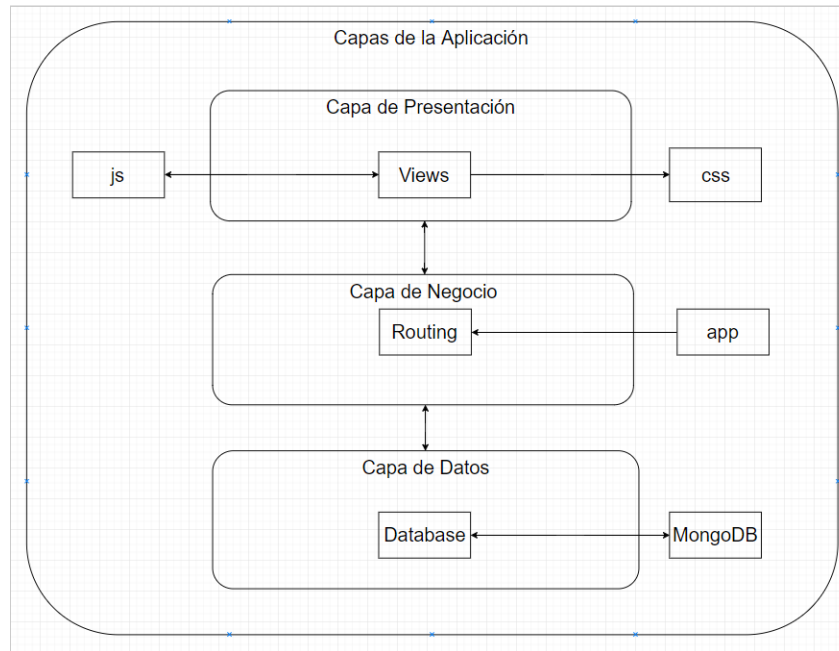


Figura 19: Capas de la aplicación

5.4. Implementación

5.4.1. Estructura de la aplicación

Una vez definidas todas las herramientas y frameworks que se usarán se prepara el proyecto en Sublime con la siguiente jerarquía de ficheros:

- **App**
 - **Public:** Los contenidos de este directorio son accesibles directamente desde la web. Son archivos que pueden ser solicitados por el navegador. En él se incluye:
 - **css:** Las hojas de estilo desarrolladas.
 - **codebase:** Son los archivos de estilos pertenecientes al calendario de eventos.
 - **images:** Directorio con los recursos gráficos, logos e iconos.

- **fonts:** Archivos de estilos pertenecientes a Font Awesome.
- **js:** En él se encuentran scripts que se añaden a las vistas.
- **Server:** Nada contenido en este directorio se ve desde el lado de cliente. Se trata del código Node.js de la aplicación, encargado de procesar las peticiones y mostrar los resultados.
 - **Database:** Código que contiene la definición de los elementos de la base de datos y su manejo.
 - **Email_System:** Código que contiene el sistema de envío de correo electrónico.
 - **Routing:** Código que sirve las páginas relacionadas con cuentas: administración, gestión de usuarios, gestión de eventos, listado, sala de conferencia...
 - **Views:** Contiene las plantillas Jade que generan las páginas HTML de la aplicación.
- **Certs:** En él se encuentran los certificados para la realización de https.
- **MongoDB:** En él se encuentra el script para la creación de la base de datos.
- **app.js:** Compone la entrada de código de la aplicación, encargada de llamar a todas las dependencias y arrancar el servidor Express.

5.4.2. Relación de recursos en la videoconferencia

Vistos ya en el apartado anterior los ficheros de la aplicación es necesario explicar cómo interactúan entre ellos. En este caso para mantenerlo simple se muestran las relaciones desde los propios paquetes y no desde todos los ficheros.

La aplicación se ejecuta desde el archivo `app.js`. En este archivo se emplean los certificados para HTTPS que se encuentran en el paquete `Certs`. El fichero principal de la aplicación, donde se encuentra todo el funcionamiento se encuentra en el paquete `Routing`. En él se hace uso de las vistas que están en el paquete `Views`, del servicio de correo almacenado en el paquete `Email_System` y también de los ficheros que permiten interactuar con la base

de datos almacenados en el paquete Database. Las vistas hacen uso de los scripts, css y el framework del calendario almacenado en los paquetes js, css, codebase y fonts.

5.4.3. MongoDB

En lo que se refiere a la estructura de la base de datos de MongoDB, se creó una base de datos con nombre Aplicación que consta de 4 colecciones:

- **Users:** En esta colección se almacenan todos los empleados creados, tanto de soporte como de administración. Está formada por los siguientes datos: username (nombre del empleado), password (contraseña del empleado), email (un email único para cada empleado), recoverPassword (un identificador que se genera para la recuperación de contraseña), recoverPasswordExpires (una fecha de expiración de recoverPassword) y rol (un enumerado con los valores de Support y Admin en el que se establece el rol del usuario creado).
- **Rooms:** En esta colección se almacenan todas las salas creadas. Está formada por los siguientes datos: id (un identificador proporcionado por Licode cuando se crea la sala), name (email del empleado bajo el que se crea la sala) y destination (email del cliente que solicitó la atención técnica).
- **Tokens:** En esta colección se guardan los tokens generados para los empleados y clientes. Está formada por los siguientes datos: email (del empleado o cliente al que se le creó el token), token (un identificador generado por la aplicación cuando se solicita un token de una sala creada) y room (el identificador de una sala creada).
- **Events:** En esta colección se guardan los eventos de todos los calendarios. Está formada por los siguientes datos: nameUser (nombre del empleado que creó el evento), text (información o asunto del evento), start_date (fecha de inicio del evento) y end_date (fecha de finalización del evento).

5.4.4. Problemas encontrados

En esta sección se explicarán brevemente los problemas encontrados en el desarrollo de la aplicación y su solución:

1. **Problemas con los datos en MongoDB** En cierto momento se observó que MongoDB almacenaba ciertos caracteres (@) de manera incorrecta en la base de datos. Después de consultar la documentación la solución recomendada era actualizar MongoDB a la última versión.
2. **Problemas con el framework del calendario** En este caso el problema encontrado se debía a que el framework en el momento en que se creaba, actualizaba o eliminaba un evento del calendario establecía varias veces la cabecera de la página y se producía un error. Se optó por modificar el código del framework para que solo pudiera setear las cabeceras una vez.

Otro problema encontrado fue en el momento de añadir la funcionalidad al administrador de modificar los calendarios de todos los empleados. Para ello la idea fue que al introducir un email y pulsar un botón se realice una petición POST y en ella se cargue el nuevo calendario. Esto presentaba un problema y es que la funcionalidad del framework no permitía ninguna petición POST que no fuera al framework. Dicho esto se optó por modificar el código del framework y que dentro de la propia funcionalidad de este se realizara la petición POST que queríamos.

3. **Problemas con Licode** Con el framework de Licode se produjeron los siguientes errores:
 - **Sin conexión con la API** Se optó primero por una reinstalación de Licode, lo cual no solucionó el error, y después por cambiar el script de dependencias creado por el alumno. Finalmente se encontró el error: se debía a que se necesitaba importar un archivo de Licode llamado `nuve.js` en el código.
 - **Compartir la webcam** Para esto se tuvo que instalar en la máquina virtual las guest additions de virtualbox, que permite activar la webcam desde la propia máquina.

■ Compartición de la pantalla

Para la compartición de la pantalla Licode necesitaba que sus comunicaciones funcionaran sobre HTTP. Al principio se desarrolló la base de la aplicación sin SSL, con un simple servidor HTTP, sabiendo que en Node.js cambiar ese servidor, por otro HTTPS, resultaba trivial. Con esa aproximación Licode funcionaba correctamente, pero más adelante cuando se quiso poner SSL, requisito básico para que la aplicación pudiera ser desplegable en un entorno real, la comunicación con Erizo no funcionó. Se comprobó que el soporte para SSL de Licode era experimental, motivo por el cual no funcionaba, pero necesario para su funcionamiento. Dicho esto se probaron las siguientes soluciones para su implementación:

- a) En primer lugar se trató de cambiar los certificados generados para el HTTPS. Se cambiaron de SHA-1 a SHA-2 para tratar de solucionar el problema pero no se obtuvo ninguna mejora.
- b) Se intentó un “workaround” montando un servidor Nginx como proxy inverso, de tal manera que la comunicación del cliente con el proxy era en HTTPS y del proxy a Erizo en HTTP, con lo que no perdería seguridad si se ejecutaban ambos en el mismo equipo. También se trató de hacer lo mismo pero con un servidor TURN sin conseguir nada. Este último intento también fracasó, por ello la única solución viable que se encontró fue migrar de plataforma.

5.5. Pruebas

En este capítulo se recoge la información relativa a los distintos tipos de validación que se realizaron dentro del proyecto. Como se indica en la metodología de Scrum, al final de cada sprint se realizan un conjunto de pruebas que deben ser superadas para validar el correcto funcionamiento de las funcionalidades. En caso de que la herramienta no sea capaz de superar dichas pruebas y existan errores en la implementación, estos deben de ser corregidos antes de pasar a la siguiente iteración ya que la mayoría de los comportamientos implementados en un sprint dependen del anterior para funcionar correctamente.

Las pruebas son una parte fundamental del desarrollo software. Estas evalúan la fiabilidad y constituyen una excelente manera de comprobar que se cumple con los objetivos marcados. A través de las pruebas se pueden identificar errores en la implementación, calidad o funcionalidad de una aplicación y de esa manera poder arreglarlos. Para probar la aplicación desarrollada durante el proyecto, se han realizado pruebas de unidad, integración, sistema y aceptación en la mayor parte del desarrollo. De esta forma, se asegura el buen funcionamiento del sistema.

Como ya se ha mencionado, las pruebas realizadas son de los siguientes tipos:

- **Pruebas unitarias:** Validan el correcto funcionamiento de cada una de las funcionalidades implementadas.
- **Pruebas de integración:** Validan el correcto funcionamiento de la herramienta para un conjunto de funcionalidades.
- **Pruebas de sistema:** Validan el correcto funcionamiento de la herramienta completa como sistema.
- **Pruebas de aceptación y validación:** Pruebas ejecutadas por el director antes de la entrega del sprint que validan el funcionamiento adecuado de las funcionalidades.

5.5.1. Pruebas unitarias

Una prueba unitaria es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Esto habilitará la correcta evaluación de los subsistemas formados por varios módulos.

Se llevaron a cabo pruebas sobre los métodos de los diferentes módulos del sistema que compone el proyecto. Estas pruebas forman el primer filtro para localizar errores ya que son ejecutadas sobre un elemento concreto.

5.5.2. Pruebas de integración

Las pruebas de integración son aquellas que se realizan en el ámbito de desarrollo software una vez que se pasaron las pruebas unitarias. Consiste en realizar pruebas para verificar que todas las partes del subsistema software funcionan correctamente juntas. Las pruebas de integración es la fase de pruebas en la cual los módulos individuales son combinados y probados como un grupo.

Las pruebas de integración deben contemplar todas las actividades de la aplicación y todas las acciones posibles. Se realizaron pruebas sobre los siguientes casos:

- Interacción entre los distintos módulos de manejo de los datos de la BD.
- Interacción con el sistema de envío de correos de la aplicación.
- Interacción con el login y la recuperación de contraseña.
- Interacción entre el módulo de gestión de las rutas con el framework de EasyRTC.

Una vez concluido y superado el proceso de pruebas de integración se puede garantizar el correcto funcionamiento de cada subsistema.

5.5.3. Pruebas de sistema

Una vez que se han pasado todas las pruebas de integración con éxito, se comprueba la funcionalidad de la herramienta en su totalidad. El objetivo es comprobar el funcionamiento global del sistema, verificando que cumple todos los requisitos iniciales para los que ha sido diseñado.

Para la realización de estas pruebas se ejecutaron conjuntamente todos los subsistemas posibles, comprobando que todo funciona como se esperaba.

5.5.4. Pruebas de aceptación y validación

Una vez superadas todas las pruebas anteriores garantizando el buen funcionamiento del sistema final, se efectúan las pruebas de aceptación y validación. Estas pruebas tienen la finalidad de comprobar que se cumple con las especificaciones marcadas en el análisis y que los resultados obtenidos son satisfactorios.

El resultado final tiene que cumplir con lo que se marcó al inicio del sprint. Los tiempos de respuesta deben ser aceptables y la interfaz amigable e intuitiva. En este caso la herramienta fue analizada por el director, lo que dio a lugar a nuevos requisitos surgidos de la propia evaluación del producto final.

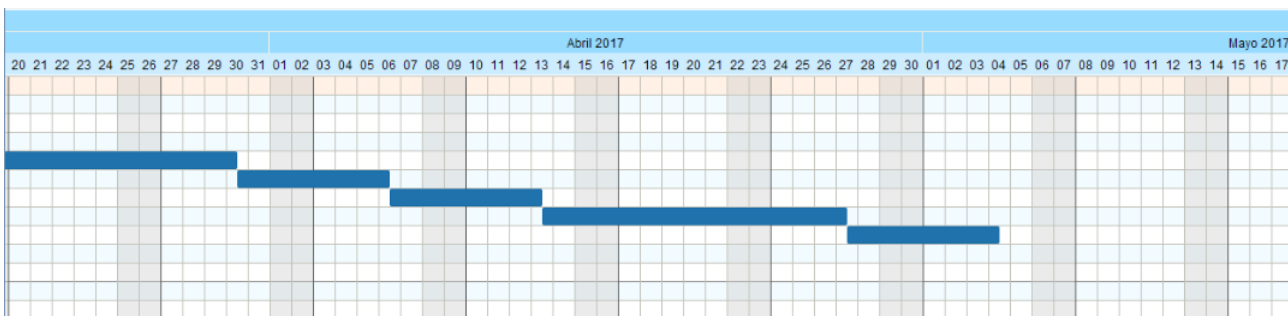


Figura 21: Diagrama de Gantt Parte 2

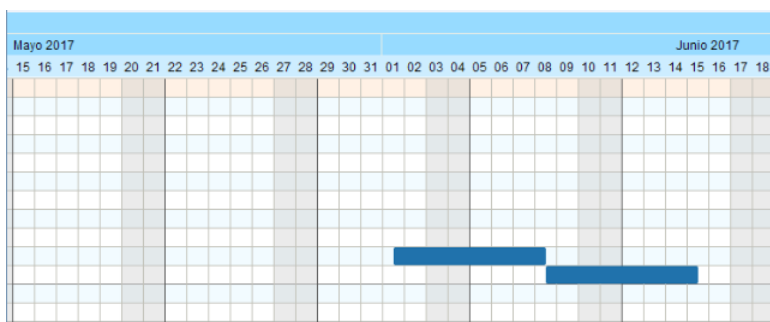


Figura 22: Diagrama de Gantt Parte 3

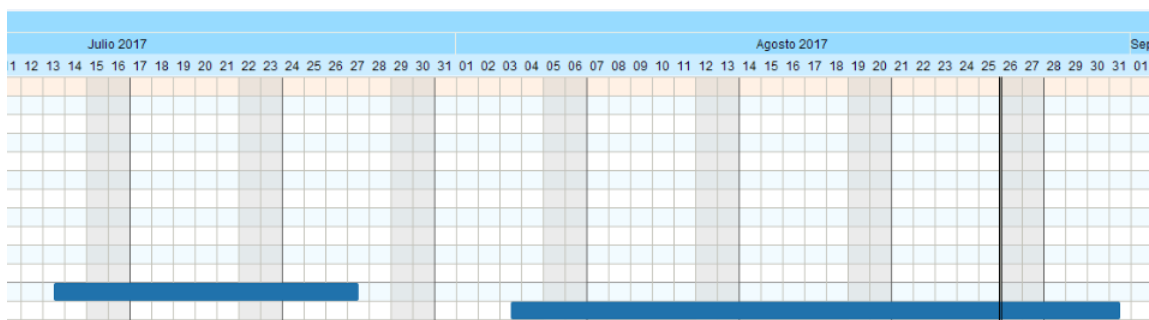


Figura 23: Diagrama de Gantt Parte 4

7. Conclusiones y trabajos futuros

7.1. Conclusiones

Se ha desarrollado la aplicación cumpliendo los objetivos presentados al inicio de este documento, siguiendo una metodología personalizada basada en Scrum. Dicho esto la aplicación cumple los objetivos definidos y permite solucionar los problemas del soporte técnico de manera rápida y eficaz.

WebRTC es una tecnología innovadora que se encuentra en continuo desarrollo. Esto provoca cierto rechazo en los fabricantes y por el momento queda relegada a desarrollos más independientes. El desarrollo del proyecto se ha basado en la tecnología WebRTC y concretamente en la implementación EasyRTC. Como se ha comentado anteriormente esta tecnología aporta como gran ventaja la disponibilidad en navegadores ampliamente extendidos, pero presenta como inconveniente cierta inestabilidad por no estar disponible una versión definitiva. Con respecto a los dos frameworks empleados primeramente Licode y posteriormente EasyRTC se puede observar que, al ser una tecnología en continuo desarrollo, se pueden llegar a producir, como en el caso de Licode, que ciertas partes de este no funcionen o aún no se hayan adaptado al cambio de sus dependencias.

En el proyecto se emplea NoSQL con MongoDB lo que proporciona un enfoque distinto al adquirido durante la realización de los estudios de Ingeniería, que se centró fundamentalmente en bases de datos relacionales.

Cabe destacar que el proyecto fue realizado siguiendo la metodología de desarrollo ágil Scrum. Esto permitió establecer una base de la cual ir añadiendo nuevas funcionalidades a medida que se desarrolló el proyecto. Esto también requirió una organización y diseño del código para que se adaptara a este modelo incremental.

Este proyecto establece una base para tratar de solucionar el problema subyacente. El resultado final de este desarrollo es un producto funcional que aunque no estaría preparado para ser comercializado constituye una muy buena base para desarrollos posteriores, de la cual se podrían realizar mejoras en un futuro.

Como conclusión personal este proyecto/trabajo ha supuesto gran esfuerzo, debido a múltiples problemas tanto del software empleado como de investigación. Todo este camino recorrido me ha servido para aprender de una gran cantidad de nuevas tecnologías y conceptos que en la carrera no se ven o no se profundiza lo suficiente.

7.2. Trabajos futuros

Por falta de tiempo se han recortado aspectos que se podrían incluir en el proyecto final, además de mejoras que se pueden añadir en un futuro:

- La aplicación actual solo almacena la información justa y necesaria para el funcionamiento. Sería muy útil recoger también datos de los clientes.
- Obtener un certificado digital de una Autoridad Certificadora y no uno realizado y firmado por el alumno.
- El diseño de las salas se pueden mejorar añadiendo elementos estéticos y de manejo.
- La calidad de las videoconferencias es baja ya que EasyRTC limita el framerate máximo.
- Se podría mejorar la seguridad de los accesos a la BD cifrando la información.
- En lo referente al calendario de cada empleado sería una buena idea añadir búsquedas múltiples, es decir, poder ver desde el administrador el calendario de uno o más empleados (con distintos colores para así distinguirlos).

8. Bibliografía

Referencias

- [1] **Skype**, “Microsoft y Skype Technologies.” <https://www.skype.com/es/>, 2017.
- [2] **Hangouts**, “Google.” <https://hangouts.google.com/?hl=es>, 2017.
- [3] **Mikogo**, “Snapview.” <http://www.mikogo.es/>, 2017.
- [4] **Cisco WebEx**, “WebEx.” <https://www.webex.es/>, 2017.
- [5] **GoToMeeting**, “LogMeIn.” <https://www.gotomeeting.com/es-es>, 2017.
- [6] **TeamViewer**, “TeamViewer.” <https://www.teamviewer.com/es/>, 2017.
- [7] **FaceTime**, “Apple.” <https://itunes.apple.com/es/app/facetime/id414307850?mt=12>, 2017.
- [8] **Microsoft Lync**, “Microsoft.” <https://www.microsoft.com/es-es/download/details.aspx?id=35451>, 2017.
- [9] **WebRTC**, “WebRTC.” <https://webrtc.org/architecture/>, 2017.
- [10] **Licode**, “Open Source WebRTC Communications Platform.” <http://lynckia.com/licode/>, 2017.
- [11] **EasyRTC**, “Priologic Software Inc..” https://easyrtc.com/docs/easyrtc_faq.php.
- [12] **Node.js**, “Interactive.” <https://nodejs.org/es/>, 2017.
- [13] **JavaScript**, “JS.” <https://www.javascript.com/>, 2017.
- [14] **Sublime Text**, “Sublime Text.” <https://www.sublimetext.com/>, 2017.
- [15] **SCRUM**, “Desarrollo de software.” [https://es.wikipedia.org/wiki/Scrum_\(desarrollo_de_software\)](https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)).

-
- [16] **SCRUM**, “What is SCRUM?.” <https://proyectosagiles.org/que-es-scrum/>.
 - [17] **Smart-IPX**, “WebRTC.” <http://smartipx.com/demand-for-in-app-drive-webrtc/>.
 - [18] **Michael James**, “Scrum.”
 - [19] **Kurento**, “WebRTC media server .” <http://www.kurento.org/>, 2017.
 - [20] **Express**, “Node.js.” <http://expressjs.com/es/>, 2017.
 - [21] **NPM**, “Npm Enterprise.” <https://www.npmjs.com/>.
 - [22] **Body-Parser**, “Node.js.” <https://www.npmjs.com/package/body-parser-json>, 2017.
 - [23] **Jade**, “Node.js.” <https://www.npmjs.com/package/jade>, 2017.
 - [24] **Cookie-Parser**, “Node.js.” <https://www.npmjs.com/package/cookie-parser>, 2017.
 - [25] **Crypto**, “Node.js.” <https://nodejs.org/api/crypto.html>, 2017.
 - [26] **EasyRTC NPM**, “Npm Enterprise.” <https://www.npmjs.com/package/easyrtc>.
 - [27] **Error-Handler**, “Node.js.” <https://www.npmjs.com/package/error-handler>, 2017.
 - [28] **Express-Flash**, “Node.js.” <https://www.npmjs.com/package/express-flash>, 2017.
 - [29] **Express-Session**, “Node.js.” <https://www.npmjs.com/package/express-session>, 2017.
 - [30] **Method-Override**, “Node.js.” <https://www.npmjs.com/package/method-override>, 2017.
 - [31] **Moment**, “Node.js.” <https://www.npmjs.com/package/moment>, 2017.
 - [32] **Mongodb**, “Node.js.” <https://www.npmjs.com/package/mongodb>, 2017.

-
- [33] **Mongoose**, “Node.js.” <https://www.npmjs.com/package/mongoose>, 2017.
 - [34] **Morgan**, “Node.js.” <https://www.npmjs.com/package/morgan>, 2017.
 - [35] **Nodemailer**, “Node.js.” <https://www.npmjs.com/package/nodemailer>, 2017.
 - [36] **Sprint-js**, “Node.js.” <https://www.npmjs.com/package/sprintf-js>, 2017.
 - [37] **Stylus**, “Node.js.” <https://www.npmjs.com/package/stylus>, 2017.
 - [38] **Fs**, “Node.js.” <https://nodejs.org/api/fs.html>, 2017.
 - [39] **Https**, “Node.js.” <https://www.npmjs.com/package/http-https>, 2017.
 - [40] **JavaScript: The Good Parts**, “Douglas Crockford.” <https://www.crockford.com/javascript/>, 2017.
 - [41] **Balsamiq Mockups**, “balsamiq.” <https://balsamiq.com/products/mockups/>, 2017.
 - [42] **Dhtmlx Scheduler**, “Scheduler.” <https://docs.dhtmlx.com/scheduler/>, 2017.
 - [43] **Bootstrap**, “Bootstrap.” <http://getbootstrap.com/>, 2017.
 - [44] **Font Awesome**, “Font Awesome.” <http://fontawesome.io/>, 2017.
 - [45] **GitHub**, “Repositorio de la Práctica.” <https://github.com/Diego-Fic/SATR>, 2017.

I. Glosario de términos

En él se definen términos que no se han explicado en la propia memoria, es decir que no son propios de la aplicación sino que se hace referencia a ellos.

- **API**

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de llamadas a bibliotecas de datos que proveen de una cierta funcionalidad y abstraen de la implementación concreta de esas funciones.

- **Autoridad de Certificación**

La Autoridad de Certificación (CA) es una entidad de confianza, responsable de emitir y revocar los certificados digitales. Utilizados en la firma electrónica.

- **Diagrama de Gantt**

El diagrama de Gantt es una herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

- **Framework**

Es una estructura tecnológica compuesta de varios módulos de software que sirven de base para el desarrollo software.

- **JSON**

Acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente.

- **Middleware**

Se llama al Middleware a la vía que conecta dos aplicaciones y pasa los datos entre ellas.

- **P2P**

Consiste en una red de nodos neutrales que se comunican entre sí, sin que existan elementos de control intermedios, como servidores. Los datos se transmiten de uno a otro a través de la red en una comunicación directa.

- **Token**

En materia de seguridad, es una cadena de texto única que se utiliza como identificación.

- **VoIP**

Es un conjunto de recursos que hacen posible que la señal de voz viaje a través de Internet empleando el protocolo IP (Protocolo de Internet). Esto significa que se envía la señal de voz en forma digital, en paquetes de datos, en lugar de enviarla en forma analógica a través de circuitos utilizables solo por telefonía convencional.

II. Manual de instalación

A continuación se va a explicar tanto los requisitos necesarios para el hardware y software como los que emplea la aplicación. Se le indicara de manera clara como instalar la aplicación para de esta manera no cometer ningún error.

II.1. Requisitos

Los requisitos se refieren al servidor en el que se instalará la aplicación, los usuarios solo necesitan un navegador compatible con WebRTC y una conexión de banda ancha a Internet.

II.1.1. Hardware

Requisitos del servidor:

	Procesador	Almacenamiento	Memoria
Mínimo	Intel Centrino	16GB	2GB
Recomendado	Intel Core	32GB	4GB

Requisitos de usuario:

- Conexión de banda ancha
- Cámara web
- Micrófono

II.1.2. Software

Como condición inicial sólo se requiere tener instalado un sistema operativo con un navegador web. También se solicita que la empresa disponga de un servicio de correo electrónico al cual enviar las URL al solicitar el servicio técnico. También se debería tener actualizada la configuración de los periféricos (webcam y micrófono).

II.1.3. Requisitos del servidor

Como condición inicial sólo se requiere tener instalado el sistema operativo Ubuntu 14.04. Es el Sistema Operativo (S.O) recomendado por los desarrolladores de EasyRTC. También se recomienda que se trate de un sistema recién instalado, con los paquetes por defecto de la distribución, para evitar posibles incompatibilidades durante la instalación de EasyRTC.

II.1.4. Requisitos globales de la aplicación

- Localizar el ratón en las conferencias, para así facilitar el trabajo de el técnico.
- Visualización de escritorio remoto.
- Gestión de calendario para los empleados (que técnico asigno a qué cliente y qué horario).

NOTA

Es muy importante comprobar que, una vez realizada la instalación, se dispone de las versiones correctas. Se puede dar el caso de que cualquier cambio en ellas pueda provocar algún error. Para ello es conveniente revisar en el sistema que las versiones instaladas se corresponden a las que aparecen en el package.json de la aplicación.

II.2. Instalación

Antes de instalar y probar la aplicación necesitamos realizar una serie de tareas:

- Instalar un sistema operativo Ubuntu 14.04, a ser posible una instalación por defecto.
- Disponer de un navegador Chrome, a ser posible con una versión anterior a la 56.
- Instalar el plugin para Chrome <https://chrome.google.com/webstore/detail/screen-capturing/ajhifddimkapgcifgcodmmfdlknahffk>.

A continuación, se explicara la instalación paso a paso:

1. Primero debemos instalar MongoDB con el comando `sudo apt-get install mongodb`.
2. Despues debemos descargar el manejador de paquetes para Node.js con el comando `sudo apt-get install npm`.
3. A continuación instalamos Node.js, este paso es importante pues necesitamos instalar un versión superior a la 6.1 de este. Introducimos los siguientes comandos `curl -sL https://deb.nodesource.com/setup-6.x --sudo -E bash -` y `sudo apt-get install -y nodejs`.
4. Debemos descargar la aplicación del repositorio (`git clone https://github.com/Diego-Fic/SATR.git`), `cd SATR/` e instalar las dependencias mediante `npm install`.
5. Por último creamos la base de datos ejecutando el archivo NoSQL.js mediante el comando `mongo < MongoDB/NoSQL.js`.

Una vez hemos instalado las dependencias podemos ejecutar nuestra aplicación:

- Ejecutamos `node SATR/app.js` .
- Iniciamos el navegador web en <https://localhost:8443>.

III. Manual de usuario

En esta sección se explica cómo utilizar la aplicación para el personal de soporte, que serán los que se ponga a trabajar con la aplicación.

III.1. Descripción de la aplicación

La aplicación permite la atención técnica al cliente mediante videoconferencia. No requiere la instalación de ningún software en el equipo cliente, se realiza mediante el uso del navegador. El cliente accede a la videoconferencia a través de una URL que se le envía al correo. Los empleados puede crear/modificar/eliminar sus propias salas y además añadir eventos en su calendario.

III.2. Funcionalidades del usuario

Las cuentas de usuario son administradas por la empresa de soporte técnico mediante el rol de administrador. Es esta quién le solicitará sus datos y le entregará un correo electrónico y una contraseña para así poder iniciar sesión.

III.2.1. Acceder a la aplicación

Cuando un usuario que todavía no está autenticado en la página hace una petición al servidor este le redirige a la página de login. En el login debe introducir su email (1) y contraseña (2) y después hacer click en el botón de login (3). Una vez los datos son validados por el servidor es redirigido a la pantalla de inicio propia de cada usuario. En caso de marcar la casilla de Keep me logged in (4) se guardan las cookies del usuario en el navegador y de esta manera siempre se loguea automáticamente. En caso de pulsar Forgot your password? (5) se redirige a la pantalla de recuperación de la contraseña.

SATR

Email: 1

Password: 2

Forgot your password? 5

☐ Keep me logged in 4

Login 3

Figura 24: Pantalla de Login

Cuando inicie la sesión el sistema le redirigirá a la página principal, que es el calendario de eventos. Si estuviese siguiendo un enlace a otro recurso la redirección será a esa pantalla, siempre que tenga permiso para acceder.

III.2.2. Cerrar Sesión

Una vez dentro de la aplicación un usuario puede cerrar su sesión haciendo click en la opción “Logout” de la barra de navegación. Una vez se cierra la sesión cualquier petición al servidor tendrá como resultado una redirección a la página de login.

III.2.3. Recuperar contraseña

Para recuperar la contraseña se debe pulsar en la pantalla de "Login" sobre el texto "Forgot your password or user?". Esto nos redirige a una nueva pantalla. En esta nueva pantalla debe introducir el email de la empresa (1), una vez introducido, se debe pulsar el botón "Reset Password" (2). En poco tiempo llegará al correo un mensaje con un enlace, haga click en el enlace y le llevará a una pantalla donde debe introducir una nueva contraseña. Si quiere volver a la pantalla de login pulse el botón "Login" (3).

Figura 25: Pantalla de Recuperación de la Contraseña

III.2.4. Crear eventos en el calendario

Cuando inicia sesión se encuentra en la pantalla del calendario de eventos. En esta pantalla puede añadir, modificar o borrar cualquier evento. Para ello solo necesita hacer doble click en el día para añadir un evento nuevo o hacer doble click sobre un evento ya creado para modificarlo o eliminarlo. Tanto soporte como administradores pueden añadir eventos en su calendario.

The screenshot displays the SATR application interface. At the top, there is a navigation bar with links for 'SATR', 'Users', 'Rooms', and 'Logout'. Below this, the user's email 'Employee: diegoderus@gmail.com' is shown next to a search bar. The main area features a calendar grid. A modal window is open, titled '00:00 - 00:05 New event'. It contains a 'Description' field with the text 'New event'. At the bottom of the modal, there is a 'Time period' section with dropdown menus for start and end times (00:00 to 00:05), dates (31 August 2017), and a 'Save' button. There are also 'Cancel' and 'Delete' buttons. The background calendar shows dates from 25 to 01, with a time slot of 15:2 highlighted.

Figura 26: Pantalla de Modificación del Calendario

III.2.5. Listar salas

Para acceder a la lista de salas se tiene que hacer click en el texto "Rooms" de la barra de navegación, en la parte superior de la pantalla. En este caso el soporte también se encuentra limitado pues solo pueden ver sus propias salas mientras que los administradores pueden ver todas las salas creadas. A continuación veremos ambas para comparar.

Empezamos por la lista de salas del soporte. Esta presenta las siguientes características: RoomID (1), que es el identificador de la sala, Name (2), que es el nombre del empleado al cual está asignada la sala, Destination Room (3), que es el email del cliente que solicita el servicio, el botón New Room (4), que sirve para acceder a la pantalla de creación de sala, el del planeta (5) permite acceder a la sala, el icono del lápiz (6) permite modificar la sala y el icono de la X (7) elimina la sala.

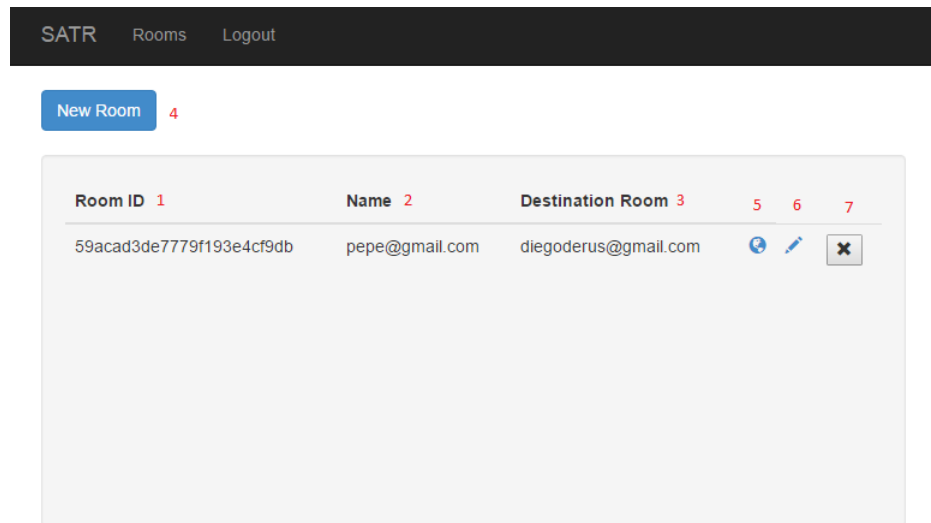
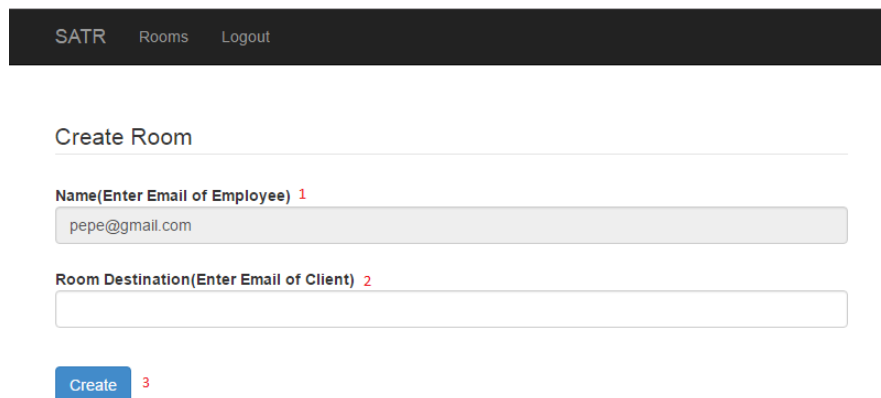


Figura 27: Pantalla de Lista de Salas del Soporte

III.2.6. Crear salas

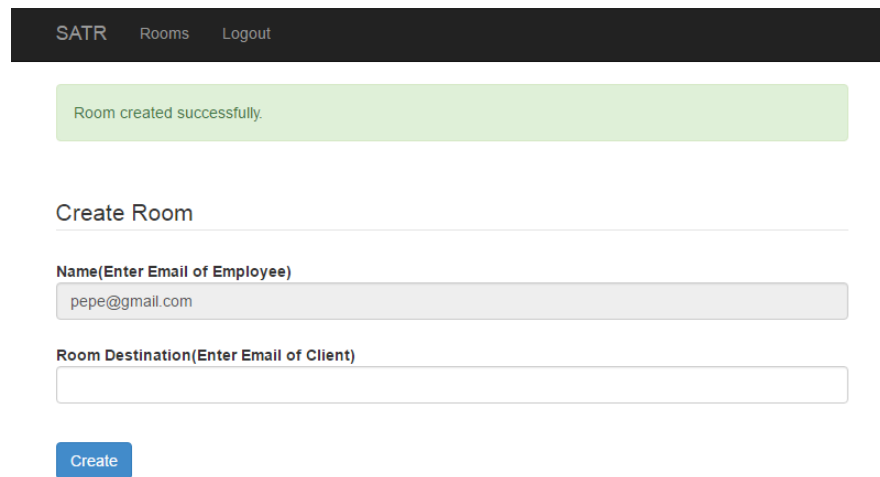
Tanto administradores como el soporte pueden crear salas y modificarlas. El soporte está limitado y solo puede crear salas bajo su nombre, mientras que el administrador puede crear salas a quien quiera. A continuación veremos como es la creación de salas por el soporte, el campo Name (1) se encuentra bloqueado con el email del soporte pero en el caso del administrador no. Rellenando el campo Destination (2) con el email del cliente y realizando click sobre el botón Create (3) se crea la sala y se le enviara al cliente un correo con la URL para incorporarse a la sala.



The screenshot shows the 'Create Room' form in the SATR application. At the top, there is a dark navigation bar with the text 'SATR', 'Rooms', and 'Logout'. Below this, the form is titled 'Create Room'. It contains two input fields: the first is labeled 'Name(Enter Email of Employee) 1' and contains the text 'pepe@gmail.com'; the second is labeled 'Room Destination(Enter Email of Client) 2' and is empty. At the bottom of the form is a blue button labeled 'Create' with a red '3' next to it.

Figura 28: Pantalla de Crear Salas Soporte

Cuando una sala se crea correctamente se envía una advertencia al igual que si se produce algún error. A continuación podemos ver la advertencia en el caso de que la sala se cree correctamente.



SATR Rooms Logout

Room created successfully.

Create Room

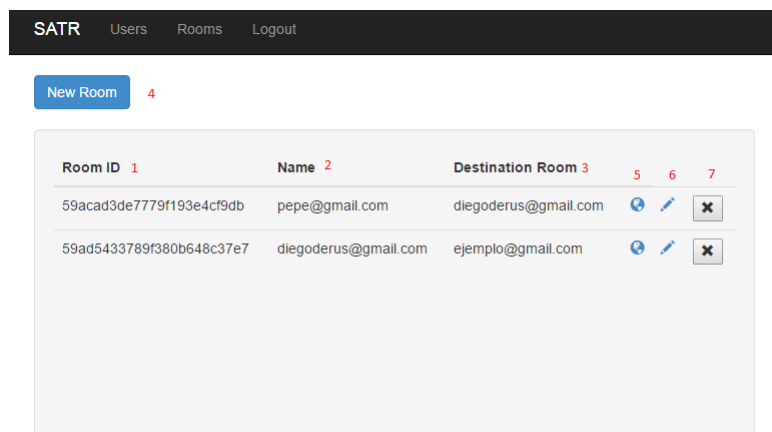
Name(Enter Email of Employee)
pepe@gmail.com

Room Destination(Enter Email of Client)

Create

Figura 29: Advertencia Sala Creada Correctamente

En este caso podemos ver la lista de salas del administrador, este puede ver todas las salas creadas (tanto las del soporte como las del propio administrador). Las funcionalidades que presenta esta pantalla son las mismas que tiene el soporte.



SATR Users Rooms Logout

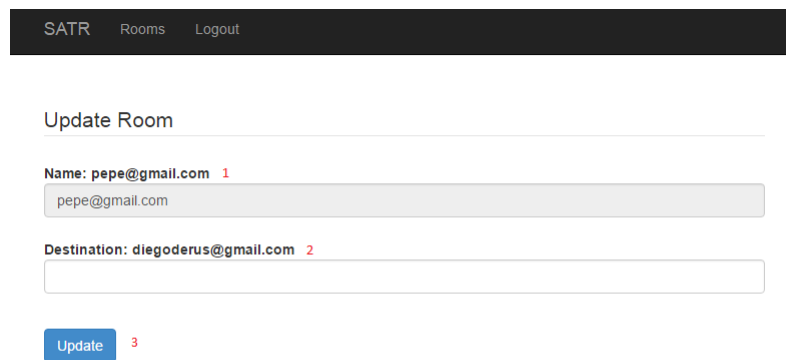
New Room 4

Room ID 1	Name 2	Destination Room 3	5	6	7
59acad3de7779f193e4cf9db	pepe@gmail.com	diegoderus@gmail.com	🔄	✎	✖
59ad5433789f380b648c37e7	diegoderus@gmail.com	ejemplo@gmail.com	🔄	✎	✖

Figura 30: Pantalla de Lista de Salas Administrador

III.2.7. Modificar salas

Realizando click en el icono del lápiz (6) nos redirige a la pantalla de modificación de la sala. Como podemos ver en la modificación tanto de soporte como de administradores, solo se pueden modificar el nombre del empleado al que le pertenece la sala (1) y a quién está dirigida la sala (2). Los cambios se guardan en el momento de hacer click en el botón Update (3).



The screenshot shows the 'Update Room' form in the SATR application. At the top is a dark navigation bar with 'SATR', 'Rooms', and 'Logout' links. The form title 'Update Room' is centered. Below it, the 'Name' field is labeled 'Name: pepe@gmail.com' with a red '1' next to it, and the input box contains 'pepe@gmail.com'. The 'Destination' field is labeled 'Destination: diegoderus@gmail.com' with a red '2' next to it, and the input box is empty. At the bottom is a blue 'Update' button with a red '3' next to it.

Figura 31: Pantalla de Modificación Sala Soporte

III.2.8. Eliminar salas

Para modificar las salas se tiene que hacer click en el icono de la X (7) que está al lado de cada sala en la lista. Cada sala se elimina individualmente. Una vez se elimina la sala no es posible recuperarla.

III.2.9. Listar usuarios

Para acceder a la lista de usuarios se tiene que iniciar sesión con el rol del administrador. Una vez iniciado, nos redirige a la pantalla del calendario de eventos del administrador. Para acceder a la lista de usuarios debemos hacer click en el Users^{en} la barra de navegación, parte superior. Una vez hacemos click nos redirige a la pantalla donde se encuentra la lista de usuarios. Esta se puede ordenar haciendo click en los nombres en negrita (Username (1), Email (2), Rol (3)). Para crear un usuario se tiene que pulsar el botón "New User"(4), para acceder a la modificación en el icono del lápiz (5) y para eliminar el usuario en el icono de la X (6).

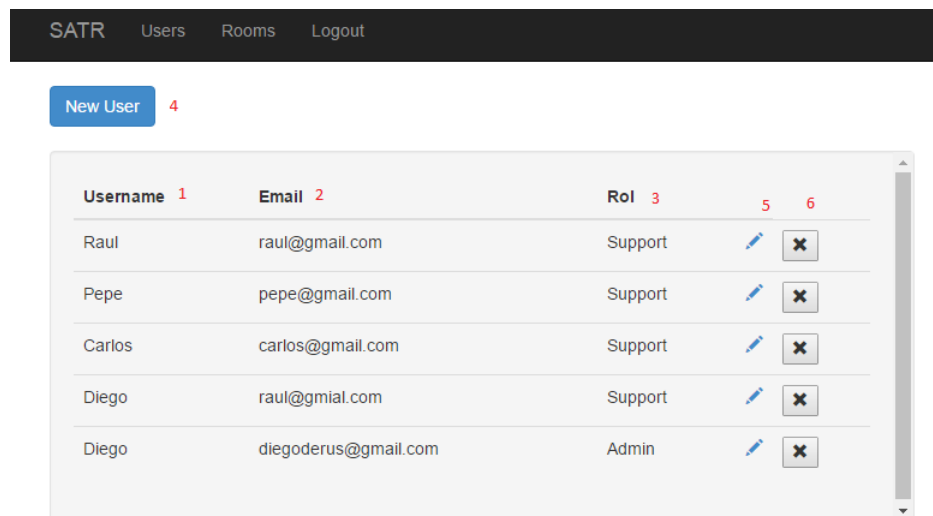
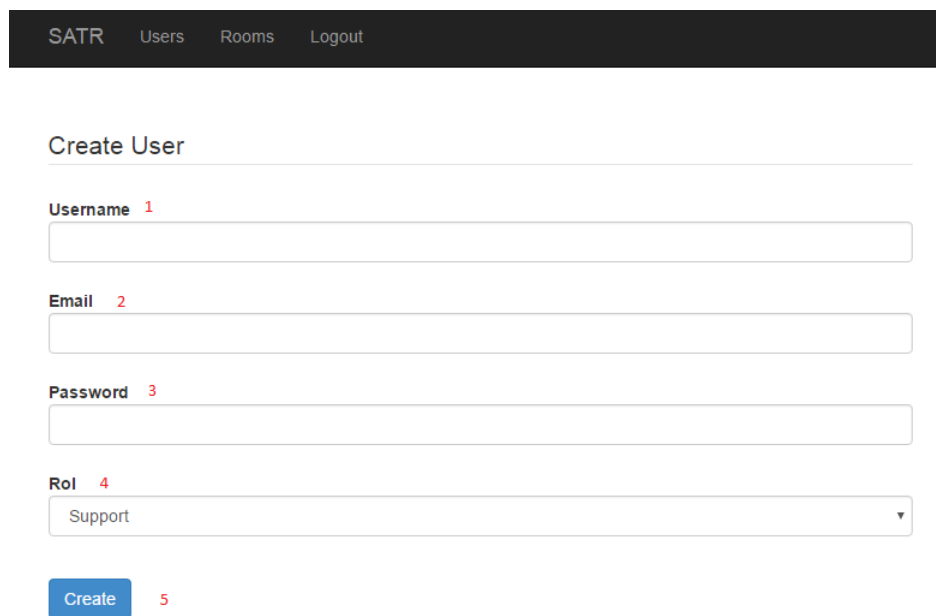


Figura 32: Pantalla de Lista de Usuarios

III.2.10. Crear usuario

Se accede pulsando el botón "New User." en la lista de usuarios. El administrador es el único con la funcionalidad de crear empleados. Para crearlos, necesita rellenar los siguientes campos: Username (1), nombre del empleado, Email (2), correo de la empresa que debe ser único para cada empleado, Password (3), contraseña, Rol (4), Support o Admin. Para confirmar la creación del usuario se tiene que pulsar el botón "Create" (5).

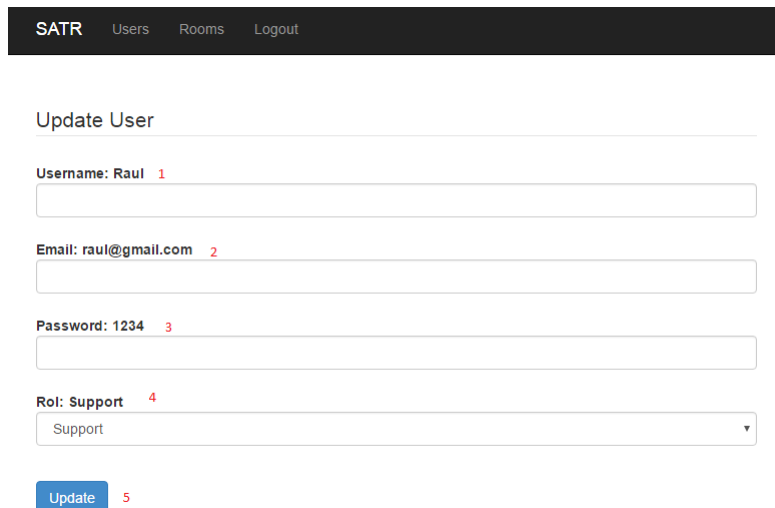


The screenshot shows the 'Create User' form in the SATR application. At the top is a dark navigation bar with links for 'SATR', 'Users', 'Rooms', and 'Logout'. Below the navigation bar, the title 'Create User' is displayed. The form consists of four input fields: 'Username' (labeled 1), 'Email' (labeled 2), 'Password' (labeled 3), and 'Rol' (labeled 4). The 'Rol' field is a dropdown menu currently showing 'Support'. At the bottom of the form is a blue 'Create' button (labeled 5).

Figura 33: Pantalla de Creación de Usuarios

III.2.11. Modificar usuarios

Para modificar los usuarios se tiene que hacer click en el icono del lápiz que está al lado de cada usuario en la lista. Cada usuario se modifica individualmente. Una vez hacemos click se nos redirige a una pantalla donde debemos introducir los datos tal y como se pedían en la creación. A los empleados se les pueden cambiar todos los campos: Username (1), Email (2), Password (3), Rol (4). Los cambios se guardan haciendo click en el botón de Update (5).



SATR Users Rooms Logout

Update User

Username: Raul 1

Email: raul@gmail.com 2

Password: 1234 3

Rol: Support 4

Support

Update 5

Figura 34: Pantalla de Modificación de Usuarios

III.2.12. Eliminar usuarios

Para eliminar los usuarios se tiene que hacer click en el icono del X (6) que está al lado de cada usuario en la lista. Cada usuario se elimina individualmente. Una vez se elimina el usuario no se puede recuperar.

III.2.13. Ver calendario de otros usuarios

Los administradores pueden modificar los calendarios de todos los empleados. Para esto, en la pantalla de su calendario se muestra un botón para buscar el calendario de quien desee. Rellenando el campo (1) con el email de un empleado y haciendo click en el icono de la lupa (2) nos redirige a la misma pantalla pero con el calendario del empleado que hemos introducido. Podemos ver en todo momento qué empleado estamos viendo, ya que se muestra a la izquierda de la pantalla (3). Para modificar los eventos de los calendarios se realiza de la misma manera que si fueran el propio calendario del administrador (haciendo doble click sobre el día o evento).

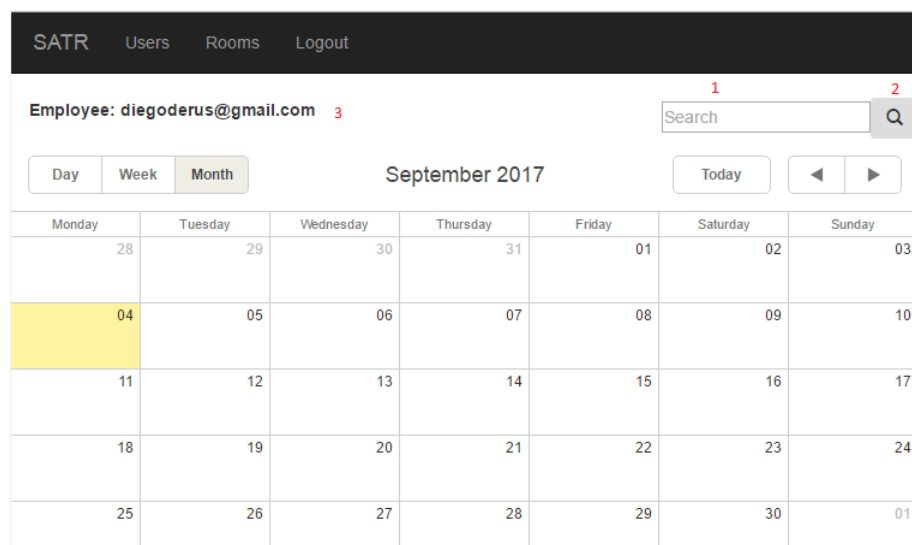
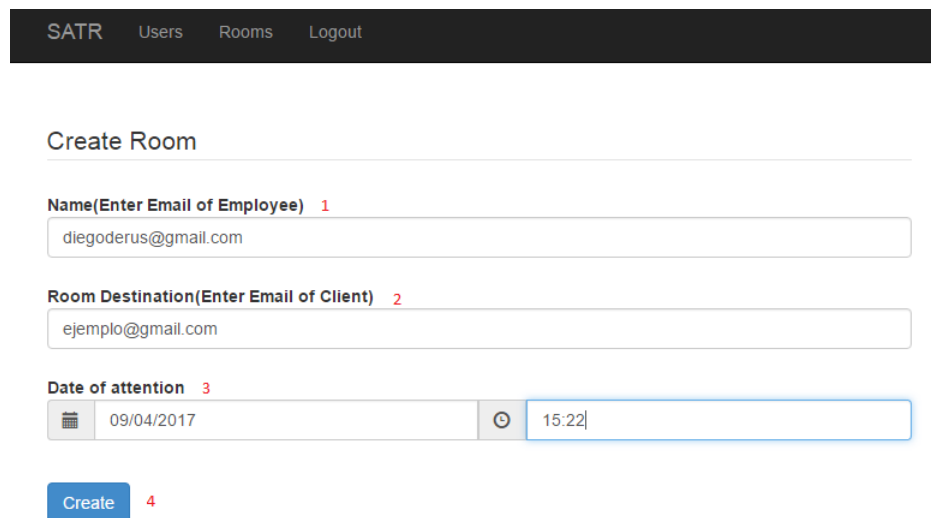


Figura 35: Pantalla de Calendario

III.2.14. Crear salas administrador

Cuando el administrador quiere crear una sala este tiene más opciones que el de soporte. Como podemos ver en la siguiente imagen, puede añadir el nombre del empleado que quiera en el campo Name (1), el email del cliente en Room Destination (2) y en Date of Attention (3) el administrador puede elegir una hora y día para crear la sala y la atención al cliente. De esta forma distinguimos entre el soporte que solo puede crear salas bajo su nombre y que son creadas para atender en el momento a los clientes y los administradores que pueden crear salas bajo el nombre de quien deseen y en un día y hora que quieran.



SATR Users Rooms Logout

Create Room

Name(Enter Email of Employee) 1
diegoderus@gmail.com

Room Destination(Enter Email of Client) 2
ejemplo@gmail.com

Date of attention 3
09/04/2017 15:22

Create 4

Figura 36: Pantalla de Creación de Salas Administrador

III.2.15. Videoconferencia

A continuación vamos a separar el manejo de la videoconferencia en lado cliente y lado de soporte.

1. Lado del soporte

Se accede a la pantalla de videoconferencia al pulsar el icono del mundo que se encuentra al lado de cada sala en la pantalla de lista de usuarios. Se nos muestra los siguiente: un recuadro grande (1) que pertenece al escritorio compartido, dos recuadros pequeños (1 para la webcam del cliente y 2 para la webcam del soporte), el identificador del usuario conectado a la sala (4), el estado de la sala (5), y el recuadro donde se muestra el chat (6).

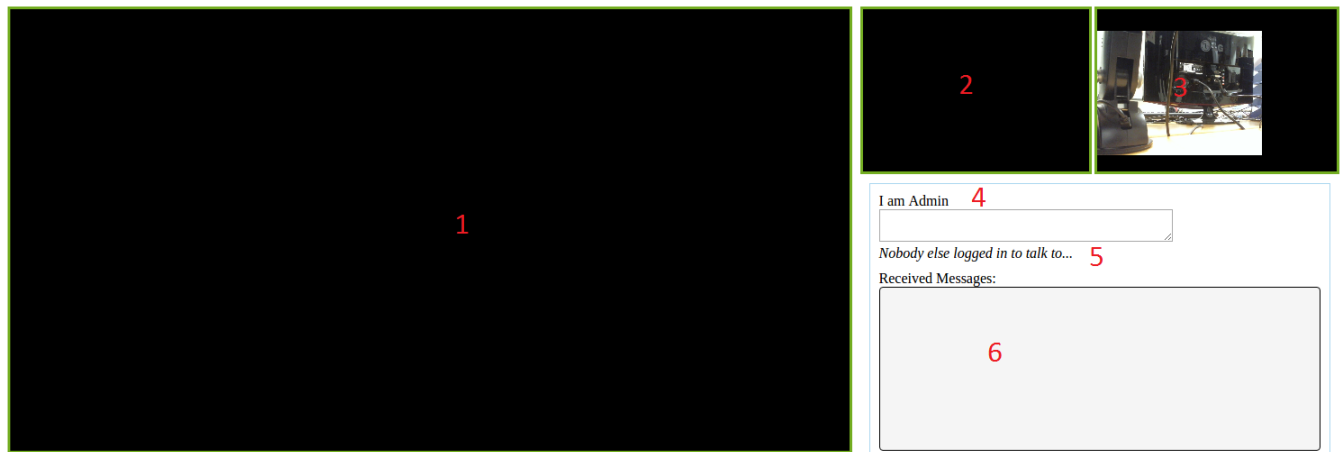


Figura 37: Acceso a la sala del soporte

2. Lado del cliente

Se accede mediante la URL enviada a su correo. En esta pantalla podemos observar que lo primero que nos aparece es elegir qué pantalla debemos compartir (1).

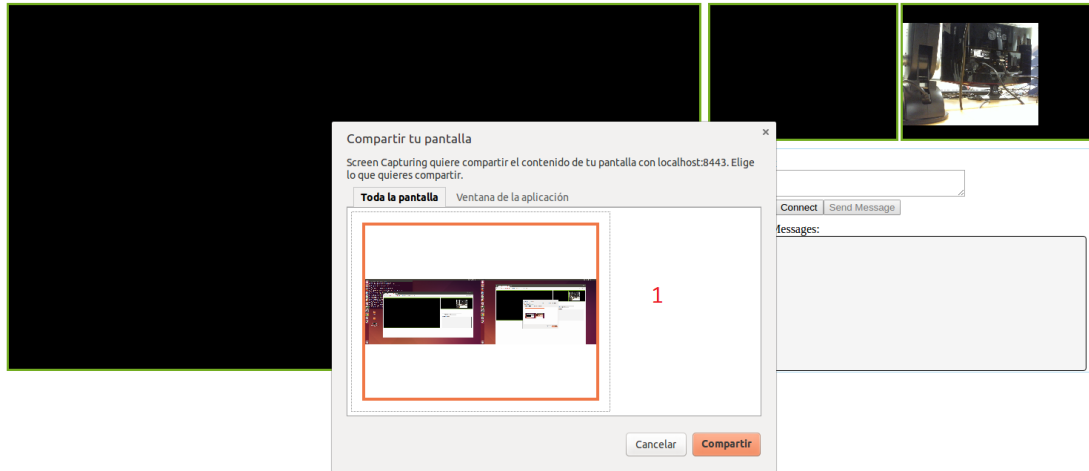


Figura 38: Solicitud de compartición de pantalla

En la siguiente imagen se muestran los otros elementos que tiene también la videoconferencia de soporte. Para establecer la comunicación se pulsa el botón Connect (4).

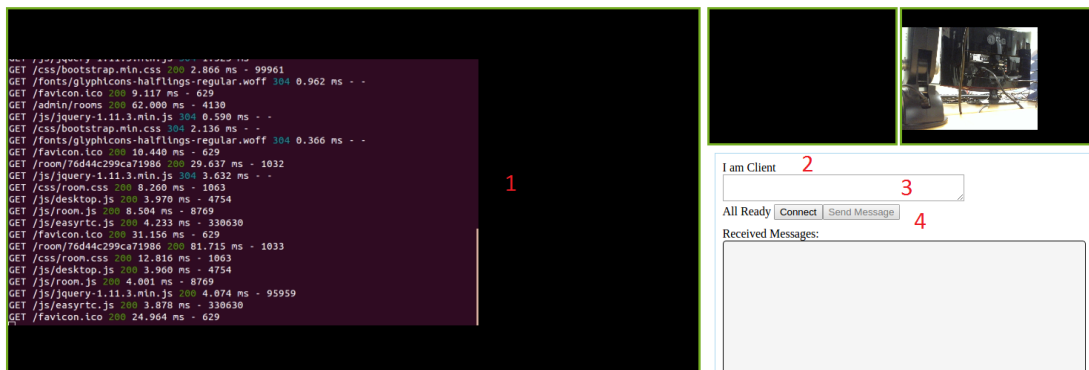


Figura 39: Acceso a la sala del cliente

Una vez la comunicación está establecida ambos usuarios puede emplear el chat, puesto que disponen de una zona para escribir el mensaje (1). Pulsando el botón Send Message (2) se envían al otro usuario. Podemos ver los mensajes en el recuadro del chat (3).

■ Vista del soporte

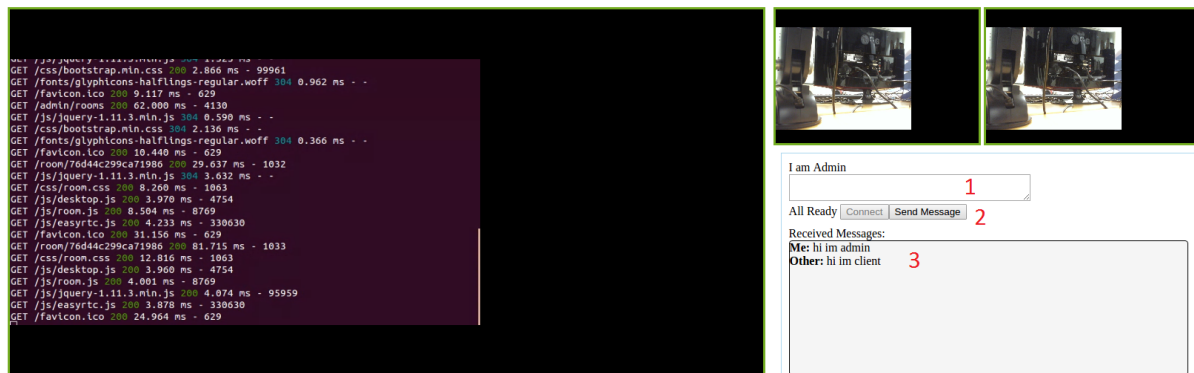


Figura 40: Pantalla de videoconferencia del soporte

■ Vista del cliente

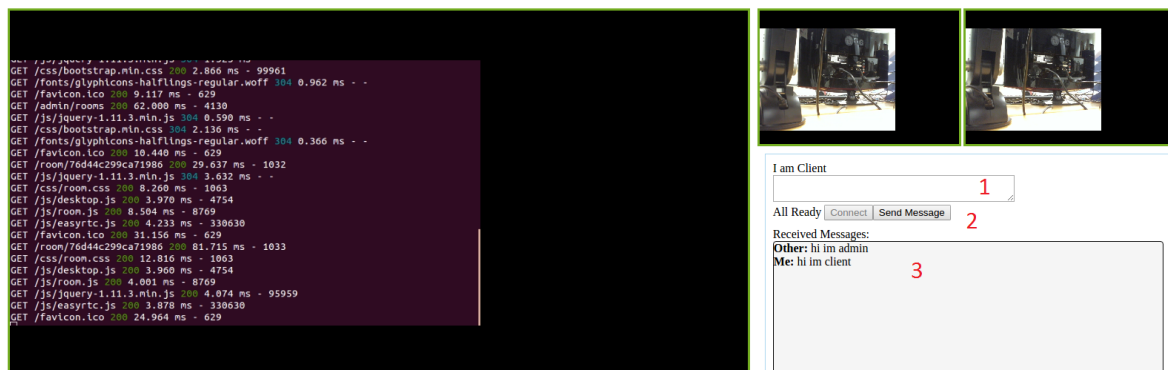


Figura 41: Pantalla de videoconferencia del cliente