

peeker

Aplicación web de videoconferencia

20 de junio de 2018

Mario garcía ramos

Desarrollo de aplicaciones web – IES Virgen de la Paz (2017 – 2018)



ÍNDICE

[ÍNDICE 1](#_Toc515951082)

[Justificación del proyecto y objetivos 2](#_Toc515951083)

[Introducción 2](#_Toc515951084)

[Metodología y desarrollo del proyecto 2](#_Toc515951085)

[Tecnologías 2](#_Toc515951086)

[Software empleado 3](#_Toc515951087)

[Comunicación WebRTC 3](#_Toc515951088)

[Resultados y discusión 6](#_Toc515951089)

[Conclusiones 6](#_Toc515951090)

[Bibliografía y referencias 6](#_Toc515951091)

[Anexos/ Otros 6](#_Toc515951092)

[ANÁLISIS DEL SISTEMA DE INFORMACIÓN 7](#_Toc515951093)

[Definición y alcance del sistema 7](#_Toc515951094)

[ESTABLECIMIENTO DE REQUISITOS 7](#_Toc515951095)

[Definición de interfaces de usuario 9](#_Toc515951096)

[Especificación de principios generales de la interfaz 9](#_Toc515951097)

[DISEÑO DE LA ARQUITECTURA DE LA APLICACIÓN 9](#_Toc515951098)

[ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS 10](#_Toc515951099)

[WEBGRAFÍA 13](#_Toc515951100)

TODO: Asignar un subtítulo a cada figura y sustituir figura X en cada una de las referencias por el número asignado.

# Justificación del proyecto y objetivos

Escogí este proyecto, porque desde siempre me ha interesado la tecnología que hace posible la comunicación entre dos personas. De esta curiosidad y de otras inquietudes como el aprendizaje de tecnologías o *frameworks* como Angular y Node JS ha nacido este proyecto.

En este proyecto se pretenden llevar a cabo las siguientes funcionalidades:

* Comunicación por videoconferencia
* Sistema de autenticación de usuarios
* Lista de amigos por usuario
* Integración de la aplicación de forma nativa para iOS y Android mediante IONIC.

# Introducción

Aquí puedes explicar más sobre el proyecto, las tecnologías que vas a utilizar, etc.

## Tecnologías

En este proyecto se van a emplear las siguientes tecnologías

* Angular (angular 4) – Google. Para gestionar la aplicación en el lado del cliente .
* Node JS – Node JS Foundation. Para emplearse a modo de servidor web.
* Express JS – Node JS Foundation. Para simplificar el desarrollo de las rutas.
* Ionic Framework – Para integrar la aplicación de forma nativa en los sistemas operativos iOS y Android.
* WebRTC – Para la comunicación multimedia que hace posible la videoconferencia.
* Socket.io – Para la comunicación entre clientes (intercambio de información sobre los ICE Candidates, sobre desconexiones, llamadas entrantes, etc).
* MongoDB – Como base de datos.
* Mongoose – Para gestionar la base de datos desde Node JS.
* BCryptJS – Para la encriptación de las contraseñas
* Jwt.io (JSON Web Token) – Para la gestión de las sesiones.
* Local Storage – Para almacenar el token en el lado del cliente

PROYECTO MVC POR ANGULAR Y NODEJS

# Metodología y desarrollo del proyecto

En este apartado se describen las metodologías y tecnologías empleadas en el desarrollo del proyecto.

## Software empleado

El software empleado para la realización de este prototipo es el siguiente:

* Microsoft Office 2016 (Word y Power Point): Para la redacción de este documento y la elaboración de la presentación.
* Node JS
* Angular 4
* Visual Studio Code. Para la codificación de la página y la gestión de los ficheros de la misma.
* Express. Para facilitar el uso del desarrollo en el lado del servidor.
* Google Chrome, Mozilla Firefox, Microsoft Edge y Opera para la visualización del contenido web y para la resolución de errores mediante el debugging.
* Windows 10 como sistema operativo en el que se ha desarrollado el prototipo.

## Comunicación WebRTC

WebRTC es una tecnología de comunicación en tiempo real (Real-Time Communications o RTC) que hace posible la comunicación entre clientes mediante llamadas de voz, chat de vídeo, compartición de mensajes o incluso de archivos ***P2P*** (*Peer to peer*).

Para que la comunicación sea posible, primero han de intercambiar algunos datos como información sobre el dato a transferir (dimensiones y resolución de la cámara, códec de audio, si se transmite audio y video o sólo uno de los dos…), además de las direcciones IPs para la posterior conexión *P2P*.

A continuación, se explica la secuencia de acontecimientos que hacen posible la comunicación en este proyecto (ver también la figura 1).

Se presenta el siguiente escenario: Un **usuario 1** llama a un **usuario 2**:

1. Al conectarse un usuario a la aplicación, se guardan en una lista de usuarios online su nombre junto con un objeto ***socket*** asociado al flujo de datos que apunta al cliente.
2. Cuando el usuario 1 llama al usuario 2, se le solicita permiso para poder obtener el flujo de datos de la videocámara y el micrófono que estén conectados.
3. Al aceptar, el cliente del usuario 1 envía mediante la librería sockets.io un mensaje al servidor, el cual comunica con el usuario 2 en caso de estar presente en la lista de usuarios online.
4. Una vez el usuario 2 ha sido notificado, y coge la llamada, se realiza el mismo procedimiento que con los datos de cámara y vídeo del usuario 1.
5. Tras esto, el usuario 2 envía un mensaje al servidor para informar de que ya ha cogido la llamada, y este les comunica a los dos usuarios que ambos están listos para comunicar la comunicación.
6. Ambos usuarios crean un objeto **PeerConnection** y el usuario 1, al ser el que llamó, crea una oferta (**offer**) y se la envía al usuario 2 a través del servidor mediante el mismo sistema de mensajes basado en web sockets que se ha mencionado antes.
7. El usuario 1 genera un **ICE Candidate** (**Interactive Connectivity Establishment**) y es enviado a través del servidor al usuario 2, cuyo fin es establecer la conexión P2P.
8. Cuando el usuario 2 recibe el ICE Candidate, este envía el suyo a través de la creación de una respuesta (**answer**), la cual es enviada al usuario 1 a través del servidor.
9. Una vez ambos tienen los datos de conexión, la comunicación P2P comienza e intercambian entre ellos los datos y se pueden ver y escuchar el uno al otro.

**Nota**: en el apartado Anexos / Otros se profundiza más en el tema a nivel técnico.

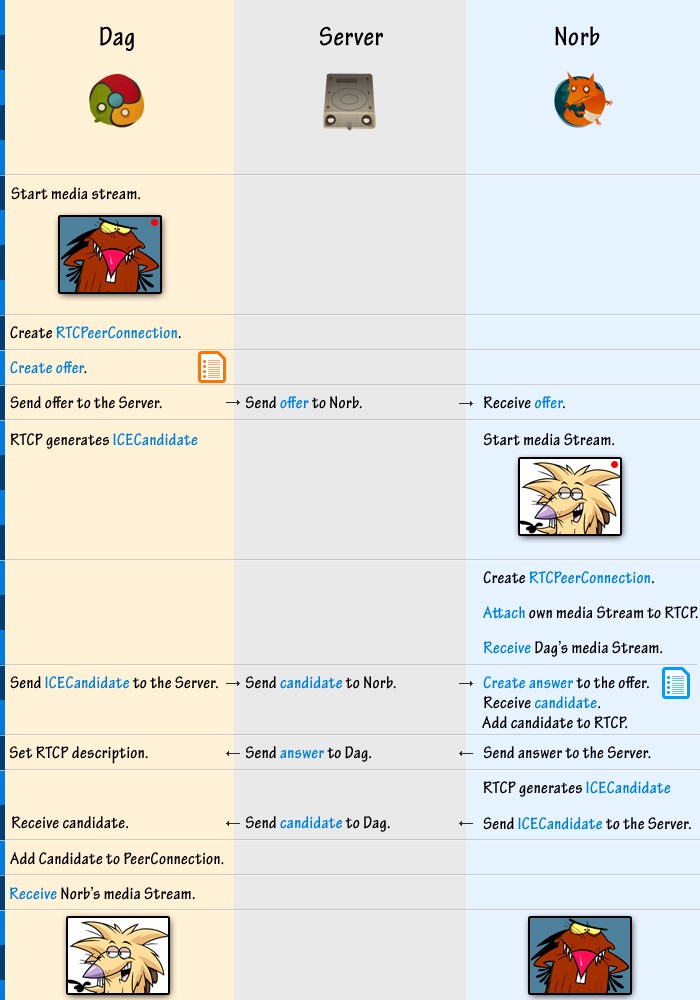


Figura 1

# Resultados y discusión

Aquí puedes hablar de si has cumplido la planificación inicial, si por algún problema no has podido terminar alguno o si has tenido que realizar algún cambio

# Conclusiones

Este apartado uno de los más importantes del proyecto, pues resumes todo lo que has realizado, la experiencia con las tecnologías nuevas, hacia donde podría continuar el proyecto en el futuro, etc.

# Bibliografía y referencias

Toda la webgrafía y biografía utilizada para el proyecto

* Angular (angular 4) – Google (<https://angular.io/>)
* Node JS – Node JS Foundation (<https://nodejs.org/es/>)
* Express JS – Node JS Foundation (<http://expressjs.com/es/)>
* Ionic Framework – Ionic (<https://ionicframework.com/>)
* WebRTC – WebRTC (<https://webrtc.org/>)
* MongoDB – MongoDB (https://www.[mongodb](https://www.mongodb.com/).com/)
* Mongoose – Mongoose (<http://mongoosejs.com/>)
* Mongoose unique
* <https://www.npmjs.com/package/bcryptjs>
* Jwt.io (user tokens) (json web token
* **USING LOCAL STORAGE**
* **fontawesome**

# Anexos/ Otros

Aquí puedes incluir código, capturas de pantallas más extensas de la aplicación etc.

## Estructura de ficheros

En este apartado se explica en detalle la estructura de carpetas y de ficheros que conforma el proyecto.

## Comunicación cliente – servidor mediante web sockets

La comunicación entre el cliente y el servidor para establecer las videoconferencias, están basadas en *web-sockets.*

Ambos pueden ponerse a la escucha gracias a la librería socket.io (tanto del lado del cliente, como del servidor).

### Servidor

Como ya se menciona en el apartado anterior [Estructura de ficheros](#_Estructura_de_ficheros), el fichero *www*, contiene la gestión de los sockets entre otras cosas.

Lo primero es importar las dependencias para poder usar los objetos deseados (ver figura x).



1. La librería os es requerida para obtener la dirección IPv4 para mandar un mensaje al socket cliente en caso de que pregunte por la dirección IP actual (ver figura x).



1. El módulo de node static permite crear una ruta estática virtual existente o no en el sistema que facilita el acceso a los ficheros a la hora de servirlos.
2. Por último, el módulo que contiene la librería de web sockets ***socket.io***, gracias a él, se podrá poner a la escucha de nuevos clientes. Crea una ruta propia que es de donde se sirve el cliente a la hora de solicitar la API del lado del cliente (ver apartado [Cliente](#_Cliente)).

En el lado del servidor, esta librería basa su funcionalidad en listeners. Se pone a la escucha una instancia de http.Server y en función de qué evento se emita en el lado del cliente, realizará una función u otra.

Una vez dicho esto, ¿cómo gestiona la aplicación la comunicación del chat con este mecanismo? De la siguiente manera:

Lo primero que hace el usuario al iniciar la aplicación es conectarse al servidor mediante el método connect del objeto io de la librería socket la cual se obtiene del archivo socket.io.js.



presentarse al servidor, de manera que sea añadido a una lista de usuarios online. Lo hace mediante un mensaje “presentation”.

### Cliente



# ANÁLISIS DEL SISTEMA DE INFORMACIÓN

## Definición y alcance del sistema

En este apartado se refleja la definición del alcance del sistema, es decir, se plantea el problema y los objetivos a alcanzar en el desarrollo de este proyecto.

El problema planteado es el siguiente. Se desea desarrollar una aplicación que consista en la comunicación de los usuarios mediante la videoconferencia, la funcionalidad principal es la siguiente:

* Establecer comunicación entre usuarios mediante videoconferencias
* Establecer comunicación entre usuarios mediante llamadas de audio
* Establecer comunicación entre usuarios mediante mensajes instantáneos de texto
* Altas y bajas de usuarios
* Proporcionar una lista de contactos del usuario que ha iniciado sesión con los que comunicarse.

Los objetivos a destacar son:

* Conocer las tecnologías citadas en el siguiente listado.
* Realizar un proyecto web multiplataforma desde cero realizando una combinación de tecnologías poco habitual e intentar sacar el máximo rendimiento y partido de ello.

## ESTABLECIMIENTO DE REQUISITOS

En esta tarea se procede a analizar los requisitos del sistema. Para se definen los siguientes tipos de requisitos:

1. Funcionales (RF): aquellos requisitos dedicados exclusivamente a definir funcionales del sistema, es decir, cómo va a funcionar nuestro programa
2. Técnicos (RT): aquellos requisitos que no definen como funciona el programa sino cómo ha de estar hecho.
3. De interfaz (RI): aquellos requisitos que especifican como ha de ser la interfaz de usuario.

El catálogo de requisitos obtenido es el siguiente, estando en negrita los resueltos en el prototipo a presentar:

**Requisitos funcionales (RF)**

* + **RF1.** Se deben poder realizar videoconferencias.
  + **RF2.** Se deben poder realizar conferencias de audio.
  + **RF3.** Se deben poder realizar comunicación por medio de mensajes instantáneos de texto.
  + **RF4.** Los usuarios podrán darse de alta e ingresar en la aplicación.
  + **RF5.** Cada usuario podrá tener una lista de \*amigos\* con los que podrá contactar.
  + **RF6.** Se debe poder colgar la llamada una vez se ha establecido.
  + **RF7.** Se debe poder pausar la cámara una vez se ha establecido la llamada.
  + **RF8.** Se debe poder silenciar el micrófono una vez se ha establecido la llamada.
  + **RF8.** Se debe poder colgar la llamada cuando un usuario llama a otro, antes de que se haya establecido.
  + **RF9.** Se debe poder colgar la llamada cuando un usuario es llamado por otro, antes de que se haya establecido.
* **Requisitos técnicos (RT)**
  + **RT1.** En toda videoconferencia se deben percibir ambas cámaras.
  + **RT2.** Sólo se recibirá el audio del usuario remoto tanto en videoconferencias como en llamadas de audio.
  + **RT3.** Los mensajes instantáneos deben estar limitados a 200 caracteres.
  + **RT4.** Los usuarios accederán a la funcionalidad de la aplicación una vez se hayan registrado.
* **Requisitos de interfaz (RI)**
  + **RI1.** Al cargar la aplicación se debe mostrar una animación en la que el logo esté rodeado de un borde circular ambos de color gris con un color recorriendo este último.
  + **RI1.** La aplicación debe ser “responsive”, flexible al tamaño del dispositivo en el que se va a mostrar.

## Definición de interfaces de usuario

(Teoría)

## Especificación de principios generales de la interfaz

(Mockups + resultado final)

# DISEÑO DE LA ARQUITECTURA DE LA APLICACIÓN

En este apartado se explica la arquitectura de la comunicación entre cliente y servidor, así como la comunicacion entre usuarios peer-to-peer.

(diagramas)

# ESPECIFICACIÓN TÉCNICA DEL PLAN DE PRUEBAS

En esta actividad se desea realizar un análisis de las pruebas a las que se va a someter el software una vez esté finalizado el periodo de desarrollo del mismo. Puesto a que son varias las partes en las que se divide el proyecto

|  |  |  |
| --- | --- | --- |
| **Id** | **Descripción** | **Resultado obtenido** |
| **1** | Un usuario está llamando a otro. |  |
| **1.1** | Se muestra la pantalla de llamada con sus correspondientes botones |  |
| **1.2** | Se solicitan los permisos para la obtención de datos a través del micrófono y la videocámara. |  |
| **1.3** | Si el usuario acepta los permisos se procede a mostrar por pantalla el vídeo capturado en directo. |  |
| **1.4** | Una vez obtenidos los datos de vídeo y audio y mostrados por pantalla, se procede a llamar al usuario en cuestión. |  |
| **2 - A** | El usuario a llamar recibe la llamada. |  |
| **2.1** | Al descolgar se notifica sobre ello al otro usuario que llamó, después de haber aceptado los permisos para la obtención de la cámara. |  |
| **2.1.1** | El usuario que llamó recibe el mensaje y está listo para la videoconferencia. |  |
| **2.2** | Al descolgar, se solicitan los permisos de la cámara y micrófono |  |
| **2.3** | Al aceptar los permisos habiendo recibido una llamada se establece el vídeo y el audio locales correctamente. |  |
| **2.4** | Una vez se han aceptado los permisos y tenemos video y audio locales funcionando, se envía un mensaje al servidor y este responde a los dos usuarios con un estado de “listo”. |  |
| **2.5** | Una vez el canal de comunicación socket está listo se procede a crear la conexión p2p (peer to peer) en la que se incluye el flujo de datos de la cámara y micrófono y el usuario que llamó crea una oferta. |  |
| **2.6** | Una vez se crea la oferta, el servidor recibe el mensaje con los datos para reenviársela al otro usuario. |  |
| **2.7** | El otro usuario recibe la oferta con los datos deseados y crea una respuesta correctamente. |  |
| **2.8** | Una vez se crea la respuesta, el servidor recibe el mensaje con los datos para reenviársela al otro usuario. |  |
| **2.8** | Una vez establecida la comunicación peer to peer, los usuarios ven su propio video y ven y escuchan el vídeo del otro usuario. |  |
| **2 - B** | El usuario está desconectado. Se muestra un mensaje al que llamó y se para la llamada. |  |
| **3** | Habiendo una llamada en curso el usuario cuelga y se cierra el flujo de datos relativo a la cámara y el micrófono. |  |
| **3.1** | Habiendo una llamada en curso el usuario cuelga desaparece la pantalla de llamada. |  |
| **4** | Habiendo una llamada en curso un usuario cuelga y para el otro participante de la llamada desaparece la pantalla de llamada. |  |
| **4.1** | Habiendo una llamada en curso un usuario cuelga y el flujo de datos relativo a la cámara y el micrófono del otro participante finaliza. |  |
| **5** | En una llamada en curso se presiona el botón de silenciar micrófono y el flujo de datos se corta. El otro participante deja de escucharle. |  |
| **6** | En una llamada en curso se presiona el botón de pausar vídeo y el flujo de datos se corta. El otro participante deja de verle. |  |
| **7** | Tras realizar una llamada el usuario vuelve a llamar al mismo usuario y no se comparte el flujo de datos hasta que se acepta la llamada. |  |
| **8** | Si un usuario llama a otro y este rechaza la llamada, la pantalla de llamada desaparece |  |
| **8.1** | Si un usuario es llamado por otro y rechaza la llamada, la pantalla de llamada recibida desaparece. |  |
| **9** | Si un usuario es llamado por otro y el que llamaba cancela la llamada, la pantalla de llamada recibida desaparece. |  |
| **9.1** | Si un usuario llama a otro y el que llamaba cancela la llamada, la pantalla de llamada desaparece. |  |
| **10** | Cuando se muestra la pantalla de llamada al llamar a otro usuario, no se muestra el vídeo de la llamada anterior. |  |
| **11** | Cuando el otro participante de la llamada cierre la aplicación, la llamada finalizará |  |
| **12** | Cuando se coja una llamada desaparece el panel de llamada recibida y aparece el de llamada en curso. |  |
| **13** | Al haber eliminado el mensaje por habitación, se establece correctamente la llamada mediante la comunicación por el método emit. (se comunican mediante el sendmessage y a parte mediante el on call emit etc del socket) |  |
| **14** | Al haber silenciado el micrófono en la llamada anterior no se mantiene en la siguiente. |  |
| **15** | Al haber pausado la cámara en la llamada anterior no se mantiene en la siguiente. |  |
| **16** | En una llamada en curso, cuelga el usuario que llamó y la llamada finaliza para ambos. |  |
| **17** | En una llamada en curso, cuelga el usuario que ha sido llamado y la llamada finaliza para ambos. |  |

# WEBGRAFÍA