Sistema de Gestión de Reuniones y Eventos Empresariales usando Programación Concurrente y Sincronizada

Johan Calderon Valenzuela, Estudiante Universitario, Arnold Condori Mamani, Estudiante Universitario, Roycer Cordova Cartagena, Estudiante Universitario, Freddy Valdivia Berrio, Estudiante Universitario, Max Banda Yanqui, Estudiante Universitario.
Universidad Nacional de San Agustín, Perú, jcalderonva@unsa.edu.pe, mcondorimama@unsa.edu.pe, rcordovac@unsa.edu.pe, fvaldiviab@unsa.edu.pe, maxx252@gmail.com.

Resumen- El problema de Programación de Reuniones (PR) es complicado ya que requiere mucho tiempo de solución, encontrar la combinación adecuada para satisfacer las disponibilidades, preferencias y privacidad de todos los asistentes es un desafío latente. El objetivo de nuestro artículo es resolver la programación de reuniones o eventos empresariales proponiendo servicios en tiempo real que resuelven los desafíos de la concurrencia y sincronización. Estas características se logran estructurando la descripción de los servicios en la nube en un modelo jerárquico y poblando en una estructura de árbol que contienen referencias a los servicios y datos en tiempo real. Gracias a la estructura jerárquica de los servicios que usamos se consiguió en el sistema una buena sincronización con el acceso de datos y manteniendo siempre un canal de notificación en tiempo real. Concluimos que el aprovechamiento de servicios en tiempo real facilita el acceso a datos y a los cambios basados en notificaciones, a fin de garantizar la entrega de los servicios a las partes interesadas en un tiempo real.

Palabras Clave: concurrencia, sistema, reuniones, sincronización, tiempo real.

I. INTRODUCCIÓN

Cuando se desea realizar una reunión o evento, se busca tener un tiempo en común para que los participantes puedan asistir y participar, logrando hacer una programación del horario del evento y que los invitados principales estén de acuerdo, además de la necesidad de una herramienta que facilite el envío del evento y respuesta de su participación.

Varios esfuerzos de investigación, que tratan de los problemas de PR, se llevaron a cabo y todos tratan de llegar a la mejor solución. Entre ellos, Chun [8] propuso un enfoque basado en la optimización de la reunión basada en la programación y centrado en las preferencias de los usuarios.

Los autores elaboraron algoritmos basados en la negociación de agentes y redujeron el espacio de búsqueda a intervalos de tiempo limitados por la disponibilidad de los participantes. La principal contribución de su trabajo es considerar la búsqueda como un proceso de negociación entre los agentes de software que actúan en nombre de los participantes en una reunión. Por lo tanto, el enfoque es completamente automatizado y las responsabilidades de planificación se confían totalmente a los agentes de software.

En el trabajo propuesto por Hassine [9], los autores manejan esta situación como un Problema de Satisfacción de Restricción de Valor Dinámico y propusieron el enfoque MSRAC (for meeting scheduling with reinforcement of arc consistency) en un entorno multi- agente y dinámico. Los autores consideraron las preferencias de los usuarios y la importancia de las reuniones. Los agentes tienen un objetivo local; que satisface sus limitaciones locales; y un objetivo global que es maximizar el número de reuniones planificadas. El trabajo anterior de los autores se centró en mantener la consistencia del arco mientras que relajaba las preferencias del usuario, preservando a lo más el aislamiento de usuarios implicados y minimizando el número de mensajes intercambiados.

El problema de PR pertenece a la clase NP de los problemas. Incluso si la dificultad depende de las instancias, para casos más restringidos, encontrar una solución aceptable puede ser muy tedioso. Sin embargo, para pequeñas instancias; como la reunión de equipo; el proceso de resolución puede ser realizado mediante el uso de un algoritmo completo como en el trabajo de Hassine [9]. De hecho, el ABT (Asynchronous backtracking) para pequeñas instancias (menos de 15 reuniones). Las investigaciones más importantes se centraron en tratar el problema de PR de manera distribuida para mantener la privacidad de los usuarios. Sin embargo, la mayoría de estos esfuerzos son insuficientes para resolver sobre casos restringidos y han considerado sólo un pequeño número de reuniones y participantes. En efecto para un mayor número de participantes, el número de mensajes intercambiados crece exponencialmente y consecuentemente penaliza el tiempo de ejecución.

Digital Object Identifier: (to be inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).

Formalización de Programación de Reuniones

- Un iniciador: Una persona que propone las reuniones
- Una duración: En cada reunión se especula una duración
- Una oficina: La oficina de reunión está limitado a la capacidad de la habitación.
- Prioridad: Cada reunión tiene una importancia
- Participantes: Cada participante tiene un orden de preferencia para las reuniones expresadas en el calendario.

BuMeet, es una aplicación que nos permitirá crear reuniones, donde el creador de un evento, podrá agregar a los participantes; los interesados en participar podrán confirmar su asistencia.

Nuestro objetivo es la implementación de un sistema de gestión de reuniones en la nube para un almacenamiento y procesamiento de datos eficientes y así lograr una aplicación de multiplataforma y en tiempo real. Por ello nuestra propuesta de implementación se realizará sobre FireBase [6], una plataforma de almacenamiento de datos en tiempo real basada en la nube, la cual posee un árbol de servicios que admiten accesos basados en notificaciones para garantizar la entrega de los servicios a los usuarios en tiempo real.

El presente artículo estará conformado por 5 Secciones. En la Sección 2 se detalló los trabajos relacionados. En la Sección 3 se especifican las herramientas que se usó para la implementación del sistema y daremos hincapié a la explicación de la implementación del sistema. En la Sección 4 se pondrán los resultados sobre la ventaja y desventaja de usar FireBase, además de que se logró con la implementación del Sistema de Gestión de Reuniones y Eventos. En la Sección 5 daremos nuestras conclusiones finales sobre el sistema y sobre trabajos futuros

II. ESTADO DEL ARTE

El problema de la programación de Reuniones define dónde y cuándo una reunión debe realizarse. El objetivo es encontrar la combinación adecuada para satisfacer las necesidades de todos los asistentes para programar una reunión que concuerde con todos, considerando disponibilidad, preferencias y privacidad.

Se han propuesto diferentes enfoques para dar solución a este problema como en el artículo [1] el cual propone un enfoque basado en Inteligencia de Enjambre adaptándola al problema de gestión de reuniones, con la cual pretenden maximizar las preferencias de los usuarios y la importancia de las reuniones. En su etapa experimental se enfocó en probar su escalabilidad, evaluando comparativas en programar reuniones semanales en una empresa, con lo que supera los problemas más restringidos poniendo a prueba la concurrencia.

En el artículo [2] toma el enfoque de un sistema de programación de reuniones basados en votos, utilizando sistemas distribuidos, donde el voto de los principales participantes tiene mayor prioridad al escoger el día y hora de la reunión, y así con Inteligencia Artificial programar satisfactoriamente las reuniones y así obtener la fecha de la misma.

En el artículo “Sistema de Horarios de Reuniones” [4] se refiere a un método, también un medio programado y un sistema en el cual un usuario tiene una opción para dar prioridad a las reuniones y a las personas que tienen acceso al calendario del usuario.

El sistema determina automáticamente el nivel de prioridad del usuario y muestra el calendario de otros usuarios invitados, mostrando sólo otras reuniones programadas que tienen un nivel de prioridad más alto, permitiendo así la programación automática del nivel de prioridad para todos los usuarios del sistema (más específicamente el módulo de calendario).

En el artículo [7] explican un plan de proyecto de un sistema de programación de reuniones, donde intentan dar solución al problema con paralelismo, donde los desarrolladores quieren lograr la programación de varias reuniones a la vez y que el sistema responda dando la mejor solución o la mejor propuesta para el creador del evento, y pueda invitar a los participantes que desea que participen sobre las reuniones y que los horarios sean los mejores para todos.

El artículo [10] toma otro enfoque, donde priorizan la automatización de un sistema de reuniones empresariales, pero usando agentes, donde el creador del evento solo debe de crear la reunión y designar los invitados importantes, entonces la agente ira con los agentes de los participantes de la empresa y podrán negociar para obtener el mejor horario donde todos puedan participar, usando negociaciones heurísticas para que los agentes no rompan ninguna regla establecida.

En el artículo "Información visual compartida del tráfico vehicular en tiempo real a través de la nube" [3] se utiliza la plataforma FireBase para dar solución al problema de congestión vehicular.

El aporte de dicho artículo es aliviar la congestión del tráfico a través del uso de sistemas inteligentes utilizando multitud de datos de tráfico, recopilando la velocidad de

tráfico y con ayuda de la tecnología FireBase identificar y/o analizar sus condiciones.

Cuando se proporciona a los sistemas de navegación, esta información, se genera y presenta una lista de rutas recomendadas para la planificación de viajes.

III. MATERIALES Y MÉTODOS

1. Plataformas

a. Heroku

Heroku es una plataforma de computación en la Nube que soporta distintos lenguajes de programación como Java, Node.js, Scala, Clojure y Python y PHP. La base del sistema operativo es Debian o en la nueva plataforma, el sistema basado en Debian Ubuntu, sus principales características son [11]:

- Elasticidad y crecimiento. La cantidad de Dynos asignados a una aplicación se puede cambiar en cualquier momento a través de la línea de comandos o el dashboard.
- Tamaño. Heroku ofrece diferentes tipos de Dynos, cada uno con diferentes capacidades de procesamiento y memoria.
- Routing. Internamente los routers realizan un seguimiento de la ubicación de los Dynos que estén corriendo, y redirigen el tráfico de acuerdo a la misma.
- Seguimiento. Existe un manejador de Dynos, el cual monitorea de forma continua los Dynos que se estén ejecutando. En caso de una falla en un Dyno, este es eliminado y creado nuevamente.
- Distribución y redundancia. Los Dynos se encuentran aislados uno del otro. Esto implica que, de existir fallos en la infraestructura interna de alguno de ellos, los otros Dynos no se ven afectados, y consecuentemente tampoco la aplicación.

b. Microsoft Azure

Microsoft Azure es una creciente colección de servicios en la nube integrados, que los desarrolladores y los profesionales de TI utilizan para crear, implementar y administrar aplicaciones a través de su red global de centros de datos [12].

Con Azure, obtiene la libertad de crear e implementar donde quiera, utilizando las herramientas, las aplicaciones y los marcos que prefiera, sus características son:

- Proceso: el servicio de proceso de Windows Azure ejecuta aplicaciones basadas en Windows Server. Estas aplicaciones se pueden crear mediante .NET Framework en lenguajes como C# y Visual Basic, o implementar sin .NET en C++, Java y otros lenguajes.
- Almacenamiento: objetos binarios grandes (blobs) proporcionan colas para la comunicación entre los componentes de las aplicaciones de Windows Azure y ofrece un tipo de tablas con un lenguaje de consulta simple.
- Servicios de infraestructura: posibilidad de desplegar de una forma sencilla máquinas virtuales con Windows Server o con distribuciones de Linux.
- Controlador de tejido: Windows Azure se ejecuta en un gran número de máquinas. El trabajo del controlador de tejido es combinar las máquinas en un solo centro de datos de Windows Azure formando un conjunto armónico. Los servicios de proceso y almacenamiento de Windows Azure se implementan encima de toda esta eficacia de procesamiento.
- Red de entrega de contenido: el almacenamiento en caché de los datos a los que se accede frecuentemente cerca de sus usuarios agiliza el acceso a esos datos.
- Connect: organizaciones interactúan con aplicaciones en la nube como si estuvieran dentro del propio firewall de la organización.
- Administración de identidad y acceso: La solución Active Directory permite gestionar de forma centralizada y sencilla el control de acceso y la identidad. Esta solución es perfecta para la administración de cuentas y la sincronización con directorios locales.

c. FireBase:

La tendencia de "datos abiertos", junto con el reciente avance en tecnologías de desarrollo web y la proliferación de marcos de JavaScript ha ayudado a popularizar la programación de aplicaciones web interactivas. Sin embargo, algunas de las características comunes de las aplicaciones web actuales que acceden a datos de los propios almacenes de datos o de los servicios web requieren una cantidad significativa de conocimientos de programación y, por lo tanto, dificultan a los desarrolladores a prototipar rápidamente aplicaciones e iterar soluciones.

FireBase es un servicio capaz de proveernos de un Backend en la nube con una fuente de datos en tiempo real y librerías para acceder a ella desde aplicaciones web, iOS y Android.

FireBase es un claro ejemplo de las posibilidades de desarrollo en la nube. A partir de un servicio web nos

ofrecen la posibilidad de programar aplicaciones con datos que se sincronizan en tiempo real a través de múltiples dispositivos, evitando muchas de las tareas engorrosas en las que tendríamos que dedicar tiempo al programar.

Los datos de la base de datos FireBase se almacenan como JSON y se sincronizan en tiempo real con cada cliente conectado. De esta manera, FireBase permite un desacoplamiento de alto nivel entre los productores de datos y los consumidores de datos, mientras que la estructura para describir los datos juega un papel clave en proporcionar una percepción significativa de los elementos de datos individuales almacenados en FireBase.

Acceso a los datos por referencia son enlaces con tu modelo de datos que apuntan a un nodo en concreto del JSON. A través de esas referencias podemos acceder a un documento completo o navegar por los hijos para acceder a otros nodos dependientes. Estas referencias se crean a través de una llamada a la función Firebase(), indicando como parámetro el nodo de nuestro modelo que queremos referenciar. Los nodos están compuestos por una ruta definida con el nombre de nuestra aplicación (http:// bumeet-roycercordova.rhcloud.com), con lo que podemos acceder a un nodo más interior, como se aprecia en la figura 1.

```
var ref = new Firebase("http:// bumeet-
roycercordova.rhcloud.com/");
```

Fig. 1 Referencia al proyecto

Para dirigirse a uno de sus hijos usamos el método child(), como se aprecia en la figura 2.

```
ref.child("appName")
```

Fig. 2 Hijos en FireBase

La sincronización a datos en FireBase se realiza con la suscripción a un evento, cuando accedes a un dato puede tener un estado determinado, pero sin embargo al ser una aplicación en tiempo real, ese estado puede cambiar en cualquier instante. Para lo cual se disparará un evento cada vez que se presente un nuevo valor alojado en el dato.

Nos suscribimos a eventos con el método on(). El evento al que nos vamos a suscribir es "value", como se observa en la figura 3.

```
ref.child("appName").on("value",
function(snapshot){
console.log(snapshot.val());
  document.getElementById("titular").textContent =
snapshot.val();
});
```

Fig. 3 Suscripción a eventos con FireBase

En FireBase las conexiones se realizan mediante sockets, por eso es tan rápido en recibir los cambios que hayan realizado en el modelo. Cuando un cliente abre una página donde se usa FireBase se mantiene abierto un canal de comunicación con el servidor bidireccional, que permite comunicarse desde el cliente al servidor y desde el servidor a todos los clientes [6].

2. AngularFire

"AngularJS no es solo una biblioteca, más bien es un framework JavaScript que ayuda a extender el HTML a un formato más expresivo y legible. Le permite decorar su HTML con un marcado especial que se sincroniza con su JavaScript apoyando a escribir su propia lógica de aplicación en lugar de actualizar manualmente las vistas. Ya sea que esté buscando aumentar las aplicaciones JavaScript existentes o aprovechar toda la potencia del marco para crear ventanas más interactivas, Angular puede ayudarle a escribir un código más limpio y eficiente." [5]

AngularFire es la biblioteca de AngularJS para FireBase. Esta unión permite asociar FireBase y referencias con modelos angular para que puedan ser transparente e inmediatamente se mantengan sincronizados con la base de datos y con todos los demás clientes que actualmente utilizan la aplicación.

El enfoque de esta biblioteca es mucho más abstracto de la plantilla que participan en la creación de enlaces de angular a FireBase, y para que sea fácil crear servicios que se sincronizan con la base de datos.

Sin embargo, esta biblioteca no intenta reemplazar las funciones de la API de toda la biblioteca del cliente FireBase y que todavía puede ser adecuado para algunos usos avanzados. También se puede utilizar los métodos proporcionados por la biblioteca cliente Firebase junto con la unión AngularFire

*$firebaseObjec*t

El $firebaseObject realiza una referencia a FireBase o FireBase Query y devuelve un objeto JavaScript que contiene los datos proporcionados por FireBase y algunos campos extras de AngularFire.

Puede utilizar la función $loaded() para ser notificado cuando los datos se ha cargado. Este servicio mantiene

automáticamente los objetos locales en sincronía con los cambios realizados en la base de datos remota FireBase.

3. Aplicación

Debido a la necesidad de trabajar con varios usuarios los cuales tienen que estar en sincronía con la base de datos con respecto a las reuniones que se les ha sido asignado, es necesario uso de correctas herramientas que nos permitan resolver la concurrencia, para ello nos apoyaremos de "AngularFire" el cual nos proporciona un conjunto de métodos que detallaremos en los siguientes puntos.

3.1. Arquitectura

Como se muestra en Fig. 4, la arquitectura del sistema está basado en una arquitectura Modelo Vista Presentador (MVP), el motivo de este tipo de arquitectura se debe a que no se implementa un servidor dedicado a manejar el modelo de negocio de la aplicación, ya que este proceso se realiza en el presentador (en el cliente).

Esta arquitectura nos permite que el sistema pueda ser accedido desde los smartphones como también computadoras, donde se podrá ingresar a la página web desde cualquier navegador, posteriormente este se conectara con los servidores para mostrar la web service donde pedirá la autenticación para entrar a la aplicación, esta misma se encontrará alojada en la nube, donde se tiene el acceso a la base de datos.

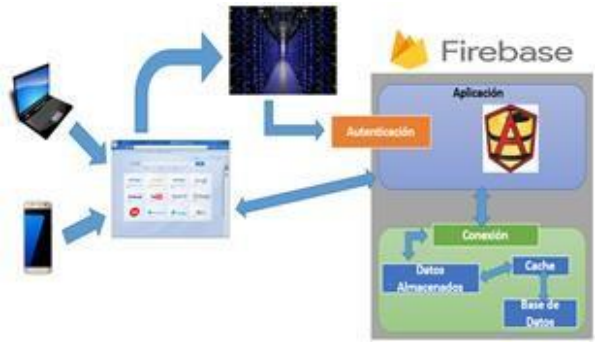

Fig. 4 Arquitectura de BuMeet

3.2 Autenticación

Para verificar la identidad del usuario para el acceso al sistema, se hizo uso de proveedores de identidad (Google, Facebook, Twitter y GitHub), también se usó cuentas basadas en contraseñas, pero debido a la necesidad de autenticar dichas cuentas decidimos obviar este método y solo considerar el uso de los proveedores externos, acelerando de esta manera el ingreso al sistema.

Como se muestra en la Fig. 5, la secuencia inicia cuando el usuario ingresa al Login del sistema y escoge un proveedor de identidad; el Login del sistema solicita un Token de autenticación, para ello el usuario ingresa con su cuenta del proveedor seleccionado, y da su consentimiento de compartir su información de dicho proveedor con nuestro sistema; El proveedor de identidad nos responde con un token de autenticación el cual lo validamos constantemente hasta que el usuario se retire del sistema.

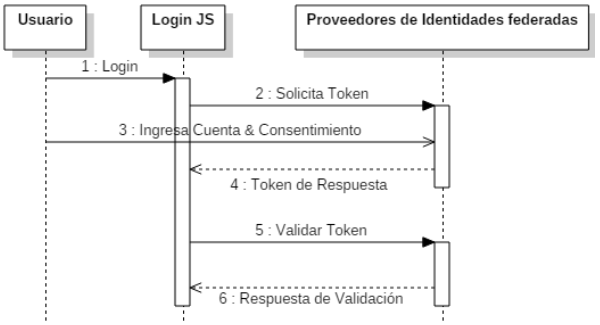

Fig. 5 Diagrama de Secuencia de Autenticación

FireBase nos proporciona mediante el método "singInWithPopup()" iniciar sesión con una ventana emergente, el cual recibe como argumento el proveedor seleccionado por el cliente, esta implementación se puede apreciar en la Fig. 6:

```
112     firebase.auth().signInWithPopup(provider)
113     .then(function(result) {...})
116     .catch(function(error) {...});
122
```

Fig. 6 Código de solicitud de token usando FireBase

3.3 Sesiones y Rutas

Como se muestra en la Fig. 7, este proceso inicia cuando el usuario ingresa a una opción del sistema, ya sea por url o por hipervínculo, para ello router provider analiza esta petición y redirecciona a una ruta definida cargando su vista correspondiente, pero antes de ello verifica si el usuario sigue en sesión, en caso su sesión haya finalizado se redirecciona al Login.

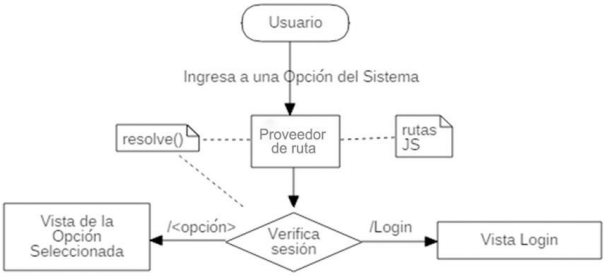

Fig. 7 Diagrama de Flujo de Sesiones y Rutas

Para verificar que el usuario se ha autenticado se aprovechó el método "resolve()" dentro de los router angulares y dos métodos auxiliares de AngularFire:

"$waitForSignIn()" y "$requireSignIn()", los cuales pueden ser apreciados en la Fig. 8.

```
app.config(["$routeProvider", function($routeProvider) {
  $routeProvider
  .when("/", {
    controller: "CtrlReuniones",
    templateUrl: "reuniones.html",
    resolve: {
      "currentAuth": ["Auth", function(Auth) {
        return Auth.$requireSignIn();
      }]
    }
  })
  .when("/reuniones", { ... })
  .when("/calendario", { ... })
  .when("/login", { ... })
  .when("/register", { ... })
  .when("/salir", { ... })
  .otherwise({
    redirectTo: '/'
  });
}]);
```

Fig. 8 Código de redirección de rutas y verificación de sesiones usando router provider

3.4 Notificaciones

Para notificar a los usuarios que ha sido asignado a una reunión, se añadió FireBase Cloud Messaging (FCM) el cual nos asegura que los mensajes o notificaciones lleguen a los usuarios correspondientes.

FireBase para ello nos proporciona el método "messaging()", el cual solicita permiso al navegador del usuario para enviar las notificaciones correspondientes, esta implementación se puede apreciar en la Fig. 9:

```
const mensaje = firebase.messaging();
mensaje.requestPermission()
.then(function(){ ... })
.catch(function(error){ ... });
```

Fig. 9 Código para envío de Notificaciones al usuario

3.5 Reuniones

Para mostrar las reuniones del usuario hacemos uso de objetos de tipo JSON los cuales se sincronizan con la base de datos, para ello hacemos una referencia a una base de especifica utilizando el método "database()" de FireBase, como se aprecia en la Fig. 10:

```
const refbd = firebase.database().ref();
const dbrefReu = refbd.child('reuniones').child(currentAuth);
$scope.reuniones=$firebaseArray(dbrefReu);
```

Fig. 10 Código de Reuniones

IV. RESULTADOS

En esta sección discutiremos los resultados de nuestro prototipo que implementamos para el sistema propuesto usando en diferentes dispositivos (desktop, móviles, Ipad). El prototipo fue implementado usando el servicio de FireBase que tiene características destacables para tratar la concurrencia y sincronización de un proyecto web de manera destacable; el framework Angular JS que es una herramienta que permite declarar vistas dinámicas, adaptable al flujo de trabajo y a las necesidades de las características necesarias.

Cada participante requiere de un usuario y contraseña para acceder a la aplicación. Ya que, al hacerlo, permitirá que su información se guardará en privado en la base de datos de la nube. Tal autentificación tiene como objetivo aumentar la seguridad del sistema y evitar la manipulación de datos.

Una vez que el usuario ha iniciado sesión, puede acceder al sistema donde puede crear una reunión y asignar a los participantes. La aplicación registrará los datos que recibe el sistema y los guardará bajo las credenciales del usuario en una base de datos NOSQL que se encuentra en la nube y es operada por FireBase, los datos se guardan en formato JSON lo cual permite una sincronización en tiempo real gracias a los crawlers que maneja FireBase. Para la creación de eventos o reuniones, creadas por los usuarios se llenará un formulario con el asunto del evento, el día, la hora de inicio y fin, una descripción breve y los usuarios a participar como se visualiza en la figura 11.

Figura 11: Creación de Reuniones en BuMeet

Una vez creada las reuniones, los usuarios serán notificados de dichas reuniones, observando el tema principal, la fecha, una descripción breve y los participantes invitados, como se aprecia en la figura 12.

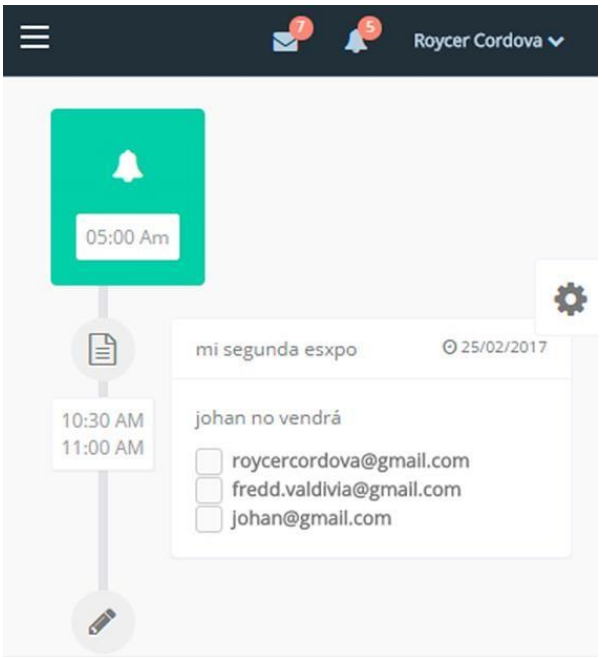

Figura 12: Mostrar todas las reuniones en BuMeet.

Además, en los resultados se comprobó la potencialidad de FireBase frente a los desafíos de concurrencia y escalabilidad ya que hace uso de sincronización de datos en tiempo real y la sincronización de datos sin conexión. si un usuario modifica un dato determinado, este mismo dato será actualizado para el resto de usuarios al que se esté mostrando automáticamente y en tiempo real. usuarios podrán modificar o almacenar datos, aunque por algún motivo se hayan quedado sin conexión. FireBase se encargará de almacenarlos en memoria local y lo sincronizará con la base de datos en cuanto la aplicación consiga conexión.

V. CONCLUSIONES

Con la realización de la aplicación BuMeet se puede conectar a gente con un mismo interés protegiendo su información individual, lo cual resuelve el problema inicial de una programación de reuniones.

La seguridad que proporciona BuMeet se basa en la utilización del framework AngularFire el cual no permite hacer enlaces de angular con FireBase, framework de alto nivel, así desarrollando aplicaciones con más facilidad.

Los framework de alto nivel resuelven la solicitud de varios usuarios a un mismo servicio, la concurrencia mediante la sincronización de las respectivas solicitudes; haciendo la actualización de los datos en tiempo real sin la necesidad de refrescar el servicio. Es así que comprobamos el rendimiento del framework. BuMeet es un software que responde ante los desafíos planteados en este artículo de concurrencia y sincronización.

Se podría realizar pruebas de escalabilidad para la herramienta BuMeet como aumentar módulos de calificación de las reuniones, hacer conexión con google calendar. Se podría también probar añadir videoconferencia al software BuMeet.

VI. REFERENCIAS

[1] H. Salema, A. B. Hassine, "Meeting Scheduling based on Swarm Intelligence" Knowledge-Based and Intelligent Information & Engineering Systems 19th Annual Conference, Volume 60, 2015, Pages 1081-1091.

[2] V. Antila, J. Mfultyjiirvi, V. Kononen, "Optimizing Meeting Scheduling in Collaborative Mobile Systems through Distributed Voting", Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on, 2010.

[3] Badgujar, M., Shaikh, S., Lahitkar, M., & Wadje, M., "Sharing Real-Time Visual Traffic Information via Vehicular Clouds", International Journal for Research in Engineering Application & Management (IJREAM), Vol-02, Issue 08, Nov 2016.

[4]Martinez Vencia Addae, Fang Lu, Billerica, MA (US); Vandana Mallempati, Austin, TX (US). "Meeting Scheduling System"

[5] Nilesh Jain, Priyanka Mangal and Deepak Mehta; "AngularJS: A Modern MVC Framework in JavaScript" ; Volume 5, No. 12, December 2014 Journal of Global Research in Computer Science

[6] Firebase Cloud Platform, http://www.firebase.com/.

[7] A. Gupta, H. Rajagopalan, K. Grover, K. Kulak, N. Priyadarshini, P. Priya, S. Singh, S. Sridhar, "Web Based Meeting Sheduler System", CS 6361,SPRING 2010 Advanced Requeriments Engineering.

[8] Chun, A., H. Wai, and R. Y. M. Wong, 2003, Optimizing agent-based meeting scheduling through preference estimation: Engineering Applications of Artificial Intelligence, v. 16, p. 727-743.

[9] Ben Hassine, A., and T. B. Ho, 2007, An agent-based approach to solve dynamic meeting scheduling problems with preferences: Engineering Applications of Artificial Intelligence, v. 20, p. 857-873.

[10] S. Sen, "An automated distributed meeting scheduler", IEEE Computer Society, 06 Agosto, 2002.

[11] Documentación de Heroku, https://devcenter.heroku.com/.

[12] Documentación de Microsoft Azure, https://docs.microsoft.com/en-us/azure/.