**DOCUMENTO**

**EVENTHUB-EVENTS / EVENTHUB-ASSISTANTS**

URLs de los repositorios:

<https://github.com/abernalmar/EventHub.git>

<https://github.com/abernalmar/events-service.git>

<https://github.com/jorsilman/EventHub-Frontend.git>

<https://github.com/jorsilman/EventHub-Assitants.git>

**Una captura de pantalla de un celular con texto e imagen

Descripción generada automáticamente con confianza media**

**Miembros del grupo**: Ángela Bernal, Francisco Andrés Caro y Jorge Sillero

**Nivel de acabado**: Nivel Hasta ***7*** Puntos. Como microservicios avanzados, hemos llevado a cabo la integración con el microservicio de asistente, incorporando un servicio externo (api.random.org) que genera un código aleatorio para cada nuevo asistente creado. Además, hemos implementado un frontend unificado que se alinea con los demás microservicios de la aplicación. La documentación detallada de las APIs de ambos microservicios se encuentra disponible en Swagger. Asimismo, hemos establecido una API Gateway utilizando Docker para optimizar la gestión y el enrutamiento de las peticiones.

Contenido

[Descripción de la aplicación 3](#_Toc157007192)

[Descomposición en microservicios 3](#_Toc157007193)

[Customer Agreement 3](#_Toc157007194)

[Análisis de la capacidad 3](#_Toc157007195)

[TCO 3](#_Toc157007196)

[Descripción API REST 4](#_Toc157007197)

[EventHub-Assistants 4](#_Toc157007198)

[EventHub-Events 5](#_Toc157007199)

[Documento de requisitos 6](#_Toc157007200)

[EventHub-Assistants 6](#_Toc157007201)

[Listar asistentes de un evento 6](#_Toc157007202)

[Añadir un asistente a un evento 6](#_Toc157007203)

[Editar un asistente 9](#_Toc157007204)

[Eliminar un asistente 10](#_Toc157007205)

[EventHub-Events 11](#_Toc157007206)

[Listar eventos 11](#_Toc157007207)

[Crear un evento 14](#_Toc157007208)

[Editar un evento 15](#_Toc157007209)

[Eliminar un evento 17](#_Toc157007210)

[Análisis de esfuerzos 18](#_Toc157007211)

## Descripción de la aplicación

La aplicación consiste en una plataforma de gestión de eventos, en la cual los usuarios podrán crear eventos, añadir asistentes a dichos eventos y tener comunicaciones entre los distintos usuarios.

## Descomposición en microservicios

La aplicación se ha dividido en los siguientes cuatro microservicios:

* **Users:** este microservicio se encarga de todas las operaciones relacionadas con los usuarios, como pueden ser, el registro, el login, el listado, la visualización del perfil, etc.
* **Communications:** este microservicio se encarga de las distintas comunicaciones posibles de la aplicación, estas pueden ser, comunicaciones entre los distintos usuarios, o envío de correos con las invitaciones a los asistentes.
* **Events:** este microservicio se encarga de todo lo relacionado con los eventos, es decir, la creación de estos, la edición, el listado o la eliminación. Como se explicará más adelante en el documento, este microservicio hace uso del microservicio de Users y el de Assistants.
* **Assistants:** este microservicio se encarga de todo lo relacionado con los asistentes, es decir, de la creación, edición, listado y eliminación. Como se explicará más adelante en el documento, este microservicio hace uso del microservicio de Events, Communications y de un servicio externo, api.random.org.

En nuestro caso, al ser 3 componentes los que formamos el grupo, nos hemos encargado del desarrollo de los microservicios de Assistants y Events.

## Customer Agreement



## Análisis de la capacidad



## TCO



## Descripción API REST

### EventHub-Assistants

* GET:
  + ‘/’: Obtiene todos los asistentes de la base de datos.
    - Respuesta Exitosa (200 OK):
      * Cuerpo: Lista de objetos de asistentes.
    - Respuesta Fallida (500 Internal Server Error):
      * Cuerpo: Problema con la base de datos.
  + ‘/:id’: Obtiene un asistente dado el id.
    - Parámetros de Ruta: id (string) - ID del asistente.
    - Respuesta Exitosa (200 OK):
      * Cuerpo: Objeto del asistente.
    - Respuesta Fallida (404 Not Found):
      * Cuerpo: Asistente no encontrado.
    - Respuesta Fallida (500 Internal Server Error):
      * Cuerpo: Problema con la base de datos.
  + ‘/event/:eventId’: Obtiene los asistentes de un evento dado el eventId.
    - Parámetros de Ruta: eventId (string) - ID del evento.
    - Respuesta Exitosa (200 OK):
      * Cuerpo: Lista de objetos de asistentes.
    - Respuesta Fallida (500 Internal Server Error):
      * Cuerpo: Problema con la base de datos.
* POST:
  + ‘/’: Crea un nuevo asistente.
    - Autenticación: Bearer Token
    - Parámetros de Solicitud:
      * name (string): Nombre del asistente.
      * surname (string): Apellido del asistente.
      * email (string): Correo electrónico del asistente.
      * eventId (string): ID del evento al que se registra el asistente.
      * username (string): Nombre de usuario del asistente.
    - Respuesta Exitosa (201 Created):
      * Cuerpo: Ninguno.
    - Respuesta Fallida (400 Bad Request):
      * Cuerpo: Error de validación.
    - Respuesta Fallida (500 Internal Server Error):
      * Cuerpo: Problema con la base de datos o al enviar el correo electrónico.
* PUT:
  + ‘/:id’: Actualiza un asistente dado su id.
  + Autenticación: Bearer Token
  + Parámetros de Ruta: id (string) - ID del asistente.
  + Parámetros de Solicitud: Datos actualizados del asistente.
  + Respuesta Exitosa (204 No Content):
    - Cuerpo: Ninguno.
  + Respuesta Fallida (404 Not Found):
    - Cuerpo: Asistente no encontrado.
  + Respuesta Fallida (500 Internal Server Error):
    - Cuerpo: Problema con la base de datos.
* DELETE:
  + ‘/:id’: Elimina un asistente dado su id.
  + Autenticación: Bearer Token
  + Parámetros de Ruta: id (string) - ID del asistente.
  + Respuesta Exitosa (204 No Content):
    - Cuerpo: Ninguno.
  + Respuesta Fallida (404 Not Found):
    - Cuerpo: Asistente no encontrado.
  + Respuesta Fallida (500 Internal Server Error):
    - Cuerpo: Problema con la base de datos.

**API Documentada en Swagger:**

<https://app.swaggerhub.com/apis/JORGESILLEROUNI/Eventhub-Assistants/1.0.0>

### EventHub-Events

API de Eventos (EventHub-Events):

* **GET:**
  + **/**: Obtiene todos los eventos de la base de datos.
    - Respuesta Exitosa (200 OK):
      * Cuerpo: Lista de objetos de eventos.
    - Respuesta Fallida (500 Internal Server Error):
      * Cuerpo: Problema con la base de datos.
  + **/:name?**: Obtiene un evento dado el nombre si se especifica, de lo contrario, devuelve todos los eventos.
    - Parámetros de Ruta: name (string) - Nombre del evento (opcional).
    - Respuesta Exitosa (200 OK):
      * Cuerpo: Objeto del evento o lista de objetos de eventos.
  + **/id/:id**: Obtiene un evento dado su id.
    - Parámetros de Ruta: id (string) - ID del evento.
    - Respuesta Exitosa (200 OK):
      * Cuerpo: Objeto del evento.
    - Respuesta Fallida (404 Not Found):
      * Cuerpo: Evento no encontrado.
    - Respuesta Fallida (500 Internal Server Error):
      * Cuerpo: Problema con la base de datos.
* **POST:**
  + **/**: Crea un nuevo evento con los siguientes parámetros:
    - name
    - place
    - date
    - description
    - category
    - Autenticación: Bearer Token
    - Respuesta Exitosa (201 Created):
      * Cuerpo: Ninguno.
    - Respuesta Fallida (400 Bad Request):
      * Cuerpo: Error de validación.
    - Respuesta Fallida (500 Internal Server Error):
      * Cuerpo: Problema con la base de datos o al enviar el correo electrónico.
* **PUT:**
  + **/:name**: Actualiza un evento dado su nombre.
    - Autenticación: Bearer Token
    - Parámetros de Ruta: name (string) - Nombre del evento.
    - Respuesta Exitosa (200 OK):
      * Cuerpo: Objeto del evento actualizado.
    - Respuesta Fallida (404 Not Found):
      * Cuerpo: Evento no encontrado.
    - Respuesta Fallida (500 Internal Server Error):
      * Cuerpo: Problema con la base de datos.
* **DELETE:**
  + **/:name**: Elimina un evento dado su nombre.
    - Autenticación: Bearer Token
    - Parámetros de Ruta: name (string) - Nombre del evento.
    - Respuesta Exitosa (204 No Content):
      * Cuerpo: Ninguno.
    - Respuesta Fallida (404 Not Found):
      * Cuerpo: Evento no encontrado.
    - Respuesta Fallida (500 Internal Server Error):
      * Cuerpo: Problema con la base de datos.

**API Documentada en Swagger:**

<https://app.swaggerhub.com/apis/JORGESILLEROUNI/Eventhub-Events/1.0.0>

## Documento de requisitos

### EventHub-Assistants

### Listar asistentes de un evento

Como usuario del sistema, quiero poder listar los asistentes del evento para poder ver las personas que están invitadas.

1. En el backend se ha creado una función que permite, mediante llamadas a la API, obtener los asistentes dado el id de un evento:

Texto

Descripción generada automáticamente

1. Por otra parte, en el frontend, se ha creado un componente, ListAssistants, que, mediante la llamada a la API, obtiene todos los asistentes de un evento y los muestra:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

### Añadir un asistente a un evento

Como usuario del sistema, quiero poder añadir un asistente a un evento para poder enviarle una invitación.

1. En el backend se ha creado una función que permite realizar una petición POST a la API. El funcionamiento es el siguiente:
   1. Se ha creado una función getRandomCode(), que hace uso de un Servicio Externo, en concreto api.random.org. Esta función realiza una petición a la API y obtiene un código aleatorio.
   2. Dentro de la función que realiza el POST, primero se obtiene el código haciendo uso de la función descrita anteriormente.
   3. Luego, se obtiene el evento haciendo uso del microservicio de Evenhub-Events.
   4. Después, se envía un correo electrónico al asistente con la invitación, haciendo uso del microservicio de Eventhub-Communications.
   5. Y por último, se guarda el asistente en la base de datos.

Texto

Descripción generada automáticamente

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Texto

Descripción generada automáticamente

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

1. En el frontend, se ha creado un componente, CreateAssistant que muestra un formulario para añadir un asistente a un evento:

Texto

Descripción generada automáticamente  
Texto

Descripción generada automáticamente

### Editar un asistente

Como usuario del sistema, quiero poder editar un asistente para poder modificar algunos de sus datos.

1. En el backend, se ha creado la función para permitir el método PUT de la API:

Texto

Descripción generada automáticamente

1. En el frontend, se ha creado un componente, EditAssistant, para realizar la llamada a la API y mostrar un formulario para modificar los campos necesarios:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

### Eliminar un asistente

Como usuario del sistema, quiero poder eliminar un asistente, para que no esté en el evento.

1. En el backend, se ha creado la función para permitir el método DELETE:

Texto

Descripción generada automáticamente

1. En el frontend, dentro del componente AssistantItem, que se encarga de mostrar los datos del asistente dentro del listado, se ha desarrollado la funcionalidad de eliminar el asistente haciendo la llamada a la API.

Texto

Descripción generada automáticamente  
Texto

Descripción generada automáticamente

### EventHub-Events

### Listar eventos

Como usuario del sistema, quiero poder listar los eventos disponibles a los que poder acudir.

1. En el backend se ha creado una función que permite, mediante llamadas a la API, obtener los eventos disponibles:

Texto

Descripción generada automáticamente

También obtener un evento por id y por nombre:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1. Por otra parte, en el frontend, se ha creado un componente, **Events** que mediante la llamada a la API, obtiene todos eventos y los muestra:

Texto

Descripción generada automáticamente

También métodos para obtener el evento por nombre e id, para que otros microservicios puedan usarlos, como por ejemplo el microservicio de Asistentes que añade un asistente a un evento determinado por el id.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

### Crear un evento

Como usuario del sistema, quiero poder crear un evento con sus campos.

1. En el backend se ha creado una función que permite realizar una petición POST a la API:

Texto

Descripción generada automáticamente

1. En el frontend, se ha creado un componente, *CreateEvent* que muestra un formulario para añadir un evento:

Texto

Descripción generada automáticamente  
Texto

Descripción generada automáticamente

### Editar un evento

Como usuario del sistema, quiero poder editar un evento para poder modificar algunos de sus datos.

1. En el backend, se ha creado la función para permitir el método PUT de la API:

Texto

Descripción generada automáticamente

1. En el frontend, se han creado los componentes, *EditEvent* y *EditableEvent*, para realizar la llamada a la API y mostrar un formulario para modificar los campos necesarios:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

### Eliminar un evento

Como usuario del sistema, quiero poder eliminar un evento.

1. En el backend, se ha creado la función para permitir el método DELETE:

Texto

Descripción generada automáticamente

1. En el frontend, dentro del componente EventItem, que se encarga de mostrar los datos del evento dentro del listado, se ha desarrollado la funcionalidad de eliminar el evento haciendo la llamada a la API.

Texto

Descripción generada automáticamente

## Análisis de esfuerzos

El análisis de esfuerzos de cada miembro está más detallado en los documentos adjuntos, esto sería un resumen:

* Jorge Sillero Manchón: 48:45h
* Ángela Bernal Martín: 73:03h
* Francisco Andrés Caro Albarrán: 47:45h

  