IIC3103 - Taller de Integración
Departamento Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica

Enunciado Tarea 2

Objetivo

El objetivo de esta tarea es replicar un sistema con comunicación basada en eventos. El trabajo a realizar consistirá en enviar y recibir eventos en tiempo real utilizando *WebSockets* como capa de comunicación. Adicionalmente, se deberá construir una capa de visualización que permita interactuar con el sistema. Toda la tarea deberá ser implementada ocupando algún framework web. Algunos de estos frameworks son React, Vue, Gatsby, Angular, Svelte, o similar. También es posible utilizar HTML, CSS y Javascript puros (sin framework).

Trabajo a realizar

Deberán **conectarse a un *WebSocket*** que entregará eventos sobre posiciones de aviones las cuales se van actualizando en tiempo real.
Cada uno deberá construir un sitio web que permita desplegar esta información en un mapa, que muestre todas las rutas disponibles, y posición de los aviones la cual se debe ir actualizando en la medida que se van recibiendo eventos.
Adicionalmente, se debe implementar un sistema de chat, que permita enviar y recibir mensajes de texto, tanto del resto de los usuarios conectados como de los vuelos en curso.

En esta tarea no se debe guardar información en bases de datos, solo se debe mostrar la información que se empezó a consumir desde que se inició la conexión con el *WebSocket*.

Índice

Websocket

Para esta entrega se ocupará únicamente la información entregada por el *WebSocket*. Para conectarse a este se ocupan los siguientes datos:

Conexión con Websocket

- **Protocolo:** wss://

- **Servidor:** tarea-2.2024-2.tallerdeintegracion.cl[1]
- **Ruta:** /connect

Descripción detallada de eventos

Eventos emitidos por el Websocket

Estos serán los eventos que deberán **recibir** como cliente para el correcto funcionamiento de su aplicación.
Los eventos emitidos son:

- FLIGHTS: Entrega un listado de los vuelos y rutas activas.
- PLANE: Entrega una actualización sobre el estado de un avión, asociado a un vuelo.
- TAKE-OFF: Notifica cuando se produce un despegue.
- LANDING: Notifica cuando se produce un aterrizaje.

[1] si no está disponible, probar con https://tarea-2-2024-2-prod-902587603657.us-central1.run.app

- MESSAGE: Notifica un nuevo mensaje de texto.

La descripción técnica de cada evento y más información, se detalla a continuación.

FLIGHTS

Este evento, cuando se emite, entrega todos los vuelos activos, como un arreglo de objetos `FlightsEvents` según la definición que se muestra a continuación.

Frecuencia de envío

Este evento se lanza automáticamente con una frecuencia predeterminada.

Descripción técnica del evento

```
type Coordinates = {      // Coordenadas, representan un punto en el mapa
 lat: number,             // Latitud de las coordenadas
 long: number             // Longitud de las coordenadas
}

type Country = {
 id: string,        // Id del país
 name: string       // Nombre del país
}

type City = {
 id: string,        // Id de la ciudad
 name: string,      // Nombre de la ciudad
 country: Country   // País donde se encuentra la ciudad
}

type Airport = {
 id: string,             // Id del aeropuerto
 name: string,           // Nombre del aeropuerto
 city: City,             // Ciudad donde se encuentra el aeropuerto
 location: Coordinates   // Coordenadas geográficas del aeropuerto
}


type Flight = {
 id: string,           // Id del vuelo
 departure: Airport,   // Aeropuerto de salida
 destination: Airport, // Aeropuerto de destino
 departure_date: Date, // Fecha de salida
```

```
}

type FlightsEvents = {
 type: 'flights',         // Nombre de evento
 flights: {
   flight_id (string): Flight, // Diccionario con los vuelos activos,
                               // donde la llave corresponde al id
 }
}
```

Ejemplo

```
{
 "type": "flights",
 "flights": {
   "HAMMRCORY": {
     "id": "HAMMRCORY",
     "departure": {
       "id": "HAM",
       "name": "Hadha Almatar Muzayaf Airport (HAM)",
       "city": {
         "id": "C4",
         "name": "Fez",
         "country": {
           "id": "MR",
           "name": "Morocco"
         }
       },
       "location": {
         "lat": 38.851497,
         "long": -77.04796
       }
     },
     "destination": {
       "id": "ORY",
       "name": "Orly International Airport (ORY)",
       "city": {
         "id": "C5",
         "name": "Paris",
         "country": {
           "id": "FR",
           "name": "France"
         }
       },
       "location": {
         "lat": 38.851497,
```

```
            "long": -77.04796
          }
        },
        "departure_date": "2022-10-20 04:59:59.086628"
      }
    }
  }
```

PLANE

Evento que envía una actualización de posición de un avión asociado a un vuelo.

Frecuencia de envío

Este evento se lanza automáticamente con una frecuencia predeterminada.

Descripción técnica del evento

```
type Airline = {
 id: string,        // Id de la aerolínea
 name: string       // Nombre de la aerolínea
}

type Coordinates = {      // Coordenadas, representan un punto en el mapa
 lat: number,             // Latitud de las coordenadas
 long: number             // Longitud de las coordenadas
}

type Plane = {            // Avión
 flight_id: string,       // Vuelo actual de este avión
 airline: Airline,        // Aerolínea del vuelo
 captain: string          // Nombre del Capitán del vuelo
 position: Coordinates,   // Posición actual del avión
 heading: Coordinates,    // Dirección hacia donde se dirige
 ETA: number,             // Fecha de llegada estimada
 distance: number,        // Distancia estimada al destino
 arrival: Date,           // Fecha original de llegada
 status: 'take-off'|'flying'|'crashed'|'arrived'
 // Status del avión (enum)
}

type PlaneEvent = {       // Evento de avión
 type: 'plane',           // Nombre de evento
 plane: Plane,            // Objeto avión
}
```

Ejemplo

```
{
 "type": "plane",
 "plane": {
   "flight_id": "HAMMRCORY",
   "airline": {
     "id": "SKYY",
     "name": "Sky Airlines"
   },
   "captain": "Margaret Johnson",
   "position": {
     "lat": 38.851497,
     "long": -77.04796
   },
   "heading": {
     "lat": 38.851497,
     "long": -77.04796
   },
   "ETA": 1.1966245115986072,
   "distance": 1029.0970799748022,
   "arrival": "2022-10-20 06:11:46.934870",
   "status": "flying"
 }
}
```

TAKE-OFF

Evento que notifica el despegue de un avión asociado a un vuelo. En la notificación, se indica el vuelo id de vuelo que acaba de despegar.

Frecuencia de envío

Este evento se lanza automáticamente cada vez que se produce un despegue.

Descripción técnica del evento

```
type TakeoffEvent = {
 type: 'take-off'           // Nombre de evento
 flight_id: string       // Id del vuelo que acaba de despegar
}
```

Ejemplo

```
{
 "type": "take-off",
 "flight_id": "SCLSKYDCA"
}
```

LANDING

Evento que notifica el aterrizaje de un avión asociado a un vuelo. En la notificación, se indica el vuelo id de vuelo que acaba de aterrizar.

Frecuencia de envío

Este evento se lanza automáticamente cada vez que se produce un aterrizaje.

Descripción técnica del evento

```
type LandingEvent = {
 type: 'landing'          // Nombre de evento
 flight_id: string        // Id del vuelo que acaba de aterrizar
}
```

Ejemplo

```
{
 "type": "landing",
 "flight_id": "SCLSKYDCA"
}
```

CRASHED

Evento que notifica el accidente aéreo de un avión asociado a un vuelo. En la notificación, se indica el vuelo id de vuelo que acaba de sufrir un accidente aéreo.

Frecuencia de envío

Este evento se lanza automáticamente cada vez que se produce un accidente aéreo.

Descripción técnica del evento

```
type CrashedEvent = {
 type: 'crashed'          // Nombre de evento
 flight_id: string        // Id del vuelo que acaba de accidentarse
}
```

Ejemplo

```
{
 "type": "crashed",
 "flight_id": "SCLSKYDCA"
}
```

MESSAGE

Evento que notifica la recepción de un mensaje en el chat común. En este evento se recibirán:

- Mensajes enviados por los demás alumnos (cuando ellos envían un evento CHAT)
- Mensajes de los capitanes volando los aviones.

Frecuencia de envío

Este evento se lanza cada vez que se recibe un evento CHAT.

Descripción del evento

```
type Message = {
 flight_id: string,       // Id del vuelo que envía el mensaje
 level: 'info'|'warn',    // Tipo de mensaje (enum)
 name: string,            // Nombre de quién envía el mensaje
 date: Date,              // Fecha y hora del mensaje
 content: string          // Contenido del mensaje
}
```

```
type MessageEvent = {
 type: 'message',          // Nombre de evento
 message: Message          // Mensaje
}
```

Ejemplo

```
{
 "type": "message",
 "message": {
   "flight_id": "SCLSKYDCA",
   "level": "info",
   "name": "Pete Mitchell",
   "date": "2022-11-15 03:30:30.409315",
   "content": "No Points For Second Place"
 }
}
```

Eventos que recibe el Websocket

Estos serán los eventos que deberán **emitir** como cliente para el correcto funcionamiento de su aplicación.

JOIN

Este es el primer evento que debes enviar para comenzar a recibir los demás eventos. Debes enviar obligatoriamente tu número de alumno como identificador, y opcionalmente un nombre de usuario (Por defecto tomará tu nombre real).

Frecuencia de envío

Debes enviar este evento cuando se inicia una nueva sesión con el servidor.

Descripción técnica del evento

```
type JoinEvent = {
 type: 'join'         // Nombre de evento
```

```
 id: string,          // id de usuario
 username: string     // nombre de usuario (opcional)
}
```

Ejemplo

```
{
 "type": "join",
 "id": "096c3f23-d352-4962-bf02-f4bbcd56a8ee",
 "username": "Maverick",
}
```

CHAT

Permite enviar un mensaje de texto al resto de las personas conectadas al sistema. Cada vez que se envía un mensaje, este se propaga a todos los usuarios conectados como un evento MESSAGE.

Frecuencia de envío

Este evento se debe enviar cada vez que un usuario ingresa un nuevo mensaje en la funcionalidad de chat.

Descripción técnica del evento

```
type ChatEvent = {
 type: 'chat'            // Nombre de evento
 content: string         // Contenido del mensaje
}
```

Ejemplo

```
{
  "type": "chat",
  "content": "Some message"
}
```

Visualización

Elementos mínimos

Cada estudiante deberá implementar una visualización web que contenga, a lo menos, los elementos mostrados a continuación.

1. Mapa con los siguientes elementos:
    a. Aeropuertos y rutas
        i. Marcador con la ubicación de todos los aeropuertos que se encuentren en el evento FLIGHTS. Al hacer click en el marcador, debe mostrar toda la información relevante sobre el aeropuerto (id vuelo, Nombre, país, ciudad). Debe haber una distinción entre un aeropuerto de partida y uno de destino, por ejemplo, distintos íconos o colores. (si un aeropuerto está como partida y destino al mismo tiempo, mostrar cualquiera de los 2).
        ii. Líneas sobre el mapa que muestre todas las rutas activas que se listan en el evento FLIGHTS. La línea se debe dibujar como la ruta más corta entre el aeropuerto de origen y de destino (en línea recta).
    b. Aviones en tiempo real
        iii. Marcador con la posición en tiempo real de todos los aviones que se van actualizando mediante el evento PLANE. Al hacer click en un avión, debe mostrar toda la información relevante sobre el avión. (id vuelo, aerolínea, capitán, ETA, estado, entre otros)
        iv. Línea que muestre el desplazamiento realizado por cada avión, desde que el usuario se conecta al sitio.
    c. Otros elementos en mapa
        v. Realce visual con la ubicación de todos los eventos LANDING y TAKE-OFF.
        vi. Realce visual (con duración cercana a 1 minuto) en la última posición registrada de la posición de un vuelo en estado CRASHED, según se informa en evento CRASHED.
        vii. Las notificaciones o alertas deben mostrarse durante un tiempo acotado para no colapsar el mapa con demasiados elementos.

Todos Los elementos de cada vuelo deben desaparecer cuando deje de aparecer en el evento FLIGHTS.

2. Tabla informativa de vuelos:
    d. Tabla que muestre todos los vuelos, según información proveniente del evento FLIGHTS. La tabla debe estar ordenada por aeropuerto de origen y luego por aeropuerto de destino.

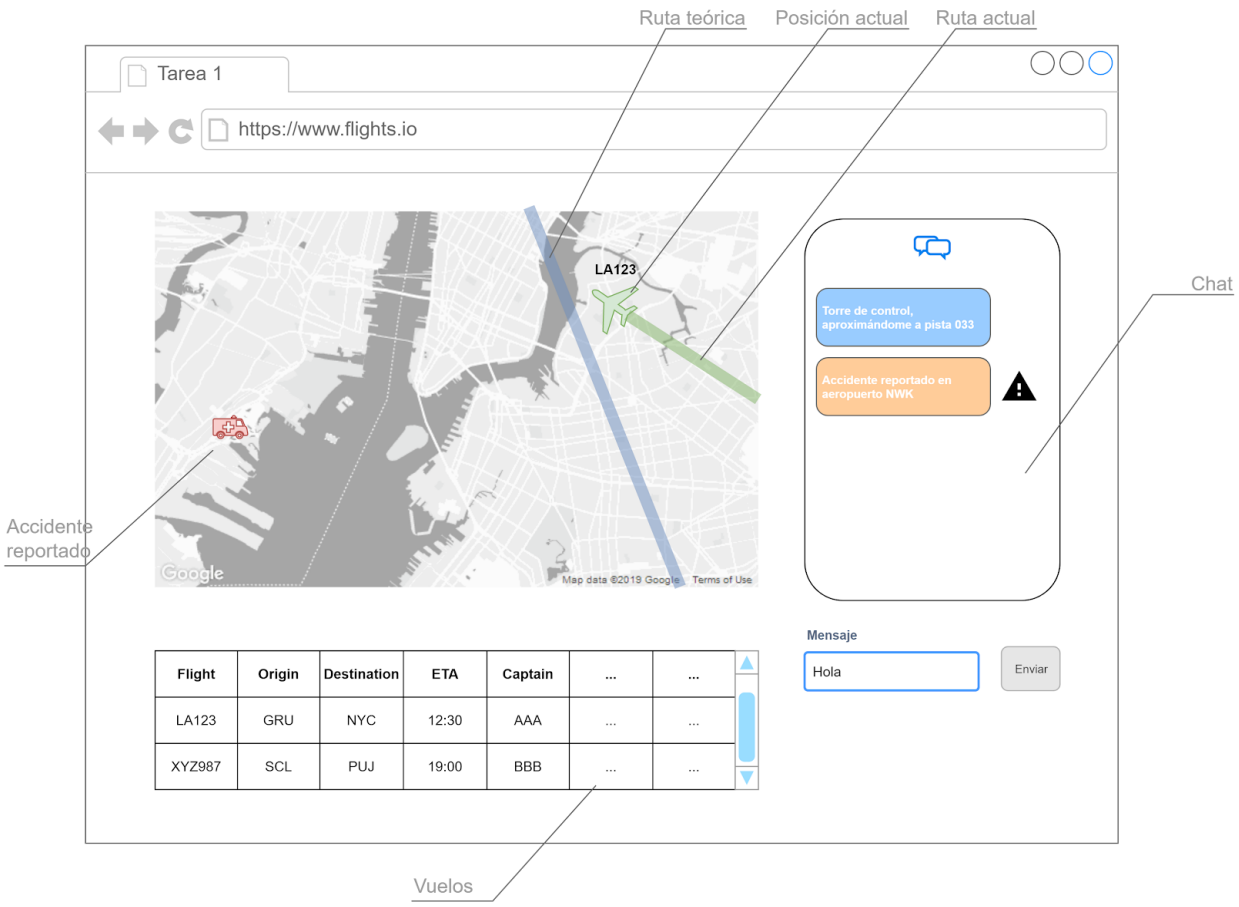
3. Chat:
    e. Panel que muestre todos los mensajes entrantes, así como los salientes, en una interfaz gráfica tipo chat, identificando fecha y hora de los mensajes, así como el emisor de estos.
    f. Cuadro de texto que permita escribir y enviar un nuevo mensaje.
    g. Bonus: Mostrar mensajes con level "warn" en color rojo.

Elementos gráficos

Se espera una interfaz gráfica con cierto grado de usabilidad. Si bien no se espera el uso de recursos de diseño gráfico y/o ilustración, si se espera que el sitio implementado tenga elementos básicos de usabilidad que permitan su uso y navegación.

Mockup de referencia

El mockup a continuación es una interfaz de referencia. Recuerde incluir todos los elementos de la sección Elementos mínimos.



Versionamiento de código

El sitio y todo su código fuente deberá estar versionado en un repositorio en GitHub, creado en el classroom del curso:

- Link para creación de repositorio: https://classroom.github.com/a/nSS4Oah1

Entregables

Para esta tarea, deberán entregar el link a su tarea desplegada, disponible públicamente en internet, donde se pueda entrar a probar las funcionalidades pedidas en la tarea.

Para esto, se recomienda usar la capa gratuita de servicios tipo Netlify, Vercel, Github Pages, entre otros.

Fecha de entrega

La tarea debe entregarse antes del 10 de octubre antes de las 18:00, y deberá estar desplegada y disponible durante el periodo de corrección (2 semanas).

Requisitos mínimos

Las tareas que no cumplan con las siguientes condiciones no serán corregidas y serán evaluados con la nota mínima:

- La página web deberá ser pública, accesible desde cualquier dispositivo conectado a internet.
- El código deberá estar versionado en su totalidad en un repositorio Git.
- El sitio implementado debe corresponder al código entregado. Para la revisión, más de una tarea serán corridas localmente por los ayudantes para comprobar el cumplimiento de este punto.
- En caso que el código entregado no represente fielmente al sitio entregado, se calificará la tarea con la nota mínima.

Penalizaciones

Se descontarán 0,2 puntos de la nota de la tarea por cada hora de atraso en la entrega, contados a partir de la fecha estipulada en el punto anterior.

En caso de no poder desplegar la API, se evaluará con nota máxima 5,0.

Cualquier intento de copia, plagio o acto deshonesto en el desarrollo de la tarea, será penalizado con nota 1,1 de acuerdo a la política de integridad académica del DCC.

Versiones del documento

- **Versión 1:**
  - 5/09/2024

Anexo 1 - Sobre el uso de Socket.io

Para esta tarea deberán investigar cómo conectarse a un *WebSocket* en el framework de su elección. Ahora, como equipo docente les recomendamos fuertemente no utilizar la librería Socket.Io, dado que como lo indica su documentación, **Socket.Io NO es una implementación de *WebSocket*.** Este paquete, si bien simplifica algunos aspectos de *WebSocket*, solo puede comunicarse con *WebSocket* hechos en Socket.Io, lo cual no es el caso para el servidor de la tarea.

Anexo 2 - Librerías recomendadas

Para el desarrollo de esta tarea, está permitido el uso de cualquier librería open source o de uso libre, siempre y cuando la licencia de la librería permitan los casos de uso propuestos en esta tarea.

Mapas

Para el gráfico de mapas, se recomienda utilizar Leaflet, librería de mapas interactivos open-source, basada en Open Street Map.

No se recomienda utilizar Google Maps, ya que esta requiere la creación de un token en un proyecto de Google Cloud con las funciones de facturación habilitadas.

Elementos gráficos

Para la generación de la interfaz gráfica, se recomienda (pero no obligatorio) el uso de alguna librería para elementos gráficos. Actualmente, existen varias librerías de este tipo. Algunas de estas se detallan a continuación:

Bootstrap

La librería de elementos gráficos más común. Contiene múltiples elementos gráficos pre-cargados, tales como botones, funciones de layout, figuras, banners, entre otros. Posee versiones para múltiples frameworks.

Ventajas: múltiples recursos gráficos, así como interacciones y otros elementos de código. Existen versiones para los frameworks Javascript más comunes.

Desventajas: Librería con muchos componentes de Javascript, lo que puede generar algunas incompatibilidades con algunos frameworks. Dependiendo del nivel de personalización que se incluya, puede requerir un proceso de compilación para publicar.

Foundation Framework

Una de las compensaciones de Bootstrap. Al igual que Bootstrap, contiene múltiples elementos gráficos pre-cargados, así como ciertas interacciones y elementos para Javascript.

Pure CSS

Librería compuesta solo por elementos CSS. No contiene ningún componente con Javascript.

Ventajas: Extremadamente pequeña en peso, por lo que agrega muy poco overhead. No contiene Javascript por lo que debería tener compatibilidad con una mayor cantidad de frameworks y librerías.

Desventajas: No posee ningún tipo de interacción o animación que requiera código Javascript.

Frameworks Javascript

Existen múltiples frameworks de Javascript que ayudan a construir interfaces responsivas y dinámicas.

Estos frameworks posibilitan la generación de aplicaciones complejas, mejoras en performance, server-side rendering, arquitecturas de microfrontends, manejo de máquinas de estados y una serie de otras funcionalidades, que probablemente no sean utilizadas en el alcance de esta tarea.

Algunos de estos frameworks son:

- React: Framework creado por Facebook, basado en componentes reutilizables. Tiene la versión Native, que permite ser compilada a lenguajes nativos de aplicaciones móviles.
- Vue: Framework desarrollado en forma independiente con foco en la velocidad y performance de los sitios web.
- Angular: Framework creado por Google para el desarrollo de aplicaciones web, con foco en PWA, velocidad y performance.

Por lo general, la mayor desventaja de estos frameworks son las curvas de aprendizaje, por lo que no se recomienda si no se tiene experiencia con alguno de ellos.