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

Enunciado **Tarea 2**

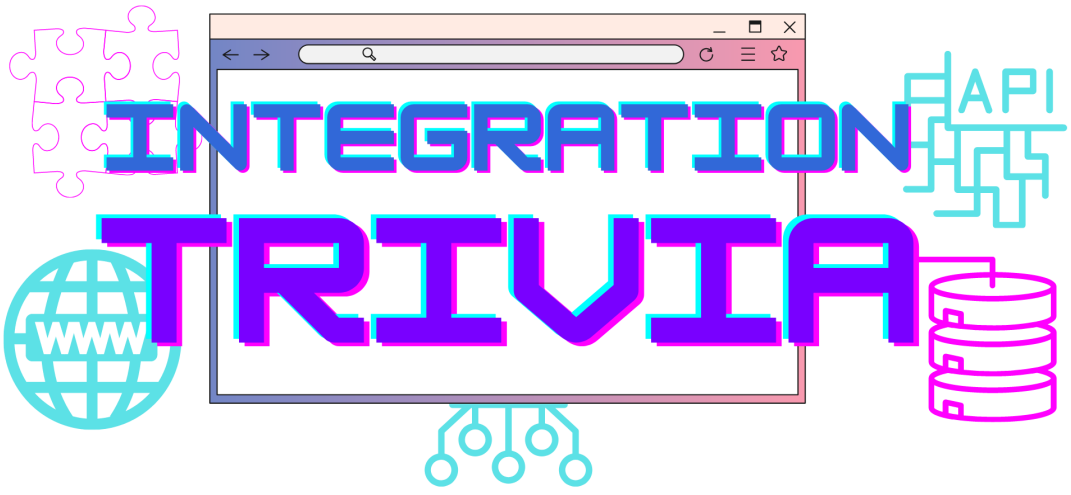
Objetivo

El objetivo de esta tarea es enviar y recibir eventos en tiempo real usando *WebSockets*[1]. Esta tarea debe ser resuelta ocupando algún framework de frontend tales como React, Vue, Gatsby, Angular, Svelte, o en su defecto, simple HTML, CSS y Javascript.

Trabajo a realizar

Deberán **conectarse a un *WebSocket*** que entregará eventos sobre una trivia en tiempo real. En su página web deberán mostrar el número de la trivia que se está jugando, el lobby de espera con los jugadores conectados y el tiempo restante para empezar a jugar, la pregunta actual de la trivia y su interfaz para responder, los puntajes de los distintos jugadores, el tiempo que le queda a la pregunta actual, y el jugador con mejor *streak*[2] hasta el momento. Finalmente, deberán desplegar un ranking de los ganadores del juego.

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*.



[1] Lectura recomendada: Documento oficial del protocolo WebSocket.
[2] Racha de preguntas correctas consecutivas.

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:** trivia.tallerdeintegracion.cl
- **Ruta:** /connect

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 que no usen 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.

Descripción detallada

Toda la lógica del juego descansa en el servidor implementado para la tarea, será tu misión interactuar con dicho servicio para desplegar en pantalla lo que se te pide y poder participar en la trivia.

Reglas

Es importante que se sigan las reglas del juego, de lo contrario, el servidor los desconectará de la partida. A continuación se describen las reglas más importantes.

¿Cómo unirse a la trivia?

A cada alumno se le asignará un **uuid** para esta tarea. Deben usar este identificador para conectarse al servidor. Opcionalmente[3], pueden indicar un **username** para definir su nombre dentro del juego. Con estos datos deben enviar un evento tipo **JOIN** al servidor. Los jugadores podrán conectarse en cualquier momento, una vez conectados existen dos escenarios:

- No hay una partida en curso. En dicho caso, los usuarios pasarán a un lobby o sala de espera.
- Hay una partida en curso. En este caso, los jugadores ingresarán en mitad del juego, partiendo con 0 puntos.

[3] Por defecto, su username será su nombre real.

Preguntas

Timer

Cada pregunta tendrá asociado un tiempo para ser respondida, que estará dado por el evento **TIMER**. Deberán mostrar en pantalla el tiempo restante de la pregunta actual. Una vez pasada la ventana de tiempo, se enviará la siguiente pregunta a ser respondida por el jugador y así sucesivamente hasta finalizar la trivia. Si se responde una pregunta fuera de su ventana de tiempo, serán expulsados de la partida.

Tipos de preguntas

Las preguntas serán enviadas a los clientes con el evento **QUESTION**. Estas pueden ser de 3 tipos, cuyos comportamientos son distintos y se detallan a continuación:

Button

En este tipo de preguntas, los jugadores deberán desplegar la pregunta junto a todas las opciones de respuesta entregadas por el servidor. El usuario debe poder seleccionar la alternativa que cree correcta enviarla apretando un botón. Todos los alumnos podrán obtener el puntaje en caso de responder correctamente la pregunta. Para este tipo de preguntas, solo se permite responder 1 vez[4]. Si se intenta responder más de una vez, el servidor los expulsará de la partida.

Text

Los alumnos deberán desplegar un cuadro de texto donde el usuario pueda escribir su respuesta. Todos los alumnos podrán obtener el puntaje en caso de responder correctamente la pregunta. Para este tipo de preguntas, solo se permite responder 1 vez. Si se intenta responder más de una vez, el servidor los expulsará de la partida.

Chat

Los alumnos deberán hacer display de un chat con el resto de jugadores. En dicho chat los alumnos deben poder ver las respuestas[5] del resto de los jugadores y al mismo tiempo enviar su respuesta a la pregunta. El puntaje será asignado al primer decil de jugadores que respondan correctamente. Además dicho puntaje no será distribuido uniformemente, sino que irá descendiendo de acuerdo al orden de respuesta. Para este tipo de preguntas, se permite responder más de una vez.

---

[4] Se recomienda que en las preguntas tipo *button* y *text* deshabiliten los inputs de la interfaz luego de responder para impedir enviar otra respuesta.

[5] Estas vendrán en los eventos tipo **CHAT**. El servidor no enviará las respuestas correctas.

Respuestas

Una vez respondida la pregunta, todos los alumnos deberán mostrar si su respuesta fue correcta o incorrecta, para eso tendrán que hacer uso del evento **RESULT**.

Puntajes

En paralelo a lo anterior, deberán mostrar los puntajes obtenidos por cada jugador durante toda la trivia. Dicha tabla debe estar ordenada de mayor a menor, donde el usuario ubicado en la primera fila es el usuario con mayor puntaje. Para el acceso de dichos puntajes deberán hacer uso del evento **SCORE**.

Streaks

Deberán mostrar al jugador que esté en la mejor racha, es decir, aquel que haya respondido más preguntas correctas seguidas. Es el evento **STREAK** el que detalla dicho jugador y su puntaje.

Highscore

Una vez finalizada la trivia actual, deberán mostrar los tres jugadores que alcanzaron el mejor puntaje. Para ello, haga uso del evento **HIGHSCORE**.

Eventos

Eventos emitidos por el Websocket

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

ACCEPTED

Envía la respuesta cuando un usuario es aceptado en una trivia

```
{
        "type": "accepted",
        "trivia_id": str,
        "joined": int,
}
```

Donde **trivia_id** hace referencia al identificador de la trivia que se está jugando y **joined** corresponde al número de jugadores que se han unido.

DENIED

Evento encargado de notificar que se rechazó la conexión del jugador y su razón.

```
{
        "type": "denied",
        "trivia_id": str,
        "reason": str,
}
```

Donde **trivia_id** es el identificador de la trivia en juego, y **reason** es la razón del rechazo de dicha conexión.

LOBBY

Envía información de la sala de espera

```
{
        "type": "lobby",
        "message": str,
        "seconds_remaining": int,
        "players": [str, ...]
}
```

Donde **message** corresponde a un mensaje de espera, **seconds_remaining** indica el tiempo restante para que inicie la partida y **players** es un array de los nombres de usuario de los jugadores que se encuentran en la sala de espera.

SCORE

Envía lista de participantes con sus respectivos puntajes.

```
{
        "type": "score",
        "scores":
        {
                "username_1" : int,
                "username_2" : int
        }
}
```

Donde **scores** corresponde a un objeto, donde las llaves son los participantes de la trivia, y los valores los puntajes de cada uno. Esta información debe ser mostrada en una tabla ordenada según el puntaje de los competidores. La tabla debe ir actualizándose cada vez que reciban este evento.

STREAK

Envía el jugador con mejor *streak* hasta el momento[6].

```
{
        "type": "streak",
        "username": str,
        "streak": int
}
```

Donde **username** corresponde al nombre de usuario del jugador con mejor racha y **streak** a la cantidad de respuestas correctas seguidas. Como se ha explicado, deberán mostrar el último jugador que haya tenido el mejor streak y la cantidad de preguntas que ha contestado correctamente al hilo.

QUESTION

Envía los datos de la pregunta a responder

```
{
        "type": "question",
        "question_id" : int,
        "question_type" : str,
        "question_title" : str,
        "question_points" : int,
        "question_options" : {
                int: str
        }
}
```

Donde **question_id** es el id de la pregunta de la trivia, **question_type** es el tipo de pregunta (chat, button, text), **question_title** la pregunta en sí, **question_points** los puntos de la pregunta y **question_options** son las opciones de respuesta (solo aplican a preguntas tipo “button”).
Al responder una pregunta tipo “button”, deben responder con la llave de la respuesta (1, 2, ...).

[6] El primero en llegar al streak máximo actual

TIMER

Indica cuánto tiempo le queda a la pregunta actual

```
{
        "type": "timer",
        "question_id": int,
        "seconds_remaining": int,
}
```

Donde **question_id** corresponde al id de la pregunta actual y **seconds_remaining** corresponde al tiempo que le queda a la pregunta actual.

RESULT

Envía el resultado de la respuesta del jugador. (Respuesta del servidor frente a evento **ANSWER**).

```
{
        "type": "result",
        "question_id": int,
        "correct": bool
}
```

Donde **question_id** es el id de la pregunta y **correct** tendrá valor **true** si la respuesta es correcta y **false** en caso contrario.

CHAT

Envía las respuestas de otros jugadores durante una pregunta tipo **chat**

```
{
        "type": "chat",
        "question_id": int,
        "message": str,
        "username": str
}
```

Donde **question_id** corresponde al id de la pregunta actual, **message** corresponde al mensaje que se muestra en el chat y **username** al usuario que envió el mensaje. Deberán mostrar en el chat los mensajes junto con el username y la hora en la que se envió el mensaje.

HIGHSCORE

Envía los puntajes de los 3 ganadores del primer, segundo y tercer lugar.

```
{
        "type": "highscore",
        "winners":
        [
                {
                        "username": str,
                        "score": int,
                        "streak": int,
                },
                ...
        ]
}
```

Donde **winners** es una lista que contiene los tres ganadores (en orden).

DISCONNECTED

Evento de desconexión

```
{
        "type": "disconnected",
        "trivia_id": str,
        "message": str,
}
```

Este evento notifica al cliente que se ha desconectado del servidor. Donde **trivia_id** es el id de la trivia que se está jugando y **message** es el mensaje de desconexión.

Eventos que recibe el Websocket

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

JOIN

Solicita al servidor unirse a la trivia.

```
{
        "type": "join",
        "id": str,
        "username": str
}
```

Donde **id** corresponde al identificador que te fue asignado[7] para esta tarea y **username** hace referencia al nombre de usuario elegido por el cliente que se está uniendo a la trivia. Luego de enviar este evento el servidor les responderá con un evento tipo **ACCEPTED** o **DENIED.**

ANSWER

Envía la respuesta del jugador que está participando en la trivia

```
{
        "type": "answer",
        "question_id": int,
        "value": str,
}
```

Donde **question_id** hace referencia al identificador de la pregunta que se está respondiendo, **value** corresponde al valor de la respuesta entregada por el cliente. Tras enviar este evento el servidor responderá con un evento tipo **RESULT.**

Visualización

A continuación se presentan ejemplos de cómo se podrían visualizar los diferentes componentes que debe contener su aplicación. Además, dan una idea del flujo al cual deben estar sujetas sus implementaciones.

[7] Puedes encontrar el **id** que te fue asignado en la siguiente planilla.

Figura 1: Ejemplo de *Landing page*

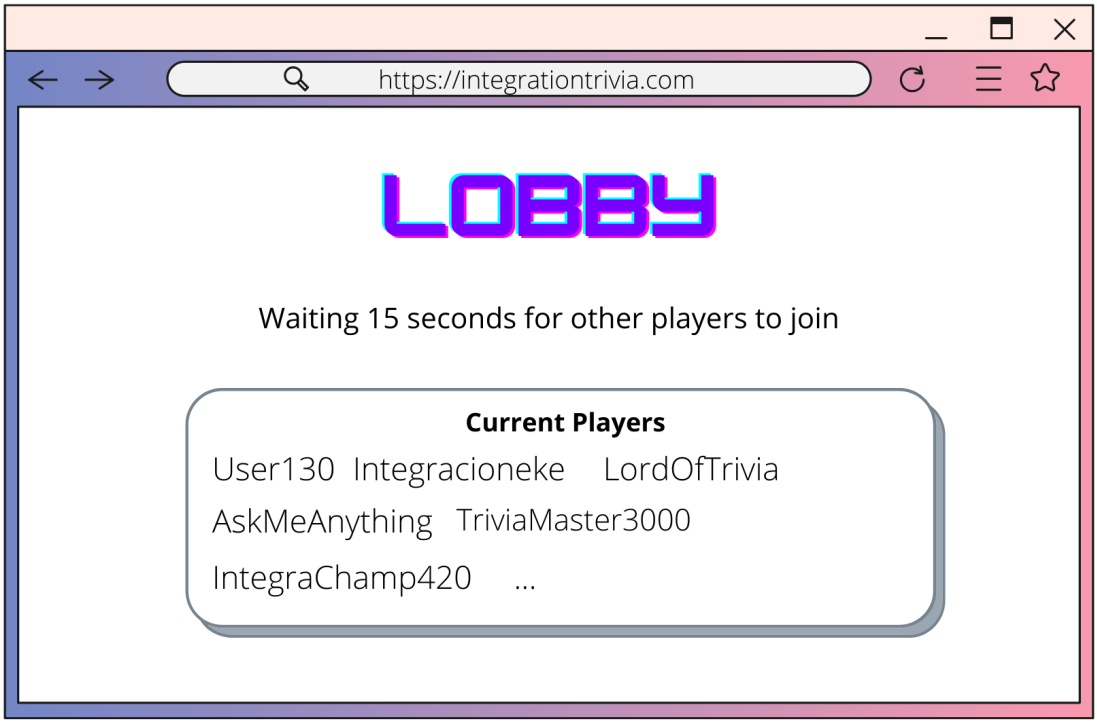


Figura 2: Ejemplo de *Lobby*

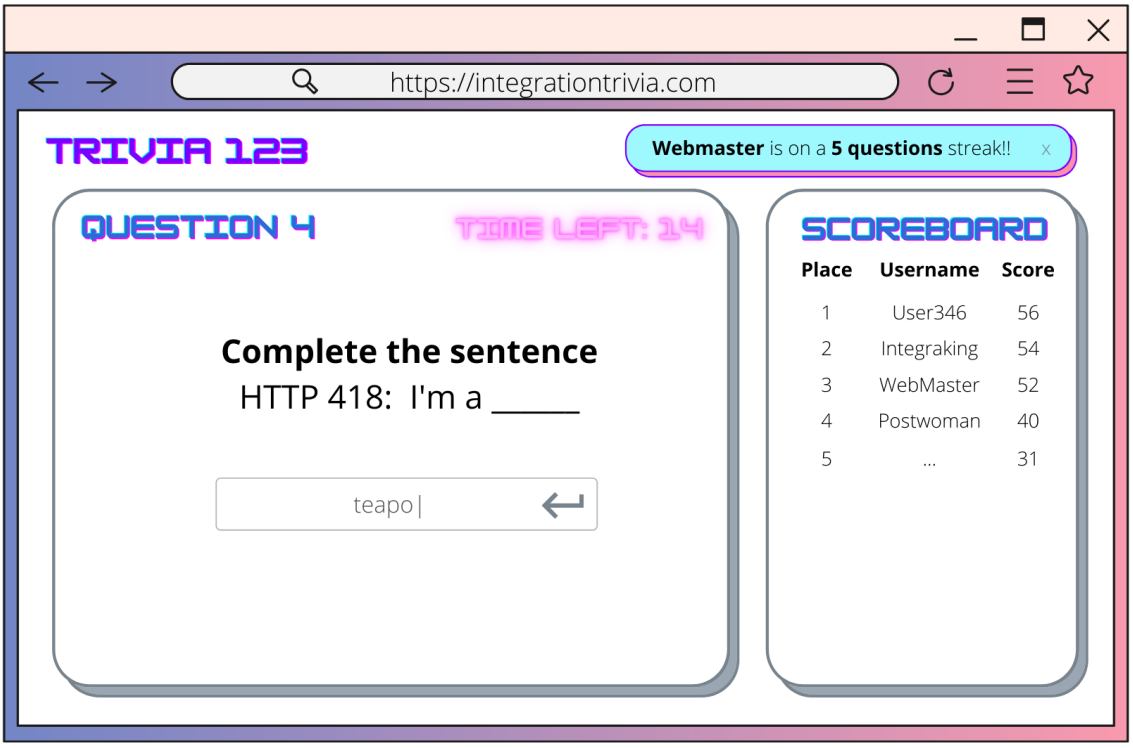


Figura 3: Ejemplo de pregunta tipo texto

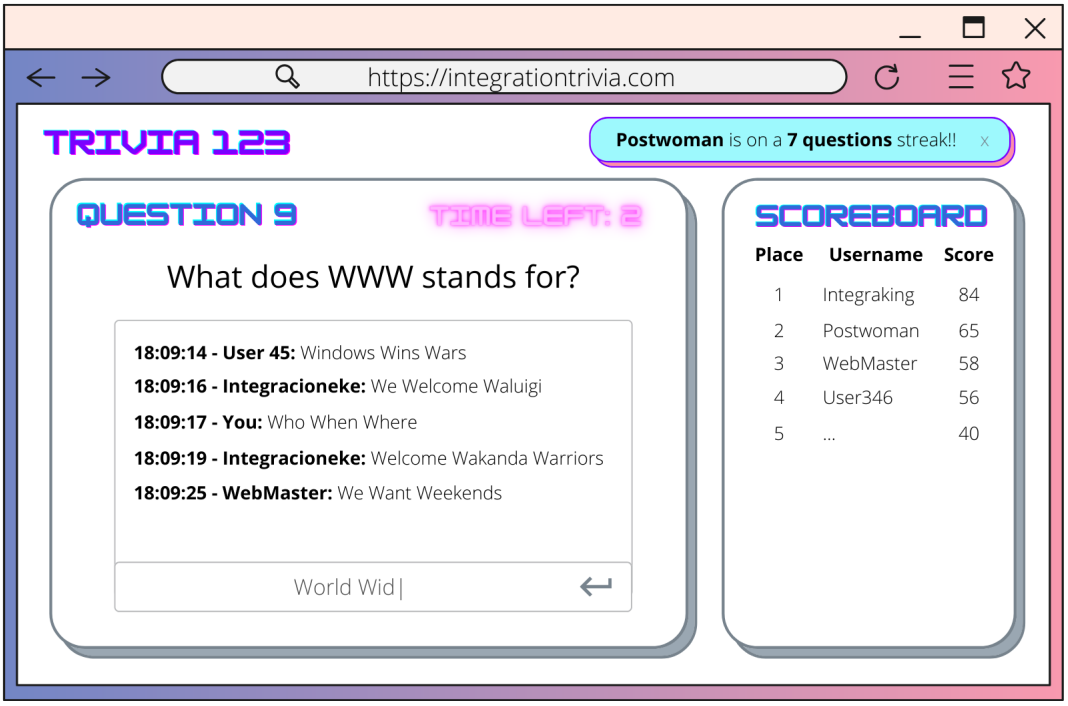


Figura 4: Ejemplo de pregunta tipo chat

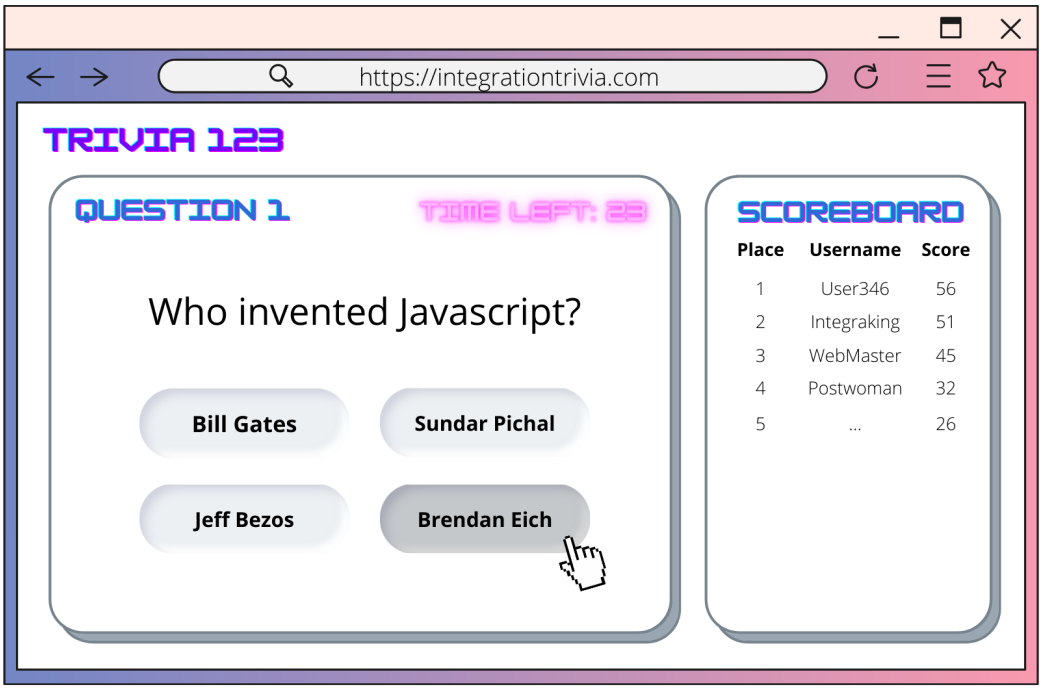


Figura 5: Ejemplo de pregunta button

Figura 6: Ejemplo de Final Scoreboard

Como se muestra en la figura 1, su aplicación debe solicitar un *username* a los diferentes clientes que quieren participar en la trivia. Una vez conectado, el jugador puede entrar al lobby (figura 2) o directamente a un juego en curso.

Dentro de un juego en curso se debe desplegar una vista (figuras 3, 4 y 5) que contenga todos los elementos solicitados, incluyendo una ventana que contenga un ranking de las posiciones de los participantes (basada en el *score* respectivo), otra que contenga la pregunta emitida por el servidor (puede apreciar que debe desplegar una interfaz de respuesta acorde al tipo de pregunta) y por último una ventana que muestre al jugador con la mejor racha y el largo de esta última. Además, su página debe informar sobre el número de la trivia que se está jugando, el número de la pregunta actual y el tiempo que queda para responder.

Finalmente, terminada la trivia actual, su aplicación debe mostrar a los ganadores del juego (🥇, 🥈 y 🥉) como se muestra en la figura 6..

Cabe mencionar que los ejemplos anteriores son solo una referencia gráfica y son libres de posicionar o diseñar los elementos a su gusto, mientras mantengan la misma funcionalidad.

Entregables

Cada alumno deberá entregar, mediante un formulario publicado en el sitio del curso, las siguientes *url's*:

- URL de acceso a sitio desplegado[8].
- URL del repositorio a GitHub

Entrega

Fecha de entrega

La tarea deberá ser entregada a más tardar el día viernes 22 de abril antes de las 18:00.

Evaluación

Además de la entrega de las *url's* mencionadas en el artículo Entregables y que debe concretarse en la fecha estipulada en el ítem anterior, en un horario que será acordado más adelante los alumnos deberán conectarse desde sus clientes para jugar la trivia con el resto de los estudiantes. Participar en esa evaluación online y la puntuación que logren obtener en dicha trivia, serán parte del puntaje para esta tarea, es decir, mientras más preguntas correctas respondan, podrán optar a una mayor nota.

[8] pueden usar servicios como Vercel, Netlify, Heroku, Firebase, entre otros.

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 debe reflejar fielmente el código entregado en el repositorio. Para la revisión, más de una tarea serán corridas localmente por los ayudantes para comprobar el cumplimiento de este punto.

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.

Se descontará 1 punto por cada revisión entre pares no contestada.

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.