Pontificia Universidad Católica de Chile
Escuela de ingeniería
Departamento de ciencias de la computación
Profesor: Hernán Cabrera

Tarea 5

IIC2513 Tecnologías y aplicaciones WEB

Segundo semestre 2020
Fecha límite de entrega de tarea: martes 17 de noviembre 21:30 hrs.

Fecha límite de evaluación de pares: viernes 20 de noviembre 23:59 hrs.
(https://forms.gle/tV2fXvtLRXXgzSHo7)

Las tareas tienen como objetivo que enfrenten un desafío de diseño y programación que les permita practicar los conocimientos adquiridos durante este semestre.

Las tareas, si bien siguen un orden pedagógico relacionado a los contenidos del curso, están vinculadas entre sí y la última de ellas tendrá como resultado una aplicación totalmente funcional con un lado cliente y otro servidor. Es decir, ustedes deberán cumplir con objetivos intermedios de la tarea de manera de llevar un diseño e implementación inicial a un "producto" totalmente funcional. Lo anterior revela la importancia de **no atrasarse ni saltarse una entrega** y procurar mantenerse al día.

Los enunciados dan la línea general de la funcionalidad que deben implementar, pero sin entrar en mayores detalles ni puntos específicos de tal manera que ustedes demuestren su capacidad de trabajo en equipo y resolución de problemas, y **discutan sus posibilidades y criterios de evaluación con su ayudante de seguimiento**.

Salvo excepciones, tendrán total libertad en cuanto a la interfaz, el diseño y la implementación de su trabajo.

Fecha límite de entrega de tarea 5, martes 17 de noviembre 21:30 hrs.

Indicaciones

El objetivo de esta entrega es ***vincular el lado cliente (navegador/browser) con el lado servidor (lo que está en NodeJs)*** . Podrá de todas formas avanzar y mejorar aspectos de usabilidad del lado cliente, el protocolo de envío de jugadas, el modelo de base de datos y la gestión de reglas en el lado servidor.

En esta entrega es indispensable y obligatorio que entreguen:

1. **Aspectos mínimos de entregas anteriores**
2. Registro de usuarios en el servidor desde el cliente.
3. Autenticación de un usuario en la plataforma.

   OBS: Se pueden usar cookies como mecanismo básico (por sí solas no bastan para el 7), pero pueden no usar cookies si usan un sistema más avanzado. Por ejemplo, si usan JWT pueden usar el *Bearer* o *Authotization Header* sin cookies, lo cual puede ser recomendable para evitar problemas con CORS. Se permite exigir el token en cada *request*.

4. Manejo de seguridad en base de datos y en inicio de sesión. Uso de *crypto* o *bcrypt*. Guardar contraseñas encriptadas. Ocultar de lado navegador la password al ingresarla.
5. Uso **pertinente** de middlewares, ya sea con Koa (recomendado) o Express.
6. Gestión del **flujo completo de las partidas** para grupo de jugadores. Entendiendo una partida como un juego completo desde su configuración inicial (escenario de partida en el tablero) y adición de jugadores, hasta el fin del juego con la determinación del jugador ganador y otros resultados.
7. Capacidad de multijugadores, es decir, varios jugadores conectados al mismo tiempo consultando al servidor. Los jugadores pertenecen todos a la misma partida. **No se exige "multipartidas"**, pero se entiende que lograr un manejo de muchas partidas simultáneas es un plus.
8. Mejora y profundización del comportamiento (acciones) a nivel de servidor y la documentación clara para su entendimiento.
9. Intercambio de jugadas entre cliente y servidor: Vale decir que los jugadores hagan su jugada en el navegador y luego tras una acción explícita (por ejemplo un botón "enviar jugada") la jugada se envía al server sin más intervención usuaria.
10. Entrega de jugadas desde el servidor. Vale decir, cuando el servidor ha procesado todas las jugadas, entrega a cada jugador el resultado del juego.

    OBS: el servidor debe poder hacer jugadas a pesar de no recibir las jugadas de todos los usuarios. Recordar que la entrega de jugadas tiene una fecha límite.

11. **Documentación** básica **de cada endpoint de la API** expuesta por el servidor. Documentar tanto *requests* como *responses*.

Otra opción para un plus es que el modelo guarde todas las partidas del juego y que un usuario pueda ver el historial de sus juegos, no sólo del actual si no de anteriores. Otro “plus” posible, que las jugadas terminadas puedan ser animadas en el tablero, es decir, que el usuario baje la “animación” y al darle play pueda ver todas las jugadas (propias y de otros jugadores) secuencialmente, desde el inicio al fin.

Con esta nueva entrega, el servidor levantado será capaz de integrarse con el lado cliente y procesar las jugadas generadas y entregadas por el cliente. Tras procesarlas, generará un archivo JSON (o equivalente) que el servidor enviará a los jugadores específicos con la información relevante para cada jugador y de acuerdo a las reglas de “visibilidad” que ustedes declararon.

La entrega de todos los archivos se hará en el repositorio de Github creado para su grupo.

El servidor (NodeJs) con su BDD (Postgres) se corregirá únicamente según sea usable en Heroku (o equivalente). No basta con que esté el código en Github. Deben entregar en el Readme.md las credenciales y rutas necesarias para que el ayudante pueda conectarse y ejecutar el juego en el sitio.

NO se aceptarán:

- Entregas por mail (ya sea al profesor o ayudantes)
- Entrega en otro sistema que no sea el que se ha provisto para estos efectos (el repositorio Github entregado por el coordinador y deploy con Heroku o equivalente)
- Respuestas atrasadas para la **evaluación de pares**

Condiciones y restricciones

1. Se debe entregar una aplicación que permita generar jugadas a nivel cliente, emitir un archivo desde el cliente y *enviar* los archivos de jugadas al servidor, el cual las almacenará en la BDD y posteriormente entregará archivos de resultados (uno por jugador) que deberán ser *recibidos* en el lado cliente y visualizar el resultado de la jugada.
2. La recepción de la jugada ha de ser a solicitud del cliente desde su aplicación.
3. El framework que recomendado será Koa.js, aunque puede utilizar Express también (pero sólo uno de esos dos)
4. El archivo de jugadas estará en un formato de protocolo que ustedes definan, es libre. Se sugiere explorar JSON aunque otros formatos son aceptados (como XML)
5. El mecanismo de interoperación entre cliente y servidor lo definen ustedes. Deben documentarlo de tal forma que un tercero pueda utilizar ese

mecanismo sin problemas para hacer jugadas a nivel de cliente y ejecutar los turnos a nivel servidor.

6. Las reglas del juego pueden sufrir variaciones con respecto a la entrega anterior, estas modificaciones **deben señalarlas** en su archivo README.md

Glosario:

- **Partida:** Corresponde al ciclo de jugadas desde que inicia el juego con una configuración inicial, hasta que termina con la determinación de un ganador o abandono.
- **Jugada**: son las operaciones permitidas por el juego, que un usuario realiza. La Jugada es esencialmente algo que se realiza en el lado cliente. Un jugador puede enviar al servidor cuantas jugadas quiera por turno, pero el servidor utilizará sólo el último envío como válido al procesar el turno. *De todas formas, si ustedes determinaron por regla que un jugador podía enviar solo una jugada por turno, eso es válido también*.
- **Turno**: Corresponde a un periodo de tiempo establecido por las reglas del juego y controlado por el servidor, en el que todos los jugadores, inscritos en el juego, pueden enviar sus jugadas para que estas sean procesadas por el servidor. El turno termina cuando el servidor envía las respuestas con los resultados de la jugada a cada jugador. Una vez enviados los resultados, comienza automáticamente el siguiente turno.
- **Elementos(piezas) de un jugador**: Corresponde a todos los elementos (terreno, recurso, personajes, otros) de los cuales dispone un jugador para poder realizar una jugada.
- **Archivo de jugada**: Corresponde a la serie de comandos, que un jugador realiza en su jugada, y que son organizados en una estructura previamente definida de tal forma que el servidor entienda las acciones que el jugador realizó en su jugada.Al respecto el arhivo de jugada puede perfectamente NO ser un archivo físico sino que alguna estructura de datos ad-hoc al juego programado.
- **Archivo de turno o de servidor**: Corresponde a la descripción de la situación que quedan los elementos (piezas) de todos los jugadores una vez concluido el turno. El archivo de servidor hace una descripción detallada de cada jugador indicando la situación en la que quedó y , dependiendo del diseño del juego, mostrar la situación completa o parcial en la que quedaron el resto de los jugadores.

Recomendaciones

1. Si el protocolo es complejo o tienen aspectos de modelo de datos o de usabilidad que los complican, no teman de realizar cambios y ajustes necesarios para entregar con éxito su tarea. Redefinir o rediseñar no está mal

en la medida que cumplan con los requerimientos establecidos en las entregas.

2. Implementen todos los puntos de “contacto” (como quien dice, los “end points”) que requieran y todas las APIs necesarias, no tiene que se una sola y recomendamos que sean más atomizadas.

3. Aprenderán mucho más si trabajan colaborativamente en su grupo, como equipo en lugar de repartirse el trabajo y realizarlo como unidades independientes.

4. Recuerden que hay una evaluación de pares la cual no es un castigo al que no trabaje sino más bien una protección a los que sí se esfuerzan.

5. Si hay problemas con su compañero(a) y no lo pueden resolver, comuníquense con el profesor o con el coordinador

6. Diseñen muy bien las reglas, el entorno,los turnos, la resolución de empates y conflictos

7. Planifiquen el trabajo para que les permita la colaboración entre los integrantes del equipo

8. Pregunten y consulten, usen foro, colaboren entre ustedes (NO COPIEN). Los ayudantes están para apoyarlos

9. **Trabajen con tiempo,** no esperen a último momento para **comenzar con la tarea** o **despejar dudas.**

10. Siempre podrán, justificadamente, cambiar alguna regla, mejorar algún aspecto del juego, etc.

Dudas

Para que todo el curso se vea beneficiado, hagan sus preguntas sobre el material del curso, sobre tecnologías web, y sobre el proyecto a través de los **foros del curso** dispuestos para estos efectos. No se responderá ninguna duda de tareas por e-mail.