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

Tarea 4

IIC2513 Tecnologías y aplicaciones WEB

Segundo semestre 2020
Fecha límite de entrega de tarea: lunes 02 de noviembre 21:30 hrs.

Fecha límite de evaluación de pares: viernes 06 de noviembre 23:59 hrs.
(https://forms.gle/JR3Yqs3hVv996vuy9)

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 4, 02 de noviembre 21:30 hrs.

Indicaciones

El objetivo de esta entrega es ***comenzar con la programación de reglas de negocio (en este caso las reglas del juego) y creación del modelo de datos que***

soportará su juego. Además podrá de todas formas avanzar y mejorar aspectos de usabilidad del lado cliente.

En esta entrega es indispensable y obligatorio que entreguen:

1. Comportamiento (acciones) a nivel de servidor y la documentación clara para su entendimiento. En este caso el servidor recibirá jugadas en formato establecido según protocolo y, de acuerdo a las reglas que ustedes establecieron, actuará como "árbitro" dirimiendo los resultados, los puntos a otorgar, victorias, empates y condiciones en los que quedarán los elementos de cada jugador. Se entiende por elementos los personajes, territorios, recursos, etc. que tenga su juego.

   OBS: El código del servidor deberá estar alojado en el **nuevo repositorio** de la organización de Github habilitado para cada grupo con este propósito. En el Readme.md de este nuevo repositorio basta con que hagan **referencia al repositorio principal (cliente) que contendrá toda la documentación.**

   OBS: El servidor deberá ser en **NodeJs, versión *12.x* o *13.x***. Se aconseja última versión estable.

   OBS: El servidor debe poder **simular que recibe jugadas** (ingresadas **manualmente**, por ahora) en formato JSON (o equivalente) y **procesarlas** apropiadamente.

   OBS: El servidor debe **distinguir quién es el jugador** que le está enviando el mensaje JSON, pero no es necesario todavía un protocolo de login seguro.

   OBS: El servidor debe realizar el procesamiento del turno una vez que tenga la información de todos los jugadores. En esta entrega **pueden asumir que todos los jugadores responderán** al turno.

   OBS: El servidor solo cumple funciones de ***backend***, no expone interfaz de usuario.

2. Documentación del **modelo de datos** que se utilizará para su base de datos (modelo entidad relación).

   OBS: Pueden usar como base el diseñado para la entrega 1, pero debe tener las actualizaciones pertinentes, si aplica.

3. Implementación del modelo de datos en una **BDD** Postgres.

   OBS: Pueden usar el ORM **Sequelize** o pueden utilizar "**pg**" (driver)

4. Base de datos y servidor **desplegados en Heroku** o similar.
5. En el **Readme.md** del repositorio principal (cliente), deberán indicar el enlace a su repositorio de servidor y a su aplicación en Heroku y junto con cualquier información necesaria para su corrección. También deben estipular cualquier cambio o decisión relevante para la corrección.

Para el caso del lado cliente deberán realizar la entrega de esta versión utilizando todas y cada una de las consideraciones para lado cliente que se entregaron para la tarea número 3 (estructura de archivos, archivos mínimos requeridos, etc.), Como **recordatorio**, esto considera, entre otros:

1. La capacidad de procesar en el cliente la información para el turno desde el servidor según protocolo establecido (JSON recomendado), cargándola a LocalStorage
2. Editar en el cliente las decisiones del turno actualizando las visualizaciones y la información en LocalStorage
3. Generar en el cliente el archivo de decisión de jugada en formato según protocolo para ser procesado por servidor

Con esta nueva entrega, se levantará un servidor capaz de procesar estas jugadas generadas con el cliente (ingresadas manualmente al servidor, por ahora). Tras procesarlas, generará un archivo JSON (o equivalente) que se puede ingresar manualmente (por ahora) al cliente, según la entrega 3.

Cabe aclarar que los **clientes solo reciben la información estrictamente necesaria** para que su jugador (login, actualmente solo simulado) pueda informarse del estado del juego y tomar una decisión, según las limitaciones de las reglas del juego. Solo deben recibir la información de otros jugadores a la que tengan permitido el acceso.

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 cargar los archivos de jugadas en el servidor, el cual las almacenará en la BDD y posteriormente entregará archivos de resultados (uno por jugador) que deberán poder ser cargados en el lado cliente y visualizar el resultado de la jugada.

2. El framework que se se utilizará (y será exigido a partir de la siguiente tarea NO en esta) será Koa.js
3. El archivo de jugadas estará en un formato de protocolo que ustedes definan, es libre. Se sugiere explorar JSON.
4. El mecanismo de carga y descarga tanto desde cliente como de 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.
5. Las reglas del juego pueden sufrir variaciones con respecto a la entrega anterior, estas modificaciones **deben señalarlas** en su archivo README.md

Glosario:

- **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. Establezcan claramente las reglas del juego. Diseñen con cuidado como el servidor interactuará con n-jugadas a la vez, entendiendo, por ejemplo, que un territorio pudiese ser invadido por dos o tres jugadores al mismo tiempo, si

eso es así ¿Quién se queda con el territorio? En el fondo el servidor será el árbitro, pero para que el árbitro dirima, debe haber claridad en todas las reglas y las situaciones que pueden ocurrir.

2. Recomendamos que separe las reglas por ciertas acciones que deben tener precedencia sobre otras. Por ejemplo, pueden decidir que primero se realicen todas las acciones de “sanación” de personajes y luego las acciones de batalla o viceversa. Este tipo de decisiones claramente afectarán el resultado de una jugada.

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. No traten de resolver aún detalles específicos de integración o comunicación con el servidor. Sin embargo, ya **es momento que vayan pensando el uso que darán a la base de datos como reserva de jugadas. Por ejemplo, guardarán todas las jugadas? solo la última y la penúltima? etc. Vayan también pensando y diseñando el manejo de la sesión, encriptación y seguridad de datos y manejo de cookies.**

11. 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.