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

Entrega 2

IIC2513 Tecnologías y aplicaciones WEB

Fecha límite de entrega: viernes 13 de mayo 23:00 hrs.

Recuerden que los enunciados sólo 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, viernes 13 de mayo 23:00 hrs.

Indicaciones

El objetivo de esta entrega es ***comenzar con la programación de reglas de su juego, junto con la creación del modelo de datos que soportará su juego y el protocolo que se usará para comunicar jugadas y el resultado de éstas***. Además podrá de todas formas avanzar y mejorar aspectos de usabilidad del lado cliente.

En esta entrega se espera que como mínimo tengan implementado/elaborado:

1. Documentación clara y completa de las reglas del juego. Esta documentación estará online y deberá ser “navegable” por el usuario.

2. Comportamiento (acciones) a nivel de servidor de su juego. Para probar las reglas implementadas, el servidor que ustedes programen (en NodeJs, usando Javascript), recibirá las jugadas usando JSON, en algún protocolo de jugadas establecido por ustedes. Este JSON puede ser creado a partir de una estructura de datos (objetos, arreglos, etc.) que se debe ingresar y completar en algún archivo .js o también puede ser un archivo de sistema que el programa leerá desde disco (esto es altamente recomendable (para uso de archivos no olviden que existe el módulo “fs”)

3. Las jugadas se enviarán al servidor utilizando un request de HTTP con POST. Este envío será a través de su aplicación cliente en una página (HTML) que ustedes crearán solo para ese propósito, entendemos que la parte cliente estará bastante incompleta.

4. Las acciones pueden ser simuladas parcial o totalmente (todavía no conectamos completamente el cliente con el servidor!) pero sin embargo estas jugadas simuladas deben tener ya la **definición del protocolo de jugada** que ustedes implementarán. Vale decir, ustedes deberán tener un JSON que defina la jugada. Este JSON debe incluir al menos un identificador del jugador (todavía no es exigible la autenticación de usuario) y un conjunto de movimientos permitidos según su jugada.

   Los movimientos permitidos deberán tener en cuenta el contexto de la jugada, vale decir si por alguna razón una "pieza" o personaje, no se puede mover y el conjunto de instrucciones dice que si se puede mover, entonces el programa debe retornar como inválido dicho movimiento e impedirlo. Así pues, su servidor actuará como un árbitro que decidirá el resultado de los movimientos entregando puntos, otorgando victorias, empates y las condiciones y/o estado en los que quedarán los elementos (piezas, personajes, recursos, etc.) de cada jugador.

5. La "respuesta" del Servidor a la jugada con un protocolo, también diseñado y documentado por ustedes, que retorne código JSON que ustedes deberán mostrar en el lado cliente (puede mostrarse el JSON completo como texto, no es necesario, para esta entrega, tener una respuesta gráfica en el lado de cliente)

   Es decir, en resumen, el servidor debe poder **simular que recibe jugadas** mediante POST, en formato JSON, **procesarlas** apropiadamente y responder al cliente en un formato JSON también.

   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.

   NOTA: Recuerden, el servidor solo cumple funciones de ***backend***, no expone interfaz de usuario.

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

7. Implementación del modelo de datos en una **BDD** Postgres, en Heroku (o similar).

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

8. Base de datos y servidor **desplegados en Heroku** o similar.
9. 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.

10. Implementar, en el lado cliente, una página de ingreso a su aplicación que será un archivo index.html, el cual será servido estáticamente por su servidor. Desde este index.html ustedes podrán navegar por el resto de su aplicación. Creen una página especial que permita comunicarse con el servidor y enviar las jugadas.(Acá habrá HTML, CSS y JS)

    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 entrega anterior (estructura de archivos, archivos mínimos requeridos, etc.)

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.

Cabe aclarar que los **clientes solo reciben la información estrictamente necesaria** para que su jugador (login, puede ser simulado aunque puede no existir también) pueda informarse del estado del juego y tomar una decisión, según las limitaciones de las reglas del juego.

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)

Condiciones y restricciones

1. El framework que se se utilizará será Koa.js
2. Pueden utilizar, si lo desean, el template del curso.
3. Las reglas del juego pueden sufrir variaciones con respecto a la entrega anterior, estas modificaciones **deben señalarlas** en su archivo README.md

Las reglas también pueden modificarse, de manera justificada, en las entregas que siguen.

Recomendaciones

1. Establezcan claramente las reglas del juego. Diseñen con cuidado como el servidor interactúa con todos los jugadores a la vez. 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. Diseñen muy bien las reglas, el entorno,los turnos, la resolución de empates y conflictos

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

6. No traten de resolver aún detalles específicos de integración o comunicación con el servidor. Sin embargo, ya **es momento de 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.**

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

8. 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.

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

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.