



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

Entrega 3 de proyecto

IIC2513 Tecnologías y aplicaciones WEB

Segundo semestre 2022

Fecha límite de entrega: Viernes 4 de noviembre, 23:59 hrs.

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.**

Fecha límite de entrega : Viernes 04 de noviembre, 23:59 hrs.

Indicaciones

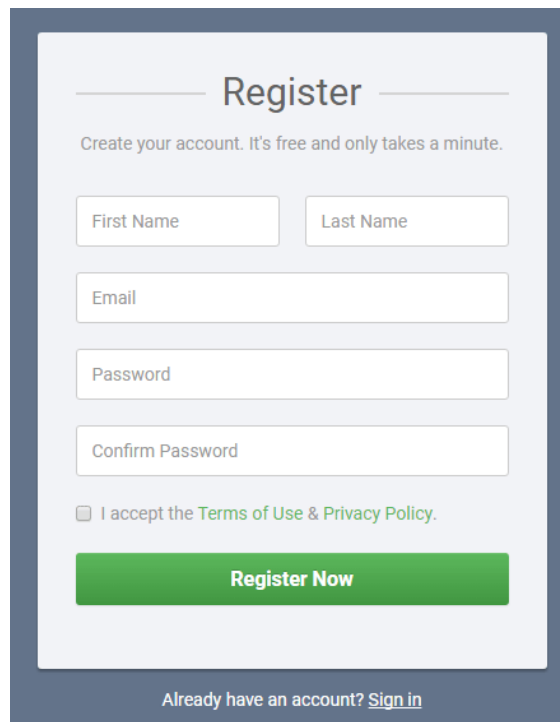
El objetivo de esta entrega es unir de forma completa las partes cliente y servidor de su aplicación. Consideren esta la primera versión funcional que cumple con todos los aspectos de usabilidad (después veremos temas de testing y seguridad).

En esta entrega es obligatorio que entreguen el comportamiento integral (acciones/reglas de negocio) a nivel de front-end, es decir en el navegador (browser). Su juego aplicación será ahora totalmente interactiva con el usuario, permitirá una navegación completa y la gran mayoría de las funciones ya deberán estar disponibles, con esto nos referimos a que habrá algún tipo de respuesta ante una acción del usuario frente a la interfaz. Aclaramos que algunos acciones que provienen desde el servidor pueden estar aún incompletas pero es relevante que el usuario lo sepa (por ejemplo, un mensaje que diga que esa funcionalidades en particular estará disponible en la versión 2.0 de la aplicación)

Se espera que su aplicación al menos sea capaz de:

1. Presentar una página inicial (landing page estática) que introduzca las características principales de su aplicación entregando información suficiente y clara al usuario para que éste pueda saber de qué se trata el sitio Web.

2. Permitir a un usuario cualquiera registrarse en la aplicación. El registro debe permitir a cualquier persona crear una cuenta con nombre de usuario y clave. Como regla, no se debe permitir la repetición del nombre de usuario.
3. Permitir a un usuario registrado ingresar al sitio con su login y password.
4. Enviar mensajes de error adecuados al tratar ya sea de registrarse o de ingresar a la aplicación. Por ejemplo, indicar al registro que el nombre de usuario ya existe, poner ciertos criterios o validaciones mínimas (a nivel de front y back) para crear un nombre de usuario (prohibir el uso de ciertos caracteres, permitir solo letras y números, etc.)
5. Validar correctamente el ingreso (login) de un usuario, para ello debe validar correctamente la dupla nombre usuario-clave de usuario e iniciar sesión.
6. Solicitar al usuario, al momento del registro, el reingreso de su clave para comprobar (confirmar) que no hayan errores y el usuario pueda replicar su clave. En la siguiente figura se muestra un formulario de ingreso adecuado que puede servir de ejemplo a lo que se espera:



Register

Create your account. It's free and only takes a minute.

First Name Last Name

Email

Password

Confirm Password

☐ I accept the [Terms of Use & Privacy Policy](#).

Register Now

Already have an account? [Sign in](#)

7. Manejar la sesión de usuario, para ello se espera utilicen *koa-session* o similar (en caso de express)
8. Utilizar, para el manejo de accesos privilegios, roles, etc. JSON Web Token (JWT). Se espera que utilicen intercambio de tokens vía JWT.

*OBS: Se pueden usar cookies como mecanismo básico para sesión (**por sí solas no bastan para la banda A**), pero se recomienda un sistema más avanzado. Por ejemplo, para 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.

9. Cifrar, obligatoriamente las claves (passwords) en la base de datos, usando hash. Al lado del front end se debe ocultar la password al momento de ingresar.
10. Tener un usuario admin (administrador) que permita administrar todos los aspectos de su aplicación, como por ejemplo eliminar usuarios, borrar jugadas o simulaciones, eliminar usuarios y mantener los aspectos relevantes de su aplicación relativos a su configuración y uso. Estos recursos solo estarán disponibles para este tipo de usuario.
11. Presentar una usabilidad de su aplicación WEB “amigable” es decir, de diseño cuidado, navegación intuitiva, con títulos, menús, sub páginas y secciones que orienten al usuario en el uso y en los resultados que se obtienen. Se valorará (banda A) ayudas online sensibles al contexto. Esto significa que existan tutoriales o bien tooltips que guíen al usuario en el uso de la aplicación.
12. Gestionar el **flujo completo de las simulaciones o partidas** (según sea el caso de su aplicación) para cada grupo de usuarios (no olvidar que existen usuarios admin). Esto significa que su aplicación debe ser funcional desde la configuración inicial (escenario de partida) hasta el fin del proceso (simulación o juego) con la determinación de la salida esperada después de completado el flujo, por ejemplo determinación del jugador ganador, evidencia de la simulación y sus resultados, etc. Piense también en algunas estadísticas que pueda entregar, esto último no será evaluado ahora pero tengan presente un diseño pues la entrega de esa información se podrá solicitar en alguna de las bandas para la última entrega (no esta).
13. Administrar la concurrencia de varios usuarios conectados al mismo tiempo consultando al servidor. Los usuarios, para esta entrega, pueden actuar todos a la misma partida o simulación. Entregar un manejo de muchas partidas/simulaciones simultáneas es otro criterio para la banda A.
14. Entregar (documentado) los endpoints mínimos con el uso de al menos: GET, POST (o PUT), DELETE. Se valorará la **coherencia** entre usar PUT/POST/PATCH (eso se determinará con justificación que ustedes hagan al respecto. Justificaciones documentadas, claro está). También se valorará el usar PATCH (que no es exigible como vieron) así como también se valorará usar, adecuadamente, otro tipo de request HTTP (como HEAD).
15. Entregar **Documentación básica de cada endpoint de la API** expuesta por el servidor. Documentar tanto *requests* como *responses*.
16. Utilizar elementos del DOM que impliquen eventos como botones para acciones, radio buttons, listas desplegables, etc. las cuales DEBERÁN ejecutar TODAS las acciones que puedan ser ejecutadas a nivel de cliente (browser).

17. Se espera el uso de REACT en la parte frontEnd, por lo tanto, documenten sus componentes. Se valorará el uso eficiente de REACT entregando una SPA. No se espera tener REACT para páginas estáticas que perfectamente se pueden servir usando html.
18. Se valorará positivamente el uso de buenas prácticas de programación, estructuras de archivos y directorios, diseño modular y escalable, reutilización de código, código documentado y otros aspectos de calidad en entrega. Por ejemplo, los códigos gigantes y monolíticos no son bien vistos.
19. Se valorará el uso de Github como elemento de calidad: Uso de branches que se asocien a una funcionalidad específica y commits descriptivos, además de la revisión de PRs.

USEN: <https://github.com/airbnb/javascript> como guía de estilo

20. Recuerden su archivo **README.md** que contendrá:
 - a. Consideraciones sobre su entregable en lo relativo al diseño de su HTML y mejoras/cambios con respecto a la entrega anterior.
 - b. Consideraciones sobre las reglas (cambios que hayan implementado)
 - c. Consideraciones sobre su archivo .js y las acciones que se pueden realizar a nivel de navegador (client-side)
 - d. Detalle de las mejoras que introdujeron a su diseño visual con el HTML y CSS así como usabilidad y acciones, respecto a la entrega 2
 - e. Cualquier tipo de supuesto, restricción, o información relevante para corregir su entrega

ASPECTO ADICIONAL FUERA DE ESTA ENTREGA (Pero parte de la banda A para la siguiente entrega)

Como medida de SEGURIDAD se les pide que implemente un mecanismo sencillo, de respaldo de información sensible de usuarios de su aplicación.

Este mecanismo opera de la siguiente forma:

1. En Heroku u otro servicio análogo (Linode, Digital Ocean, AWS, por ejemplo) ustedes tendrán una pequeña base de datos que guarda la información sensible de sus usuarios (**determine ustedes que es esa información**, de partida user-password. ¿Algo más? ¿El estado de uso de la aplicación? ¿Alguna estadística? ¿Datos personales?)
2. En Heroku (u otro) definan un endpoint que tendrá un único "servicio" dado por PUT/POST/PATCH (ustedes deciden el método) y que se encargará de recibir un Json con la información que ustedes envíen desde su aplicación. La información de este Json será lo que se guardará en la BDD del punto 1.
3. Para enviar esa información a Heroku, **solo** el administrador de su aplicación podrá hacerlo. Diseñen una página que le permita hacerlo, basta con un botón que diga "respaldar" o algo parecido, en el panel de administración que ustedes han construido.

4. El administrador, por tanto, enviará la información de respaldo desde su servidor (localhost) hacia el servicio en la nube (Heroku) usando la API expuesta y la información se guardará en la BDD de Heroku (u otro)
5. En OTRA página estática SEPARADA de su aplicación y NO en su localhost, sino que desplegada en **Netlify** o Vercel, habrá un pequeño panel (tendrá título, estilos, una breve descripción). En él, un usuario podrá pedir a Heroku (GET) toda la información respaldada y se podrá desplegar en el panel mostrando los datos recibidos.

Nota 1: sólo por propósitos de simplificar el desarrollo no se exige que este panel de respaldo tenga mecanismo de ingreso vía usuario-password, sin embargo el hecho de tenerlo es un criterio que se puede considerar para la banda A.

Nota 2: Se insiste en el uso de Heroku dado que no hay riesgo de cobros al no exigir -por ahora- una tarjeta de crédito. Por lo mismo, a partir del 28 de noviembre se les solicita e insiste que dicha cuenta sea bajada (a menos que quieran pagar bajo las nuevas condiciones comerciales entregadas por Heroku).

Entrega:

La entrega de todos los archivos se hará en los repositorios de su grupo.

Estructura de archivos:

Una estructura apropiada para su repositorio tras esta entrega **podría** verse como el siguiente ejemplo:

```
-
+-- documents/
+-- src/
|   +-- assets/
|   |   +-- styles/
|   |   +-- imgs/
|   |   +-- scripts/
|   +-- views/
|   |
|   +-- lib/
+-- index.html
+-- index.js
+-- .gitignore
+-- README.md
-
```

Condiciones y restricciones

1. Se debe entregar una aplicación **totalmente funcional a nivel de cliente** que se comunique de manera efectiva con el servidor y se calculen las jugadas

según sus reglas de juego (puede mejorar aspectos de manejo cliente en las siguientes entregas) con estructura, estilo y comportamiento.

2. Las reglas del juego pueden sufrir variaciones con respecto a la entrega anterior, estas modificaciones **deben señalarlas** en su archivo README.txt.
3. Debido a que Heroku cambiará el 28 de noviembre la forma de entrega de servicio, es que se ha permitido el levantar el servicio WEB en su localhost. Si ustedes quieren utilizar Heroku y desplegar su aplicación allí (o en otro lugar como AWS) se puede hacer voluntariamente (tengan cuidado con los cobros!)

Recomendaciones

1. Pongan atención a los mensajes de su página, el manejo de errores, las validaciones, la navegación y a la mecánica de uso de su aplicación.
2. Recuerden mirar y utilizar la guía de estilo entregada en este enunciado (<https://github.com/airbnb/javascript>)
3. Diseñen adecuadamente sus componentes para usar en REACT.
4. Aprenderán mucho más si trabajan colaborativamente en su grupo, como equipo en lugar de repartirse el trabajo y realizarlo como unidades independientes.
5. Si sienten que quisieron abarcar demasiado en su aplicación (muchas veces pasa), ajusten su aplicación, cortando funcionalidades y características que no sean tan relevantes y que su remoción no quite el espíritu de su aplicación en el servicio que pretende prestar. De todas formas comuniquen estas simplificaciones a su ayudante y traten que no sea una simplificación que previamente se haya acordado como necesaria.
6. Pregunten y consulten, usen foro, colaboren entre ustedes (NO COPIEN). Los ayudantes están para apoyarlos

7. Trabajen con tiempo, no esperen a último momento para comenzar con la entrega o despejar dudas.

8. 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 entrega por e-mail.