Pontificia Universidad Católica de Chile Escuela
de ingeniería
Departamento de ciencias de la computación

Entrega 3

IIC2513 Tecnologías y aplicaciones WEB

Primer semestre 2024

Profesor: Hernán Cabrera

Los enunciados de proyecto dan instrucciones y aspectos genéricos para la construcción de su sitio WEB, pero sin entrar en detalles ni puntos específicos de tal manera que ustedes demuestren su capacidad de análisis, trabajo en equipo, resolución de problemas y puedan exponer sus ideas y así acordar los criterios de evaluación con su ayudante de seguimiento.

LEAN CON ATENCIÓN EL ENUNCIADO COMPLETO

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

Fecha límite de entrega: lunes 3 de junio, 23:59 horas

Indicaciones

Hemos visto que un sitio WEB se compone, al menos, de una parte, cliente, llamada frontend, que se accede por medio de un browser y otra parte de backend, que reside en algún servidor en alguna parte del mundo.

Para esta entrega será relevante la aplicación de conceptos de de HTML, CSS, JS en el browser y DOM, conocimientos que hemos visto ligeramente pero que los seguiremos profundizando junto con un manejo introductorio de REACT.

El objetivo de esta entrega es transformar su diseño inicial en un conjunto de archivos HTML con CSS y algo de JavaScript-DOM, que den vida de manera visual y funcional de su aplicación en frontend (es decir la capa cliente, que es lo que ve y maneja el usuario en el browser/navegador). Para esto, deberán entregar las páginas estáticas [1] HTML que serán parte de su sitio WEB. A esas páginas, ustedes le incorporarán un diseño y estilo (CSS) e implementarán una navegación básica por las distintas páginas estáticas.

[1] por “estáticas” nos referimos a que su entrega sólo será, por el momento, algo de carácter “local”, es decir en el PC de usuario.

Su sitio WEB debe incluir, entre otras cosas: imágenes, texto, secciones, instrucciones de su aplicación, preguntas frecuentes y una página especial de “nosotros” que relata cuál es el equipo de trabajo. Por supuesto pueden crear otras secciones con toda otra información que consideren relevante.

En particular, se deberá entregar:

1. Un **mockup** de su sitio web cliente, es decir un diagrama esquemático inicial, un “borrador” de su interfaz de cliente
2. Un archivo **index.html** que será su landing page, es de contenido estático y presenta su juego, con imágenes relativas al juego y una barra de navegación
3. La **barra de navegación** permitirá navegar por las distintas secciones de su sitio, en particular tendrá:
    - **Página de instrucciones** (html estático)
    - **Página de “nosotros”** que describe el equipo de desarrollo
    - **Página de “ir a partida”** que llevará a un juego
4. Además habrá botones para **“login”** y para **“registro”**
5. Se deberán crear las páginas estáticas de instrucciones y de **“nosotros”**
6. Uno o más archivo con estilos **.css**
7. La posibilidad de navegar entre distintas páginas HTML (usen la etiqueta de anchor para ello)
8. La página de inicio de su aplicación WEB, que sea “amigable” es decir, de navegación obvia, con títulos, menús, subpáginas y secciones que orienten al usuario.
9. La página de partida debe ser de diseño modular, es decir que conceptualice módulos reutilizables para permitir el uso de **React**.
10. La página de juego debe también permitir seleccionar de un listado, **TODOS sus endpoints programados** y poder conectarse a su lado servidor, enviar datos y recibir a su vez la respuesta del servidor.
11. Para el punto anterior, deberán proporcionar un formato, en su página de juego, para ingresar la información que recibirá el endpoint. **Documente aquello**.
12. Deberá documentar las consideraciones, guía y reglas detalladas para el uso de su aplicación, a nivel de usuario como de administrador (al menos esos niveles se han de considerar).
13. Debe implementar el login, manejo de sesión y el registro de usuarios. La interfaz puede ser muy básica, en un HTML estático, pero debe ser plenamente funcional (es decir debe permitir registrar usuarios y permitir hacer login al espacio del usuario)
14. La página debe desplegar en una esquina superior (la derecha o izquierda, ustedes elijan) el nombre del usuario “logeado”.
15. Se debe permitir el cierre de sesión y por lo tanto cambiar con ello la visualización del nombre de usuario.
16. TIP: para el punto 14 y 15, eso háganlo con reglas de JS.-cliente, es decir, ocultar/mostrar “divs” uno que sea genérico (usuario no registrado) y el otro que cambie un innerText una vez que ocurra el login. (FÁCIL!)
17. De ser necesario, deberán entregar también, en el directorio correspondiente, todas

las imágenes y otros recursos requeridos para el despliegue correcto de su sitio.

18. La barra de navegación deberá estar presente siempre en su página (en la parte superior) que permitirá ir a “home” o a cualquiera de las secciones que ustedes entreguen (recomendación: cada sección diseñada debiese corresponder a una página HTML)
19. Deberán entregar Javascript de lado cliente (archivos .js) que permitan hacer algunas acciones básicas sobre sus páginas, por ejemplo, usar elementos HTML como un tag selection y que ante la selección de una opción cambie algo en la página, o el uso de botones que realicen acciones locales, etc. Recuerden que esto es parte integral de vuestro diseño de frontend.

Adicionalmente **deben entregar** un archivo README.md. (en el repositorio grupal) que incluya toda la información relativa al diseño de su HTML, consideraciones sobre las reglas (cambios que hayan implementado), detalle de las mejoras que introdujeron a su diseño visual con el HTML y CSS respecto a la entrega cero, cualquier tipo de supuesto, restricción, o información relevante para corregir su entrega.

Todo lo descrito anteriormente es lo mínimo que se espera en su entrega.

¿Qué cosas se evaluarán positivamente para mejorar la banda (escala) del mínimo?

- Uso de Flex y Grid de manera adecuada.
- Un diseño atractivo de las páginas (cuiden uso de colores, textos, ortografía, etc.)
- Un diseño modular adecuado (cuando corresponda) para el uso de React. Por ejemplo, la selección de elementos del juego (personajes, recursos, etc.) al estilo "cartas" o bien ya los primeros prototipos de su tablero.

Importante: Usen lo que ya conocen de ingeniería de software y otros cursos en lo relativo a diseño de software. En particular haga su trabajo orientado a la reutilización de código, a código modular y a buenas prácticas de programación.

👉 GUIA DE ESTILO QUE USAREMOS EN EL CURSO: Leanla, es obligatorio su uso. Se aplicará tanto para JavaScript como para React.

https://github.com/airbnb/javascript

¿Se quedaron en blanco y quisieran ver un ejemplo de páginas más bien estáticas?

https://phylum.io/ https://mystaticself.com/

https://loops.wannathis.one/

¿No se le ocurren buenas combinaciones de colores?

https://color.adobe.com/create/color-wheel

¿Alguna guía rápida de JS-HTML-CSS?

https://htmlcheatsheet.com/

https://web.stanford.edu/group/csp/cs21/htmlcheatsheet.pdf

http://www.simplehtmlguide.com/cheatsheet.php

https://ilovecoding.org/blog/htmlcss-cheatsheet

https://fundamentals.generalassemb.ly/11_unit/dom-cheatsheet.html

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

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 GitHub entregado por el coordinador)

Estructura de archivos:

Una estructura apropiada para su repositorio de frontend tras esta entrega podrá verse como el siguiente ejemplo (no olviden los módulos React en su folder “components”)

```
-
+-- documents/
+-- src/
 | +-- components
 | +-- assets/
 | | +-- styles/

 | | +-- imgs/
 | +-- views/
+-- index.html
+-- .gitignore
+-- README.md
```

Condiciones y restricciones

Para esta entrega, cada equipo de estudiantes deberá trabajar en completar la implementación del backend, que da soporte a las funcionalidades del juego, y completar el frontend, que es la interfaz por medio de la que el juego es utilizable. Ambas piezas de software se deberán integrar para trabajar en conjunto.

Las componentes de software a implementar en esta entrega son:

1. **Base de datos en PostgreSQL**: Se debe diseñar un modelo apropiado para el contexto, además de documentar cómo levantar la base de datos para probar la aplicación. Para interactuar con la base de datos, se deberá utilizar Sequelize.
2. **API RESTful** para interactuar con la base de datos: Este servicio deberá estar disponible para el frontend, para consultar y almacenar los datos relevantes al modelo del punto anterior. Esta API deberá ser implementada en Koa. En caso de que existan elementos pendientes de desarrollar de la entrega anterior, estos deberán ser completados para esta entrega.
3. **Frontend como UI para la interacción con usuarios**: Se debe implementar un frontend utilizando React para actuar como interfaz de usuarios (UI). Para su

funcionamiento, deberá conectarse a la API (backend). En caso de que existan elementos pendientes de desarrollar de la entrega anterior, estos deberán ser completados para esta entrega. Se debe entregar una aplicación con estilos, es decir colores, fonts, control de despliegue (es decir ancho, alto de los elementos, etc.). La parte React será sólo preliminar, el objetivo es que incorporen desde ya en su diseño el manejo de módulos y de creación de contenido dinámico.

Al igual que en la entrega anterior, cada equipo deberá desarrollar documentación apropiada para cada pieza de software. Esta puede ser incluida en el README de cada repositorio utilizado. Este documento deberá incluir:

Backend:

- Instrucciones para levantar la base de datos, indicando comandos y programas a utilizar, junto con una breve descripción.
- Instrucciones para instalar dependencias de la API, indicando comandos a ejecutar y una breve descripción de lo que hacen.
- Documentación de la API, indicando (como mínimo) para cada endpoint: método HTTP a utilizar, ruta del endpoint, argumentos que recibe (y su formato) y lo que retorna este endpoint (y su formato). Se recomienda el uso de **Swagger** para documentar la API (ejemplo relevante a Koa) o **Postman** e incluir ejemplos en la documentación.

Frontend:

- Instrucciones para construir entorno de desarrollo de frontend, indicando comandos a ejecutar y una breve descripción de lo que hacen.
- Instrucciones para levantar la interfaz de la aplicación y conectarse a la API (backend).
- No se puede utilizar frameworks como Bootstrap, Tailwind, Bulma o similares. Cualquier librería que se quiera incorporar en el proyecto debe ser consultada a su ayudante y aprobada por este.

Entregables

Como equipo deberán entregar:

- Implementación del backend del juego, incluyendo la base de datos en PostgreSQL y la API en Koa.
- Implementación del frontend del juego, desarrollado utilizando React. Documentación

  para levantar la aplicación y montar la base de datos (backend).

- Documentación para levantar la interfaz de la aplicación y conectarse a la API

- (frontend). Documentación y/o ejemplos de uso de los endpoints implementados.

Bandas

A continuación, se describe el criterio para entrar a cada banda. Tengan en cuenta que los criterios son acumulativos y que para caer dentro de una banda es necesario tener todo lo de las bandas anteriores. La nota correspondiente a cada una de estas bandas se encuentra descrita en el enunciado general del proyecto.

- **Banda C**: En backend, se deberá completar la implementación de la lógica de juego. En frontend, deberá incluir la implementación de tablero básico funcional. Para que el tablero sea funcional, se deberá conectar la aplicación a los endpoints implementados en la entrega anterior. Entrega documentación solicitada para frontend y backend.
- **Banda B:** En backend deberá incluir manejo de sesión y permisos/autorización por medio de JWT (JSON Web Tokens), utilizando koajs/jwt y se deberá utilizar bcrypt para hashear las contraseñas de los usuarios.
- **Banda A:** Uso de JWT por medio de headers en requests que llegan al servidor (o por medio de http-only cookies). Para completar la banda A, deberá completar una (o más) de las siguientes opciones (acordar con el ayudante). Uso de ESlint en manejo de errores y estilo.

Recomendaciones

1. Las reglas de negocio de su aplicación pueden sufrir variaciones con respecto a la primera entrega, sin embargo, deben señalar esto en su archivo README.
2. Piensen en distintas opciones de presentación y la mejor forma de abordar su entrega. No se queden con la primera idea que le venga a la cabeza.
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, comuníquese con el profesor o con el coordinador.
6. Planifiquen el trabajo para que les permita la colaboración entre los integrantes del equipo.
7. Pregunten y consulten, usen foro, colaboren entre ustedes (NO COPIEN). Los ayudantes están para apoyarlos.
8. Trabajen con tiempo, no esperen a último momento para comenzar con la tarea o despejar dudas.
9. Comiencen con una interfaz de usuario (UI) simple; no es necesario que su aplicación se vea excepcionalmente "bonita" o “completa” desde el comienzo. Una vez que cubren lo básico, la pueden ir mejorando.
10. No traten de resolver aún detalles específicos de integración, comunicación, consultas a BDD u otros temas de implementación ajenos al alcance de esta entrega.
11. Siempre podrán, justificadamente, cambiar alguna regla de negocio, mejorar algún aspecto de su aplicación, etc.
12. Pueden usar ChatGpt. PERO, deben indicar qué usaron de ChatGpt (que es lo mismo que hacer una “cita” de bibliografía) y deben señalar el código entregado y qué modificaron ustedes para su entrega, es decir, deben mostrar qué valor agrega.