PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2513 — Tecnologías y Aplicaciones Web

Tarea 3

Entrega

- **Fecha y hora:** Viernes 19 de mayo de 2023, 20:00 hrs
- **Lugar:** Buzón de Tareas en Canvas

1. Indicaciones

Objetivo General:

Crear una calculadora, muy simple de 4 operaciones, cuyas operaciones no se resuelven en front sino que en back-end. Es decir, el usuario abre la calculadora en su navegador (browser), ingresa una operación y dicha operación es atendida por una API que ustedes programaran usando un middleware, proveerán las rutas end points y tendrán 2 operaciones que usarán GET y otras 2 que usarán POST (este diseño es para comprobar que pueden usar ambos "verbos")

Detalles del trabajo:

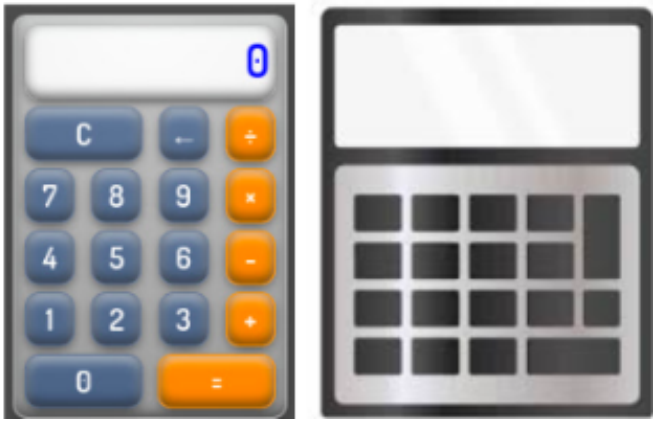
El objetivo de esta entrega es entregar una aplicación WEB que comunique front y back end utilizando APIs.

Para front: (Esta parte tiene una puntuación de 1,5 puntos)

Usted deberá crear una calculadora, usando HTML, CSS y JavaScript lado cliente (manejo de DOM) que tenga los dígitos de 0 a 9 y 4 operaciones: suma, resta, multiplicación y división junto con un visor donde muestre los números ingresados y su operación.

Si quiere puede utilizar React para construir el front, sin embargo no es requisito.

La calculadora deberá verse aproximadamente así (la imagen es solo referencial):



Consideraciones de la visualización:

- Los dígitos, operaciones y otras "teclas"de la calculadora deben desplegarse ordenada y armónicamente en la pantalla.
- Debe incluir una tecla para dar enter que sea mayor en tamaño que el resto de las teclas.
- Debe incluir una tecla de borrado de dígitos.
- La disposición de los dígitos debe seguir un orden lógico y se recomienda el uso de Flex o Grid para su despliegue.
- En el visor debe desplegarse el número a medida que se ingresen los dígitos. Si usted desea se puede ver la operación completa (por ejemplo: 1789 + 155) o mostrar un número, y luego de entrada la operación, el otro número (siguiendo el ejemplo anterior, se vería primero 1789 y luego de apretar enter se limpia el visor y sale 155).

Manejo de errores, su aplicación en front debe al menos: (0,5 puntos)

- Evitar que se ingrese cualquier otro caracter que no sea número u operador (suma, resta, multiplicación y división) también debe velar por la semántica de la operación (por ejemplo filtrar que se haga algo como *777)
- Deben manejar números negativos
- La división por cero la maneja como error el BACK, no Front

Comportamiento: (1 punto)

- El usuario debe ingresar un número, luego un operador y por último otro número utilizando para ello el teclado de la calculadora en pantalla
- Al momento de presionar "Enter" o la tecla =, usted deberá programar la invocación a fetch, que dependiendo del operador deberá invocar a una ruta específica de la operación, usando GET o POST
- Una vez recibida la respuesta, esta se desplegará en el visor
- Si presiona cualquier número, la respuesta se borrará del visor y podrá ingresar una nueva operación

Operaciones con GET: (1 punto)

Suma y multiplicación utilizarán GET, habrá una ruta para /suma, otra para /multiplicacion y los números se ingresarán como PARÁMETROS en la ruta por ejemplo puede usar (recuerde usar ctx.params):

- /suma/178/555

Operaciones con POST: (1 punto)

Resta y división, en este caso los números deberán ir en un JSON en el body (no olvide usar ctx.request.body)

Resultado: (1 punto)

- El HTTP response (que es manejado por fetch()) deberá contener en su body dos campos: status y resultado.

- En status indicarán si hubo éxito o hubo error y en resultado se dará el resultado de la operación o bien algún texto con error (el error que se considera es la división por cero)
- Obtenida la respuesta desde la API, el resultado debe verse en el visor de su calculadora

Entrega del trabajo:

- Usted deberá entregar dos carpetas, una con sus archivos que usa en front (index.html, los .css y .js, etc, usados por FRONT) y otra carpeta llamada back con los archivos para ejecutar el back (incluido index.js, todos los js de router y .env)
- Recuerde que debe levantar un servidor (local) que escuche en puerto 80 (default HTTP) usando koa (.listen)
- Deberá usar dotenv para todos los parámetros (en particular el puerto usado en app.listen deberá ser leído desde process.env)
- Para probar este programa el ayudante ejecutará index.html en su browser y todo el resto debiese funcionar sin problemas.

Algunos detalles:

- Use koa, koa-router, koa-bodyparser (o equivalente), .dotenv
- Puede usar la siguiente documentación para apoyarse en el uso de fetch.
- No olvide su archivo package.json. Si hay problemas de CORS (habitual si usa home, es decir 127.0.0.1), use koa/cors o koa-cors2.

Sobre la corrección:

- Si utiliza chatgpt o copilot u otro, indíquenos (no está prohibido). Si no lo indica y se detecta su uso, puede ser evaluado con un 1.
- Si usa chatgpt o copilot (u otro similar) y lo declara, la corrección se fijará en otros detalles de su implementación al momento de evaluar (por ejemplo, se considerarán aspectos de estilo, documentación, usabilidad, control de errores, orden en la entrega (carpetas, nombres de archivos), etc.).

2. Detalles de la entrega

- La entrega de todos los archivos se hará en un buzón de tareas en canvas.
- Se debe entregar un zip que contenga todos los archivos. En el nombre del zip se debe especificar el número de alumno y nombre y apellido, de la siguiente manera: `12345678_Perez_Juana.zip`

NO se aceptarán:

- Entregas por correo electrónico (ya sea al profesor o ayudantes).
- Entrega en otro sistema que no sea el que se ha provisto para estos efectos.

3. Recomendaciones

- Pregunten y consulten, usen el foro, colaboren entre ustedes (NO COPIEN). Los ayudantes están para apoyarlos.
- TODA referencia debe ser documentada. Por ejemplo, si usaron Stack Overflow o ChatGPT, indíquenlo. Si no lo indican, será considerado plagio y la evaluación será de un 1.0.
- Trabajen con tiempo, no esperen a último momento para comenzar con la tarea o despejar dudas.
- Recuerden que esto es programación "fullstack", "lado servidor y cliente" usando Node.js, Koa e implementando APIs.

4. 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 correo electrónico.