**CODING BOOTCAMPS ESPOL**

**REACT**

# Indicaciones

Se presenta el proyecto del módulo, una vez implementado el proyecto deberá ser publicarlo en un repositorio en Github.com y posteriormente desplegar la aplicación en Cloudflare.

Asegúrese de subir el código a GitHub y de incluir un archivo README.md (ubicado en la raíz del repositorio) con las siguientes secciones:

1. Descripción: explicación de la web app y su uso. Incluya detalles necesarios sobre la arquitectura de la web app para que alguien pueda mantener su código en el futuro.
2. Ejecución: explicación de los pasos necesarios para poner en ejecución la web app de manera local.
3. Despliegue: explicación de los pasos necesarios para el despliegue en la nube de la web app.

# Requisitos Técnicos

El estado de la aplicación deberá estar implementado utilizando Redux (Thunks), Zod, MomentJs y Tailwind. La aplicación debe estar organizada en diferentes módulos (archivos), priorizando el principio de Single Responsability, es decir, que cada módulo debe tener código enfocado a las tareas de una entidad, modelo o aspecto de la web app.

Prefieran el uso de funciones de orden superior (map, filter, reduce) en lugar de lazos imperativos. Al menos uno de sus componentes deberá utilizar el prop children

# Requisitos Funcionales

La aplicación será Walletfy, la cual permitirá a un usuario definir un dinero inicial de su wallet o billetera, crear eventos de consumo o ingreso de dinero, con la finalidad de poder visualizar el flujo de su balance a lo largo del tiempo según ocurren los eventos de ingresos o egresos.

## Entidades

* Evento: corresponde a la representación del ingreso o egreso del usuario. Sus campos deben ser los siguientes:
  + Id, obligatorio y único por cada evento. Se recomienda el uso de la librería [uuid](https://www.npmjs.com/package/uuid).
  + Nombre, obligatorio y de máximo 20 caracteres.
  + Descripción, opcional y de máximo 100 caracteres.
  + Cantidad, obligatorio y debe ser un número entero o decimal positivo.
  + Fecha, obligatorio y debe ser una fecha válida
  + Tipo, un String que solo puede ser “egreso” o “ingreso”.

## Secciones de la app

La web app constará de las siguientes secciones o pantallas.

### Flujo Balance

Esta sección servirá para la presentación de los eventos creados por el usuario. Los eventos deberán estar agrupados por meses. Cada componente de Mes deberá tener los siguientes datos:

* Nombre del mes y año.
* Computo total de ingresos del mes.
* Computo total de egresos del mes.
* Eventos, lista de los eventos correspondientes a ese mes. Esta lista puede ser fija o desplegable al dar clic en el mes. Cada evento deberá mostrar lo siguiente:
  + Nombre
  + Descripción, el cual deberá solo aparecer como un Tooltip al pasar el mouse por encima del evento. Como referencial pueden visitar esta documentación [React Tooltip](https://react-tooltip.com/docs/getting-started).
  + Cantidad.
  + Fecha, en el siguiente formato “DD/MM/YYYY”.
  + Tipo, texto del tipo de evento o algún otro distintivo como iconos que exprese el tipo de evento egreso o ingreso.

### Formulario de evento

Esta sección servirá tanto para la creación como para la edición de eventos creados por el usuario. Si el usuario previamente dio clic en el botón “Crear evento”, entonces el formulario deberá estar limpio y con sus valores iniciales. En caso de que el usuario dio clic al botón de edición de algún evento, entonces el formulario deberá estar inicializado con los datos del evento seleccionado.

El formulario deberá tomar en cuenta las validaciones de la entidad Evento para cada campo y mostrar un mensaje de error en el input correspondiente en caso de que el valor ingreso no cumpla con los requisitos del campo.

El botón para persistir los cambios o los datos realizados en el formulario deberá ser “Crear evento” para cuando es un nuevo evento y “Actualizar evento” para cuando se está editando un evento existente.

Al dar clic en guardar se deberá persistir los datos del formulario en el navegador (Local Storage) y redirigir al usuario a la vista Flujo del Balance la cual deberá reflejar los cambios realizados.

### Adicionales

La aplicicón deje tener un botón para intercambiar entre tema “light” y “dark”. Este valor debe estar presente en el global state y persistir si se refresca la página.

## Puntos Extras

Para la obtención de puntos extra se requerirá que pueda implementar los siguientes requisitos.

### Eventos

Para la entidad eventos se deberá implementar adicionalmente los siguientes campos:

* Adjunto (imagen): opcional, servirá para almacenar una imagen en formato de texto, codificada en base64.
* Periodicidad: el cual puede ser opcional. En caso de existir este deberá incluir los siguientes campos
  + Tipo, el tipo de periodicidad el cual puede ser diario, semanal, mensual, anual o único (para el caso por defecto de que el evento no se repetirá)
  + Cantidad, un número positivo que indica cuando veces se repetirá el evento según su tipo de periodicidad.

### Formulario

Se deberá agregar los inputs necesarios con sus respectivas validaciones para los campos nuevos de Adjunto y Periodicidad. Tomar en cuenta que el campo periodicidad es un objeto que contiene otros campos lo cuales son tipo y cantidad, por lo que deberá investigar sobre como manipular valores anidados en formularios.

### Flujo del Balance

Se deberá realizar la implementación para mostrar el tipo de periodicidad en la lista de eventos de cada mes, tomando que el texto a mostrar debe ser en el formato “N periodicidad”, donde N es el valor de Cantidad y periodicidad el valor de Tipo del campo periodicidad.

Adicionalmente, ahora al momento de dar clic en algún evento de la lista del mes, se deberá mostrar un Modal flotante donde se muestre todos los campos del Evento incluyendo la imagen del evento, en caso de que exista (usar la librería de su elección).