MANUAL DE DESAFÍOS – Módulo III

Programación Backend

CODERHOUSE

¡Bienvenidas y bienvenidos!

Qué bueno encontrarlos/as en este espacio, el cual hemos creado para que puedan conseguir en un mismo lugar, de manera rápida y ágil, todos los desafíos entregables que plantea el módulo II.

A continuación presentamos el sistema de entregas de los cursos de Coder. Luego, en un tablero, podrán ver **las 13 clases establecidas en el módulo II,** marcando con el ícono correspondiente las clases que sí tienen entregables.

De esta forma podrás tener un pantallazo del cronograma de clases y los desafíos que deberás completar.

Sistema de entregas

Desafíos entregables

Tienen una vigencia de 7 días, es decir, a partir de la fecha (de la clase) en la que se lanza el desafío, empezarán a correr 7 días continuos, para que puedas cargar tu desafío en la plataforma.

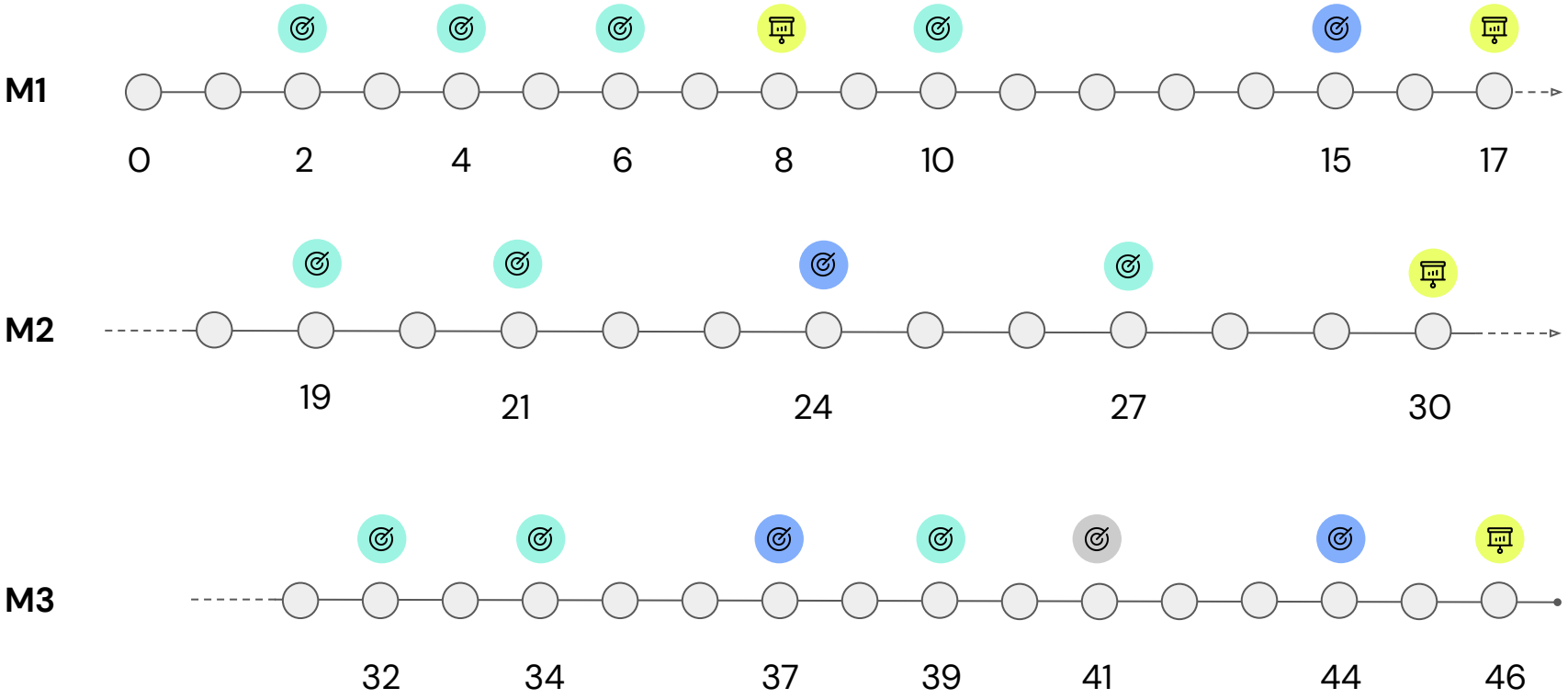
Pre-entrega del PF

También tiene una vigencia o duración de 7 días antes de que el botón de “entrega” se deshabilite. Por este motivo te recomendamos estar al día con todas las actividades planteadas.

Proyecto Final

A diferencia de los anteriores puntos, cuenta con un lapso de 20 días continuos luego de finalizada la última clase (**n° 46**). Posterior a ese tiempo, el botón de la entrega quedará inhabilitado y no será posible entregarlo o recibirlo por otros medios.

GRILLA DE ENTREGAS



MÓDULO III - CLASE 32

Optimización

CODERHOUSE

Mocking y manejo de errores

CODERHOUSE

DESAFÍO ENTREGABLE

Mocking y manejo de errores

Consigna

- ✓ Se aplicará un módulo de mocking y un manejador de errores a tu servidor actual

Formato

- ✓ Link al repositorio de github sin node_modules

Sugerencias

- ✓ Céntrate solo en los errores más comunes
- ✓ Puedes revisar el documento de testing aquí:

Aspectos a incluir

- ✓ Generar un módulo de Mocking para el servidor, con el fin de que, al inicializarse pueda generar y entregar 100 productos con el mismo formato que entregaría una petición de Mongo. Ésto solo debe ocurrir en un endpoint determinado (‘/mockingproducts’)

- ✓ Además, generar un customizador de errores y crear un diccionario para tus errores más comunes al crear un producto, agregarlo al carrito, etc.

CODERHOUSE

MÓDULO III - CLASE 34

Logging y performance

CODERHOUSE

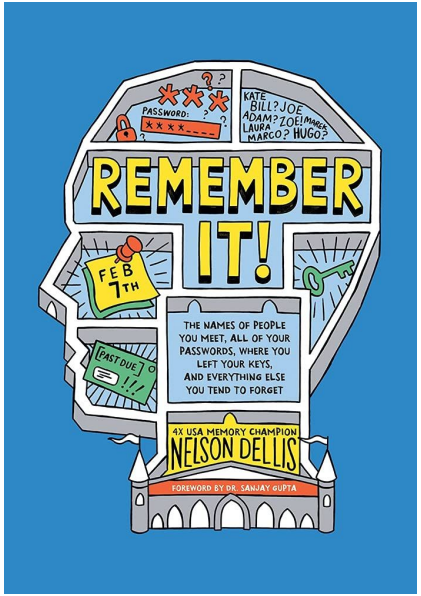
Implementación de logger

CODERHOUSE

Recordemos...

Basado en el entregable de la clase 32

- Realizamos un servicio de mocking
- Realizamos además las primeras modificaciones de modificación de nuestros servidor.



Ahora... Agregaremos un logger para tener mejores registros

CODERHOUSE

DESAFÍO ENTREGABLE

Implementación de logger

Consigna

- ✓ Basado en nuestro proyecto principal, implementar un logger

Aspectos a incluir

- ✓ Primero, definir un sistema de niveles que tenga la siguiente prioridad (de menor a mayor):
  debug, http, info, warning, error, fatal
- ✓ Después implementar un logger para desarrollo y un logger para producción, el logger de desarrollo deberá loggear a partir del nivel debug, sólo en consola
- ✓ Sin embargo, el logger del entorno productivo debería loggear sólo a partir de nivel info.
- ✓ Además, el logger deberá enviar en un transporte de archivos a partir del nivel de error en un nombre “errors.log”
- ✓ Agregar logs de valor alto en los puntos importantes de tu servidor (errores, advertencias, etc) y modificar los **console.log**() habituales que tenemos para que muestren todo a partir de winston.
- ✓ Crear un endpoint /loggerTest que permita probar todos los logs

CODERHOUSE

DESAFÍO ENTREGABLE

Implementación de logger

Formato

- ✓ link al repositorio de Github con el proyecto sin node_modules

Sugerencias

- ✓ Puedes revisar el testing del entregable Aquí
- ✓ La ruta loggerTest es muy importante para que tu entrega pueda ser calificada de manera rápida y eficiente. ¡No olvides colocarla!

CODERHOUSE

MÓDULO III - CLASE 37

Tercera práctica integradora

CODERHOUSE

Práctica de integración sobre tu ecommerce

CODERHOUSE

DESAFÍO COMPLEMENTARIO

Consigna

Con base en el proyecto que venimos desarrollando, toca solidificar algunos procesos

Aspectos a incluir

- ✓ Realizar un sistema de recuperación de contraseña, la cual envíe por medio de un correo un botón que redireccione a una página para restablecer la contraseña (no recuperarla).
  - link del correo debe expirar después de 1 hora de enviado.
  - Si se trata de restablecer la contraseña con la misma contraseña del usuario, debe impedirlo e indicarle que no se puede colocar la misma contraseña
  - Si el link expiró, debe redirigir a una vista que le permita generar nuevamente el correo de restablecimiento, el cual contará con una nueva duración de 1 hora.
- ✓ Establecer un nuevo rol para el schema del usuario llamado “premium” el cual estará habilitado también para crear productos
- ✓ Modificar el schema de producto para contar con un campo “owner”, el cual haga referencia a la persona que creó el producto
  - Si un producto se crea sin owner, se debe colocar por defecto “admin”.
  - El campo owner deberá guardar sólo el correo electrónico (o _id, lo dejamos a tu conveniencia) del usuario que lo haya creado (Sólo podrá recibir usuarios premium)
- ✓ Modificar los permisos de modificación y eliminación de productos para que:
  - Un usuario premium sólo pueda borrar los productos que le pertenecen.
  - El admin pueda borrar cualquier producto, aún si es de un owner.

DESAFÍO COMPLEMENTARIO

Aspectos a incluir

- ✓ Además, modificar la lógica de carrito para que un usuario premium NO pueda agregar a su carrito un producto que le pertenece
- ✓ Implementar una nueva ruta en el router de api/users, la cual será **/api/users/premium/:uid** la cual permitirá cambiar el rol de un usuario, de “user” a “premium” y viceversa.

Formato

- ✓ Link al repositorio de GitHub con el proyecto completo (No incluir node_modules).

Sugerencias

- ✓ Te recomendamos testear muy bien todas las políticas de acceso. ¡Son la parte fuerte de este entregable!

MÓDULO III - CLASE 39

Documentación de API

Documentar API

CODERHOUSE

DESAFÍO ENTREGABLE

Documentar API

Consigna

- ✓ Realizar la configuración necesaria para tener documentado tu proyecto final a partir de Swagger.

Aspectos a incluir

- ✓ Se debe tener documentado el módulo de productos.
- ✓ Se debe tener documentado el módulo de carrito
- ✓ No realizar documentación de sesiones

Formato

- ✓ Link al repositorio de Github sin node_modules

Sugerencias

- ✓ Recuerda que es un proceso de documentación, ¡Hay que ser lo más claros posibles!

CODERHOUSE

MÓDULO III - CLASE 41

Testing avanzado

CODERHOUSE

Módulos de Testing para proyecto final

CODERHOUSE

DESAFÍO COMPLEMENTARIO

Módulos de testing para proyecto final

Consigna

- ✓ Realizar módulos de testing para tu proyecto principal, utilizando los módulos de mocha + chai + supertest

Aspectos a incluir

- ✓ Se deben incluir por lo menos 3 tests desarrollados para
  - Router de products.
  - Router de carts.
  - Router de sessions.
- ✓ NO desarrollar únicamente tests de status, la idea es trabajar lo mejor desarrollado posible las validaciones de testing

Formato

- ✓ link del repositorio en github sin node_modules

Sugerencias

- ✓ Ya que el testing lo desarrollarás tú, no hay una guía de test por leer. ¡Aplica tu mayor creatividad en tus pruebas!

MÓDULO III - CLASE 44

Cuarta práctica integradora

CODERHOUSE

Práctica de integración sobre tu ecommerce

CODERHOUSE

DESAFÍO COMPLEMENTARIO

Consigna

Con base en el proyecto que venimos desarrollando, toca solidificar algunos procesos

Aspectos a incluir

- ✓ Mover la ruta suelta ***/api/users/premium/:uid*** a un router específico para usuarios en /api/users/

- ✓ Modificar el modelo de User para que cuente con una nueva propiedad “documents” el cual será un array que contenga los objetos con las siguientes propiedades
  - name: String (Nombre del documento).
  - reference: String (link al documento).

  No es necesario crear un nuevo modelo de Mongoose para éste.
- ✓ Además, agregar una propiedad al usuario llamada “last_connection”, la cual deberá modificarse cada vez que el usuario realice un proceso de login y logout

DESAFÍO COMPLEMENTARIO

Aspectos a incluir

- ✓ Crear un endpoint en el router de usuarios ***api/users/:uid/documents*** con el método POST que permita subir uno o múltiples archivos. Utilizar el middleware de Multer para poder recibir los documentos que se carguen y actualizar en el usuario su status para hacer saber que ya subió algún documento en particular.

- ✓ El middleware de multer deberá estar modificado para que pueda guardar en diferentes carpetas los diferentes archivos que se suban.
  - Si se sube una imagen de perfil, deberá guardarlo en una carpeta ***profiles***, en caso de recibir la imagen de un producto, deberá guardarlo en una carpeta ***products***, mientras que ahora al cargar un documento, multer los guardará en una carpeta ***documents***.
- ✓ Modificar el endpoint ***/api/users/premium/:uid*** para que sólo actualice al usuario a premium si ya ha cargado los siguientes documentos:
  - Identificación, Comprobante de domicilio, Comprobante de estado de cuenta

DESAFÍO COMPLEMENTARIO

Aspectos a incluir

En caso de llamar al endpoint, si no se ha terminado de cargar la documentación, devolver un error indicando que el usuario no ha terminado de procesar su documentación.
(Sólo si quiere pasar de user a premium, no al revés)

Formato

- ✓ Link al repositorio de GitHub con el proyecto completo (no incluir node_modules).

Sugerencias

- ✓ Corrobora que los usuarios que hayan pasado a premium tengan mayores privilegios de acceso que un usuario normal.

MÓDULO III - CLASE 46

Pasarelas de pago

CODERHOUSE

Backend de una aplicación ecommerce

CODERHOUSE

Backend de una aplicación ecommerce

Desde el router de /api/users, crear tres rutas:

- ✓ GET / deberá obtener todos los usuarios, éste sólo debe devolver los datos principales como nombre, correo, tipo de cuenta (rol)

- ✓ DELETE / deberá limpiar a todos los usuarios que no hayan tenido conexión en los últimos 2 días. (puedes hacer pruebas con los últimos 30 minutos, por ejemplo). Deberá enviarse un correo indicando al usuario que su cuenta ha sido eliminada por inactividad

Crear una vista para poder visualizar, modificar el rol y eliminar un usuario. Esta vista únicamente será accesible para el administrador del ecommerce

ENTREGA DEL PROYECTO FINAL

Backend de una aplicación ecommerce

Modificar el endpoint que elimina productos, para que, en caso de que el producto pertenezca a un usuario premium, le envíe un correo indicándole que el producto fue eliminado.

Finalizar las vistas pendientes para la realización de flujo completo de compra. NO ES NECESARIO tener una estructura específica de vistas, sólo las que tú consideres necesarias para poder llevar a cabo el proceso de compra.

No es necesario desarrollar vistas para módulos que no influyan en el proceso de compra (Como vistas de usuarios premium para crear productos, o vistas de panel de admin para updates de productos, etc)

Realizar el despliegue de tu aplicativo en la plataforma de tu elección (Preferentemente Railway.app, pues es la abarcada en el curso) y corroborar que se puede llevar a cabo un proceso de compra completo.

Última entrega

Objetivos generales

- ✓ Completar el proyecto final

Objetivos específicos

- ✓ Conseguir una experiencia de compra completa
- ✓ Cerrar detalles administrativos con los roles.

Formato

- ✓ Link al repositorio sin node_modules
- ✓ Link del proyecto desplegado..

Sugerencias

- ✓ Presta especial atención al template del Proyecto final. ¡Es crucial para alcanzar la nota que esperas!
- ✓ Debido a la complejidad de frontend requerida para poder aplicar una pasarela de pago, el PF no evalúa la pasarela de pago.

¡Éxitos!

#DemocratizandoLaEducación