MANUAL DE DESAFÍOS – Módulo II

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.

CODERHOUSE

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.

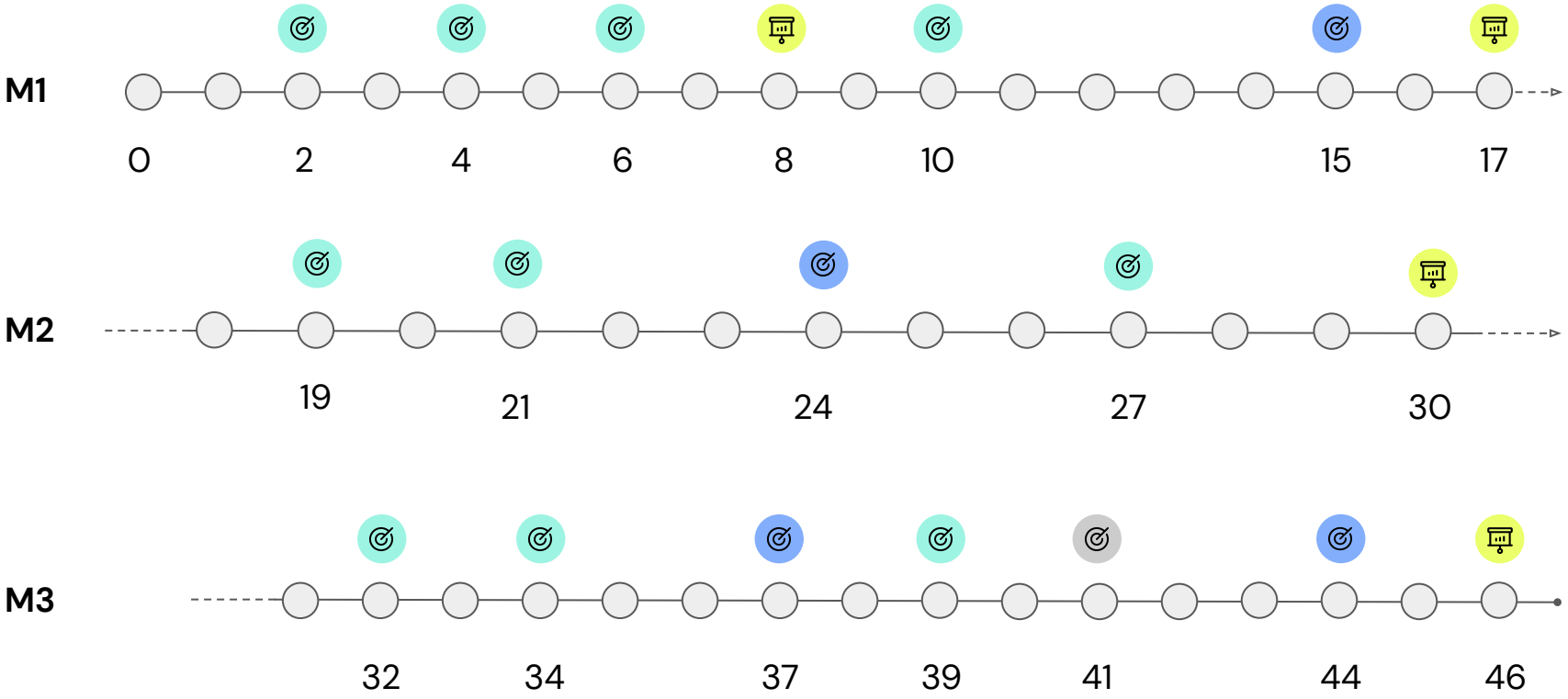
Pre-entrega del PF

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° 47**). 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 II - CLASE 19

Cookies, Sessions & Storage II

CODERHOUSE

Implementación de login.

CODERHOUSE

DESAFÍO ENTREGABLE

Implementación de login

Consigna

- ✓ Ajustar nuestro servidor principal para trabajar con un sistema de login.

Aspectos a incluir

- ✓ Deberá contar con todas las vistas realizadas en el hands on lab, así también como las rutas de router para procesar el registro y el login.
- ✓ En lugar de realizar una redirección a la vista “/profile”, realizar la redirección directamente a la vista de productos.
- ✓ Agregar a la vista de productos un mensaje de bienvenida con los datos del usuario
- ✓ Agregar un sistema de roles, de manera que si colocamos en el login como correo adminCoder@coder.com, y una contraseña predefinida por ti, el usuario de la sesión además tenga un campo `rol: “admin”`
- ✓ Todos los usuarios que no sean admin deberán contar con un rol “usuario”.

MÓDULO II - CLASE 21

Estrategia de autenticación por terceros + JWT

CODERHOUSE

Refactor a nuestro login

CODERHOUSE

DESAFÍO ENTREGABLE

Refactor a nuestro login

Consigna

- ✓ Con base en el login de nuestro entregable anterior, refactorizar para incluir los nuevos conceptos.

Aspectos a incluir

- ✓ Se deberá contar con un hasheo de contraseña utilizando bcrypt
- ✓ Se deberá contar con una implementación de passport, tanto para register como para login.
- ✓ Implementar el método de autenticación de Github a la vista de login.

Formato

- ✓ Link al repositorio de Github con el proyecto solicitado.

Sugerencias

- ✓ Si deseas utilizar algún método de autenticación diferente a github, deberás hablar con tu tutor sobre ello y llegar a algún acuerdo con las private keys de la aplicación

CODERHOUSE

MÓDULO II - CLASE 24

Segunda práctica integradora

CODERHOUSE

Práctica de integración sobre tu ecommerce

CODERHOUSE

DESAFIO COMPLEMENTARIO

Consigna

Continuar sobre el proyecto que has trabajado para tu ecommerce y configurar los siguientes elementos:

Aspectos a incluir

- ✓ Crear un modelo User el cual contará con los campos:
  - first_name:String,
  - last_name:String,
  - email:String (único)
  - age:Number,
  - password:String(Hash)
  - cart:Id con referencia a Carts
  - role:String(default:'user')
- ✓ Desarrollar las estrategias de Passport para que funcionen con este modelo de usuarios
- ✓ Modificar el sistema de login del usuario para poder trabajar con session o con jwt (a tu elección).
- ✓ (Sólo para jwt) desarrollar una estrategia "current" para extraer la cookie que contiene el token para obtener el usuario asociado a dicho token, en caso de tener el token, devolver al usuario asociado al token, caso contrario devolver un error de passport, utilizar un extractor de cookie.
- ✓ Agregar al router /api/sessions/ la ruta /current, la cual utilizará el modelo de sesión que estés utilizando, para poder devolver en una respuesta el usuario actual.

DESAFIO COMPLEMENTARIO

Formato

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

Sugerencias

- ✓ Te recomendamos trabajar con el modelo de sesión con el cual te sientas más cómodo (sessions / jwt)

MÓDULO II - CLASE 25

Arquitectura del servidor: diseño

CODERHOUSE

Reestructura de nuestro servidor

CODERHOUSE

DESAFÍO ENTREGABLE

Reestructura de nuestro servidor

Consigna

- ✓ Con base en las clases previamente vistas, realizar los cambios necesarios en tu proyecto para que se base en un modelo de capas.

Aspectos a incluir

- ✓ El proyecto debe contar con capas de routing, controlador, dao, con nuestras vistas bien separadas y con las responsabilidades correctamente delegadas.

Aspectos a incluir

- ✓ Además, mover del proyecto todas las partes importantes y comprometedoras en un archivo .env para poder leerlo bajo variables de entorno en un archivo config.js

Formato

- ✓ Link al repositorio de Github para poder clonar, además. adjunto el archivo .env para poder relacionar las variables de entorno.

CODERHOUSE

MÓDULO II - CLASE 30

Tercera pre entrega de tu Proyecto Final

CODERHOUSE

Tercera entrega de tu Proyecto final

Se profundizará sobre los roles de los usuarios, las autorizaciones y sobre la lógica de compra.

CODERHOUSE

Mejorando la arquitectura del servidor

Objetivos generales

- ✓ Profesionalizar el servidor

Objetivos específicos

- ✓ Aplicar una arquitectura profesional para nuestro servidor
- ✓ Aplicar prácticas como patrones de diseño, mailing, variables de entorno. etc.

Se debe entregar

- ✓ Modificar nuestra capa de persistencia para aplicar los conceptos de Factory (opcional), DAO y DTO.

Se debe entregar

- ✓ El DAO seleccionado (por un parámetro en línea de comandos como lo hicimos anteriormente) será devuelto por una Factory para que la capa de negocio opere con él. (Factory puede ser opcional)
- ✓ Implementar el patrón Repository para trabajar con el DAO en la lógica de negocio.
- ✓ Modificar la ruta /current Para evitar enviar información sensible, enviar un DTO del usuario sólo con la información necesaria.

Mejorando la arquitectura del servidor

Se debe entregar

✓ Realizar un middleware que pueda trabajar en conjunto con la estrategia “current” para hacer un sistema de autorización y delimitar el acceso a dichos endpoints:

- Sólo el administrador puede crear, actualizar y eliminar productos.
- Sólo el usuario puede enviar mensajes al chat.
- Sólo el usuario puede agregar productos a su carrito.

Se debe entregar

✓ **Crear un modelo Ticket el cual contará con todas las formalizaciones de la compra. Éste contará con los campos**

- **Id (autogenerado por mongo)**
- **code: String debe autogenerarse y ser único**
- **purchase_datetime: Deberá guardar la fecha y hora exacta en la cual se formalizó la compra (básicamente es un created_at)**
- **amount: Number, total de la compra.**
- **purchaser: String, contendrá el correo del usuario asociado al carrito.**

Mejorando la arquitectura del servidor

Se debe entregar

✓ Implementar, en el router de carts, la ruta /:cid/purchase, la cual permitirá finalizar el proceso de compra de dicho carrito.

- La compra debe corroborar el stock del producto al momento de finalizarse
  - Si el producto tiene suficiente stock para la cantidad indicada en el producto del carrito, entonces restarlo del stock del producto y continuar.
  - Si el producto no tiene suficiente stock para la cantidad indicada en el producto del carrito, entonces no agregar el producto al proceso de compra.

Se debe entregar

- Al final, utilizar el servicio de Tickets para poder generar un ticket con los datos de la compra.
- En caso de existir una compra no completada, devolver el arreglo con los ids de los productos que no pudieron procesarse.

Una vez finalizada la compra, el carrito asociado al usuario que compró deberá contener sólo los productos que no pudieron comprarse. Es decir, se filtran los que sí se compraron y se quedan aquellos que no tenían disponibilidad.

Mejorando la arquitectura del servidor

Formato

- ✓ Link al repositorio de Github con el proyecto (sin node_modules)
- ✓ Además, archivo .env para poder correr el proyecto.

Sugerencias

- ✓ Te recomendamos ver el vídeo explicativo disponible en la carpeta de clase

Muchas gracias.

#DemocratizandoLaEducación