Desarrollarás el backend de una aplicación de **e-commerce** para poder vender productos de un rubro a elección.

- Contendrá las rutas necesarias que permitan listar los productos existentes, ingresar productos nuevos, borrar y modificar sus detalles, así como interactuar con el carrito de compras.
- Se implementará una API RESTful con los verbos get, post, put y delete para cumplir con todas las acciones necesarias.
- Debe brindar al frontend un mecanismo de ingreso autorizado al sistema basado en JWT (Json Web Token).
- Los productos ingresados se almacenarán en una base de datos MongoDB.
- El usuario podrá registrar sus credenciales de acceso (email y password) para luego poder ingresar a su cuenta. Estas credenciales serán guardadas en la base de datos MongoDB encriptando la contraseña.
- El cliente tendrá una sesión activa de usuario con tiempo de expiración configurable.
- (*OPCIONAL*) Implementarás un canal de chat basado en websockets, el cual permita atender las consultas del cliente.
- La arquitectura del servidor estará basada en capas (MVC)
- El servidor podrá tomar configuraciones desde un archivo externo (development.env y production.env si usan local, si usan heroku setear en el Config Variables https://devcenter.heroku.com/articles/config-vars).
- (*OPCIONAL*) Dispondrá de una vista creada con pug, que permita ver la configuración del servidor.
- Se enviará un mail a una casilla configurable, por cada registro nuevo de usuario y con cada orden de compra generada.
- En caso de detectar algún error, el servidor enviará una vista implementada con ejs, que contenga el id y el detalle completo (Configurar para que el email sea enviado tambien a emanuelbalcazar13@gmail.com)

Te recomendamos incluir:
● Node.js ● MongoDB ● Passport JWT ● Mongoose ● Bcrypt ● Websocket ● Dotenv ● Handlebars, Pug, Ejs ● Nodemailer

Los requisitos base serán parte de los criterios de evaluación para aprobar el proyecto.
● **Inicio:** Al momento de requerir la ruta base **‘/’** ○ Permitir un menú de ingreso al sistema con email y password así como también la posibilidad de registro de un nuevo usuario. ○ El menú de registro consta del nombre completo del cliente, email y campo de password duplicado para verificar coincidencia. ○ Si un usuario se loguea exitosamente o está en sesión activa, la ruta ‘/’ hará una re dirección a la ruta de **/productos** ○ La ruta **/productos** devolverá el listado de todos los productos disponibles para la compra. ○ (*OPCIONAL*) La ruta **/productos/:categoria** devolverá los productos por la categoría requerida. ○ Los ítems podrán ser agregados al carrito de compras y listados a través de la ruta **/carrito**. ○ Se podrán modificar y borrar por su id a través de la ruta **/carrito/:id**. ○ Mínimamente, implementar la API REST, opcionalmente incluir las pantallas.

- **Flow:** Se puede solicitar un producto específico con la ruta **/productos/:id**, donde **id** es el id del item generado por MongoDB y devolver la descripción del producto (foto, precio, selector de cantidad).
- Si se ingresa a **/productos/:id** y el producto no existe en MongoDB, debemos responder un mensaje adecuado que indique algo relacionado a que el producto no existe.

- **MongoDB**:
    - Implementar al menos estas colecciones:
        - **usuarios:** clientes registrados
        - **productos:** catálogo completo
            - (*OPCIONAL*) Link para foto (puede almacenarse de modo estático en la página en una subruta **/images/:productoid**)
            - Precio unitario
            - Descripción
            - (*OPCIONAL*) Categoría

        - **(OPCIONAL) mensajes:** chat del usuario (preguntas y respuestas)
            - Email: del usuario que pregunta o al que se responde
            - Tipo (‘usuario’ para preguntas ó ‘sistema’ para respuestas)
            - Fecha y hora
            - Cuerpo del mensaje

        - **carrito:** orden temporal de compra
            - Email
            - Fecha y hora
            - Items con sus cantidades
            - Dirección de entrega

        - **ordenes:** las órdenes generadas, que deben incluir los productos, descripciones y los precios **al momento de la compra.**
            - La orden se envía cuando se confirma la compra.
            - Referencia al ID del carrito
            - Número de orden: Se extrae de la cantidad de órdenes almacenadas

- Fecha y hora
- estado (por defecto en ‘generada’ -> ‘enviada’)
- Email de quién realizó la orden
- Agregar la ruta */orden/confirmar/:idCarrito*, buscar el carrito y enviar por email la información del mismo (productos que están en el carrito y fecha y hora)

- Finalizada la orden, enviar un mail a la dirección de mi cuenta con los detalles de la orden.
- (*OPCIONAL*) Se dispondrá de un archivo de configuración externo con opciones para desarrollo y otras para producción, que serán visualizadas a través de una vista construida con handlebars. Como parámetros de configuración estará el puerto de escucha del servidor, la url de la base de datos, el mail que recibirá notificaciones del backend, tiempo de expiración de sesión y los que sea necesario incluir.
- (*OPCIONAL*) Vamos a contar con un canal de chat general donde el usuario enviará los mensajes en la ruta **/chat** y en **/chat/:email** podrá ver sólo los suyos. Se utilizará la colección **mensajes** en MongoDB. La tecnología de comunicación a utilizar será Websockets. El servidor implementará una vista, utilizando handlebars, para visualizar todos los mensajes y poder responder individualmente a ellos, eligiendo el email de respuesta.

Los requisitos extra *pro-coders* no se incluyen en los criterios de evaluación.

Los requisitos extra son funcionalidades opcionales que no se incluyen en los criterios de evaluación, pero si te falta diversión y quieres agregar valor a tu proyecto... ¡bajo la única **condición** de que **lo que incluyas debe funcionar!**

- **auth/login:** Implementar alguna de las estrategias de autenticación disponibles en passport para permitir el login con Facebook y Gmail
- **Custom item**: Posibilidad de agregar características seleccionables al producto (ej. talla, color, etc). La customización no debería modificar el precio. Las selecciones serán detalladas en el checkout. Por ejemplo: **1 x camisa (roja) $200** y **2 x camisa (verde) $400**.
- **Stock check**: Validar stock al momento de intentar generar la **orden**.

- **Mis órdenes:** El usuario podrá visualizar todas las órdenes que realizó a través de la ruta **/orden**.

No es necesario ni recomendado.

- Crear un administrador de stock, dado que puede escaparse del scope y requerir bastante trabajo extra. Podremos gestionar el stock desde la base MongoDB.
- Implementar el FrontEnd salvo que así sea deseado por parte del estudiante.