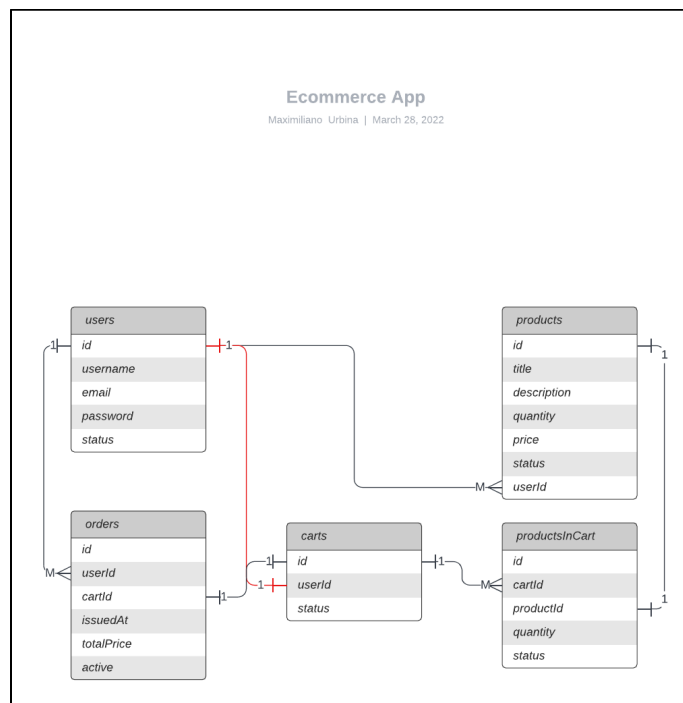


## Requerimientos para proyecto final: Ecommerce

Dentro de este proyecto se cubrirán los siguientes temas:

- Relaciones con otras tablas
- JWT
- Middlewares
- Express-validator
- Heroku

Se deben considerar los siguientes modelos y sus relaciones:



Se deben considerar los siguientes endpoints:

/api/v1/users		
HTTP Verb	Route	Description
POST	/	Crear usuario (enviar username, email, y password por req.body)
POST	/login	Iniciar sesión (enviar email y password por req.body)
GET	/me	Obtener los productos que el usuario ha creado
PATCH	/:id	Actualizar perfil de usuario (solo username y email)
DELETE	/:id	Deshabilitar cuenta de usuario
GET	/orders	Obtener todas las compras hechas por el usuario
GET	/orders/:id	Obtener detalles de una sola orden dado un ID

- Todas las rutas, excepto para crear usuario e iniciar sesión, se deben proteger por un medio de autenticación, es decir, por JWT.

- Se debe usar express-validator para el endpoint de crear usuarios.
- El endpoint **/me**, debe buscar los productos del usuario en sesión (del token que se envió), extraer el id del token y usarlo para buscar dichos productos.
- Los métodos **PATCH** y **DELETE** deben estar protegidos para que únicamente el dueño de la cuenta a modificar pueda realizar dichas acciones.
- Para los endpoints **/orders**, se debe incluir el carrito al que pertenece la compra y los productos que se compraron (status *'purchased'*)

<b>/api/v1/products</b>		
HTTP Verb	Route	Description
POST	/	Crear un producto (enviar title, description, price (INT), y quantity por req.body, adjuntar el userId de la sesión)
GET	/	Obtener todos los productos disponibles
GET	/:id	Obtener producto por id
PATCH	/:id	Actualizar producto (title, description, price, quantity) SOLO EL USUARIO QUIEN CREO EL PRODUCTO
DELETE	/:id	Deshabilitar producto. SOLO EL USUARIO QUIEN CREO EL PRODUCTO

- Todas las rutas deben estar protegidas por un método de autenticación.
- El endpoint para crear productos, debe estar protegido con express-validator.
- Los métodos **PATCH** y **DELETE** deben estar protegidos para que únicamente el dueño del producto a modificar pueda realizar dichas acciones.

<b>/api/v1/cart</b>		
HTTP Verb	Route	Description
POST	/add-product	Agregar un producto al carrito del usuario (enviar productId y quantity por req.body)
PATCH	/update-cart	Actualizar algún producto del carrito (incrementar o decrementar cantidad { productId, newQty })
DELETE	/:productId	Remover producto del carrito
POST	/purchase	Realizar compra de todos los productos en el carrito con status <b>active</b>

- Todas las rutas deben estar protegidas por un método de autenticación.
- Para el endpoint **/add-product** se debe agregar el producto seleccionado al carrito:
  - Se debe buscar si el usuario tiene un carrito en status **active**, en caso de que no, crear uno.
  - Validar que el producto a agregar no exceda la cantidad disponible de dicho producto (si el producto tiene 5 items, el usuario no puede poner 6 items en el carrito)
  - Validar si el producto ya existe en el carrito
    - Si existe, mandar error al usuario de que este producto ya fue a agregado
    - Si el producto ya se encuentra, pero con status **removed**, modificar el status a **active** y su respectiva cantidad
- Para el endpoint **/update-cart**, se debe buscar el carrito del usuario usando los datos de su token.

- Validar que el producto a agregar no exceda la cantidad disponible de dicho producto (si el producto tiene 5 items, el usuario no puede poner 6 items en el carrito)
- Si el usuario envía como *newQty* 0, marcar el producto con status **removed**, si dicho producto lo vuelve a agregar, modificar status a **active**.
- Para el endpoint **/purchase**, se debe realizar lo siguiente:
  - Buscar el carrito del usuario con status **active**, e incluir todos los productos del carrito con status **active**
  - Recorrer la lista de productos en el carrito
    - Restar la cantidad seleccionada al producto seleccionado (si el producto tiene 6 items y tenemos 3 en el carrito, hacemos la resta y el resultado la actualizamos en el producto dado su id)
    - Calcular el precio total de todo el carrito, recuerda multiplicar la cantidad por el precio del producto en donde sea necesario. (este sera usado para el modelo **orders**)
    - Marcar los productos del carrito con status **purchased**
  - Marcar el carrito con status **purchased**
  - Crear registro en el modelo **orders**