



Desarrollo de Sistemas Web (Back End)

Primer Parcial

Caso 5: Aplicación de Comercio

Grupo Undefined

Integrantes:

- Ciro Villasanti
- Daniela Carballo

Comisión: 2°B

Contenido

- Consigna 3
- Objetivos específicos: 3
- Caso 5: Aplicación de Comercio..... 5
 - Introducción..... 5
 - Arquitectura del Sistema..... 5
 - Tecnologías..... 5
 - Funcionalidades principales..... 6
 - API Endpoints..... 6
 - Diagrama de alto nivel 8
- Bibliografía 8

Consigna

Este proyecto abarca los contenidos vistos hasta el primer parcial de la materia de Desarrollo Web Backend de la carrera de programación. Podrá realizarse en grupos (de 3 a 5 personas de la misma comisión). Se enfocará en la creación de una aplicación web utilizando Node.js y Express, con una estructura de archivos, carpetas coherente al desarrollo y una base de datos.

Objetivos específicos:

1. Desarrollar una aplicación web utilizando Node.js y Express.
2. Integrar una base de datos JSON o MONGO
3. Aplicar conceptos de asincronía y manejo de promesas.
4. Módulos
5. Utilizar el motor de plantillas Pug que permite crear HTML (que sea algo muy simple)
6. Implementar un sistema de rutas dinámicas y middleware.
7. Probarla con Postman o Thunder Client (Capturar pantallas y pegarlas en la documentación)
8. Seguir buenas prácticas de desarrollo, una estructura de carpetas ordenada, incluyendo la utilización de POO.

Objetivos generales:

- Desarrollar software por encargo
- Integrar equipos de proyecto para el desarrollo
- Liderar grupos de trabajo y asumir roles especializados.
- Desempeñarse de manera autónoma en el desarrollo de sistemas de baja complejidad.


Propuestas a realizar

Este proyecto abarca los contenidos vistos hasta el primer parcial de la materia de Desarrollo Web Backend de la carrera de programación. Podrá realizarse en grupos (de 3 a 5 personas de la misma comisión). Se enfocará en la creación de una aplicación web utilizando Node.js y Express, con una estructura de archivos, carpetas coherente al desarrollo y una base de datos.

1. Aplicación de Comercio

- **Descripción:** Una plataforma de comercio electrónico para vender productos.
- **Características:**
 - **Catálogo de productos con categorías y filtros:** La aplicación permitirá a los clientes mayoristas navegar por el catálogo de productos, utilizando filtros por categorías para facilitar la búsqueda y selección de artículos.

- **Selección de productos y proceso de pago (simulado):** Los clientes podrán agregar productos al carrito de compras, revisar su selección y proceder al pago de manera segura a través de la plataforma.
- **Registro e inicio de sesión de usuarios:** La plataforma requerirá que los usuarios se registren y accedan de manera simple para poder realizar compras.
- **Panel de administración para gestionar productos, pedidos y usuarios:** Los administradores podrán gestionar el catálogo de productos, revisar y procesar pedidos, y administrar las cuentas de usuarios desde un panel centralizado.
- **Uso de una base de datos para almacenar productos, usuarios y pedidos:** La aplicación utilizará una base de datos para almacenar toda la información relacionada con los productos, clientes, y pedidos, permitiendo un acceso rápido y una gestión eficiente de los datos.

Nombre del Usuario	Nombre del Producto	Fecha	Cantidad	Estado
Juan Pérez	Auriculares	2024-10-01	2	Enviado
Ana Gómez	Laptop	2024-10-02	1	Procesando
Juan Pérez	Escritorio	2024-10-03	1	Entregado
Sofía Morales	Monitor	2024-10-04	1	Cancelado
Luis Martínez	Teclado 	2024-10-05	3	Enviado

Criterios de Evaluación

El proyecto será evaluado en base a los siguientes criterios:

1. **Funcionalidad:** La aplicación debe cumplir con todos los requisitos funcionales establecidos. No extender el proyecto a más de lo solicitado, los módulos a evaluar son del M1 a M9. Se evaluará de acuerdo a los contenidos vistos a la fecha.
2. **Código:** El código debe estar bien estructurado y seguir las buenas prácticas de desarrollo. La aplicación debe ser probada exhaustivamente y todas las pruebas deben estar documentadas.
3. **Documentación:** Deben colocar la bibliografía utilizada (Tutoriales, Libros, Videos). Deben poner las responsabilidades de cada uno de los estudiantes en el desarrollo del mismo. Explicar el funcionamiento de la aplicación. Los roles deben ser especificados que hizo cada uno y grabar un pequeño video explicando cada uno de ellos. Si quieren aprovechar contenido de otra materia en cuanto a documentación, no hay problema.
4. **Entrega:** La documentación del proyecto debe ser clara y completa en un PDF adjunto al trabajo. Deben realizar un video sobre la explicación del código y proyecto, pueden aprovechar a grabarlo en un meet o zoom y adjuntar el link en el documento (Drive, Onedrive, Dropbox, Youtube, etc con los permisos correspondientes para visualizarlo. No es necesario una edición. En caso de ser necesario se contactará con los

estudiantes a fin de evacuar dudas. **Cada estudiante debe ser responsable de la entrega individual del trabajo con su usuario del campus.**

Primer Parcial

Caso 5: Aplicación de Comercio

Introducción

Este documento presenta una descripción detallada de nuestra aplicación de comercio desarrollada utilizando Node.js, Express y TypeScript.

El código fuente de la misma se encuentra en <https://github.com/carballod/app-de-comercio>

Arquitectura del Sistema

La aplicación sigue una arquitectura (MVC) con capas adicionales para mejorar la separación de responsabilidades. Esta estructura permite una clara distinción entre la lógica de negocio, la presentación de datos y el manejo de solicitudes HTTP.

La capa de Modelo define las estructuras de datos y las relaciones entre ellas. Utilizamos interfaces TypeScript para definir claramente la forma de nuestros objetos de datos, como productos, órdenes y usuarios. Los modelos se encuentran en el directorio 'src/application/models'.

La capa de Vista está implementada utilizando el motor de plantillas Pug. Esto nos permite generar HTML dinámicamente y mantener una separación clara entre la lógica de presentación y la lógica de negocio. Las vistas se encuentran en el directorio 'src/application/views' y están organizadas por funcionalidad (productos, órdenes, usuarios).

La capa de Controlador maneja las solicitudes HTTP y coordina la interacción entre el Modelo y la Vista. Los controladores se encuentran en 'src/application/controllers' y están divididos por dominio (productos, órdenes, usuarios, vistas).

Además de estas capas tradicionales, hemos introducido una **capa de Servicios** que encapsula la lógica de negocio principal.

Tecnologías

Nuestra aplicación utiliza el stack propuesto en la consigna y algunas tecnologías adicionales que nos facilitan la entrega de un producto más robusto y escalable. Estas son:

TypeScript: se utiliza en todo el proyecto para agregar tipado estático a nuestro código JavaScript. Esto mejora significativamente la mantenibilidad del código y nos ayuda a detectar errores en tiempo de compilación.

La autenticación y el manejo de sesiones se implementan utilizando **JSON Web Tokens (JWT)**. Esto nos permite mantener un sistema de autenticación stateless y escalable.

Bootstrap se utiliza para el diseño frontend, proporcionándonos un conjunto de componentes y estilos predefinidos que aceleran el desarrollo de la interfaz de usuario.

Funcionalidades principales

Gestión de Usuarios

Nuestra aplicación permite el registro e inicio de sesión de usuarios. Implementamos la distinción entre usuarios normales y administradores, cada uno con sus propios privilegios y vistas. Los administradores tienen acceso a funcionalidades adicionales como la gestión de todos los usuarios, productos y órdenes.

Catálogo de Productos

Los usuarios pueden ver un listado completo de productos, buscar productos específicos y filtrarlos por categorías o palabra clave y ordenarlos por precios. Cada producto tiene su propia página de detalles. Los administradores pueden agregar, editar y eliminar productos.

Carrito de Compras

Implementamos un carrito de compras del lado del cliente utilizando JavaScript. Los usuarios pueden agregar productos al carrito y ver un resumen de su pedido antes de proceder al pago.

Gestión de Órdenes

Los usuarios pueden crear órdenes a partir de su carrito de compras. Una vez creada una orden, los usuarios pueden ver el estado de sus órdenes y, si aún están pendientes, tienen la opción de cancelarlas. Los administradores pueden ver todas las órdenes, editarlas y cambiar su estado.

API Endpoints

A continuación, se presenta la documentación técnica para consumir los distintos endpoints por módulo:

Autenticación

POST /api/user/login: Inicia sesión de usuario. Requiere body con username y password. Si es exitoso guarda el token en cookie y lo devuelve en el response.

POST /api/user/logout: Cierra la sesión del usuario actual.

POST /api/user/register: Registra un nuevo usuario. Requiere username, email y password en el body de la solicitud.

POST /api/user/reset-password: Permite modificar la contraseña de un usuario. Requiere username, email, código de verificación y password nueva en el body de la solicitud.

Productos

GET /api/product: Obtiene la lista de productos. Acepta parámetros de consulta para filtrado y paginación. Ejemplo:

http://localhost:3000/products?category%5B%5D=Electronics&sortBy=price_desc&keyword=

GET /api/product/:id: Obtiene los detalles de un producto específico.

POST /api/product: Crea un nuevo producto (solo para administradores).

PUT /api/product/:id: Actualiza un producto existente (solo para administradores).

DELETE /api/product/:id: Elimina un producto (solo para administradores).

Órdenes

GET /api/order: Obtiene las órdenes del usuario actual (todas las órdenes para administradores).

GET /api/order/:id: Obtiene los detalles de una orden específica.

POST /api/order: Crea una nueva orden.

PUT /api/order/:id: Actualiza el estado de una orden (solo para administradores).

POST /api/order/:id/cancel: Cancela una orden.

Usuarios (solo para administradores):

GET /api/users: Obtiene la lista de todos los usuarios.

GET /api/users/:id: Obtiene los detalles de un usuario específico.

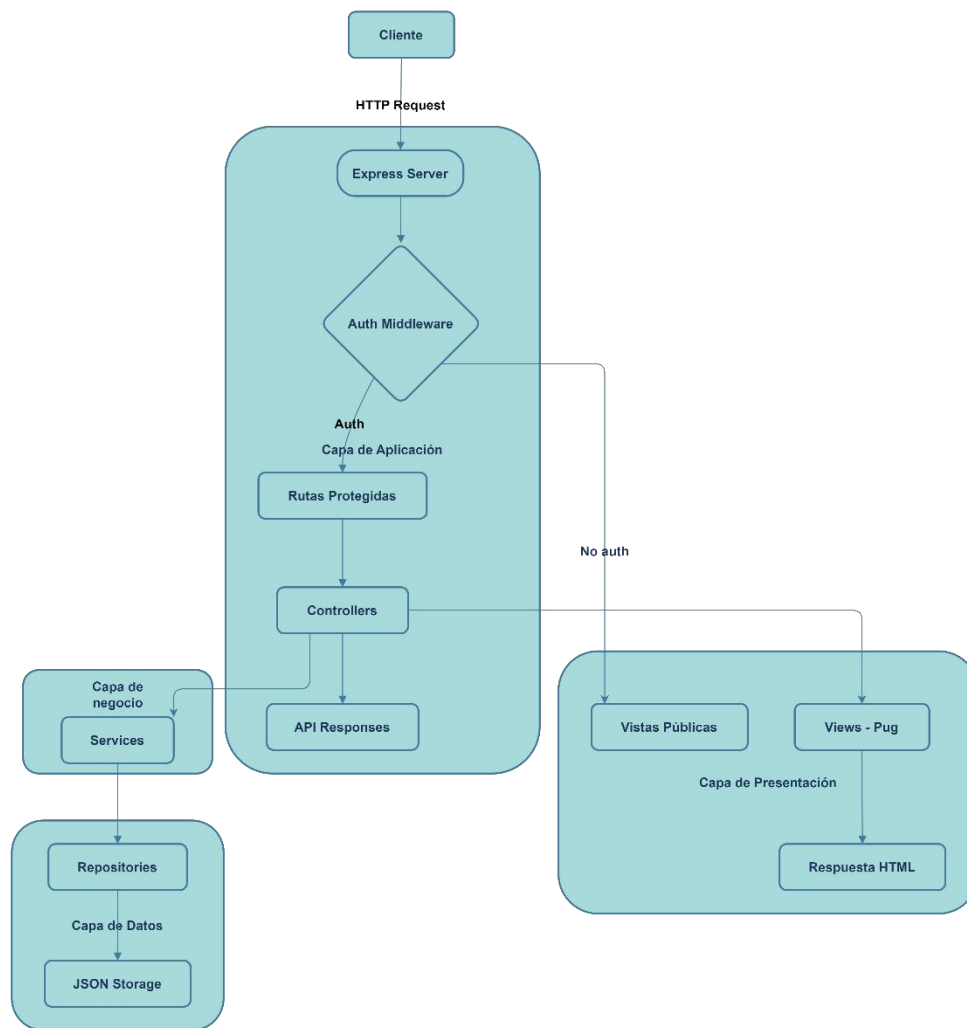
PUT /api/users/:id: Actualiza la información de un usuario.

DELETE /api/users/:id: Elimina un usuario.

Documentación Postman y ejemplos:

<https://documenter.getpostman.com/view/13303225/2sAXxY2TPU>

Diagrama de alto nivel



Bibliografía

Node.js Documentation. (2023). <https://nodejs.org/en/docs/>

Express.js Documentation. (2023). <https://expressjs.com/>

TypeScript Documentation. (2023). <https://www.typescriptlang.org/docs/>

Pug Documentation. (2023). <https://pugjs.org/api/getting-started.html>

JWT.io. (2023). JSON Web Tokens Introduction. <https://jwt.io/introduction>

Bootstrap Documentation. (2023). <https://getbootstrap.com/docs/>

MDN Web Docs. (2023). JavaScript. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

Midudev. Tutorial TypeScript con Node.js y Express. ¡Crea tu API REST con tipos estáticos DESDE CERO! (2022) <https://www.youtube.com/watch?v=ZpY5KdGQvwI>