2.6. Công nghệ sử dụng

Hệ thống quản lý bán giày là một website thương mại điện tử chuyên về các sản phẩm giày dép. Hệ thống cho phép quản trị viên quản lý danh mục, sản phẩm, đơn hàng và theo dõi hoạt động mua bán; đồng thời cung cấp cho người mua hàng một giao diện thân thiện để duyệt sản phẩm, thêm vào giỏ hàng và thanh toán trực tuyến. Mục tiêu của hệ thống là đơn giản hóa quy trình bán hàng cho quản trị viên và mang lại trải nghiệm mua sắm trực tuyến thuận tiện cho khách hàng. Hệ thống được xây dựng với kiến trúc web full-stack sử dụng Node.js cho backend (server), Express cho việc xây dựng API, MongoDB để lưu trữ dữ liệu, và React cho frontend (client-side). Ngoài ra, Pug được sử dụng làm engine template để phát triển giao diện trang quản trị (admin). Việc lựa chọn các công nghệ này nhằm đảm bảo hiệu năng, tính linh hoạt và dễ mở rộng cho ứng dụng web.

2.7. Kiến trúc hệ thống

Mô hình MVC cho backend: Backend tuân theo mô hình Model-View-Controller, tách biệt rõ phần dữ liệu, giao diện và xử lý điều khiển. Cách tiếp cận này giúp mã nguồn dễ bảo trì và phát triển mở rộng, khi Model đảm nhiệm kết nối và thao tác với cơ sở dữ liệu, View (ở đây là các template Pug cho admin) xây dựng giao diện hiển thị, và Controller xử lý các yêu cầu, tương tác giữa Model và View

Việc áp dụng MVC trong ứng dụng Node.js với Express giúp tổ chức code một cách logic và tăng chất lượng codebase

Kiến trúc RESTful API: Backend được thiết kế dưới dạng RESTful API, nghĩa là cung cấp các endpoint (URL) để client (ứng dụng React hoặc bất kỳ client nào) có thể gửi yêu cầu HTTP (GET, POST, PUT, DELETE) và nhận dữ liệu dạng JSON. Các API tuân thủ các quy tắc RESTful về URL

và phương thức HTTP, giúp cho việc giao tiếp thống nhất và dễ hiểu. Ví dụ: /api/products để lấy danh sách sản phẩm (GET) hoặc thêm sản phẩm mới (POST), /api/products/:id để cập nhật (PUT) hoặc xóa (DELETE) một sản phẩm. Kiến trúc RESTful tạo ra một giao diện thống nhất giữa frontend và backend, giúp việc tích hợp trên nhiều nền tảng (web, mobile) được thuận lợi.

Giao tiếp giữa backend và frontend: Ứng dụng React (frontend) sẽ gửi request tới các API Express (backend) qua HTTP (thường sử dụng fetch hoặc Axios), nhận dữ liệu JSON và hiển thị lên giao diện. Backend Express lắng nghe các request tại các route đã định nghĩa, xử lý logic (ví dụ truy vấn cơ sở dữ liệu MongoDB) rồi trả về JSON cho frontend. Với mô hình tách biệt này, frontend và backend hoạt động độc lập, giao tiếp qua JSON API, giúp dễ bảo trì và có thể phát triển song song. Ngoài ra, việc tách biệt còn cho phép triển khai front-end và back-end trên các máy chủ khác nhau nếu cần.

Giao diện quản trị với Pug: Bên cạnh ứng dụng React cho người dùng, hệ thống còn có phần quản trị viên được xây dựng bằng Pug (một template engine cho Node.js). Các trang admin (như trang quản lý sản phẩm, đơn hàng, người dùng) được render phía server bằng Express và Pug. Việc sử dụng Pug cho admin giúp trang quản trị nhẹ và nhanh, vì không cần tải cả ứng dụng React; đồng thời tăng cường bảo mật (admin truy cập qua giao diện nội bộ). Backend Express sẽ có các route như /admin/products, /admin/orders... trả về HTML được render từ Pug template. Mô hình này cho phép chạy đồng thời hai giao diện: client-side rendering (React) cho khách hàng và server-side rendering (Pug) cho admin, nhưng vẫn dùng chung một server Node.js. Việc này tương tự như một số trang web tách biệt phần dashboard quản trị và trang mua sắm của khách.

2.8. API Endpoints của Cline

Backend xây dựng một loạt các API endpoint (RESTful) phục vụ cho cả chức năng người dùng và quản trị. Tất cả các API yêu cầu dữ liệu và phản hồi đều dùng định dạng JSON. Dưới đây là các nhóm API quan trọng trong hệ thống:

API quản lý người dùng: Bao gồm các endpoint cho đăng ký (POST /api/auth/register) để tạo tài khoản mới, đăng nhập (POST /api/auth/login) để người dùng nhận JSON Web Token (JWT) đăng nhập. Sau khi đăng nhập, JWT sẽ được sử dụng để xác thực người dùng cho các request tiếp theo (thông qua header Authorization). Ngoài ra, có API để lấy thông tin người dùng hiện tại (GET /api/auth/me với token hợp lệ) và có thể có API để cập nhật thông tin tài khoản, đổi mật khẩu, v.v. Việc sử dụng JWT tuân theo chuẩn JSON Web Token – một chuẩn định dạng token dưới dạng JSON dùng để truyền thông tin xác thực an toàn giữa client và server

API CRUD sản phẩm: Cho phép quản trị viên quản lý danh sách sản phẩm. Bao gồm: tạo sản phẩm mới (POST /api/products), xem danh sách hoặc chi tiết sản phẩm (GET /api/products hoặc GET /api/products/:id), cập nhật thông tin sản phẩm (PUT /api/products/:id), xóa sản phẩm (DELETE /api/products/:id). Các API này thường được bảo vệ chỉ cho admin. Ngoài ra, phía người dùng có thể có các endpoint để tìm kiếm, lọc sản phẩm theo danh mục, tên, giá... (ví dụ GET /api/products?category=...). Dữ liệu phản hồi bao gồm thông tin sản phẩm và có thể kèm theo thông tin danh mục (populate) hoặc số đánh giá, xếp hạng trung bình.

API giỏ hàng (Cart): Cho phép người dùng thêm sản phẩm vào giỏ, cập nhật số lượng hoặc xóa sản phẩm khỏi giỏ. Ví dụ: POST /api/cart để thêm sản phẩm (payload bao gồm product_id và quantity), PUT /api/cart/:itemId để cập nhật số lượng một mục trong giỏ, DELETE

/api/cart/:itemId để xóa một mục. Thông tin giỏ hàng có thể được lưu tạm trong database (collection Cart) hoặc trong session. Nếu người dùng đã đăng nhập, giỏ hàng gắn với user; nếu chưa, có thể dùng session ID hoặc token tạm. Các API này đảm bảo rằng khi thêm/xóa sẽ kiểm tra tồn kho (số lượng sản phẩm còn) để không cho thêm quá số lượng hiện có.

API thanh toán đơn hàng: Tích hợp với các cổng thanh toán bên thứ ba như VNPay, PayPal. Quá trình thường gồm: tạo một đơn thanh toán và chuyển hướng người dùng đến cổng thanh toán. VD: frontend gọi POST /api/orders/checkout với thông tin đơn hàng, server tạo request URL đến cổng VNPay hoặc PayPal (kèm các tham số như số tiền, mã hóa đơn, checksum bảo mật), sau đó trả về URL đó để frontend chuyển người dùng sang trang thanh toán. Sau khi thanh toán xong, cổng thanh toán sẽ gọi lại một URL callback (vd /api/orders/vnpay_return hoặc /api/orders/paypal_return) trên server với kết quả thanh toán (thành công hoặc thất bại), server cập nhật trạng thái đơn hàng tương ứng (đã thanh toán hoặc vẫn chờ). Các API thanh toán cần xử lý bảo mật các thông tin thanh toán và kiểm tra tính hợp lệ của giao dịch (ví dụ so sánh amount, orderId với database, xác minh chữ ký số do cổng thanh toán gửi về).

API quản lý đơn hàng: Cho phép quản trị viên và người dùng theo dõi đơn hàng. Người dùng có thể gọi GET /api/orders (với token của mình) để lấy danh sách đơn hàng của họ, xem chi tiết trạng thái từng đơn (đang xử lý, đã gửi, v.v.). Quản trị viên có các endpoint để lấy tất cả đơn hàng (của mọi khách), và cập nhật trạng thái đơn hàng: ví dụ PUT /api/orders/:id để thay đổi trạng thái (chuyển từ "đang xử lý" sang "đang vận chuyển" hoặc "đã hoàn thành", v.v.). Khi cập nhật trạng thái, hệ thống có thể gửi email thông báo trạng thái mới cho khách hàng. Ngoài ra, có thể có API hủy đơn hàng (do khách yêu cầu hoặc do admin quyết định).

API quản lý danh mục: Cho phép admin thêm danh mục mới (POST /api/categories), chỉnh sửa tên/mô tả danh mục (PUT /api/categories/:id), xóa danh mục (DELETE /api/categories/:id). Danh mục có thể được lấy danh sách qua GET /api/categories (cho cả người dùng lẫn admin để biết có những loại nào). Việc xóa danh mục có thể kèm theo kiểm tra ràng buộc (không xóa nếu còn sản phẩm thuộc danh mục đó, trừ khi cho phép xóa và đồng thời xóa/cập nhật các sản phẩm liên quan).