# CHƯƠNG 4: THỰC NGHIỆM VÀ TRIỂN KHAI HỆ THỐNG

## 4.1. Phát triển Frontend (Angular)

### 4.1.1. Các module chính và quản lý state bằng dịch vụ

Trong hệ thống website Fast Food Restaurant, frontend được xây dựng bằng Angular và bao gồm các module chính như sau:

|  |  |  |
| --- | --- | --- |
| **Module** | **Đường dẫn** | **Nội dung** |
| header | src/app/components/header | Hiển thị tiêu đề trang và các liên kết điều hướng. |
| home | src/app/components/home | Hiển thị giao diện chính, danh sách món ăn nổi bật, khuyến mãi và các chương trình đang diễn ra. |
| footer | src/app/components/footer | Chứa các thông tin liên hệ, điều khoản sử dụng và bản quyền. |
| customer-profile | src/app/components/customer-profile | Quản lý thông tin cá nhân của khách hàng. |
| customer-orders | src/app/components/customer-orders | Quản lý đơn hàng của khách hàng. |
| login | src/app/components/login | Chức năng đăng nhập vào hệ thống. |
| manage | src/app/components/manage | Quản lý hệ thống, bao gồm người dùng và nội dung. |
| product-card | src/app/components/product-card | Hiển thị thông tin sản phẩm dưới dạng thẻ. |
| product-detail | src/app/components/product-detail | Hiển thị chi tiết thông tin sản phẩm. |
| register | src/app/components/register | Chức năng đăng ký tài khoản mới. |
| shopping-cart | src/app/components/shopping-cart | Quản lý giỏ hàng của khách hàng. |
| core | src/app/core | Chứa các thành phần đăng nhập và lưu token. |
| services | src/app/services | Chứa các dịch vụ hỗ trợ cho các module khác. |
| types | src/app/types | Định nghĩa các kiểu dữ liệu sử dụng trong ứng dụng. |
| … | … | … |

Danh sách module chính

Angular sử dụng RxJS (Reactive Extensions for JavaScript), đặc biệt là BehaviorSubject, để theo dõi và quản lý state. BehaviorSubject hoạt động như một nguồn dữ liệu luôn phát ra giá trị hiện tại và thông báo cho tất cả các thành phần đang lắng nghe mỗi khi có sự thay đổi. 

CartService

CartService hoạt động như một trung tâm để quản lý trạng thái giỏ hàng, bao gồm thêm, xóa và lấy thông tin các sản phẩm trong giỏ hàng.

|  |  |
| --- | --- |
| **Thành phần chính** | **Giải thích** |
| Inject HttpClient | Cho phép dịch vụ thực hiện các yêu cầu HTTP đến Backend để lấy và cập nhật dữ liệu giỏ hàng. |
| items: CartItem[] | Biến này lưu trữ danh sách các mặt hàng trong giỏ hàng. |
| init() | Hàm này gọi getCartItems() để khởi tạo dữ liệu giỏ hàng khi dịch vụ được khởi chạy. |
| getCartItems() | Hàm này thực hiện yêu cầu GET đến API backend để lấy danh sách giỏ hàng. |
| addToCart(productId: string, quantity: number) | Gửi yêu cầu POST đến API để thêm một sản phẩm vào giỏ hàng với số lượng cụ thể. |
| removeFromCart(productId: string) | Gửi yêu cầu DELETE đến API để xóa một sản phẩm khỏi giỏ hàng. |

Thành phần chính CartService

### 4.1.2. Tích hợp với REST API từ Backend.

Tích hợp REST API từ backend trong Angular là quá trình kết nối giữa giao diện người dùng (frontend) và logic xử lý dữ liệu (backend) thông qua các endpoint API. Việc này đảm bảo rằng dữ liệu được trao đổi hiệu quả giữa client (trình duyệt người dùng) và server (máy chủ backend).

Angular sử dụng HttpClient để thực hiện các yêu cầu HTTP đến API backend.

HttpClient

Các phương thức REST API trong dịch vụ:



Các phương thức REST API

|  |  |  |  |
| --- | --- | --- | --- |
| **Phương thức** | **Endpoint** | **Mô tả** | **Kết quả** |
| GET | /product | Lấy danh sách tất cả sản phẩm từ API backend | Trả về một mảng các đối tượng Product |
| GET | /product/{id} | Lấy thông tin chi tiết của một sản phẩm dựa vào id | Trả về một đối tượng Product |
| POST | /product | Gửi dữ liệu sản phẩm mới đến backend để thêm vào cơ sở dữ liệu | Trả về thông báo thành công hoặc đối tượng sản phẩm đã được tạo |
| PUT | /product/{id} | Cập nhật thông tin sản phẩm dựa vào id | Trả về thông báo cập nhật thành công. |
| DELETE | /product/{id} | Xóa sản phẩm dựa vào id | Trả về thông báo xóa thành công |

REST API

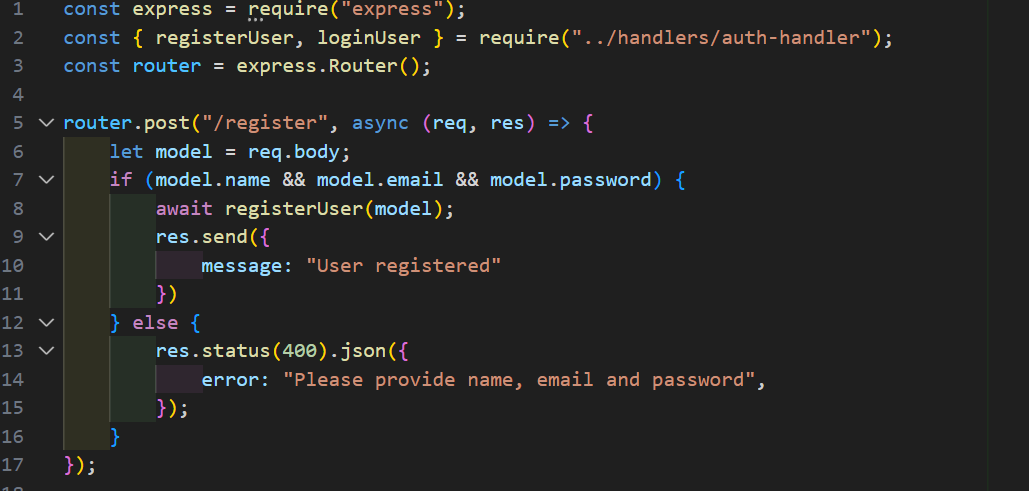
## 4.2. Phát triển Backend (NodeJS)

### 4.2.1. Xử lý logic máy chủ và API

|  |  |
| --- | --- |
| **Xử lý logic máy chủ** | **Xử lý API** |
|  |  |

Thư mục xử lý logic và xử lý API

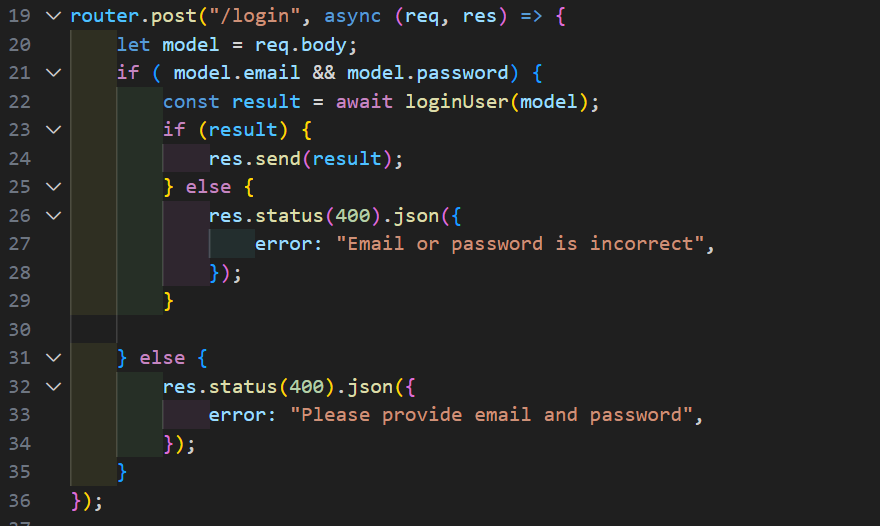
### 4.2.2. Xác thực người dùng bằng JWT

Chúng ta sẽ phân tích chi tiết quy trình xác thực người dùng bằng JWT trong file auth.js

API đăng kí

API này nằm trong endpoint POST /register và chịu trách nhiệm tạo tài khoản mới cho người dùng. Khi người dùng gửi thông tin gồm name, email, và password, hệ thống sẽ kiểm tra tính đầy đủ của các trường này. Nếu thông tin hợp lệ, hàm registerUser sẽ được gọi.

Trong hàm registerUser, mật khẩu người dùng sẽ được mã hóa bằng thư viện bcrypt trước khi lưu vào cơ sở dữ liệu. Việc mã hóa này đảm bảo rằng nếu cơ sở dữ liệu bị xâm nhập, mật khẩu thực tế của người dùng vẫn được bảo mật. Sau khi thông tin được lưu thành công, API sẽ trả về thông báo "User registered". Nếu thông tin đầu vào không hợp lệ, API sẽ trả về lỗi 400 Bad Request với thông báo chi tiết.



API đăng nhập

API này nằm trong endpoint POST /login và có nhiệm vụ xác thực người dùng khi họ muốn đăng nhập. Khi người dùng gửi thông tin email và password, hệ thống sẽ kiểm tra tính đầy đủ của thông tin này. Nếu hợp lệ, hàm loginUser sẽ được gọi để xử lý.

Hàm loginUser thực hiện các bước sau:

1. Tìm kiếm thông tin người dùng trong cơ sở dữ liệu dựa trên email được cung cấp.
2. Nếu tìm thấy, hệ thống sẽ so sánh mật khẩu người dùng gửi lên với mật khẩu đã mã hóa trong cơ sở dữ liệu bằng cách sử dụng bcrypt.compare. Nếu mật khẩu khớp, một **token JWT** sẽ được tạo ra.
3. Token này chứa thông tin cơ bản về người dùng như id, name, email, và quyền isAdmin. Token được mã hóa bằng một **secret key** và có thời hạn (ví dụ: 50 giờ).
4. Nếu xác thực thành công, API trả về token cùng thông tin người dùng. Nếu thông tin không khớp, API trả về lỗi 400 Bad Request với thông báo: "Email or password is incorrect".

## 4.3. Kết nối và xử lý dữ liệu với MongoDB

Mongoose là một thư viện Node.js giúp bạn tương tác với MongoDB dễ dàng hơn bằng cách cung cấp các công cụ để quản lý dữ liệu như schema, model, và query.

Kết nối cơ sở dữ liệu bằng MongoDB

**mongoose.connect**: Hàm này kết nối ứng dụng của bạn đến MongoDB.

**URI MongoDB**: "mongodb://localhost:27017" là địa chỉ kết nối tới máy chủ MongoDB đang chạy cục bộ (localhost) trên cổng mặc định 27017.

**dbName**: Tên cơ sở dữ liệu được sử dụng là e-comm-store-db. Nếu cơ sở dữ liệu này chưa tồn tại, MongoDB sẽ tự động tạo ra.

**await**: Đảm bảo rằng quá trình kết nối được hoàn tất trước khi tiếp tục thực thi.