**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT**

**KHOA CÔNG NGHỆ SỐ**

---🙠🕮🙢---



**ĐỒ ÁN MÔN HỌC**

**CÔNG NGHỆ XML**

**TÊN ĐỀ TÀI:**

**QUẢN LÝ CỬA HÀNG BÁN ĐỒ BÓNG ĐÁ**

**Giảng viên hướng dẫn :** Ngô Lê Quân

**Nhóm :** 15

**Sinh viên thực hiện :** Huỳnh Lê Tiến Hiển (23115053122216)

**:** Trần Minh Chiến (23115053122206)

**Lớp học phần :** 125CNX01

***Đà Nẵng, 13 tháng 11 năm 2025***

**LỜI MỞ ĐẦU**

Trong thời đại công nghệ thông tin phát triển mạnh mẽ như hiện nay, việc ứng dụng các công nghệ hiện đại vào quản lý và kinh doanh đã trở thành xu hướng tất yếu. Các cửa hàng không chỉ dừng lại ở hình thức bán hàng truyền thống mà dần chuyển mình sang các mô hình quản lý thông minh, giúp tối ưu hóa hoạt động và mang lại hiệu quả cao hơn.

Đặc biệt, trong lĩnh vực thể thao – nơi nhu cầu về trang phục, phụ kiện bóng đá ngày càng tăng – việc xây dựng một hệ thống *“Quản lý cửa hàng bán đồ bóng đá”* là rất cần thiết. Hệ thống này giúp người quản lý dễ dàng nắm bắt thông tin sản phẩm, quản lý đơn hàng, khách hàng, cũng như cập nhật số lượng hàng tồn một cách chính xác và nhanh chóng.

Trong quá trình thực hiện đồ án môn học *“Công nghệ XML”*, nhóm chúng em lựa chọn đề tài *“Quản lý cửa hàng bán đồ bóng đá”* với mục tiêu ứng dụng XML vào việc lưu trữ và trao đổi dữ liệu một cách linh hoạt, đồng thời giúp chuẩn hóa thông tin và hỗ trợ quá trình xử lý dữ liệu trong chương trình.

Đề tài không chỉ giúp chúng em củng cố kiến thức lý thuyết đã học mà còn rèn luyện kỹ năng phân tích, thiết kế và áp dụng công nghệ vào thực tế. Nhân dịp này, nhóm chúng em xin gửi lời cảm ơn chân thành đến thầy Ngô Lê Quân, người đã tận tình hướng dẫn, chỉ bảo trong suốt quá trình thực hiện đề tài.

Nhóm sinh viên thực hiện xin chân thành cảm ơn!

Mục lục

[Chương 1: Giới thiệu 5](#_Toc121256718)

[1.1. Ích Lợi Khi Sử Dụng XML 5](#_Toc121256719)

[1.2. Mục tiêu 5](#_Toc121256720)

[1.3. Phạm vi đề tài, yêu cầu 5](#_Toc121256721)

[Chương 2: Nội dung 6](#_Toc121256722)

[2.1. Khảo Sát Nghiệp Vụ 6](#_Toc121256723)

[2.1.1. Khảo sát và đánh giá hiện trạng 6](#_Toc121256724)

[2.1.2. Mô hình hoạt động nghiệp vụ của quản lý bán loa 7](#_Toc121256725)

[2.2. Phân tích thiết kế hệ thống 8](#_Toc121256726)

[2.2.1. 1. Mô tả vấn đề 8](#_Toc121256727)

[2.2.2. 2. Đặc tả yêu cầu 8](#_Toc121256728)

[2.3. Thiết kế cơ sở dữ liệu 9](#_Toc121256729)

[2.3.1. Mô hình phân cấp chức năng: 9](#_Toc121256730)

[2.3.2. Mô hình luồng dữ liệu mức khung cảnh: 10](#_Toc121256731)

[2.3.3. Mô hình dữ liệu mức đỉnh: 10](#_Toc121256732)

[2.3.4. Xác định thực thể 11](#_Toc121256733)

[2.3.5. Mô hình dữ liệu mức quan hệ 12](#_Toc121256734)

[2.3.6. Các bảng dữ liệu 13](#_Toc121256735)

[2.4. Thiết kế và cài đặt chương trình 15](#_Toc121256736)

[2.4.1. Các module chính 15](#_Toc121256737)

[2.4.2. . Class ConnectFile.cs: (Gồm phương thức kết nối SQL, tạo XML, thêm sửa xóa, hiển thị và tìm kiếm) 15](#_Toc121256738)

[2.4.3. DangNhap.cs: 17](#_Toc121256739)

[2.4.4. HoaDon.cs: 18](#_Toc121256740)

[2.4.5. Main.cs: 21](#_Toc121256741)

[2.4.6. NhanVien.cs: 22](#_Toc121256742)

[2.4.7. SanPham.cs: 23](#_Toc121256743)

[2.4.8. TaiKhoan.cs: 25](#_Toc121256744)

[2.4.9. MatKhau.cs: 26](#_Toc121256745)

[2.5. Giao diện chương trình 27](#_Toc121256746)

[2.5.1. Form Đăng nhập 27](#_Toc121256747)

[2.5.2. Form Trang chủ 28](#_Toc121256748)

[2.5.3. Form Quản lý sản phẩm 29](#_Toc121256749)

[2.5.4. Form Quản lý hóa đơn: 30](#_Toc121256750)

[2.5.5. Form thêm hóa đơn 30](#_Toc121256751)

[2.5.6. Form Quản lý nhân viên 31](#_Toc121256752)

[2.5.7. Form quản lý tài khoản: 32](#_Toc121256753)

[2.5.8. Form đổi mật khẩu: 32](#_Toc121256754)

[Chương 3: Kết luận 33](#_Toc121256755)

# Giới thiệu

## Lợi ích khi sử dụng XML

* XML cho phép tách rời dữ liệu khỏi chương trình, dữ liệu được lưu trong các tệp XML riêng biệt.
* XML có tính độc lập cao, dữ liệu ở dạng văn bản nên hầu hết các phần mềm đều có thể đọc và xử lý được.
* Có thể mô tả được thông tin của các đối tượng phức tạp mà cơ sở dữ liệu quan hệ khó thực hiện.
* Dễ dàng chia sẻ dữ liệu thông qua các tệp văn bản có cấu trúc rõ ràng, dễ hiểu.
* Khả năng đọc và phân tích dữ liệu nhanh chóng, hỗ trợ tốt cho việc trao đổi dữ liệu giữa nhiều hệ thống.
* Có thể dùng để chuyển đổi dữ liệu giữa các hệ thống không tương thích.
* Lưu trữ dữ liệu hiệu quả, giúp việc khai thác và tái sử dụng thông tin thuận tiện hơn.

## Mục tiêu

* Tìm hiểu và phân tích hoạt động kinh doanh, quản lý hàng hóa của các cửa hàng bán đồ bóng đá hiện nay.
* Xây dựng phần mềm quản lý cửa hàng bán đồ bóng đá giúp theo dõi sản phẩm, khách hàng, hóa đơn một cách hiệu quả.
* Hướng đến hệ thống thân thiện với người dùng: dễ sử dụng, giao diện trực quan, thao tác nhanh và thuận tiện cho việc quản lý.

## Phạm vi đề tài, yêu cầu

Sử dụng SQL Server để xây dựng và quản lý cơ sở dữ liệu, kết hợp với Visual Studio 2022 và các tệp XML để hoàn thành đề tài “Quản lý cửa hàng bán đồ bóng đá”.

Yêu cầu:

* Cơ sở dữ liệu
* XSD (lược đồ)
* XML
* XSLT
* Form kết nối Database → XML
* Chức năng Thêm
* Chức năng Sửa, Xóa
* Hiển thị dữ liệu trên trình duyệt
* Cập nhật dữ liệu từ XML lên Database
* Kiểm tra tính hợp lệ của dữ liệu

# Nội dung

## Khảo sát nghiệp vụ

### Khảo sát và đánh giá hiện trạng

Các mặt hàng trong cửa hàng bóng đá bao gồm nhiều loại sản phẩm như quần áo thi đấu, giày bóng đá, bóng, găng tay thủ môn, bình nước, túi thể thao và các phụ kiện khác đến từ nhiều thương hiệu nổi tiếng. Nhu cầu mua sắm các sản phẩm thể thao ngày càng tăng, đặc biệt là trong giới trẻ và các câu lạc bộ bóng đá phong trào.

Vì vậy, bài toán *“Quản lý cửa hàng bán đồ bóng đá”* là bài toán kết hợp giữa quản lý danh mục sản phẩm và quản lý hoạt động bán hàng. Chương trình được thiết kế đơn giản, dễ sử dụng, tập trung vào việc quản lý sản phẩm, khách hàng và hóa đơn, chưa mở rộng đến khâu quản lý nhà cung cấp.

**Bài toán tập trung vào các vấn đề sau:**

* *Quản lý hóa đơn bán hàng:* Cập nhật, tìm kiếm và theo dõi chi tiết từng hóa đơn.
* *Quản lý danh mục:* Quản lý các nhóm hàng như quần áo, giày, bóng, phụ kiện,… giúp phân loại sản phẩm rõ ràng.
* *Quản lý sản phẩm:*Lưu trữ, thêm, sửa, xóa và tìm kiếm thông tin sản phẩm như tên, mã, giá, số lượng, thương hiệu, và danh mục tương ứng.
* *Quản lý khách hàng:*Lưu trữ thông tin khách hàng mua hàng, phục vụ việc chăm sóc và tra cứu.
* Quản lý tài khoản người dùng phần mềm.
* Thống kê doanh thu, sản phẩm bán chạy.
* Chuyển đổi cơ sở dữ liệu: Từ SQL Server sang XML và ngược lại.

**Yêu cầu chương trình phần mềm:**

* Phần mềm phải dễ sử dụng, giao diện thân thiện, thao tác nhanh.
* Có khả năng hiển thị dữ liệu rõ ràng, hỗ trợ người quản lý tra cứu thuận tiện.
* Có thể cập nhật, chỉnh sửa dữ liệu dễ dàng và đảm bảo tính chính xác.

**Yêu cầu dữ liệu**:

* Dữ liệu phải đầy đủ, không trùng lặp, đảm bảo thống nhất giữa cơ sở dữ liệu và file XML.
* Các chức năng đọc, ghi, cập nhật dữ liệu phải hoạt động ổn định, chính xác.

### Mô hình hoạt động nghiệp vụ của quản lý ứng dụng đặt đồ ăn

* Admin có thể thực hiện các công việc quản lý danh mục và sản phẩm như thêm, sửa, xóa, và cập nhật thông tin khi cần thiết.
* Ứng dụng cho phép phân nhóm sản phẩm theo danh mục như quần áo, giày, bóng, găng tay, phụ kiện,… giúp việc tìm kiếm và quản lý thuận tiện hơn.
* Quản lý tồn kho: Theo dõi số lượng sản phẩm hiện có, cập nhật khi có nhập hoặc bán hàng, đảm bảo hàng hóa luôn đầy đủ và chính xác.
* Thống kê, báo cáo: Tổng hợp danh sách sản phẩm bán chạy, sản phẩm sắp hết hàng và hỗ trợ quản lý trong việc nhập hàng mới.

Việc phân loại và quản lý sản phẩm cần đảm bảo chính xác, dễ thao tác và tiết kiệm thời gian cho người sử dụng. Do đó, hệ thống quản lý cửa hàng bán đồ bóng đá được thiết kế nhằm đáp ứng đầy đủ các yêu cầu nghiệp vụ trên.

## Phân tích thiết kế hệ thống

### Mô tả vấn đề

Với phần mềm phục vụ cho công việc *“Quản lý cửa hàng bán đồ bóng đá”*, có thể nhận thấy:

**Mục đích của hệ thống:**

Tin học hóa quá trình quản lý sản phẩm và danh mục hàng hóa trong cửa hàng, giúp việc quản lý trở nên nhanh chóng, chính xác và hiệu quả hơn.

**Các chức năng chính:**

* Quản lý danh mục sản phẩm.
* Quản lý sản phẩm.
* Quản lý bình luận
* Quản lý đơn hàng
* Quản lí khách hàng

Đối tượng sử dụng và vai trò của mỗi đối tượng:

* **Chủ cửa hàng (Admin):** Quản lý toàn bộ hệ thống, thực hiện các chức năng thêm – sửa – xóa danh mục, sản phẩm, và xử lý dữ liệu XML/Database.

### Đặc tả yêu cầu

Hệ thống được xây dựng nhằm hỗ trợ cửa hàng kinh doanh đồ đá banh trong việc quản lý sản phẩm, đơn hàng, khách hàng và phản hồi đánh giá. Dưới đây là các yêu cầu chức năng chính.

* **Quản lý khách hàng**  
  Lưu trữ thông tin khách hàng để dễ dàng tra cứu và chăm sóc khách hàng. Thông tin khách hàng bao gồm:

+ Tên khách hàng

+ Email

+ Số điện thoại

+ Ngày sinh

+ Giới tính

+ Địa chỉ

+ Số lượng đơn hàng hoàn thành

+ Tổng tiền cho các đơn hàng hoàn thành

+ Số lượng đơn hàng đang xử lí

+ Tổng tiền cho các đơn hàng đang xử lí

+ Ngày tạo tài khoản

* **Quản lý danh mục**  
  Thêm sửa xóa các danh mục:

+ ID danh mục

+ Tên danh mục

+ Ngày tạo

+ Mô tả

* **Quản lý sản phẩm**  
  Quản lý thông tin sản phẩm đồ đá banh: thêm, sửa, xóa, cập nhật tồn kho, hiển thị hình ảnh.

+ ID Sản phẩm

+ ID Danh mục

+ Hãng

+ Kích thước

+ Màu sắc

+ Mô tả

+ Hình ảnh

+ Số lượng tồn kho

+ Đơn vị tính

* **Quản lý đơn hàng**  
  Mỗi giao dịch tạo HoaDon kèm các bản ghi ChiTietHoaDon.

**Hóa đơn**

+ ID Hóa đơn

+ ID Người dùng

+ Tổng tiền

+ Địa chỉ giao hàng

+ Ngày đặt

+ Trạng thái

+ Hình ảnh

+ Số lượng tồn kho

+ Đơn vị tính

**Chi tiết hóa đơn**

+ ID Chi tiết hóa đơn

+ ID Hóa đơn

+ ID Sản phẩm

+ Số lượng

+ Đơn giá

* **Quản lý bình luận**  
  Lưu trữ phản hồi/đánh giá của khách hàng về sản phẩm.

+ ID Bình luận

+ ID Khách hàng

+ ID Sản phẩm

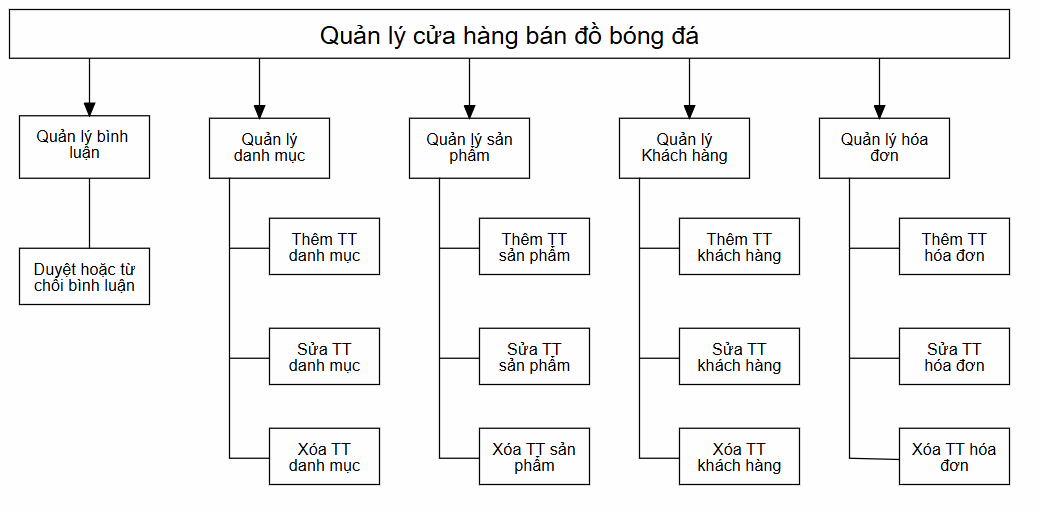
+ Nội dung

+ Ngày bình luận

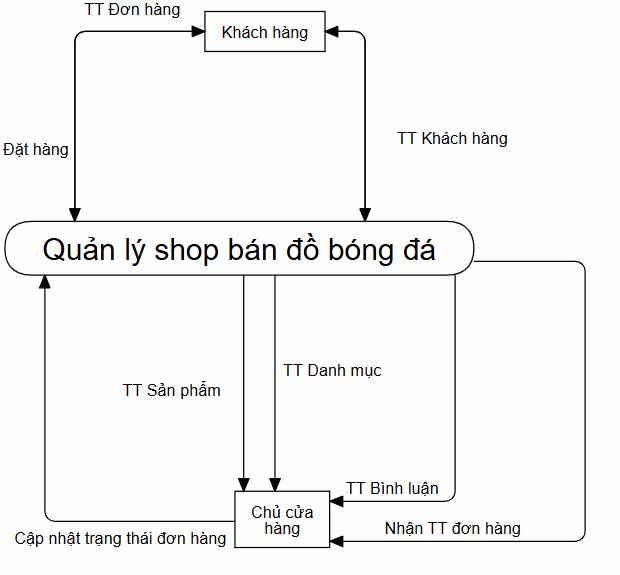
+ Tình trạng

## Thiết kế cơ sở dữ liệu

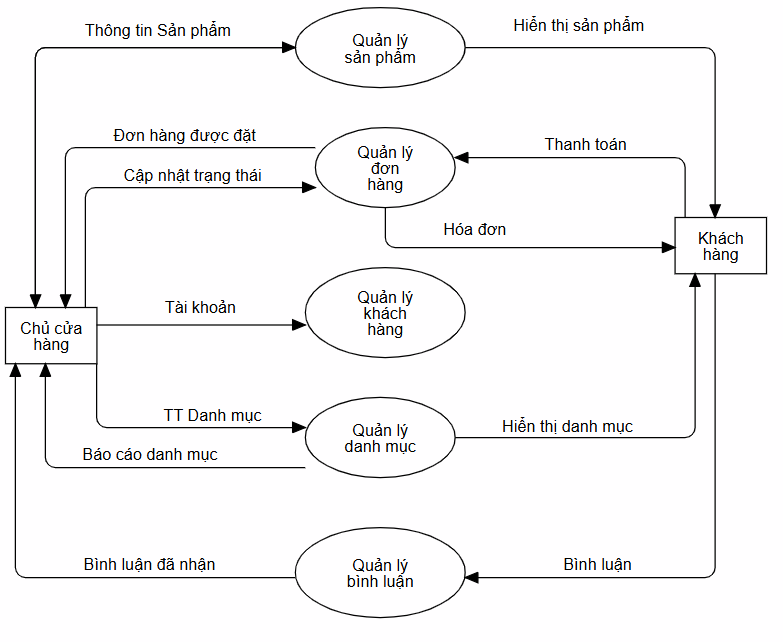
### Mô hình phân cấp chức năng:



### Mô hình luồng dữ liệu mức khung cảnh:



### Mô hình dữ liệu mức đỉnh:



### Xác định thực thể

|  |  |
| --- | --- |
| **Tên thực thể** | **Thuộc tính** |
| **Khách hàng** | ID Người dùng  Email  Mật khẩu  Số điện thoại  Địa chỉ  Đường dẫn ảnh đại diện  Vai trò  Giới tính |
| **Danh mục** | ID Danh mục  Tên danh mục  Mô tả  Ngày tạo |
| **Sản phẩm** | ID Sản phẩm  ID Danh mục  Hãng  Kích thước  Màu sắc  Mô tả  Hình ảnh  Số lượng tồn kho  Đơn vị tính |
| **Hóa đơn** | ID Hóa Đơn  ID Người dùng  Tổng tiền  Địa chỉ  Ngày đặt  Trạng thái |
| **Chi tiết hóa đơn** | ID Chi tiết hóa đơn  ID Hóa đơn  ID Sản phẩm  Số lượng  Đơn giá |
| **Bình luận** | ID Bình luận  ID Người dùng  ID Sản phẩm  Nội dung  Ngày bình luận  Tình trạng |

### Mô hình dữ liệu mức quan hệ

### Các bảng dữ liệu

Bảng người dùng

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu**  **(Độ rộng)** | **Ràng buộc** | **Khóa chính/ Khóa phụ (PK/FK)** | **Mô tả** |
|  | IDNguoiDung | INT | Not null, Identity(1,1) | PK | Mã người dùng |
|  | Email | VARCHAR(100) | Not null, Unique, Check format |  | Email |
|  | SĐT | VARCHAR(10) |  |  | Số điện thoại |
|  | DiaChi | NVARCHAR(255) |  |  | Địa chỉ |
|  | AvatarURL | VARCHAR(255) |  |  | Đường dẫn ảnh đại diện |
|  | VaiTro | VARCHAR(100) | Check IN ('Admin', 'User') |  | Phân quyền người dùng |
|  | GioiTinh | NVARCHAR(3) | Check IN (N'Nam', N'Nữ') |  | Giới tính |

Bảng danh mục

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu**  **(Độ rộng)** | **Ràng buộc** | **Khóa chính/ Khóa phụ (PK/FK)** | **Mô tả** |
|  | IDDanhMuc | INT | Not null, Identity(1,1) | PK | Mã danh mục |
|  | TenDanhMuc | NVARCHAR(255) | Not null |  | Tên danh mục |
|  | MoTa | NVARCHAR(255) |  |  | Mô tả |
|  | NgayTao | DATE |  |  | Địa chỉ |
|  | AvatarURL | VARCHAR(255) |  |  | Ngày tạo |

Bảng sản phẩm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu**  **(Độ rộng)** | **Ràng buộc** | **Khóa chính/ Khóa phụ (PK/FK)** | **Mô tả** |
|  | IDSanPham | INT | Not null, Identity(1,1) | PK | Mã sản phẩm |
|  | IDDanhMuc | INT |  | FK | Mã danh mục |
|  | Hang | NVARCHAR(100) |  |  | Hãng |
|  | KichThuoc | VARCHAR(5) |  |  | Kích thước |
|  | MauSac | NVARCHAR(100) |  |  | Màu sắc |
|  | MoTa | NVARCHAR(255) |  |  | Mô tả |
|  | HinhAnh | VARCHAR(255) |  |  | Đường dẫn |
|  | SoLuongTK | INT | Check (>= 0) |  | Số lượng |
|  | DonViTinh | NVARCHAR(255) |  |  | Đơn vị tính |

Bảng hóa đơn

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu**  **(Độ rộng)** | **Ràng buộc** | **Khóa chính/ Khóa phụ (PK/FK)** | **Mô tả** |
|  | IDHoaDon | INT | Not null, Identity(1,1) | PK | Mã hóa đơn |
|  | IDNguoiDung | INT |  | FK | Mã người dùng |
|  | TongTien | DECIMAL(18,2) | Check (>= 0) |  | Tổng tiền |
|  | DiaChi | NVARCHAR(255) |  |  | Địa chỉ |
|  | NgayDat | DATE |  |  | Ngày đặt |
|  | TrangThai | NVARCHAR(100) | Check IN ('Đã/Chờ/Hủy thanh toán') |  | Trạng thái |

Bảng chi tiết hóa đơn

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu**  **(Độ rộng)** | **Ràng buộc** | **Khóa chính/ Khóa phụ (PK/FK)** | **Mô tả** |
|  | IDChiTietHD | INT | Not null, Identity(1,1) | PK | Mã chi tiết |
|  | IDHoaDon | INT |  | FK | Mã hóa đơn |
|  | IdSanPham | INT |  | FK | Mã sản phẩm |
|  | SoLuong | INT | Check (> 0) |  | Số lượng mua |
|  | DonGia | DECIMAL(18,2) | Check (> 0) |  | Đơn giá |

Bảng bình luận

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu**  **(Độ rộng)** | **Ràng buộc** | **Khóa chính/ Khóa phụ (PK/FK)** | **Mô tả** |
|  | IDBinhLuan | INT | Not null, Identity(1,1) | PK | Mã bình luận |
|  | IDNguoiDung | INT |  | FK | Người bình luận |
|  | IdSanPham | INT |  | FK | Sản phẩm được bình luận |
|  | NoiDung | NVARCHAR(255) | Check (> 0) |  | Nội dung |
|  | NgayBL | DATE | Check (> 0) |  | Thời gian |
|  | TinhTrang | NVARCHAR(100) |  |  | Tình trạng |

## Thiết kế và cài đặt chương trình

### Các module chính

a. Kết nối database

b. Module về nhân viên

c. Module về khách hàng

d. Module về danh mục

e. Module về món ăn

f. Module về đơn hàng

g. Module về bình luận

### Kết nối database

import { Sequelize } from "sequelize";

const sequelize = new Sequelize(

    process.env.DATABASE\_NAME, //Tên database

    process.env.DATABASE\_USERNAME, //Username

    process.env.DATABASE\_PASSWORD, //Password

    {

       host: process.env.DATABASE\_HOST, //Tên địa chỉ host

       dialect: 'mysql'

    }

);

sequelize.authenticate().then(() => {

    console.log('Kết nối DB thành công');

 }).catch((error) => {

    console.error('Kết nối thất bại: ', error);

 });

export default sequelize;

### Module nhân viên ( Lấy danh sach, thêm, sửa, xóa)

import fs from 'fs';

import path from 'path';

import { parseStringPromise,Builder } from 'xml2js';

export const listStaff = async (staffList: any[], fileName: string = 'Staff.xml'): Promise<void> => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        try {

            await fs.promises.access(filePath, fs.constants.F\_OK);

            console.log(`File XML "${fileName}" đã tồn tại. Không cần tạo lại.`);

            return;

        } catch {

            console.log(`File XML "${fileName}" chưa tồn tại. Tạo file mới.`);

        }

        const builder = new Builder({

            rootName: 'staffs',

            xmldec: { version: '1.0', encoding: 'UTF-8' },

            renderOpts: { pretty: true, indent: '  ', newline: '\n' }

        });

        const formattedStaffList = staffList.map(({ 'Role.tenVaiTro': \_, idNguoiDung, ...staff }) => ({

            $: { id: idNguoiDung },

            ...staff

        }));

        const xmlData = builder.buildObject({ staff: formattedStaffList });

        await fs.promises.writeFile(filePath, xmlData, 'utf-8');

        console.log(`File XML đã được lưu tại: ${filePath}`);

    } catch (error) {

        console.error('Lỗi khi tạo file XML: ', error.message);

        throw new Error('Không thể tạo file XML');

    }

};

export const createStaff= async (newStaff: any, fileName: string = 'Staff.xml')=>{

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const parsedData = await parseStringPromise(xmlData);

        const newStaffXML = {

            $: { id: newStaff.idNguoiDung },

            hoTen: newStaff.hoTen,

            email: newStaff.email,

            sdt: newStaff.sdt,

            ngaySinh: newStaff.ngaySinh || '',

            gioiTinh: newStaff.gioiTinh,

            avatar: newStaff.avatar || '',

            trangThai: newStaff.trangThai,

            token: newStaff.token,

            tenVaiTro: newStaff.tenVaiTro

        };

        if (!parsedData.staffs.staff) {

            parsedData.staffs.staff = [];

        }

        parsedData.staffs.staff.push(newStaffXML);

        const builder = new Builder({ xmldec: { version: '1.0', encoding: 'UTF-8' }, renderOpts: { pretty: true } });

        const updatedXML = builder.buildObject(parsedData);

        await fs.promises.writeFile(filePath, updatedXML, 'utf-8');

        console.log('Đã thêm nhân viên mới vào file XML.');

    } catch (error) {

        console.error('Lỗi khi thêm nhân viên vào file XML: ', error.message);

        throw new Error('Không thể thêm nhân viên vào file XML');

    }

}

export const changStatus = async (token: string, status: string, fileName: string = 'Staff.xml') => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const jsonData: any = await parseStringPromise(xmlData);

        const staffList = jsonData.staffs.staff;

        const staff = staffList.find((s: any) => s.token[0] === token);

        if (!staff) {

            console.error('Token không tồn tại:', token);

            return;

        }

        staff.trangThai[0] = status;

        const builder = new Builder();

        const updatedXml = builder.buildObject(jsonData);

        await fs.promises.writeFile(filePath, updatedXml, 'utf-8');

        console.log('Cập nhật trạng thái thành công');

    } catch (error) {

        console.error('Lỗi khi cập nhật trạng thái: ', error);

    }

};

export const updateStaff=async (token: string, tenVaiTro: string, fileName: string = 'Staff.xml') => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const jsonData: any = await parseStringPromise(xmlData);

        const staffList = jsonData.staffs.staff;

        const staff = staffList.find((s: any) => s.token[0] === token);

        if (!staff) {

            console.error('Token không tồn tại:', token);

            return;

        }

        staff.tenVaiTro[0] = tenVaiTro;

        const builder = new Builder();

        const updatedXml = builder.buildObject(jsonData);

        await fs.promises.writeFile(filePath, updatedXml, 'utf-8');

        console.log('Cập nhật thành công');

    } catch (error) {

        console.error('Lỗi khi cập nhật: ', error);

    }

}

export const deleteStaff = async (token: string, fileName: string = 'Staff.xml') => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const jsonData: any = await parseStringPromise(xmlData);

        let staffList = jsonData.staffs.staff;

        const staffIndex = staffList.findIndex((s: any) => s.token[0] === token);

        if (staffIndex === -1) {

            console.error('Không tìm thấy nhân viên với token được cung cấp');

            return;

        }

        staffList.splice(staffIndex, 1);

        const builder = new Builder();

        const updatedXml = builder.buildObject(jsonData);

        await fs.promises.writeFile(filePath, updatedXml, 'utf-8');

        console.log('Xóa nhân viên thành công');

    } catch (error) {

        console.error('Lỗi khi xóa nhân viên:', error);

    }

}

### Module khách hàng (Lấy danh sách, sửa, xóa)

import fs from 'fs';

import path from 'path';

import { parseStringPromise,Builder } from 'xml2js';

export const listCustomer = async (customerList: any[], fileName: string = 'Customer.xml'): Promise<void> => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        try {

            await fs.promises.access(filePath, fs.constants.F\_OK);

            console.log(`File XML "${fileName}" đã tồn tại. Không cần tạo lại.`);

            return;

        } catch {

            console.log(`File XML "${fileName}" chưa tồn tại. Tạo file mới.`);

        }

        const builder = new Builder({

            rootName: 'customers',

            xmldec: { version: '1.0', encoding: 'UTF-8' },

            renderOpts: { pretty: true, indent: '  ', newline: '\n' }

        });

        const formattedStaffList = customerList.map(({ 'Role.tenVaiTro': \_, idNguoiDung, ...customer }) => ({

            $: { id: idNguoiDung },

            ...customer

        }));

        const xmlData = builder.buildObject({ customer: formattedStaffList });

        await fs.promises.writeFile(filePath, xmlData, 'utf-8');

        console.log(`File XML đã được lưu tại: ${filePath}`);

    } catch (error) {

        console.error('Lỗi khi tạo file XML: ', error.message);

        throw new Error('Không thể tạo file XML');

    }

};

export const changStatus = async (token: string, status: string, fileName: string = 'Customer.xml') => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const jsonData: any = await parseStringPromise(xmlData);

        const customerList = jsonData.customers.customer;

        const customer = customerList.find((s: any) => s.token[0] === token);

        if (!customer) {

            console.error('Token không tồn tại:', token);

            return;

        }

        customer.trangThai[0] = status;

        const builder = new Builder();

        const updatedXml = builder.buildObject(jsonData);

        await fs.promises.writeFile(filePath, updatedXml, 'utf-8');

        console.log('Cập nhật trạng thái thành công');

    } catch (error) {

        console.error('Lỗi khi cập nhật trạng thái: ', error);

    }

}

export const deleteCustomer = async (token: string, fileName: string = 'Customer.xml') => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const jsonData: any = await parseStringPromise(xmlData);

        let customerList = jsonData.customers.customer;

        const customerIndex = customerList.findIndex((s: any) => s.token[0] === token);

        if (customerIndex === -1) {

            console.error('Không tìm thấy nhân viên với token được cung cấp');

            return;

        }

        customerList.splice(customerIndex, 1);

        const builder = new Builder();

        const updatedXml = builder.buildObject(jsonData);

        await fs.promises.writeFile(filePath, updatedXml, 'utf-8');

        console.log('Xóa khách hàng thành công');

    } catch (error) {

        console.error('Lỗi khi xóa khách hàng:', error);

    }

}

### Module danh mục (Lấy danh sách, thêm, sửa, xóa)

import fs from 'fs';

import path from 'path';

import { parseStringPromise,Builder } from 'xml2js';

export const listCategory = async (categoryList: any[], fileName: string = 'Category.xml'): Promise<void> => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        try {

            await fs.promises.access(filePath, fs.constants.F\_OK);

            console.log(`File XML "${fileName}" đã tồn tại. Không cần tạo lại.`);

            return;

        } catch {

            console.log(`File XML "${fileName}" chưa tồn tại. Tạo file mới.`);

        }

        const builder = new Builder({

            rootName: 'categories',

            xmldec: { version: '1.0', encoding: 'UTF-8' },

            renderOpts: { pretty: true, indent: '  ', newline: '\n' }

        });

        const formattedCategoryList = categoryList.map(({ idDanhMuc, ...category }) => ({

            $: { id: idDanhMuc },

            ...category

        }));

        const xmlData = builder.buildObject({ category: formattedCategoryList });

        await fs.promises.writeFile(filePath, xmlData, 'utf-8');

        console.log(`File XML đã được lưu tại: ${filePath}`);

    } catch (error) {

        console.error('Lỗi khi tạo file XML: ', error.message);

        throw new Error('Không thể tạo file XML');

    }

}

export const createCategory = async (category: any, fileName: string = 'Category.xml') => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const parsedData = await parseStringPromise(xmlData);

        const newCategoryXML = {

            $: { id: category.idDanhMuc },

            tenDanhMuc: category.tenDanhMuc,

            moTa: category.moTa || '',

            slug: category.slug,

            ngayTao:category.ngayTao || '',

        };

        if (!parsedData.categories.category) {

            parsedData.categories.category = [];

        }

        parsedData.categories.category.push(newCategoryXML);

        const builder = new Builder({

            xmldec: { version: '1.0', encoding: 'UTF-8' },

            renderOpts: { pretty: true, indent: '  ', newline: '\n' }

        });

        const updatedXML = builder.buildObject(parsedData);

        // Ghi lại file XML

        await fs.promises.writeFile(filePath, updatedXML, 'utf-8');

        console.log('Đã thêm danh mục mới vào file XML.');

    } catch (error) {

        console.error('Lỗi khi thêm danh mục vào file XML: ', error.message);

        throw new Error('Không thể thêm danh mục vào file XML');

    }

}

export const updateCategory = async (slug:string,updatedFields: any, fileName: string = 'Category.xml') => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const parsedData = await parseStringPromise(xmlData);

        const categories = parsedData.categories.category;

        const categoryToUpdate = categories.find((category: any) => category.slug[0] === slug);

        if (!categoryToUpdate) {

            console.log(`Không tìm thấy danh mục với slug: ${slug}`);

            return;

        }

        if (updatedFields.tenDanhMuc) categoryToUpdate.tenDanhMuc[0] = updatedFields.tenDanhMuc;

        if (updatedFields.moTa) categoryToUpdate.moTa[0] = updatedFields.moTa;

        if (updatedFields.ngayTao) categoryToUpdate.ngayTao[0] = updatedFields.ngayTao;

        const builder = new Builder({ xmldec: { version: '1.0', encoding: 'UTF-8' }, renderOpts: { pretty: true, indent: '  ', newline: '\n' } });

        const updatedXML = builder.buildObject(parsedData);

        // Ghi lại file XML

        await fs.promises.writeFile(filePath, updatedXML, 'utf-8');

        console.log('Danh mục đã được cập nhật thành công.');

    } catch (error) {

        console.error('Lỗi khi cập nhật danh mục:', error.message);

        throw new Error('Không thể cập nhật danh mục trong file XML');

    }

}

export const deleteCategory = async (slug: string, fileName: string = 'Category.xml') => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const parsedData = await parseStringPromise(xmlData);

        const categories = parsedData.categories.category;

        const categoryIndex = categories.findIndex((category: any) => category.slug[0] === slug);

        if (categoryIndex === -1) {

            console.log(`Không tìm thấy danh mục với slug: ${slug}`);

            return;

        }

        categories.splice(categoryIndex, 1);

        const builder = new Builder({ xmldec: { version: '1.0', encoding: 'UTF-8' }, renderOpts: { pretty: true, indent: '  ', newline: '\n' } });

        const updatedXML = builder.buildObject(parsedData);

        await fs.promises.writeFile(filePath, updatedXML, 'utf-8');

        console.log('Danh mục đã được xóa thành công.');

    } catch (error) {

        console.error('Lỗi khi xóa danh mục:', error.message);

        throw new Error('Không thể xóa danh mục trong file XML');

    }

}

### Module món ăn (Lấy danh sách, thêm, sửa, xóa)

import fs from 'fs';

import path from 'path';

import { parseStringPromise,Builder } from 'xml2js';

export const listFood = async (foodList: any[], fileName: string = 'Food.xml'): Promise<void> => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        try {

            await fs.promises.access(filePath, fs.constants.F\_OK);

            console.log(`File XML "${fileName}" đã tồn tại. Không cần tạo lại.`);

            return;

        } catch {

            console.log(`File XML "${fileName}" chưa tồn tại. Tạo file mới.`);

        }

        const builder = new Builder({

            rootName: 'foods',

            xmldec: { version: '1.0', encoding: 'UTF-8' },

            renderOpts: { pretty: true, indent: '  ', newline: '\n' }

        });

        const formattedFoodList = foodList.map(({ idSanPham, idDanhMuc, 'Category.tenDanhMuc': tenDanhMuc, ...food }) => ({

            $: { id: idSanPham, idDanhMuc },

            tenDanhMuc,

            ...food

        }));

        const xmlData = builder.buildObject({ food: formattedFoodList });

        await fs.promises.writeFile(filePath, xmlData, 'utf-8');

        console.log(`File XML đã được lưu tại: ${filePath}`);

    } catch (error) {

        console.error('Lỗi khi tạo file XML: ', error.message);

        throw new Error('Không thể tạo file XML');

    }

};

export const createFood = async (newFood: any, fileName: string = 'Food.xml') => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const parsedData:any = await parseStringPromise(xmlData);

        const newFoodXML = {

            $: { id: newFood.idSanPham, idDanhMuc: newFood.idDanhMuc },

            tenDanhMuc: newFood.tenDanhMuc,

            tenSanPham: newFood.tenSanPham,

            trangThai: newFood.trangThai,

            giaTien: newFood.giaTien,

            images: newFood.images,

            moTa: newFood.moTa,

            soLuongDaBan: newFood.soLuongDaBan,

            donViTinh: newFood.donViTinh,

            ngayTao: newFood.ngayTao,

            slug: newFood.slug

        };

        if (!parsedData.foods.food) {

            parsedData.foods.food = [];

        }

        parsedData.foods.food.push(newFoodXML);

        const builder = new Builder({

            xmldec: { version: '1.0', encoding: 'UTF-8' },

            renderOpts: { pretty: true, indent: '  ', newline: '\n' }

        });

        const updatedXML = builder.buildObject(parsedData);

        await fs.promises.writeFile(filePath, updatedXML, 'utf-8');

        console.log('Đã thêm món ăn mới vào file XML.');

    } catch (error) {

        console.error('Lỗi khi thêm món ăn vào file XML: ', error.message);

        throw new Error('Không thể thêm món ăn vào file XML');

    }

}

export const changeStatus=async(slug:string,status:string, fileName: string = 'Food.xml')=>{

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const jsonData: any = await parseStringPromise(xmlData);

        const foodList = jsonData.foods.food;

        const food = foodList.find((s: any) => s.slug[0] === slug);

        if (!food) {

            throw new Error(`Không tìm thấy món ăn với slug "${slug}".`);

        }

        food.trangThai = [status];

        // Xây dựng lại XML

        const builder = new Builder({

            xmldec: { version: '1.0', encoding: 'UTF-8' },

            renderOpts: { pretty: true, indent: '  ', newline: '\n' },

        });

        const updatedXML = builder.buildObject(jsonData);

        await fs.promises.writeFile(filePath, updatedXML, 'utf-8');

        console.log('Đã cập nhật trạng thái món ăn.');

    } catch (error) {

        console.error('Lỗi khi cập nhật trạng thái món ăn:', error.message);

        throw new Error('Không thể cập nhật trạng thái món ăn.');

    }

}

export const updateFood=async(slug:string,updatedData:any,fileName: string = 'Food.xml')=>{

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const jsonData: any = await parseStringPromise(xmlData);

        const foodList = jsonData.foods.food;

        const food = foodList.find((s: any) => s.slug[0] === slug);

        if (!food) {

            throw new Error(`Không tìm thấy món ăn với slug "${slug}".`);

        }

        if (updatedData.tenSanPham) food.tenSanPham = [updatedData.tenSanPham];

        if (updatedData.idDanhMuc) food.$.idDanhMuc = updatedData.idDanhMuc;

        if (updatedData.giaTien) food.giaTien = [updatedData.giaTien];

        if (updatedData.soLuongDaBan) food.soLuongDaBan = [updatedData.soLuongDaBan];

        if (updatedData.donViTinh) food.donViTinh = [updatedData.donViTinh];

        if (updatedData.ngayTao) food.ngayTao = [updatedData.ngayTao];

        if (updatedData.moTa) food.moTa = [updatedData.moTa];

        if  (updatedData.images) food.images = [updatedData.images];

        // Xây dựng lại XML

        const builder = new Builder({

            xmldec: { version: '1.0', encoding: 'UTF-8' },

            renderOpts: { pretty: true, indent: '  ', newline: '\n' },

        });

        const updatedXML = builder.buildObject(jsonData);

        // Ghi lại XML vào file

        await fs.promises.writeFile(filePath, updatedXML, 'utf-8');

        console.log('Đã cập nhật thông tin món ăn.');

    }

    catch (error) {

        console.error('Lỗi khi cập nhật món ăn:', error.message);

        throw new Error('Không thể cập nhật món ăn.');

    }

}

export const deleteFood=async(slug:string,fileName: string = 'Food.xml')=>{

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const jsonData: any = await parseStringPromise(xmlData);

        const foodList = jsonData.foods.food;

        const food = foodList.findIndex((s: any) => s.slug[0] === slug);

        if (food === -1) {

            console.log(`Không tìm thấy món ăn với slug: ${slug}`);

            return;

        }

        foodList.splice(food, 1);

        const builder = new Builder({ xmldec: { version: '1.0', encoding: 'UTF-8' }, renderOpts: { pretty: true, indent: '  ', newline: '\n' } });

        const updatedXML = builder.buildObject(jsonData);

        await fs.promises.writeFile(filePath, updatedXML, 'utf-8');

    }catch(error){

        console.error('Lỗi khi xóa món ăn:', error.message);

        throw new Error('Không thể xóa món ăn.');

    }

}

### Module đơn hàng (Lấy danh sách, tạo thông báo)

import fs from 'fs';

import path from 'path';

import { parseStringPromise,Builder } from 'xml2js';

export const listOrder = async (orderList: any[], fileName: string = 'Order.xml'): Promise<void> => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        try {

            await fs.promises.access(filePath, fs.constants.F\_OK);

            console.log(`File XML "${fileName}" đã tồn tại. Không cần tạo lại.`);

            return;

        } catch {

            console.log(`File XML "${fileName}" chưa tồn tại. Tạo file mới.`);

        }

        const builder = new Builder({

            rootName: 'orders',

            xmldec: { version: '1.0', encoding: 'UTF-8' },

            renderOpts: { pretty: true, indent: '  ', newline: '\n' },

        });

         // Chuẩn bị dữ liệu để build XML

         const xmlData = builder.buildObject({

            order: orderList.map(order => ({

                $: { id: order.idDonHang }, // Đặt `id` trong thuộc tính của `<order>`

                tinhTrang: order.tinhTrang,

                thoiGianHoanThanh: order.thoiGianHoanThanh,

                ngayTao: order.ngayTao,

                ghiChu: order.ghiChu || '',

                diaChi: order.diaChi || '',

                tinhTrangThanhToan: order.tinhTrangThanhToan || '',

                thoiGianHuy: order.thoiGianHuy || '',

                customer: {

                    $: { id: order.idNguoiDung }, // Đặt `id` trong thuộc tính của `<customer>`

                    hoTen: order.User.hoTen,

                    avatar: order.User.avatar,

                    sdt: order.User.sdt,

                    gioiTinh: order.User.gioiTinh,

                },

                foods: {

                    food: order.OrderDetails.map(food => ({

                        $: { id: food.idSanPham }, // Đặt `id` trong thuộc tính của `<food>`

                        soLuongDat: food.soLuongDat,

                        tenSanPham: food.tenSanPham,

                        giaTien: food.giaTien,

                        images: food.images,

                        donViTinh: food.donViTinh,

                    })),

                },

            })),

        });

        await fs.promises.writeFile(filePath, xmlData, 'utf-8');

        console.log(`File XML đã được lưu tại: ${filePath}`);

    } catch (error) {

        console.error('Lỗi khi tạo file XML: ', error.message);

        throw new Error('Không thể tạo file XML');

    }

}

export const createNotification= async(data:any, fileName: string = 'Notification.xml'): Promise<void> => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        let existingData: any[] = [];

        try {

            await fs.promises.access(filePath, fs.constants.F\_OK);

            console.log(`File XML "${fileName}" đã tồn tại. Đọc dữ liệu cũ.`);

            // Đọc dữ liệu từ XML

            const xmlContent = await fs.promises.readFile(filePath, 'utf-8');

            const parsedData = await parseStringPromise(xmlContent);

            existingData = parsedData.notifications?.notification || [];

        } catch {

            console.log(`File XML "${fileName}" chưa tồn tại. Sẽ tạo mới.`);

        }

        // Thêm dữ liệu mới

        const newNotification = {

            $: { id: data.idThongBao },

            tieuDe: data.tieuDe,

            noiDung: data.noiDung,

            ngayThongBao: data.ngayThongBao,

            customer: {

                $: { id: data.idNguoiDung },

                hoTen: data.hoTen,

                avatar: data.avatar,

                sdt: data.sdt,

                gioiTinh: data.gioiTinh,

            },

        };

        const updatedNotifications = [...existingData, newNotification];

        // Xây dựng XML mới

        const builder = new Builder({

            rootName: 'notifications',

            xmldec: { version: '1.0', encoding: 'UTF-8' },

            renderOpts: { pretty: true, indent: '  ', newline: '\n' },

        });

        const xmlData = builder.buildObject({ notification: updatedNotifications });

        // Ghi dữ liệu vào file

        await fs.promises.writeFile(filePath, xmlData, 'utf-8');

        console.log(`File XML đã được cập nhật tại: ${filePath}`);

    } catch (error) {

        console.error('Lỗi khi tạo hoặc cập nhật file XML: ', error.message);

        throw new Error('Không thể tạo hoặc cập nhật file XML');

    }

}

### Module bình luận (Lấy danh sách, chấp thuận, từ chối)

import fs from 'fs';

import path from 'path';

import { parseStringPromise,Builder } from 'xml2js';

export const listComment = async (commentList: any[],fileName: string = 'Comment.xml'): Promise<void> => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        try {

            await fs.promises.access(filePath, fs.constants.F\_OK);

            console.log(`File XML "${fileName}" đã tồn tại. Không cần tạo lại.`);

            return;

        } catch {

            console.log(`File XML "${fileName}" chưa tồn tại. Tạo file mới.`);

        }

        // Chuyển đổi dữ liệu bình luận thành cấu trúc XML

        const comments = commentList.map((comment) => ({

            $: { id: comment.idBinhLuan },

            noiDung: comment.noiDung,

            ngayBinhLuan: comment.ngayBinhLuan,

            tinhTrang: comment.tinhTrang,

            customer: {

                $: { id: comment.idNguoiDung },

                hoTen: comment.user.hoTen,

                avatar: comment.user.avatar,

                sdt: comment.user.sdt,

                gioiTinh: comment.user.gioiTinh,

            },

            food: {

                $: { id: comment.idSanPham },

                tenSanPham: comment.product.tenSanPham,

                images: comment.product.images,

            },

        }));

        const builder = new Builder({

            rootName: 'comments',

            xmldec: { version: '1.0', encoding: 'UTF-8' },

            renderOpts: { pretty: true, indent: '  ', newline: '\n' },

        });

        const xmlData = builder.buildObject({ comment: comments });

        // Ghi dữ liệu XML vào file

        await fs.promises.writeFile(filePath, xmlData, 'utf-8');

        console.log(`File XML đã được tạo tại: ${filePath}`);

    } catch (error) {

        console.error('Lỗi khi tạo file XML:', error.message);

        throw new Error('Không thể tạo file XML');

    }

}

export const approveComment =async (idComment:number,fileName: string = 'Comment.xml'): Promise<void> => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const parsedData = await parseStringPromise(xmlData);

        const comments = parsedData.comments.comment;

        let isUpdated = false;

        comments.forEach((comment: any) => {

            if (parseInt(comment.$.id) === idComment) {

                comment.tinhTrang = ['Đã chấp thuận'];

                isUpdated = true;

            }

        });

        if (!isUpdated) {

            throw new Error(`Không tìm thấy bình luận với id: ${idComment}`);

        }

        // Chuyển đổi lại object thành XML

        const builder = new Builder({

            xmldec: { version: '1.0', encoding: 'UTF-8' },

            renderOpts: { pretty: true, indent: '  ', newline: '\n' },

        });

        const updatedXML = builder.buildObject(parsedData);

        // Ghi lại file XML với dữ liệu đã được cập nhật

        await fs.promises.writeFile(filePath, updatedXML, 'utf-8');

        console.log(`Bình luận với id ${idComment} đã được cập nhật thành "Đã chấp thuận".`);

    } catch (error) {

        console.error('Lỗi khi cập nhật file XML:', error.message);

        throw new Error('Không thể cập nhật trạng thái bình luận.');

    }

}

export const disapproveComment =async (idComment:number,fileName: string = 'Comment.xml'): Promise<void> => {

    try {

        const filePath = path.join(\_\_dirname, '../../data', fileName);

        const xmlData = await fs.promises.readFile(filePath, 'utf-8');

        const parsedData = await parseStringPromise(xmlData);

        const comments = parsedData.comments.comment;

        let isUpdated = false;

        comments.forEach((comment: any) => {

            if (parseInt(comment.$.id) === idComment) {

                comment.tinhTrang = ['Đã từ chối'];

                isUpdated = true;

            }

        });

        if (!isUpdated) {

            throw new Error(`Không tìm thấy bình luận với id: ${idComment}`);

        }

        // Chuyển đổi lại object thành XML

        const builder = new Builder({

            xmldec: { version: '1.0', encoding: 'UTF-8' },

            renderOpts: { pretty: true, indent: '  ', newline: '\n' },

        });

        const updatedXML = builder.buildObject(parsedData);

        // Ghi lại file XML với dữ liệu đã được cập nhật

        await fs.promises.writeFile(filePath, updatedXML, 'utf-8');

        console.log(`Bình luận với id ${idComment} đã được cập nhật thành "Đã chấp thuận".`);

    } catch (error) {

        console.error('Lỗi khi cập nhật file XML:', error.message);

        throw new Error('Không thể cập nhật trạng thái bình luận.');

    }

}

## Giao diện chương trình

### Form Đăng nhập

~~Ảnh có chứa văn bản, ảnh chụp màn hình, thiết kế đồ họa

Mô tả được tạo tự động~~

### Form trang chủ:

~~Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, vòng tròn

Mô tả được tạo tự động~~

### Form quản lý nhân viên:

~~Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Mô tả được tạo tự động~~

### Form quản lý khách hàng:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Mô tả được tạo tự động

### Form quản lý danh mục:

~~Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Mô tả được tạo tự động~~

### Form quản lý món ăn:

~~Ảnh có chứa văn bản, ảnh chụp màn hình, Trang web, phần mềm

Mô tả được tạo tự động~~

### Form quản lý đơn hàng:

~~Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Mô tả được tạo tự động~~

### Form quản lí bình luận:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Mô tả được tạo tự động

# Kết luận

Đây là lần đầu tiên nhóm chúng em áp dụng công nghệ XML vào việc thiết kế, xây dựng phần mềm để quản lý ứng dụng đặt đồ ăn. Vận dụng kiến thức đã được học ở trường, kết hợp với những tài liệu tham khảo trên mạng , sách báo, và đặc biệt là sự hướng dẫn của thầy, cô nhóm thực hiện chúng em đã hoàn thành bài tập lớn. Qua bài báo chúng em muốn gửi lời cảm ơn giảng viên hướng dẫn ThS. Đỗ Lê Quân , rất cảm ơn thầy đã nhiệt tình dạy bảo và hướng dẫn chúng em hoàn thành bài tập lớn này. Vì kiến thức còn hạn chế, điều kiện về thời gian hơi gấp nên bài báo cáo không tránh khỏi thiếu sót. Vì thế, nhóm chúng em rất mong nhận được chỉ dẫn, bổ sung, sửa chữa, những đóng góp và sự cảm thông sâu sắc của thầy, cô cũng như bạn bè để bài tập của chúng em được hoàn thiện hơn. Nhóm chúng em xin chân thành cảm ơn!