HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG CƠ SỞ TẠI TP. HCM

KHOA CÔNG NGHỆ THÔNG TIN 2

BÁO CÁO MÔN KIẾN TRÚC VÀ THIẾT KẾ PHẦN MỀM

Đề tài: Xây dựng hệ thống bán sách tại cửa hàng

Giảng viên phụ trách: Thầy Nguyễn Văn Hữu Hoàng

Lớp: D21CQCNPM01 – N

Sinh viên thực hiện:

1. Nguyễn Ngọc Thiên Phúc – N21DCCN066
2. Trần Thị Thùy Ngân – N21DCCN055

TP. Hồ Chí Minh, ngày 31 tháng 05 năm 2025

BẢNG PHÂN CÔNG CÔNG VIỆC

Thành viên	Nhiệm vụ
Nguyễn Ngọc Thiên Phúc – N21DCCN066	– Thiết kế Cơ sở dữ liệu. – Xây dựng chức năng đăng ký, đăng nhập, đổi mật khẩu, quên mật khẩu. – Xây dựng Quản lý nhân viên, phân quyền cho nhân viên. – Xây dựng chức năng bán hàng. – Làm báo cáo.
Trần Thị Thùy Ngân – N21DCCN055	– Thu thập và phân tích yêu cầu. – Xây dựng Quản lý thể loại, tác giả, nhà xuất bản, sách. – Xây dựng Quản lý khuyến mãi, khách hàng.

LỜI CÁM ƠN

Trong thời đại số, nhu cầu xây dựng các hệ thống phần mềm ngày càng cao, đòi hỏi quy trình thiết kế, lập trình và triển khai cần được thực hiện một cách bài bản để đáp ứng yêu cầu thực tiễn. Để đáp ứng xu thế này, nhóm chúng em đã thực hiện đề tài "Xây dựng hệ thống bán sách tại cửa hàng".

Trong quá trình thực hiện đề tài, nhóm chúng em không chỉ củng cố kiến thức về thiết kế hệ thống và mô hình hóa nghiệp vụ, mà còn rèn luyện kỹ năng lập trình và triển khai phần mềm theo hướng chuyên nghiệp và thực tiễn hơn.

Chúng em xin chân thành cảm ơn thầy Nguyễn Văn Hữu Hoàng đã tận tình hướng dẫn, truyền đạt những kiến thức quý báu và luôn hỗ trợ, giải đáp mọi thắc mắc trong suốt quá trình làm đề tài. Dù đã nỗ lực hết mình, nhưng do kiến thức và kinh nghiệm còn hạn chế, nhóm em khó tránh khỏi những thiếu sót trong quá trình thực hiện. Rất mong thầy có thể góp ý, chỉ bảo để nhóm em hoàn thiện sản phẩm tốt hơn.

MỤC LỤC

DANH MỤC HÌNH ẢNH

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI

1. **Tên đề tài: "Xây dựng hệ thống bán sách tại cửa hàng"**
2. **Lý do chọn đề tài:**

Trong bối cảnh các cửa hàng truyền thống vẫn giữ vai trò quan trọng, việc tin học hóa quy trình bán sách tại quầy thu ngân góp phần tăng hiệu suất và tối ưu hóa quy trình hoạt động. Hệ thống bán sách tại cửa hàng sẽ giúc việc quản lý sách, khuyến mãi, khách hàng, và thanh toán trở nên nhanh chóng, minh bạch hơn.

3. **Mục tiêu nghiên cứu:**

- Tìm hiểu và phân tích quy trình nghiệp vụ thực tế tại cửa hàng sách.
- Thiết kế hệ thống backend theo kiến trúc RESTful API với ASP.NET Core và áp dụng Repository Design Pattern.
- Thiết kế cơ sở dữ liệu quan hệ trên nền tảng SQL Server.
- Xây dựng giao diện frontend dành cho nhân viên bán hàng bằng DevExpress trên Visual Studio.
- Triển khai và kiểm thử các chức năng quản lý sách, khuyến mãi, khách hàng, đơn hàng và thanh toán nhằm đảm bảo tính thực tiễn và hiệu quả sử dụng.

CHƯƠNG 2: PHÂN TÍCH, THIẾT KẾ HỆ THỐNG

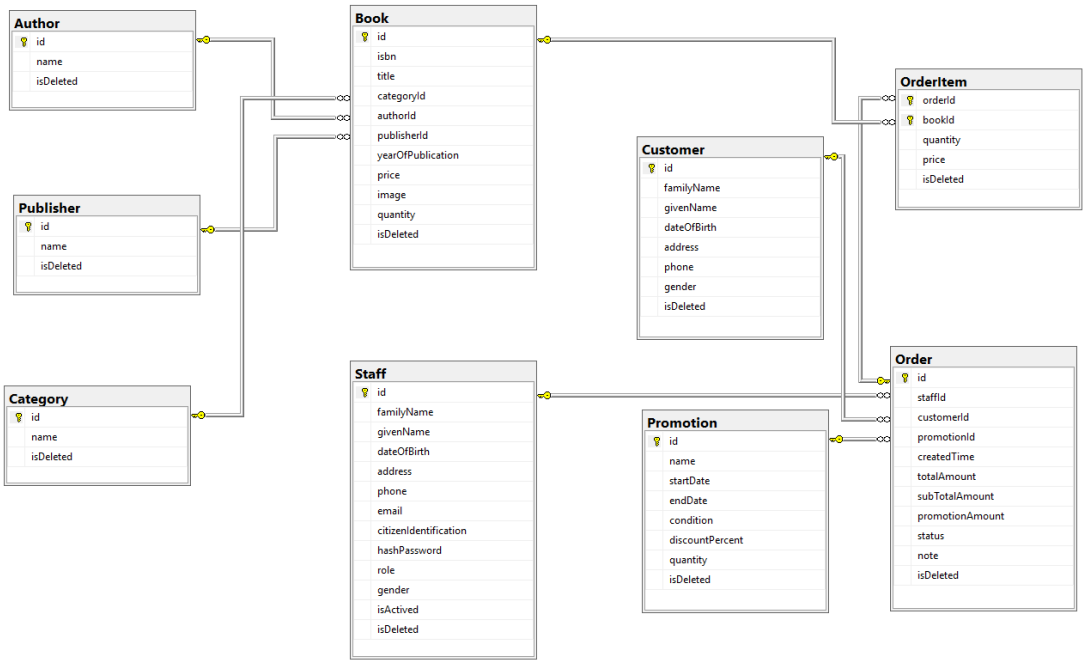
1. Mô tả hệ thống bằng ngôn ngữ tự nhiên:

Hệ thống bao gồm chức năng đăng ký, đăng nhập, đổi mật khẩu và lấy lại mật khẩu khi nhân viên quên mật khẩu thông qua hình thức gửi mail; quản lý danh mục sách, tác giả, nhà xuất bản; quản lý danh sách khách hàng; quản lý danh sách khuyến mãi; quản lý danh sách nhân viên; thêm đơn hàng khi khách mang sách ra quầy, chọn khuyến mãi nếu có, và tiến hành thanh toán. Toàn bộ tính năng backend sẽ cung cấp API cho giao diện frontend sử dụng DevExpress.

2. Công nghệ và tài nguyên sử dụng:

- Backend: ASP.NET Core MVC
- Repository Design Pattern + Entity Framework Core
- Frontend: DevExpress UI for .NET (trên Visual Studio)
- Database: Microsoft SQL Server
- IDE: Visual Studio 2022
- Ngôn ngữ: C#, Razor Page

3. Sơ đồ diagram:



Hình 1: Sơ đồ diagram

4. Từ điển dữ liệu:

– Bảng Category:

Category				
ID	**Attribute**	**Type**	**Constraint**	**Note**
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi tạo mới
2	name	nvarchar(100)	UK, NOT NULL	Tên thể loại
3	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

– Bảng Author:

Author				
ID	**Attribute**	**Type**	**Constraint**	**Note**
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi tạo mới
2	name	nvarchar(100)	NOT NULL	Tên tác giả (có thể trùng, đã có trường hợp trùng bút danh)
3	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

– Bảng Publisher:

Publisher				
ID	**Attribute**	**Type**	**Constraint**	**Note**
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi tạo mới
2	name	nvarchar(100)	UK, NOT NULL	Tên nhà xuất bản
3	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

– Bảng Book:

Book				
ID	**Attribute**	**Type**	**Constraint**	**Note**
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi tạo mới
2	isbn	char(13)	UK, NOT NULL	Mã isbn
3	title	nvarchar(100)	NOT NULL	Tên sách
4	categoryId	uniqueidentifier	FK, NOT NULL	Mã thể loại

5	authorId	uniqueidentifier	FK, NOT NULL	Mã tác giả
6	publisherId	uniqueidentifier	FK, NOT NULL	Mã nhà xuất bản
7	yearOfPublication	smallint	NOT NULL, > 1500	Năm xuất bản
8	price	decimal(8, 0)	NOT NULL, > 1000	Giá tiền ở thời điểm hiện tại
9	image	varchar(255)	NOT NULL	URL hình ảnh
10	quantity	int	NOT NULL, >= 0	Số lượng tồn
11	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

– **Bảng Customer:**

Customer				
ID	**Attribute**	**Type**	**Constraint**	**Note**
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi tạo mới
2	familyName	nvarchar(70)	NOT NULL	Họ và tên đệm
3	givenName	nvarchar(30)	NOT NULL	Tên khách hàng
4	dateOfBirth	date	NOT NULL	Ngày sinh
5	address	nvarchar(50)	NOT NULL	Địa chỉ
6	phone	char(10)	UK, NOT NULL	Số điện thoại
7	gender	bit	NOT NULL, DEFAULT = 0	Giới tính, 0 = nam, 1 = nữ
8	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

– **Bảng Staff:**

Staff				
ID	**Attribute**	**Type**	**Constraint**	**Note**
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi tạo mới
2	familyName	nvarchar(70)	NOT NULL	Họ và tên đệm
3	givenName	nvarchar(30)	NOT NULL	Tên
4	dateOfBirth	date	NOT NULL	Ngày sinh
5	address	nvarchar(50)	NOT NULL	Địa chỉ
6	phone	char(10)	UK, NOT NULL	Số điện thoại
7	email	varchar(50)	UK, NOT NULL	Địa chỉ mail
8	citizenIdentification	char(12)	UK, NOT NULL	Mã định danh

9	hashPassword	varchar(255)	NOT NULL	Mật khẩu mã hóa
10	role	bit	NOT NULL, DEFAULT = 0	Quyền, 0 = staff, 1 = admin
11	gender	bit	NOT NULL, DEFAULT = 0	Giới tính, 0 = nam, 1 = nữ
12	isActived	bit	NOT NULL, DEFAULT = 0	Trạng thái kích hoạt tài khoản, 0 = chưa, 1 = rồi
13	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

– **Bảng Promotion:**

Promotion				
ID	**Attribute**	**Type**	**Constraint**	**Note**
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi tạo mới
2	name	nvarchar(100)	UK, NOT NULL	Tên khuyến mãi, không được trùng
3	startDate	datetime2	NOT NULL	Ngày bắt đầu
4	endDate	datetime2	NOT NULL, > startDate	Ngày kết thúc
5	condition	decimal(8, 0)	NOT NULL, > 1000	Điều kiện tối thiểu
6	discountPercent	decimal(3, 2)	NOT NULL, > 0.0	Phần trăm khuyến mãi
7	quantity	smallint	NOT NULL, >= 0	Số lượng
8	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

– **Bảng Order:**

Order				
ID	**Attribute**	**Type**	**Constraint**	**Note**
1	id	uniqueidentifier	PK, DEFAULT = NEWID()	ID, tự động generate khi tạo mới
2	staffId	uniqueidentifier	FK, NOT NULL	Mã nhân viên
3	customerId	uniqueidentifier	FK, NOT NULL	Mã khách hàng

4	promotionId	uniqueidentifier	FK	Mã khuyến mãi
5	createdTime	datetime2	NOT NULL, DEFAULT = SYSDATETIME()	Thời gian tạo hóa đơn
6	totalAmount	decimal(11, 3)	NOT NULL	Tổng tiền khách phải trả
7	subTotalAmount	decimal(11, 3)	NOT NULL	Tiền trước khi giảm giá
8	promotionAmount	decimal(11, 3)	NOT NULL	Tiền giảm giá
9	status	bit	NOT NULL, DEFAULT = 1	Trạng thái hóa đơn, 0 = hóa đơn có lỗi, 1 = hóa đơn thanh toán thành công hoặc đã được giải quyết
10	note	nvarchar(MAX)		Ghi chú hóa đơn (khi gặp lỗi)
11	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

– **Bảng OrderItem:**

OrderItem				
ID	**Attribute**	**Type**	**Constraint**	**Note**
1	orderId	uniqueidentifier	PK, FK	Mã hóa đơn
2	bookId	uniqueidentifier		Mã sách
3	quantity	smallint	NOT NULL, > 0	Số lượng
4	price	decimal(8, 0)	NOT NULL, > 1000	Giá (ở thời điểm bán)
5	isDeleted	bit	NOT NULL, DEFAULT = 0	Trạng thái xóa, mặc định là false

CHƯƠNG 3: XÂY DỰNG VÀ PHÁT TRIỂN HỆ THỐNG

1. Xây dựng các API quan trọng:

- API đăng ký:

```
public async Task<bool> RegisterAsync(RegisterDTO registerDTO)
{
    var existingByEmail = await _authRepository.GetByEmailAsync(registerDTO.Email);
    var existingByPhone = await _authRepository.GetByPhoneAsync(registerDTO.Phone);
    var existingByCitizenIdentification = await
_authRepository.GetByCitizenIdentificationAsync(registerDTO.CitizenIdentification);

    if (existingByEmail != null)
        throw new InvalidOperationException("Email is already in use.");

    if (existingByPhone != null)
        throw new InvalidOperationException("Phone is already in use.");

    if (existingByCitizenIdentification != null)
        throw new InvalidOperationException("Citizen identification is already in
use.");

    if (registerDTO.Password != registerDTO.ConfirmPassword)
        throw new ArgumentException("Confirm password does not match password.");

    if (!IsOver18(registerDTO.DateOfBirth))
        throw new ArgumentException("You must be at least 18 years old.");

    string hashedPassword = BCrypt.Net.BCrypt.HashPassword(registerDTO.Password);

    var staff = new Staff
    {
        FamilyName = registerDTO.FamilyName,
        GivenName = registerDTO.GivenName,
        DateOfBirth = registerDTO.DateOfBirth,
        Address = registerDTO.Address,
        Phone = registerDTO.Phone,
        Email = registerDTO.Email,
        CitizenIdentification = registerDTO.CitizenIdentification,
        HashPassword = hashedPassword,
```

```
            Gender = registerDTO.Gender,
            Role = false,
            IsActived = false
        };

        await _authRepository.AddSync(staff);

        var saved = await _authRepository.SaveChangesAsync();

        if (saved == false)
            return false;

        var token = GenerateActivationToken(staff.Id);
        var activationLink = $"http://localhost:5208/api/auth/activate?token={token}";

        await _emailService.SendEmailAsync(staff.Email, "Activate your account",
            $"Click on the following link to activate your account.: <a
href='{activationLink}'>Verify account</a>");

        Console.WriteLine($"Send email confirm to: {staff.Email} successfully.");

        return true;
    }

    public async Task<bool> ActivateAccountAsync(string token)
    {
        var jwtKey = Environment.GetEnvironmentVariable("JWT_SECRET_KEY") ??
_config["Jwt:Key"];

        var handler = new JwtSecurityTokenHandler();

        var tokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new
SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtKey)),
            ValidateIssuer = false,
            ValidateAudience = false,
            ValidateLifetime = true
        };
```

```
    var principal = handler.ValidateToken(token, tokenValidationParameters, out var
validatedToken);
    var staffIdClaim = principal.FindFirst("staffId");
    if (staffIdClaim == null) return false;

    var staffId = Guid.Parse(staffIdClaim.Value);
    var staff = await _authRepository.GetByIdAsync(staffId);
    if (staff == null) return false;

    if (staff.IsActived) return false;

    staff.IsActived = true;
    return await _authRepository.SaveChangesAsync();
}


private string GenerateActivationToken(Guid staffId)
{
    var jwtKey = Environment.GetEnvironmentVariable("JWT_SECRET_KEY")
              ?? _config["Jwt:Key"]
              ?? throw new Exception("JWT key is missing");

    var jwtIssuer = _config["Jwt:Issuer"];
    var jwtAudience = _config["Jwt:Audience"];
    var expireMinutes = int.Parse(_config["Jwt:ExpireMinutes"] ?? "60");

    var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtKey));
    var credentials = new SigningCredentials(securityKey,
SecurityAlgorithms.HmacSha256);

    var claims = new[] { new Claim("staffId", staffId.ToString()) };

    var token = new JwtSecurityToken(
        issuer: jwtIssuer,
        audience: jwtAudience,
        claims: claims,
        expires: DateTime.UtcNow.AddMinutes(expireMinutes),
        signingCredentials: credentials
    );
```

```
    return new JwtSecurityTokenHandler().WriteToken(token);
}
```

- API đăng nhập:

```
public async Task<String?> LoginAsync(LoginDTO loginDTO)
{
    var staff = await _authRepository.GetByEmailAsync(loginDTO.Email);

    if (staff == null)
        throw new UnauthorizedAccessException("Invalid account, please check your
email or create new account.");

    if (staff.IsDeleted)
        throw new UnauthorizedAccessException("Your account has been deleted.
Please contact the administrator.");

    if (!staff.IsActived)
        throw new UnauthorizedAccessException("Your account has not been activated.
Please check your email to active.");

    bool isValidPassword = BCrypt.Net.BCrypt.Verify(loginDTO.Password,
staff.HashPassword);

    if (!isValidPassword)
        throw new UnauthorizedAccessException("Invalid password. Please try
again.");

    var token = GenerateJwtToken(staff);

    return token;
}

private string GenerateJwtToken(Staff user)
{
    var jwtKey = Environment.GetEnvironmentVariable("JWT_SECRET_KEY") ??
_config["Jwt:Key"];
    var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(jwtKey));
    var credentials = new SigningCredentials(securityKey,
SecurityAlgorithms.HmacSha256);
```

```
        var claims = new[]
        {
            new Claim(JwtRegisteredClaimNames.Sub, user.Email),
            new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),
            new Claim("staffId", user.Id.ToString()),
            new Claim("email", user.Email),
            new Claim(ClaimTypes.Role, user.Role ? "Admin" : "Staff")
        };

        var token = new JwtSecurityToken(
            issuer: _config["Jwt:Issuer"],
            audience: _config["Jwt:Audience"],
            claims: claims,
            expires: DateTime.UtcNow.AddMinutes(int.Parse(_config["Jwt:ExpireMinutes"]
    ?? "60")),
            signingCredentials: credentials
        );

        return new JwtSecurityTokenHandler().WriteToken(token);
    }
```

- API bán hàng:

```
    public async Task<bool> AddAsync(OrderCreateDTO orderCreateDTO, ClaimsPrincipal
    user)
    {
        if (orderCreateDTO.Items == null || !orderCreateDTO.Items.Any())
            throw new ArgumentException("Order must have at least one item.");

        var customer = await
    _orderRepository.GetCustomerByIdAsync(orderCreateDTO.CustomerId);
        if (customer == null || customer.IsDeleted)
            throw new ArgumentException("Customer is invalid or has been deleted.");

        var order = new Order
        {
            Id = Guid.NewGuid(),
            StaffId = CurrentUserHelper.GetStaffId(user),
            CustomerId = orderCreateDTO.CustomerId,
            PromotionId = orderCreateDTO.PromotionId,
```

```
            CreatedTime = DateTime.Now,
            Status = true
        };

        decimal subTotal = 0;

        foreach (var item in orderCreateDTO.Items)
        {
            var book = await _orderRepository.GetBookByIdAsync(item.BookId)
                ?? throw new KeyNotFoundException($"Book with id '{item.BookId}' not
found.");

            if (book.Quantity < item.Quantity)
                throw new InvalidOperationException($"The quantity of book with with
title '{book.Title}' not enough to buy.");

            if (book.IsDeleted)
                throw new InvalidOperationException($"The book with title
'{book.Title}' is deleted.");

            var orderItem = new OrderItem
            {
                OrderId = order.Id,
                BookId = item.BookId,
                Quantity = item.Quantity,
                Price = book.Price
            };

            book.Quantity -= item.Quantity;

            subTotal += (orderItem.Price * orderItem.Quantity);
            order.OrderItems.Add(orderItem);
        }

        order.SubTotalAmount = subTotal;

        if (order.PromotionId.HasValue)
        {
            var promotion = await
_orderRepository.GetPromotionByIdAsync(order.PromotionId.Value)
```

```
                ?? throw new ArgumentException("Promotion not found.");

            if (promotion.IsDeleted)
                throw new InvalidOperationException($"{promotion.Name} is deleted.");

            if (DateTime.Now < promotion.StartDate)
                throw new InvalidOperationException("Promotion has not started yet.");

            if (DateTime.Now > promotion.EndDate)
                throw new InvalidOperationException("Promotion has expired.");

            if (promotion.Quantity <= 0)
                throw new InvalidOperationException("Promotion has run out.");

            if (subTotal < promotion.Condition)
                throw new InvalidOperationException("Order does not meet promotion
condition.");

            order.PromotionAmount = subTotal * promotion.DiscountPercent;
            promotion.Quantity--;
        }

        order.TotalAmount = subTotal - order.PromotionAmount;

        await _orderRepository.AddAsync(order);

        return await _orderRepository.SaveChangesAsync();
    }
```

2. Áp dụng Design Pattern:

- Repository Design Pattern:

Trong quá trình phát triển hệ thống, nhóm đã áp dụng Repository Design Pattern nhằm tách biệt rõ ràng giữa tầng truy xuất dữ liệu (Data Access Layer) và tầng nghiệp vụ (Business Logic Layer). Điều này giúp hệ thống dễ mở rộng, dễ kiểm thử và tuân thủ nguyên lý Separation of Concerns trong thiết kế phần mềm.

Mỗi nhóm chức năng (AuthService, BookService, OrderService, ...) đều được tổ chức với cấu trúc:

+ **DTOs/**: Chứa các lớp trung gian để truyền dữ liệu giữa client và server.
+ **Interfaces/**: Định nghĩa các interface cho service.
+ **Repositories/**: Chứa các interface và lớp triển khai cụ thể cho repository để làm việc với Entity Framework thông qua ApplicationDbContext.

- **Data Transfer Object Design Pattern:**

Nhóm cũng áp dụng Data Transfer Object (DTO) Design Pattern nhằm tối ưu quá trình truyền dữ liệu giữa client và server, đồng thời đảm bảo tính bảo mật và hiệu quả trong giao tiếp giữa các tầng.

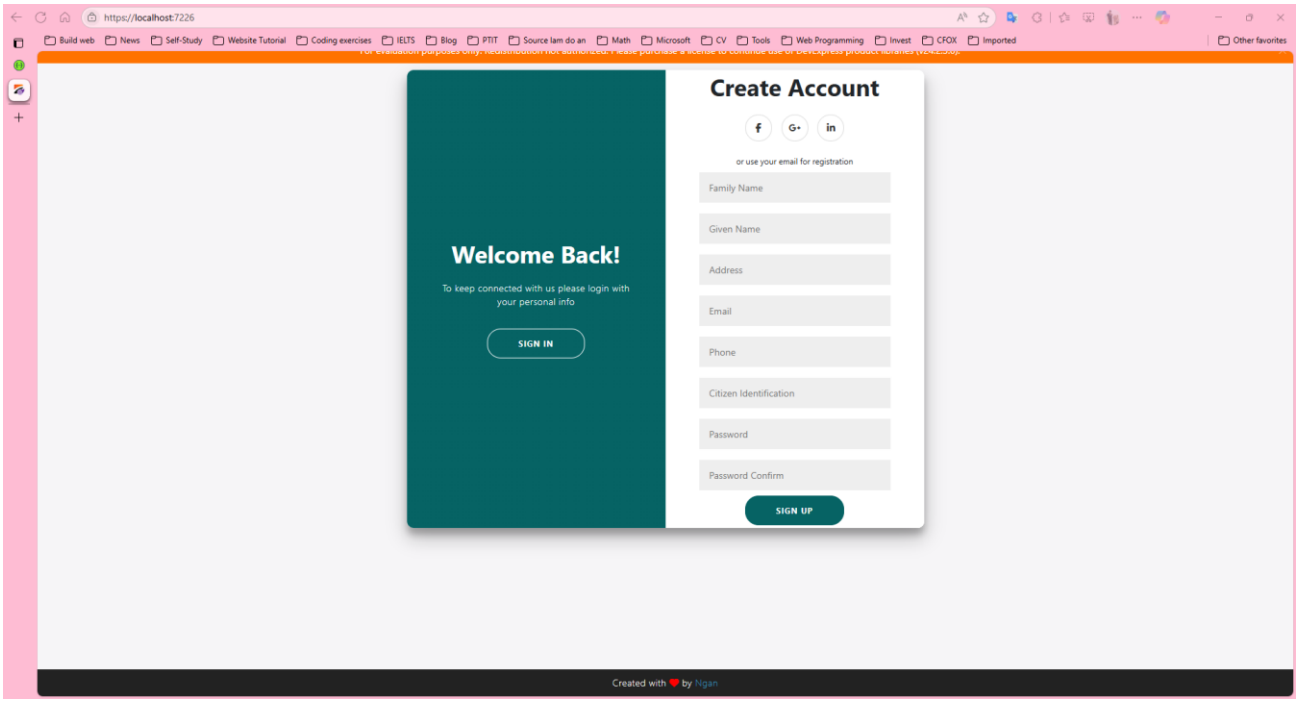
DTO là các lớp trung gian không chứa logic nghiệp vụ, được sử dụng để đóng gói và truyền tải thông tin cần thiết giữa frontend và backend mà không để lộ toàn bộ thông tin của entity trong cơ sở dữ liệu.

Cấu trúc thư mục dự án được tổ chức rõ ràng để hỗ trợ pattern này:

+ **DTOs/**: Chứa các lớp như LoginDTO, RegisterDTO, OrderDTO, ... được sử dụng để nhận và trả dữ liệu phù hợp cho từng API.
+ Việc sử dụng DTO giúp giảm thiểu dữ liệu dư thừa, tăng hiệu suất mạng và đảm bảo rằng client chỉ thao tác với những trường dữ liệu được cho phép.
+ Ngoài ra, DTO còn hỗ trợ kiểm soát luồng dữ liệu giữa UI và logic nghiệp vụ, tạo điều kiện thuận lợi cho việc kiểm thử và bảo trì hệ thống.

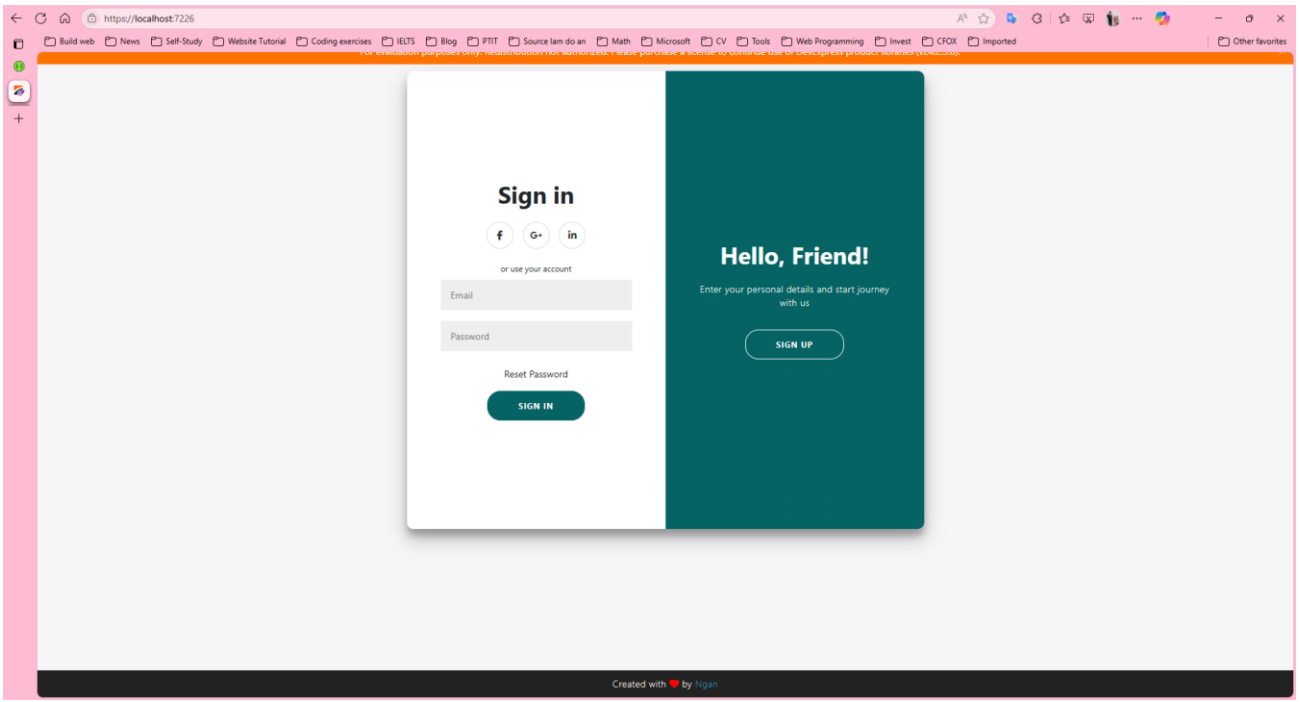
3. Giao diện:

- Giao diện đăng ký:



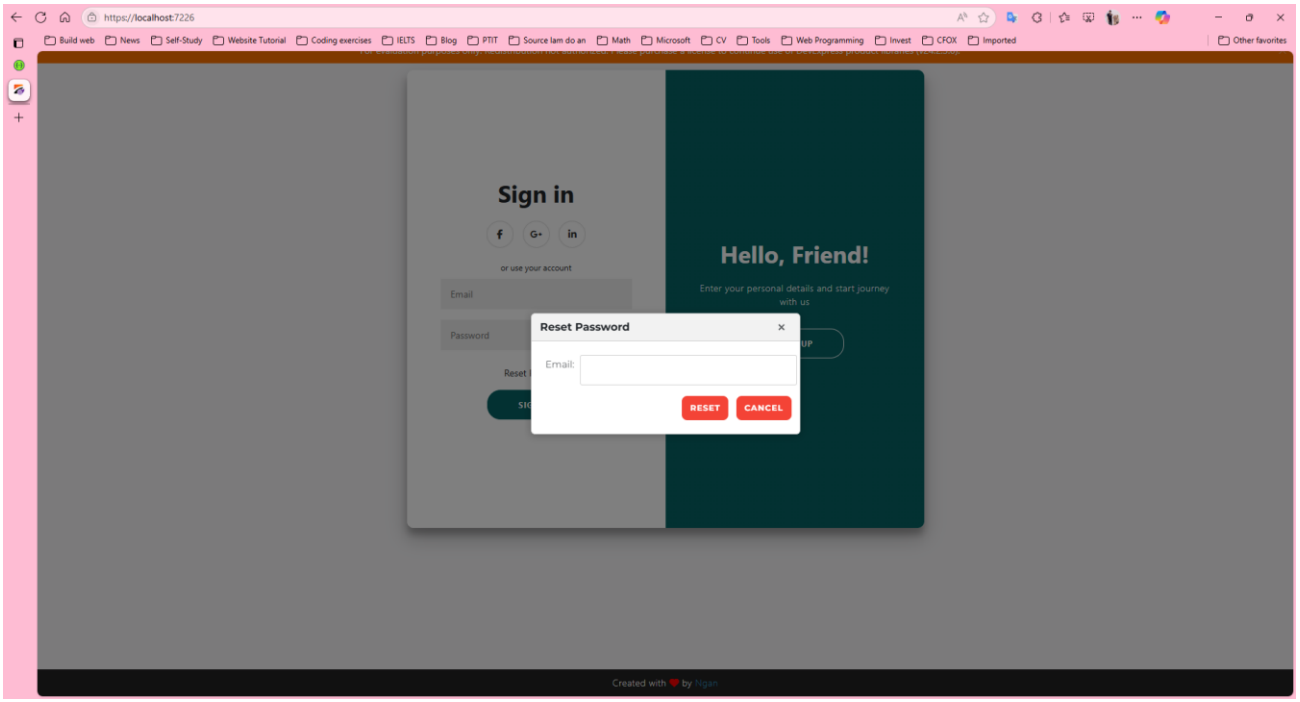
Hình 2: Giao diện đăng ký

- Giao diện đăng nhập:



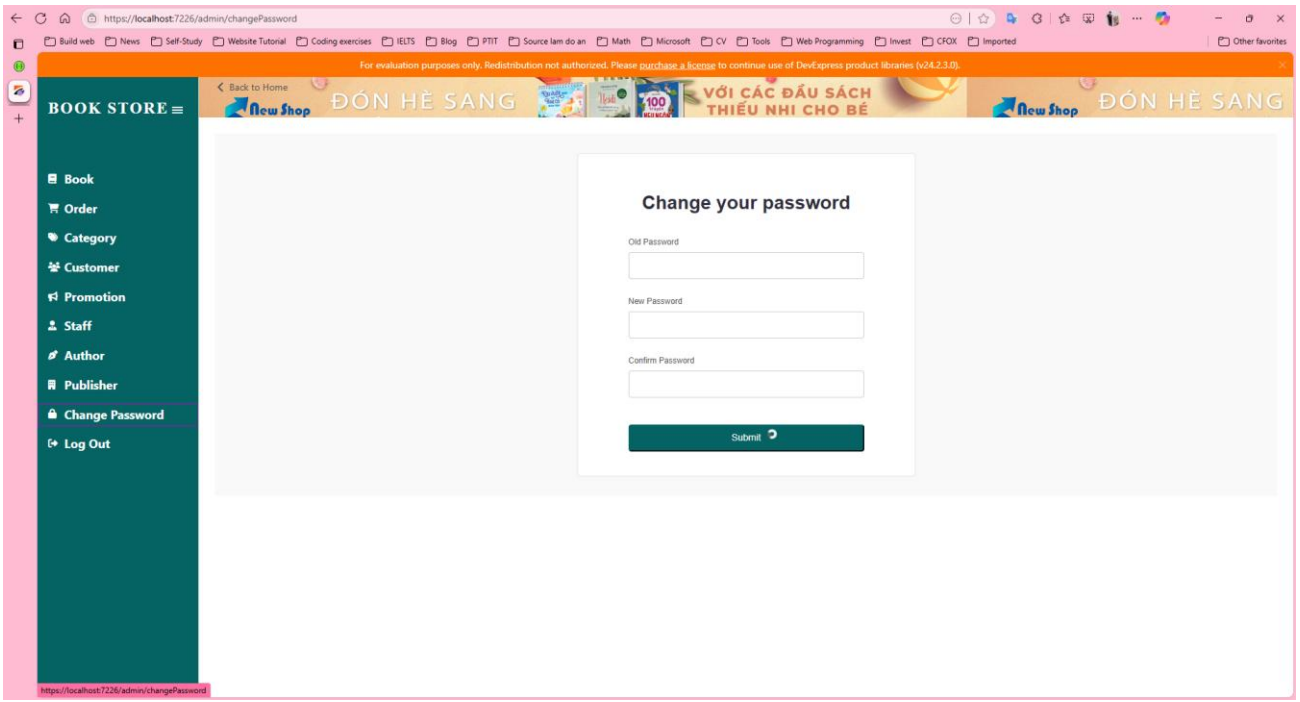
Hình 3: Giao diện đăng nhập

- Giao diện quên mật khẩu:



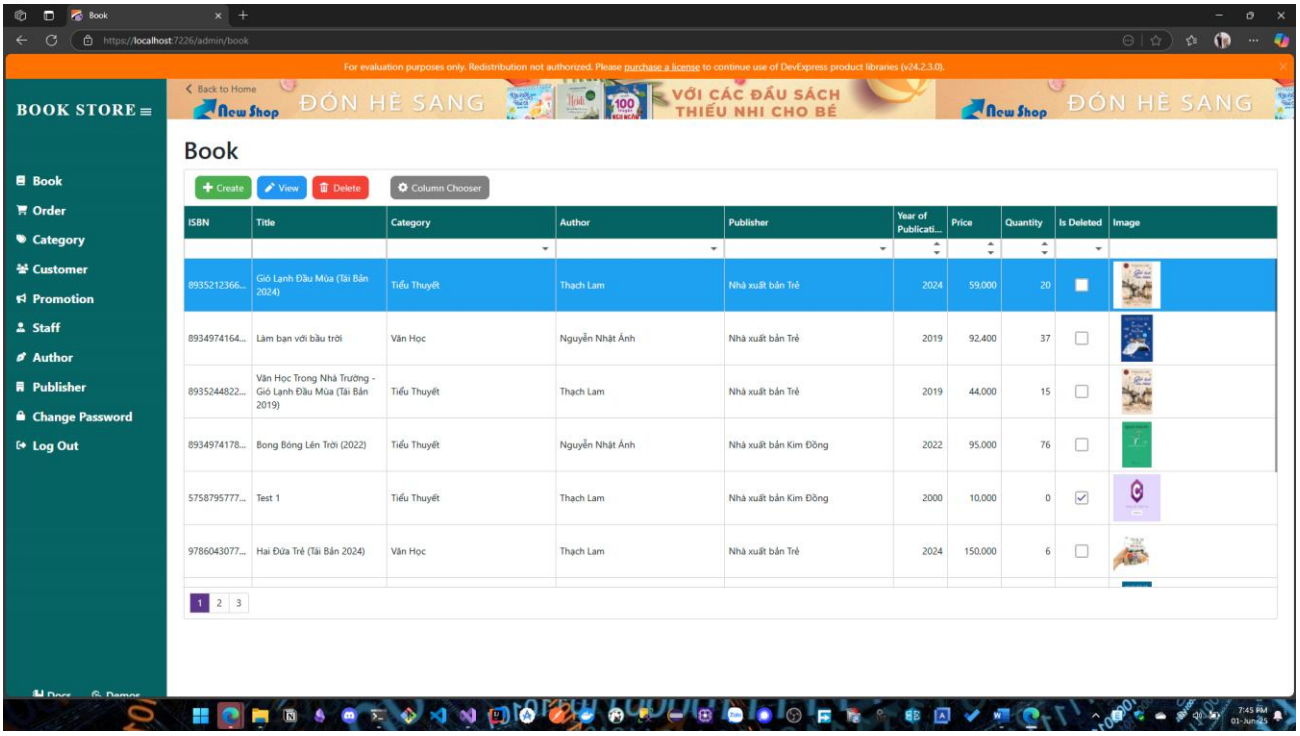
Hình 4: Giao diện quên mật khẩu

- Giao diện đổi mật khẩu:



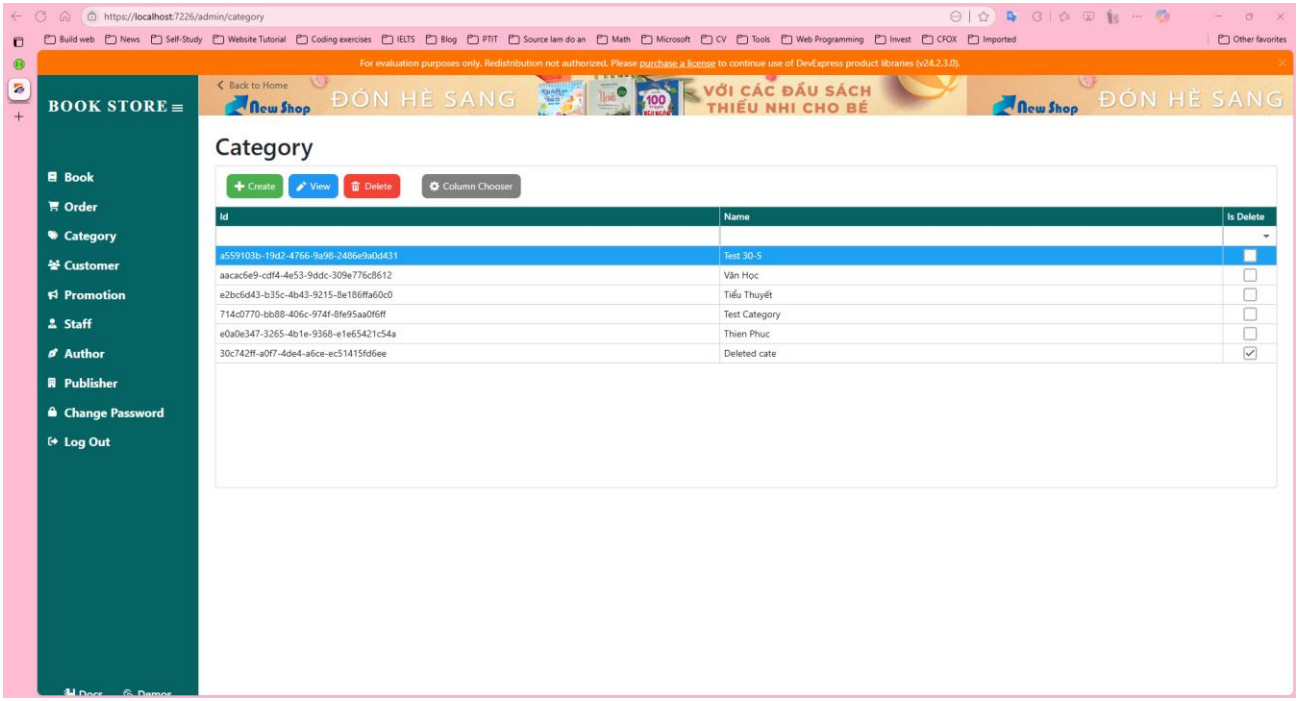
Hình 5: Giao diện đổi mật khẩu

- Giao diện quản lý sách:



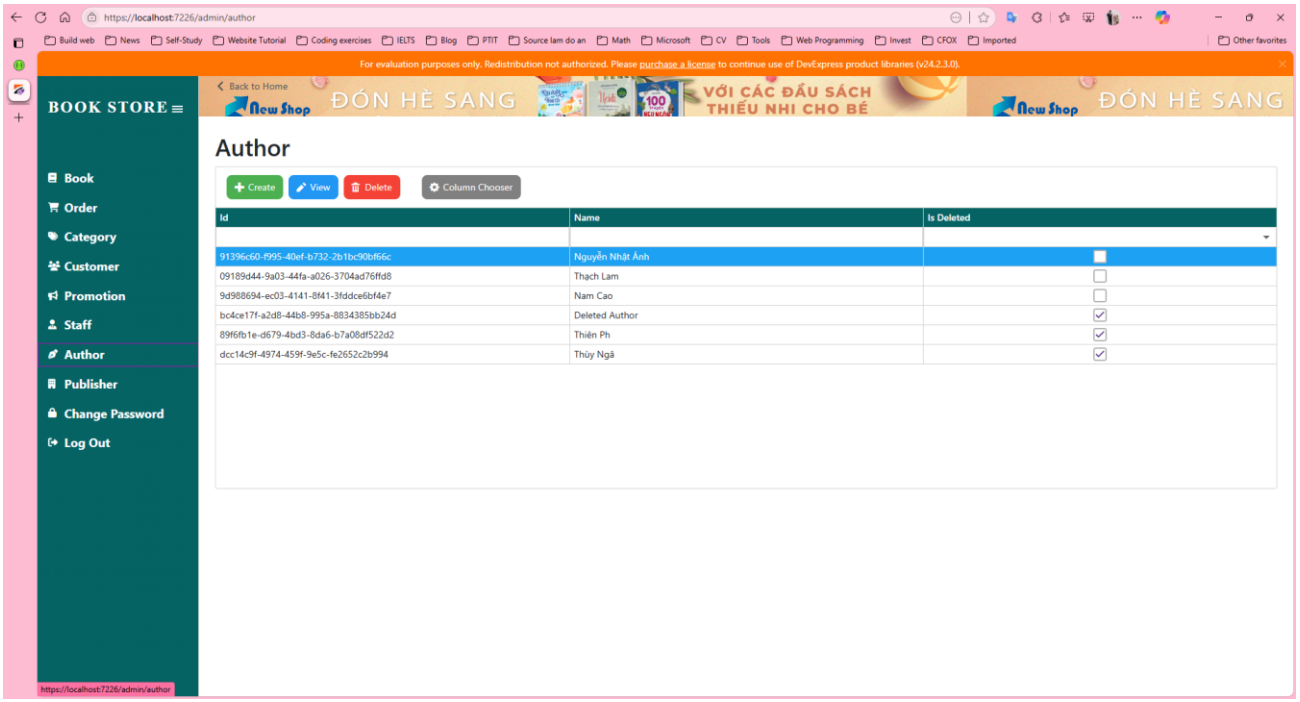
Hình 6: Giao diện quản lý sách

- Giao diện quản lý thể loại:



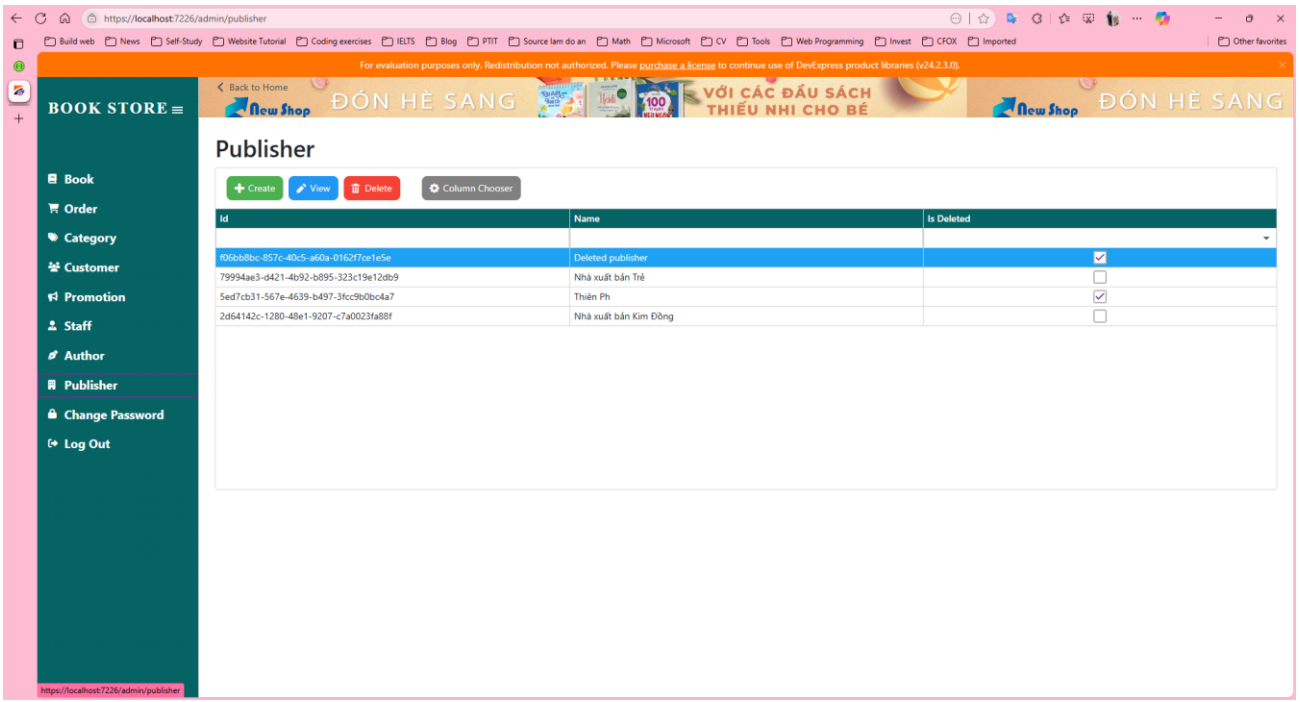
Hình 7: Giao diện quản lý thể loại

- Giao diện quản lý tác giả:



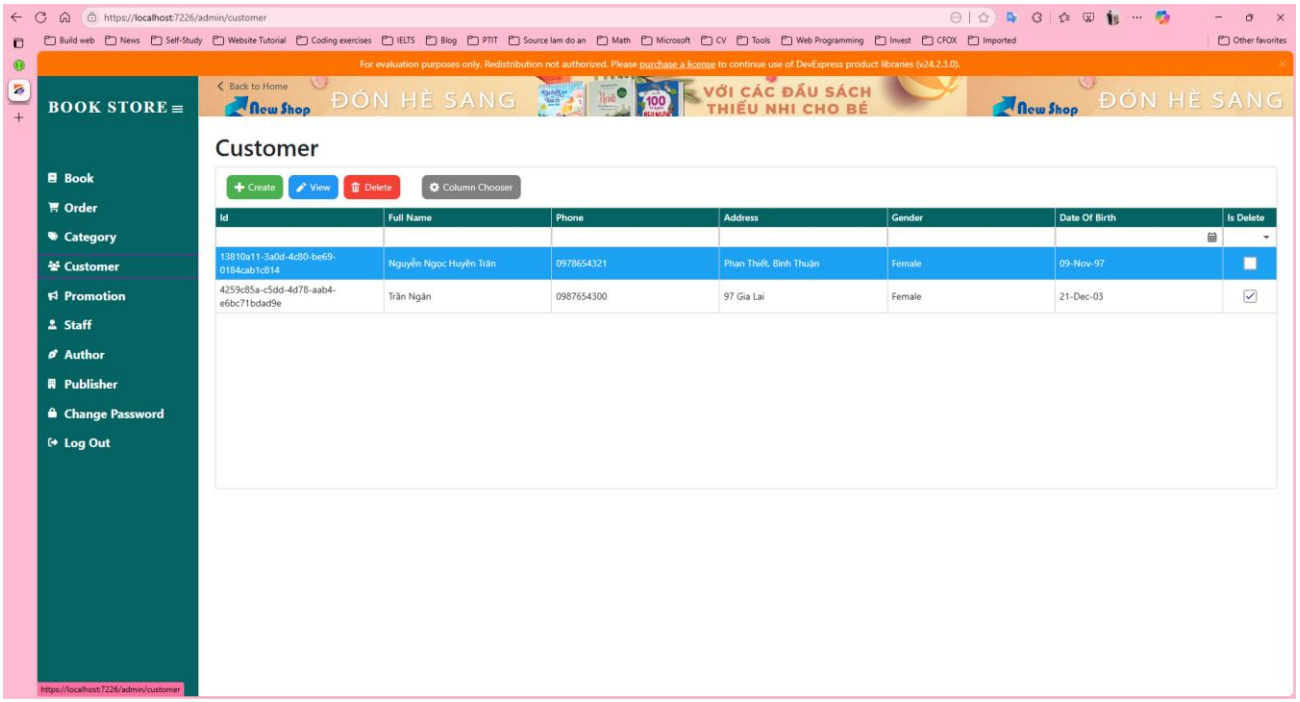
Hình 8: Giao diện quản lý tác giả

- Giao diện quản lý nhà xuất bản:



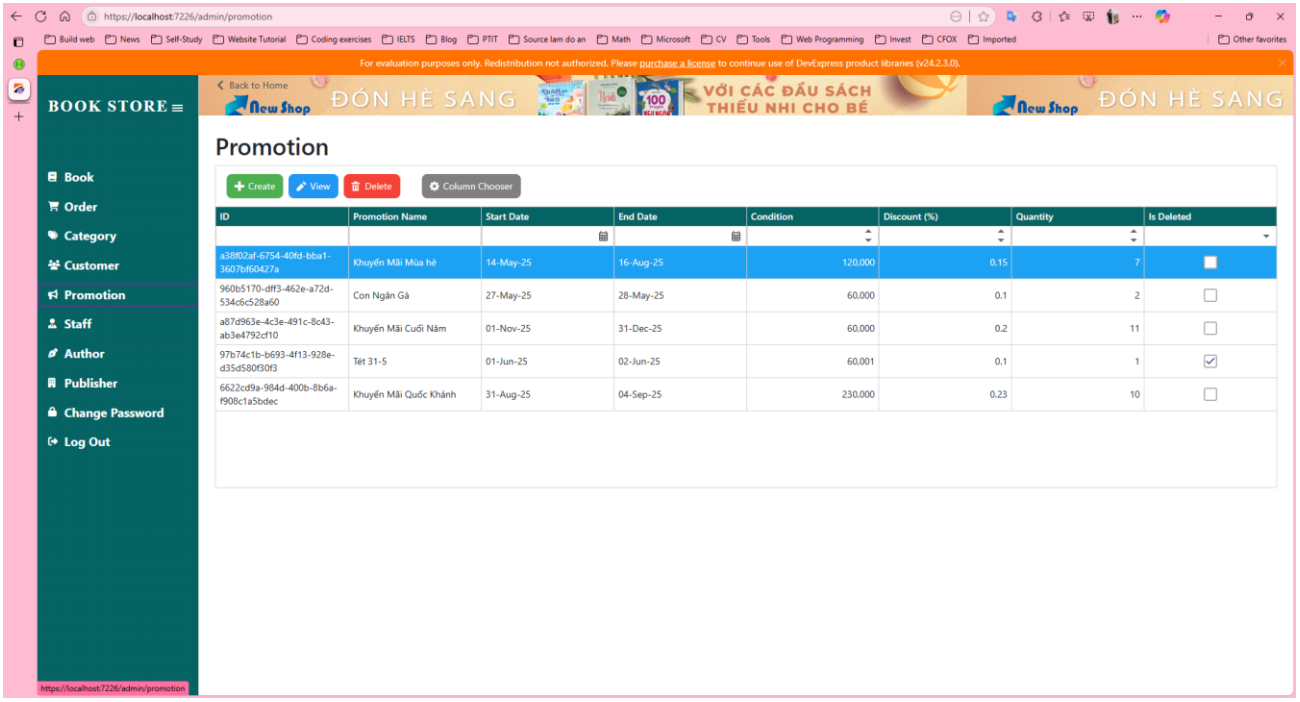
Hình 9: Giao diện quản lý nhà xuất bản

- Giao diện quản lý khách hàng:



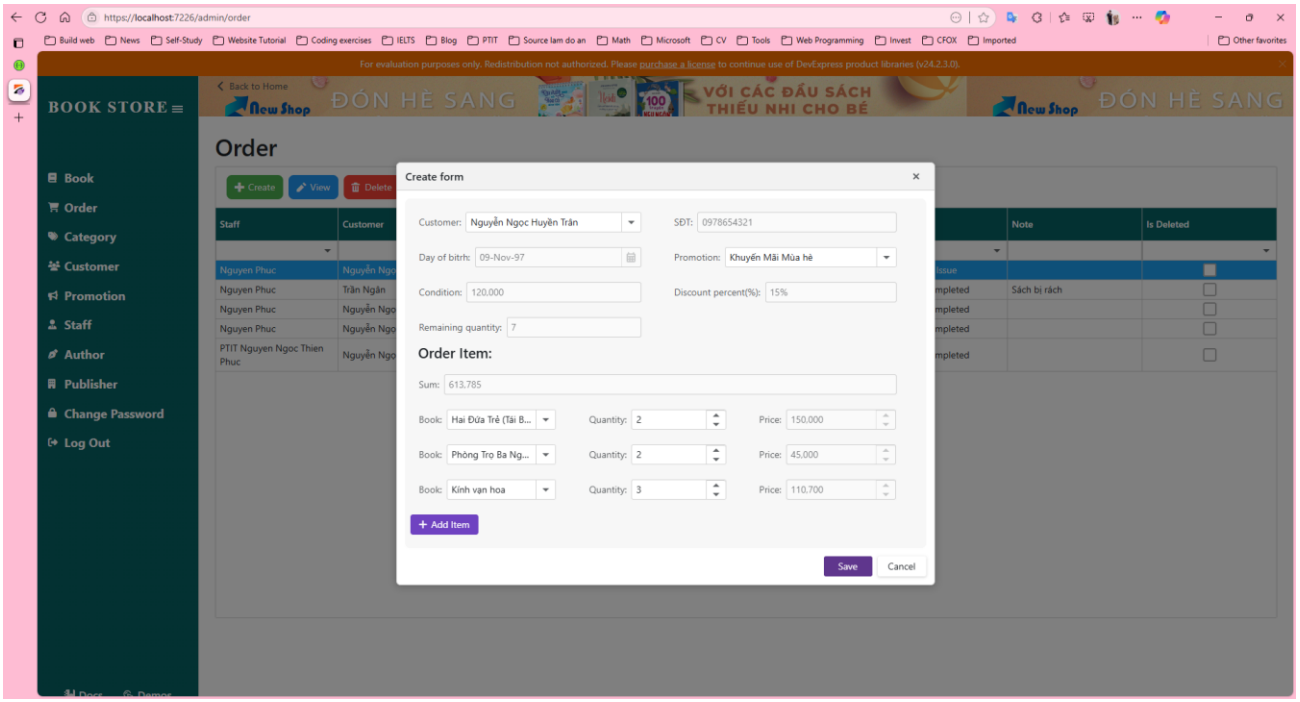
Hình 10: Giao diện quản lý khách hàng

- Giao diện quản lý khuyến mãi:

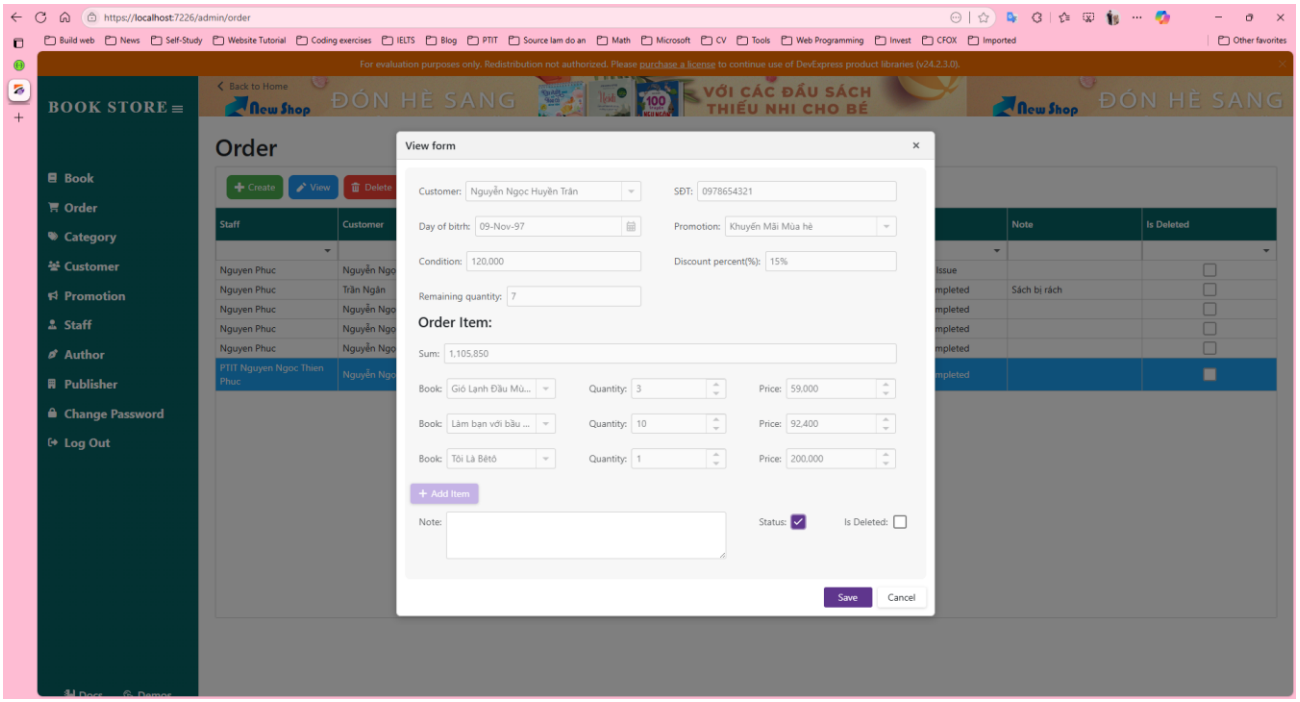


Hình 11: Giao diện quản lý khuyến mãi

- Giao diện quản lý bán hàng & hóa đơn:

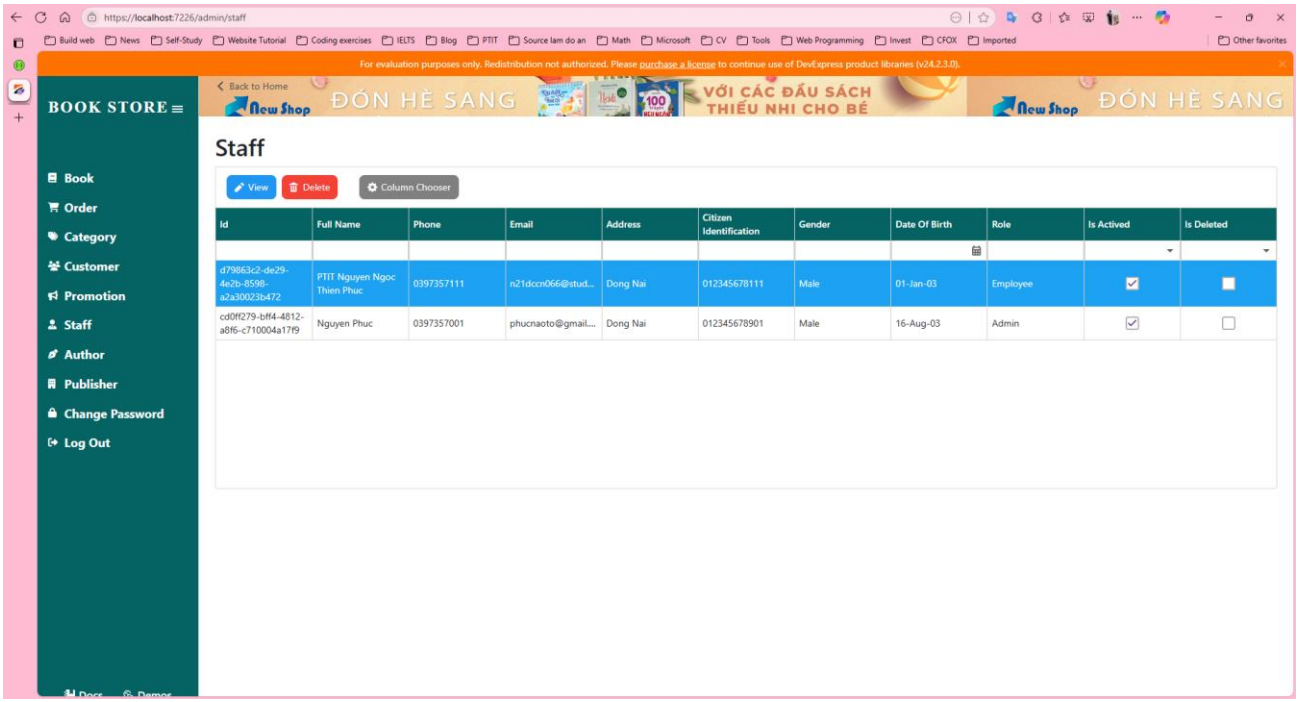


Hình 12: Giao diện tạo hóa đơn



Hình 13: Giao diện xem chi tiết hóa đơn

- Giao diện quản lý nhân viên:



Hình 14: Giao diện quản lý nhân viên

CHƯƠNG 4: KẾT LUẬN

1. Hạn chế:

- Giao diện: do sử dụng DevExpress nên còn hạn chế về mặt thẩm mỹ và khó mở rộng các chức năng theo mong muốn.
- API chưa được xây dựng đầy đủ cho tất cả các chức năng nâng cao như thống kê, báo cáo doanh thu, quản lý tồn kho chi tiết.
- Việc kiểm thử mới dừng lại ở mức chức năng cơ bản, chưa có kiểm thử hiệu năng hoặc bảo mật sâu.

2. Hướng phát triển trong tương lai:

- Xây dựng API hoàn chỉnh, hỗ trợ đầy đủ nghiệp vụ quản lý cửa hàng sách.
- Mở rộng khả năng tích hợp API với các nền tảng khác như ứng dụng Android, iOS, giúp đồng bộ hóa dữ liệu và tối ưu hóa quá trình bán hàng đa kênh.
- Phát triển thêm tính năng thống kê doanh thu, lượng tồn kho, và biểu đồ phân tích theo thời gian.
- Tích hợp cổng thanh toán điện tử (QR code, thẻ ngân hàng) để hiện đại hóa quy trình thanh toán.
- Tăng cường bảo mật hệ thống với xác thực 2 bước và mã hóa thông tin nhạy cảm.
- Nâng cấp giao diện người dùng (UI/UX) với các công nghệ frontend hiện đại như Blazor, React hoặc Angular để cải thiện trải nghiệm người dùng.