## **Các resource và ghi chú đính kèm:**

### ***a. Thư viện cần cho Entity Framework Core***

Nhớ cài 3 package sau khi làm việc với Entity Framework Core:

* Microsoft.EntityFrameworkCore
* Microsoft.EntityFrameworkCore.SqlServer
* Microsoft.EntityFrameworkCore.Tools

*Ngoài ra, các bạn chú ý version .NET Core = version Entity Framework Core (vd: cùng là v8)*

### ***b. Code lớp helper dùng cho mã hóa Hash (MD5, SHA512)***

using System.Security.Cryptography;

using System.Text;

namespace MyCommerce.Models

{

public static class DataEncryptionExtensions

{

#region [Hashing Extension]

public static string ToSHA256Hash(this string password, string? saltKey)

{

var sha256 = SHA256.Create();

byte[] encryptedSHA256 = sha256.ComputeHash(Encoding.UTF8.GetBytes(string.Concat(password, saltKey)));

sha256.Clear();

return Convert.ToBase64String(encryptedSHA256);

}

public static string ToSHA512Hash(this string password, string? saltKey)

{

SHA512Managed sha512 = new SHA512Managed();

byte[] encryptedSHA512 = sha512.ComputeHash(Encoding.UTF8.GetBytes(string.Concat(password, saltKey)));

sha512.Clear();

return Convert.ToBase64String(encryptedSHA512);

}

public static string ToMd5Hash(this string password, string? saltKey)

{

using (var md5 = MD5.Create())

{

byte[] data = md5.ComputeHash(Encoding.UTF8.GetBytes(string.Concat(password, saltKey)));

StringBuilder sBuilder = new StringBuilder();

for (int i = 0; i < data.Length; i++)

{

sBuilder.Append(data[i].ToString("x2"));

}

return sBuilder.ToString();

}

}

#endregion

}

}

Cách sử dụng: **<chuoi\_password>.ToMD5Hash()** hoặc **<chuoi\_password>.ToMD5Hash(“salt\_key”)**

### ***c. Thư viện AutoMapper dùng để map đối tượng từ kiểu này sang kiểu khác***

* Cần cài package: AutoMapper.Extensions.Microsoft.DependencyInjection
* Định nghĩa lớp Map giữa 2 kiểu dữ liệu:  
  public class AutoMapperProfile : Profile

{

public AutoMapperProfile()

{

CreateMap<RegisterVM, KhachHang>(); }

}

* Đăng ký sử dụng ở Program.cs:  
  *builder.Services.AddAutoMapper(typeof(AutoMapperProfile));*

Tham khảo thêm: <https://docs.automapper.org/en/stable/Dependency-injection.html>

### **d. Xác thực người dùng [Authentication]**

Có 2 cách xác thực: Cookie hay Token. Trong video này mình dùng Cookie cho web, bạn tham khảo thêm trên trang Microsoft: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/cookie?view=aspnetcore-8.0>

Sau khi xác thực thành công, bạn muốn protect action hay controller nào thì thêm thuộc tính [Authorize] vào trước action/controller đó.

### ***e. Thanh toán Paypal (JS SDK)***

**Chuẩn bị:**

* Đăng ký tài khoản Paypal
* Vào <https://developer.paypal.com/>
* Tạo tài khoản sandbox để demo: [Sandbox accounts - PayPal Developer](https://developer.paypal.com/dashboard/accounts) (gồm 2 loại Personal - người mua, Business - người bán)
* Tạo mới App ở mục **Apps & Credentials** [Applications - PayPal Developer](https://developer.paypal.com/dashboard/applications/sandbox) hoặc đứng ngay trong tài khoản Business để tạo.  
  Sau đó copy App ID, App secret key để dành để thêm vào project.

Tham khảo: <https://developer.paypal.com/sdk/js>

Ở trên trang Paypal docs toàn hướng dẫn cho JS không nên mình chuẩn bị sẵn cho các bạn một số lệnh:

Định nghĩa lớp **PaypalClient** để xác thực và tạo đơn hàng:

public sealed class PaypalClient

{

public string Mode { get; }

public string ClientId { get; }

public string ClientSecret { get; }

public string BaseUrl => Mode == "Live"

? "https://api-m.paypal.com"

: "https://api-m.sandbox.paypal.com";

public PaypalClient(string clientId, string clientSecret, string mode)

{

ClientId = clientId;

ClientSecret = clientSecret;

Mode = mode;

}

private async Task<AuthResponse> Authenticate()

{

var auth = Convert.ToBase64String(Encoding.UTF8.GetBytes($"{ClientId}:{ClientSecret}"));

var content = new List<KeyValuePair<string, string>>

{

new("grant\_type", "client\_credentials")

};

var request = new HttpRequestMessage

{

RequestUri = new Uri($"{BaseUrl}/v1/oauth2/token"),

Method = HttpMethod.Post,

Headers =

{

{ "Authorization", $"Basic {auth}" }

},

Content = new FormUrlEncodedContent(content)

};

var httpClient = new HttpClient();

var httpResponse = await httpClient.SendAsync(request);

var jsonResponse = await httpResponse.Content.ReadAsStringAsync();

var response = JsonSerializer.Deserialize<AuthResponse>(jsonResponse);

return response;

}

public async Task<CreateOrderResponse> CreateOrder(string value, string currency, string reference)

{

var auth = await Authenticate();

var request = new CreateOrderRequest

{

intent = "CAPTURE",

purchase\_units = new List<PurchaseUnit>

{

new()

{

reference\_id = reference,

amount = new Amount

{

currency\_code = currency,

value = value

}

}

}

};

var httpClient = new HttpClient();

httpClient.DefaultRequestHeaders.Authorization = AuthenticationHeaderValue.Parse($"Bearer {auth.access\_token}");

var httpResponse = await httpClient.PostAsJsonAsync($"{BaseUrl}/v2/checkout/orders", request);

var jsonResponse = await httpResponse.Content.ReadAsStringAsync();

var response = JsonSerializer.Deserialize<CreateOrderResponse>(jsonResponse);

return response;

}

public async Task<CaptureOrderResponse> CaptureOrder(string orderId)

{

var auth = await Authenticate();

var httpClient = new HttpClient();

httpClient.DefaultRequestHeaders.Authorization = AuthenticationHeaderValue.Parse($"Bearer {auth.access\_token}");

var httpContent = new StringContent("", Encoding.Default, "application/json");

var httpResponse = await httpClient.PostAsync($"{BaseUrl}/v2/checkout/orders/{orderId}/capture", httpContent);

var jsonResponse = await httpResponse.Content.ReadAsStringAsync();

var response = JsonSerializer.Deserialize<CaptureOrderResponse>(jsonResponse);

return response;

}

}

public sealed class AuthResponse

{

public string scope { get; set; }

public string access\_token { get; set; }

public string token\_type { get; set; }

public string app\_id { get; set; }

public int expires\_in { get; set; }

public string nonce { get; set; }

}

public sealed class CreateOrderRequest

{

public string intent { get; set; }

public List<PurchaseUnit> purchase\_units { get; set; } = new();

}

public sealed class CreateOrderResponse

{

public string id { get; set; }

public string status { get; set; }

public List<Link> links { get; set; }

}

public sealed class CaptureOrderResponse

{

public string id { get; set; }

public string status { get; set; }

public PaymentSource payment\_source { get; set; }

public List<PurchaseUnit> purchase\_units { get; set; }

public Payer payer { get; set; }

public List<Link> links { get; set; }

}

public sealed class PurchaseUnit

{

public Amount amount { get; set; }

public string reference\_id { get; set; }

public Shipping shipping { get; set; }

public Payments payments { get; set; }

}

public sealed class Payments

{

public List<Capture> captures { get; set; }

}

public sealed class Shipping

{

public Address address { get; set; }

}

public class Capture

{

public string id { get; set; }

public string status { get; set; }

public Amount amount { get; set; }

public SellerProtection seller\_protection { get; set; }

public bool final\_capture { get; set; }

public string disbursement\_mode { get; set; }

public SellerReceivableBreakdown seller\_receivable\_breakdown { get; set; }

public DateTime create\_time { get; set; }

public DateTime update\_time { get; set; }

public List<Link> links { get; set; }

}

public class Amount

{

public string currency\_code { get; set; }

public string value { get; set; }

}

public sealed class Link

{

public string href { get; set; }

public string rel { get; set; }

public string method { get; set; }

}

public sealed class Name

{

public string given\_name { get; set; }

public string surname { get; set; }

}

public sealed class SellerProtection

{

public string status { get; set; }

public List<string> dispute\_categories { get; set; }

}

public sealed class SellerReceivableBreakdown

{

public Amount gross\_amount { get; set; }

public PaypalFee paypal\_fee { get; set; }

public Amount net\_amount { get; set; }

}

public sealed class Paypal

{

public Name name { get; set; }

public string email\_address { get; set; }

public string account\_id { get; set; }

}

public sealed class PaypalFee

{

public string currency\_code { get; set; }

public string value { get; set; }

}

public class Address

{

public string address\_line\_1 { get; set; }

public string address\_line\_2 { get; set; }

public string admin\_area\_2 { get; set; }

public string admin\_area\_1 { get; set; }

public string postal\_code { get; set; }

public string country\_code { get; set; }

}

public sealed class Payer

{

public Name name { get; set; }

public string email\_address { get; set; }

public string payer\_id { get; set; }

}

public sealed class PaymentSource

{

public Paypal paypal { get; set; }

}

Phần appsettings.json thêm phần config dành cho Paypal:

"PaypalOptions": {

"Mode": "Sandbox",

"ClientId": "id",

"ClientSecret": "secret"

},

Khai báo sử dụng lớp PaypalClient ở Program.cs:

đăng ký PaypalClient dạng Singleton (có 1 thể hiện/instance duy nhất trong toàn ứng dụng)

builder.Services.AddSingleton(x =>

new PaypalClient(

builder.Configuration["PayPalOptions:ClientId"],

builder.Configuration["PayPalOptions:ClientSecret"],

builder.Configuration["PayPalOptions:Mode"]

)

);

Thực hiện thanh toán:

* Frontend: Hiện nút PAYPAL và khai báo các sự kiện createOrder, captureOrder
* Backend: Viết API cho thao tác CreateOrder và CaptureOrder.

### ***f. Thanh toán VNPAY***

[Hướng dẫn tích hợp Cổng thanh toán VNPAY · Cổng thanh toán VNPAY (vnpayment.vn)](https://sandbox.vnpayment.vn/apis/docs/huong-dan-tich-hop/)

[Kết nối Cổng thanh toán VNPAY · Cổng thanh toán VNPAY (vnpayment.vn)](https://sandbox.vnpayment.vn/apis/)

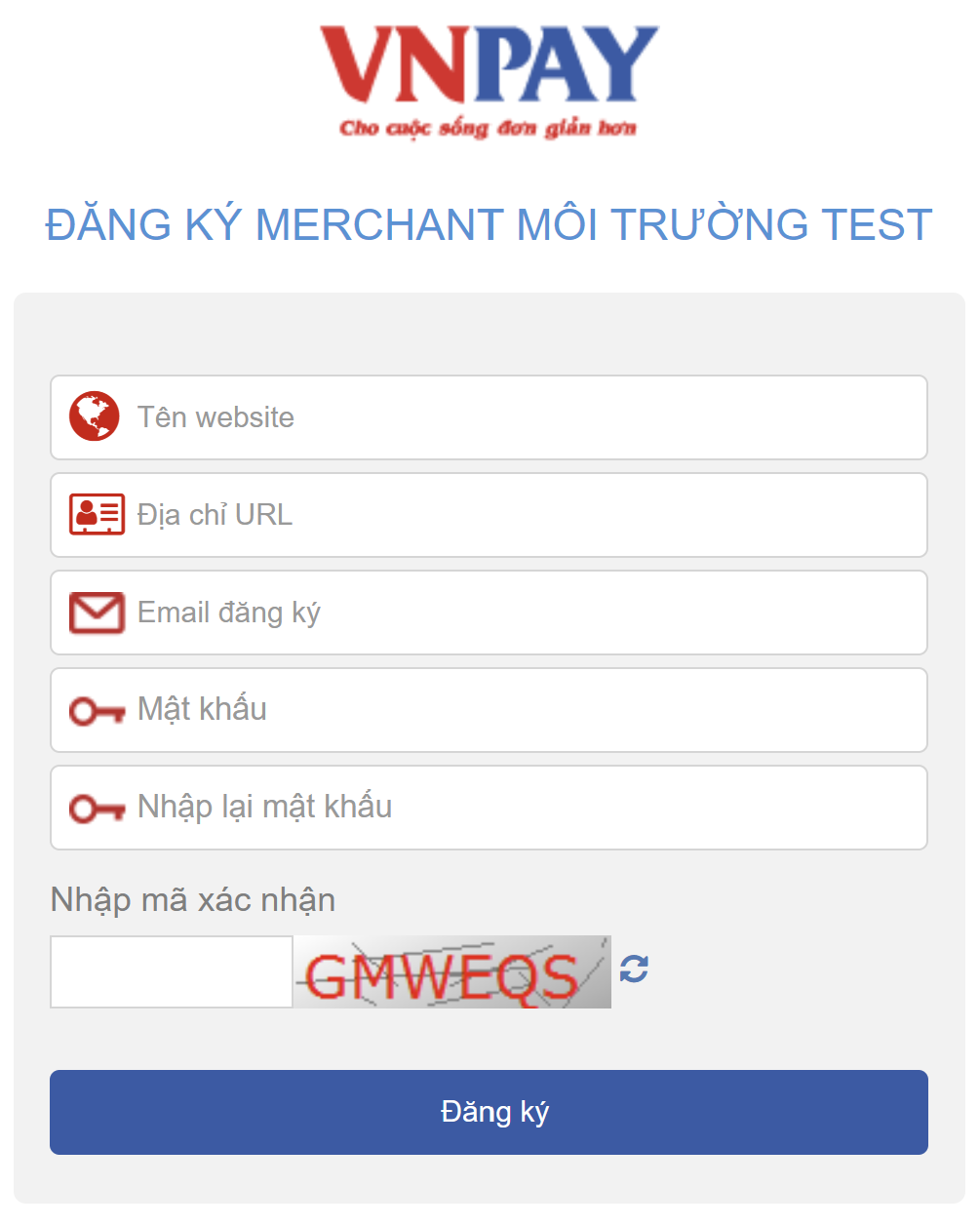
VNPAY cung cấp sẵn ví dụ nhưng C# WebForm: [Download · Cổng thanh toán VNPAY (vnpayment.vn)](https://sandbox.vnpayment.vn/apis/downloads/)

Thông tin thẻ demo: [Demo Cổng thanh toán VNPAY · Cổng thanh toán VNPAY (vnpayment.vn)](https://sandbox.vnpayment.vn/apis/vnpay-demo/)

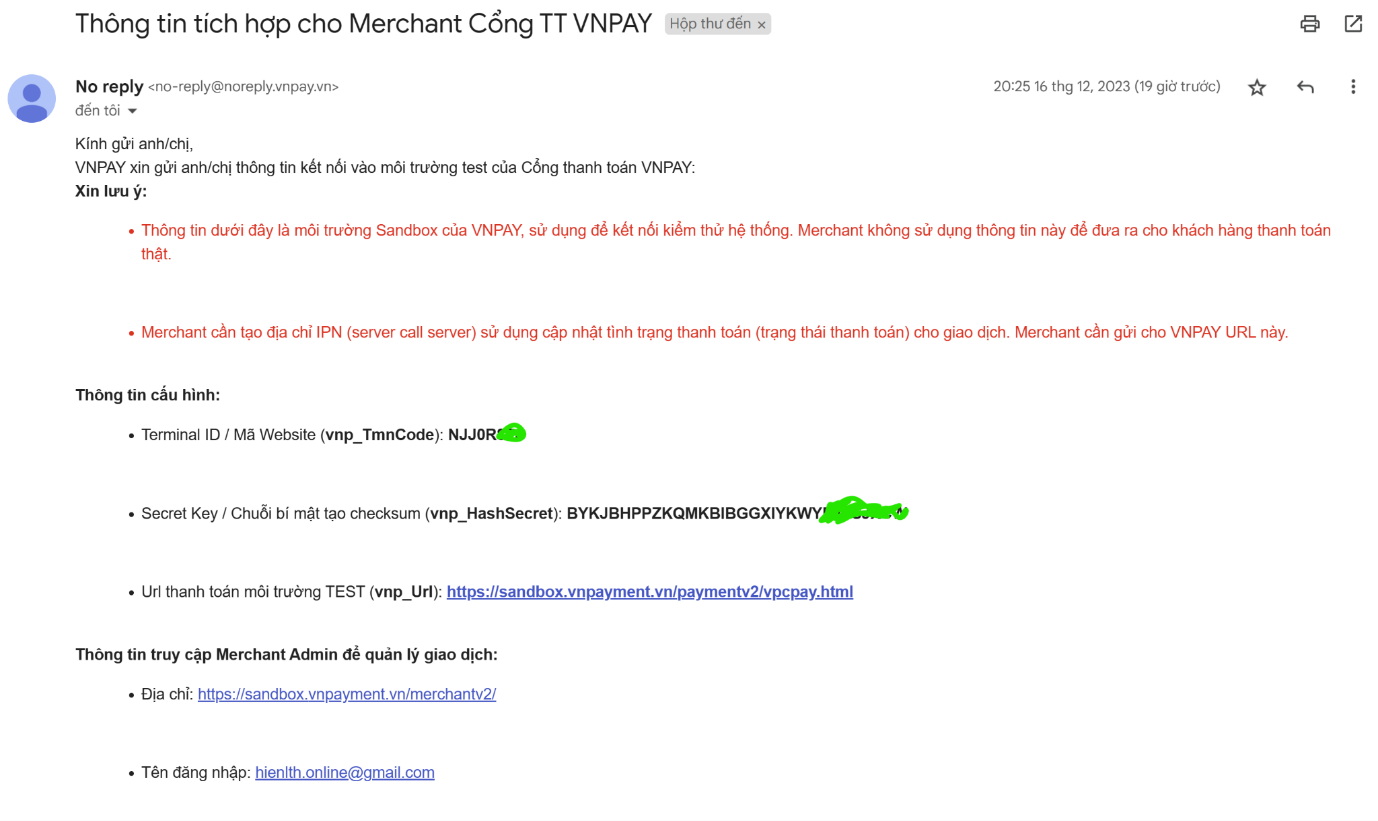
Bảng mã lỗi khi thanh toán: [Bảng mã lỗi · Cổng thanh toán VNPAY (vnpayment.vn)](https://sandbox.vnpayment.vn/apis/docs/bang-ma-loi/)

***Đăng ký***

Đăng ký tài khoản VNPAY tại địa chỉ: <https://sandbox.vnpayment.vn/devreg/>



Sau đó bạn check email để kích hoạt tài khoản và nhận 01 email chứa thông tin tích hợp:



***Tích hợp***

Thông tin lớp **VnPayLibrary** (lấy từ ví dụ mẫu nhưng có chỉnh sửa):

public class VnPayLibrary

{

private readonly SortedList<string, string> \_requestData = new SortedList<string, string>(new VnPayCompare());

private readonly SortedList<string, string> \_responseData = new SortedList<string, string>(new VnPayCompare());

public void AddRequestData(string key, string value)

{

if (!string.IsNullOrEmpty(value))

{

\_requestData.Add(key, value);

}

}

public void AddResponseData(string key, string value)

{

if (!string.IsNullOrEmpty(value))

{

\_responseData.Add(key, value);

}

}

public string GetResponseData(string key)

{

return \_responseData.TryGetValue(key, out var retValue) ? retValue : string.Empty;

}

#region Request

public string CreateRequestUrl(string baseUrl, string vnpHashSecret)

{

var data = new StringBuilder();

foreach (var (key, value) in \_requestData.Where(kv => !string.IsNullOrEmpty(kv.Value)))

{

data.Append(WebUtility.UrlEncode(key) + "=" + WebUtility.UrlEncode(value) + "&");

}

var querystring = data.ToString();

baseUrl += "?" + querystring;

var signData = querystring;

if (signData.Length > 0)

{

signData = signData.Remove(data.Length - 1, 1);

}

var vnpSecureHash = Utils.HmacSHA512(vnpHashSecret, signData);

baseUrl += "vnp\_SecureHash=" + vnpSecureHash;

return baseUrl;

}

#endregion

#region Response process

public bool ValidateSignature(string inputHash, string secretKey)

{

var rspRaw = GetResponseData();

var myChecksum = Utils.HmacSHA512(secretKey, rspRaw);

return myChecksum.Equals(inputHash, StringComparison.InvariantCultureIgnoreCase);

}

private string GetResponseData()

{

var data = new StringBuilder();

if (\_responseData.ContainsKey("vnp\_SecureHashType"))

{

\_responseData.Remove("vnp\_SecureHashType");

}

if (\_responseData.ContainsKey("vnp\_SecureHash"))

{

\_responseData.Remove("vnp\_SecureHash");

}

foreach (var (key, value) in \_responseData.Where(kv => !string.IsNullOrEmpty(kv.Value)))

{

data.Append(WebUtility.UrlEncode(key) + "=" + WebUtility.UrlEncode(value) + "&");

}

//remove last '&'

if (data.Length > 0)

{

data.Remove(data.Length - 1, 1);

}

return data.ToString();

}

#endregion

}

public class Utils

{

public static string HmacSHA512(string key, string inputData)

{

var hash = new StringBuilder();

var keyBytes = Encoding.UTF8.GetBytes(key);

var inputBytes = Encoding.UTF8.GetBytes(inputData);

using (var hmac = new HMACSHA512(keyBytes))

{

var hashValue = hmac.ComputeHash(inputBytes);

foreach (var theByte in hashValue)

{

hash.Append(theByte.ToString("x2"));

}

}

return hash.ToString();

}

// có chế biến cho .NET Core MVC

public static string GetIpAddress(HttpContext context)

{

var ipAddress = string.Empty;

try

{

var remoteIpAddress = context.Connection.RemoteIpAddress;

if (remoteIpAddress != null)

{

if (remoteIpAddress.AddressFamily == AddressFamily.InterNetworkV6)

{

remoteIpAddress = Dns.GetHostEntry(remoteIpAddress).AddressList

.FirstOrDefault(x => x.AddressFamily == AddressFamily.InterNetwork);

}

if (remoteIpAddress != null) ipAddress = remoteIpAddress.ToString();

return ipAddress;

}

}

catch (Exception ex)

{

return "Invalid IP:" + ex.Message;

}

return "127.0.0.1";

}

}

public class VnPayCompare : IComparer<string>

{

public int Compare(string x, string y)

{

if (x == y) return 0;

if (x == null) return -1;

if (y == null) return 1;

var vnpCompare = CompareInfo.GetCompareInfo("en-US");

return vnpCompare.Compare(x, y, CompareOptions.Ordinal);

}

}

Mô tả lớp kết quả trả về khi thanh toán:

public class **VnPaymentResponseModel**

{

public bool Success { get; set; }

public string PaymentMethod { get; set; }

public string OrderDescription { get; set; }

public string OrderId { get; set; }

public string PaymentId { get; set; }

public string TransactionId { get; set; }

public string Token { get; set; }

public string VnPayResponseCode { get; set; }

}