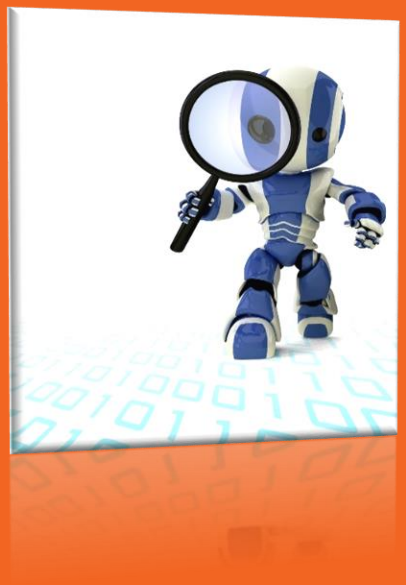
FPT Education

FPT POLYTECHNIC



LẬP TRÌNH C# 5

BÀI 7: JSON WEB TOKEN - SECURE API ENDPOINT

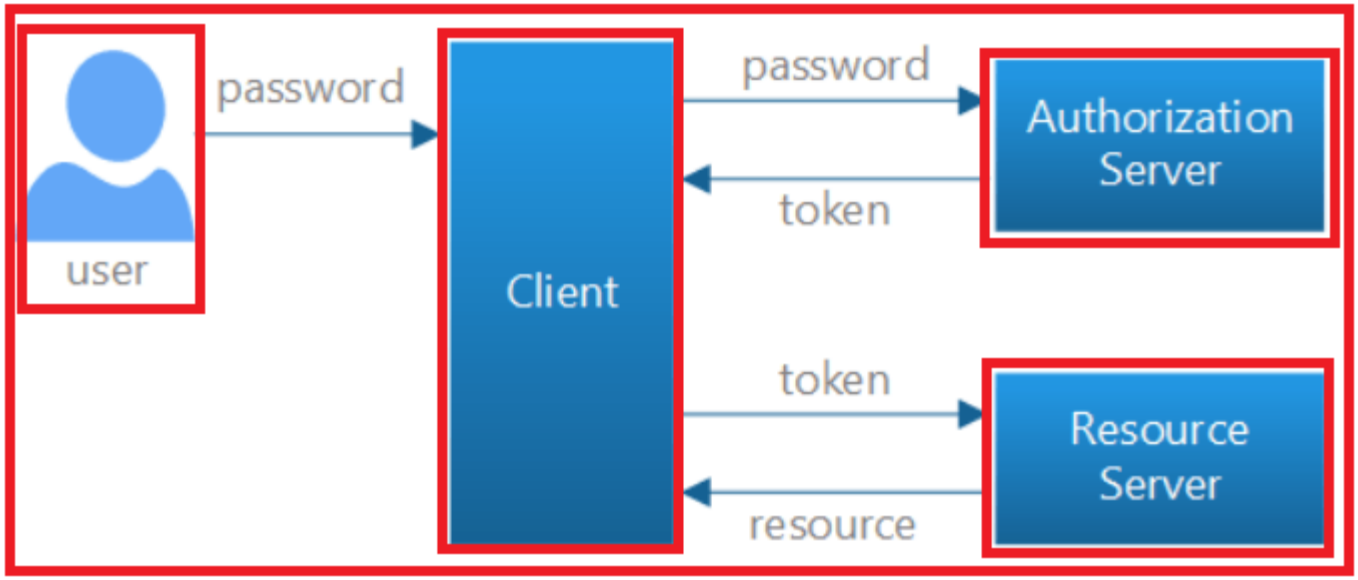
www.poly.edu.vn

Mục tiêu

- Token-based authentication
- JSON Web Token (JWT)
- Secure API endpoint - jwt

TOKEN-BASED AUTHENTICATION

- Token-based authentication là phương thức xác thực bằng chuỗi má hóa
- Một hệ thống sử dụng Token-based authentication cho phép người dùng nhập user/password để nhận về 1 chuỗi token. Chuỗi Token này được sử dụng để "xác minh" quyền truy cập vào tài nguyên mà không cần phải cung cấp lại username/password nữa.

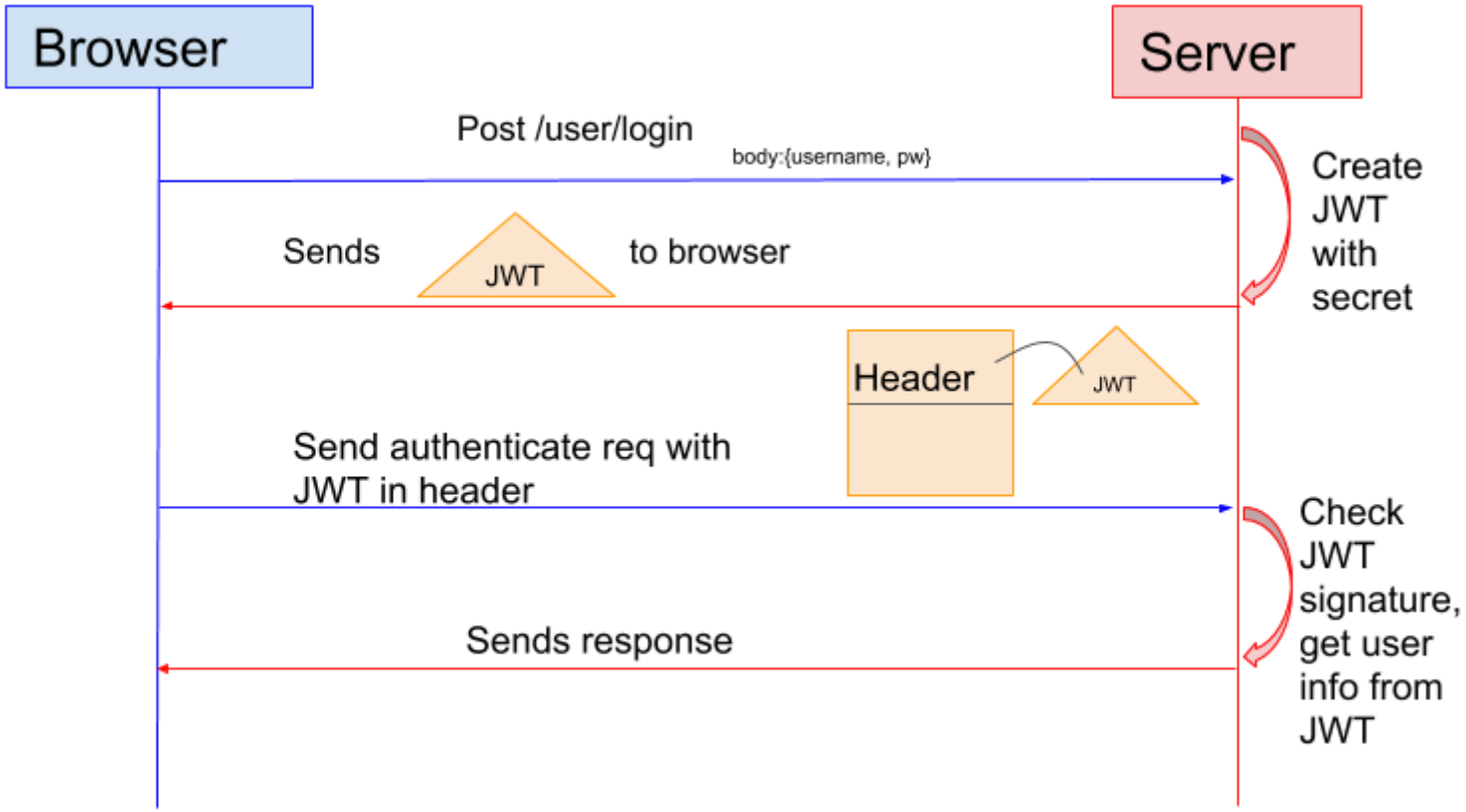


JSON Web Token (JWT)

- Là một phương pháp sử dụng Token-based authentication
- JSON Web Token (JWT) là 1 tiêu chuẩn mở (RFC 7519) định nghĩa cách thức truyền tin an toàn giữa các thành viên bằng 1 đối tượng JSON. Thông tin này có thể được xác thực và đánh dấu tin cậy nhờ nó có chứa chữ ký số (digital signature). Phần chữ ký của JWT sẽ được mã hóa lại bằng HMAC hoặc RSA
- JWT là một phương tiện đại diện cho các yêu cầu chuyển giao giữa hai bên Client – Server, các thông tin trong chuỗi JWT được định dạng bằng JSON

JSON Web Token (JWT)

- Bảo mật JWT là phương pháp xác thực quyền truy cập (Authentication) bằng JSON Web Token



JSON Web Token (JWT)

- Chuỗi Token phải có 3 phần là header , phần payload và phần signature được ngăn bằng dấu "."

1 2

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.XbPfbIHMI6arZ3Y922BhjWgQzWXcXNrz0ogtVhfEd2o 3

1 Header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

2 Payload

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

3 Signature

```
HMACSHA256(
BASE64URL(header)
.
BASE64URL(payload),
secret)
```

JSON Web Token (JWT)

- Header: Phần header sẽ chứa kiểu dữ liệu , và thuật toán sử dụng để mã hóa ra chuỗi JWT
  - *typ* – Loại token (mặc định là JWT – cho biết đây là một Token JWT)
  - alg – Thuật toán đã dùng để mã hóa (HMAC SHA256 – HS256 hoặc RSA).

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

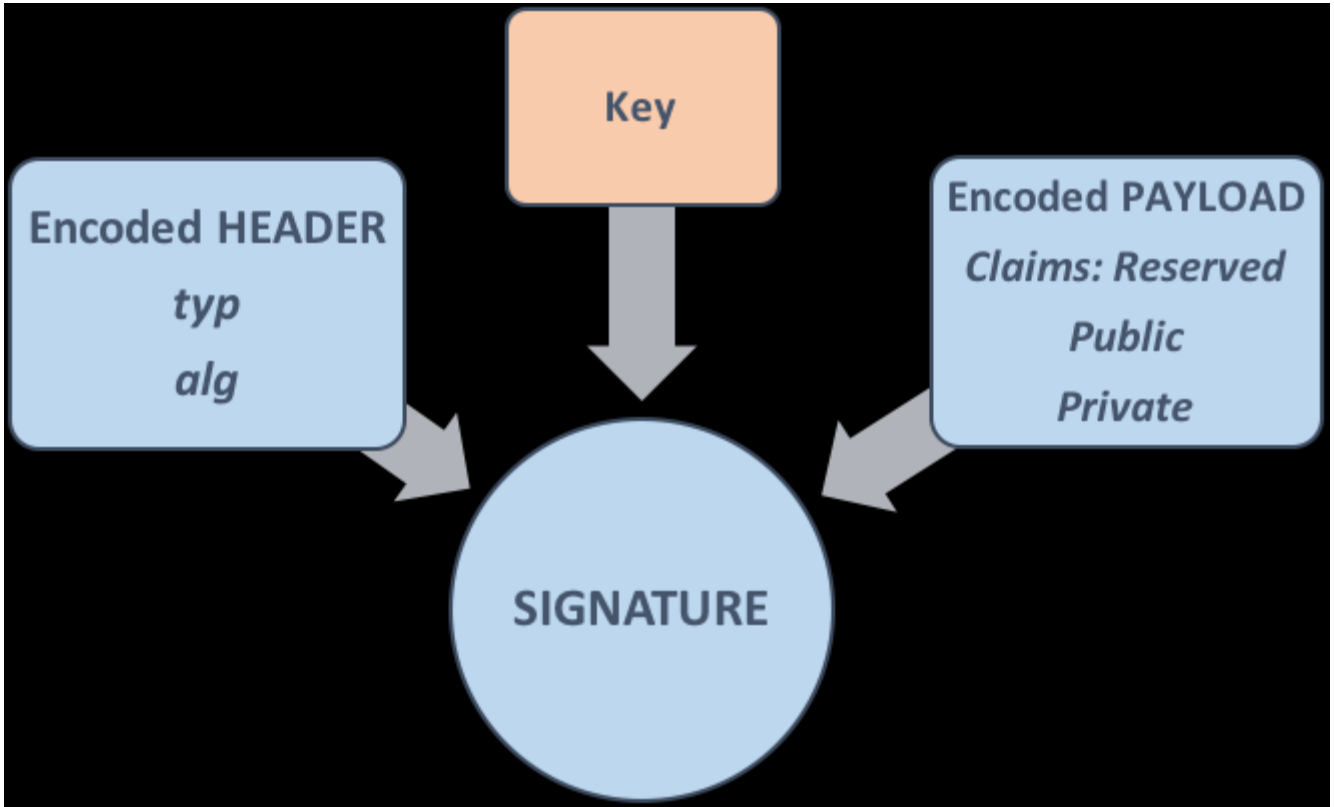
JSON Web Token (JWT)

- Payload: Là nơi chứa các nội dung của thông tin (claim). Thông tin truyền đi có thể là mô tả của 1 thực thể hoặc cũng có thể là các thông tin bổ sung thêm cho phần Header. Chúng được chia làm 3 loại: reserved, public và private
- Các thông tin Token như username , userId , author , ... ví dụ:

```
{
  "user_name": "admin",
  "user_id": "1513717410",
  "authorities": "ADMIN_USER",
  "jti": "474cb37f-2c9c-44e4-8f5c-1ea5e4cc4d18"
}
```

JSON Web Token (JWT)

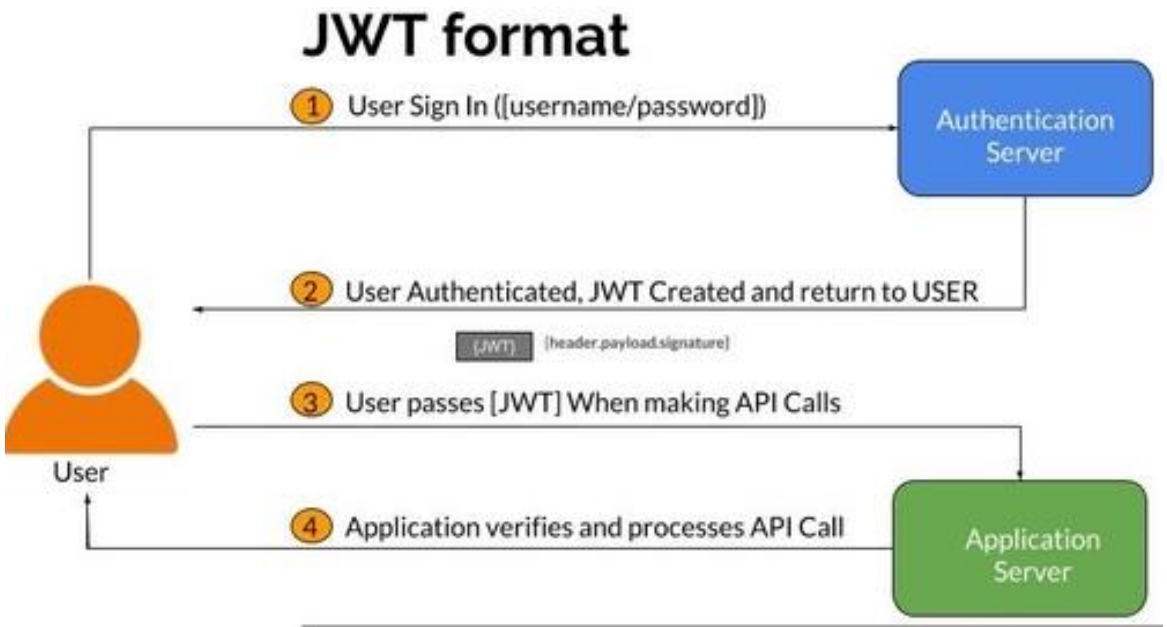
- Signature: Phần chữ ký được tạo bằng cách kết hợp 2 phần Header + Payload kèm theo một chuỗi secret (khóa bí mật), rồi mã hóa nó lại bằng 1 giải thuật encode bất kỳ ví dụ như HMAC SHA-256



FPT Education
FPT POLYTECHNIC

FLOW HỆ THỐNG SỬ DỤNG JWT

1.User thực hiện login bằng cách gửi id/password hay sử dụng các tài khoản mạng xã hội lên phía Authentication Server (Server xác thực)
2.Authentication Server tiếp nhận các dữ liệu mà User gửi lên để phục vụ cho việc xác thực người dùng. Trong trường hợp thành công, Authentication Server sẽ tạo một JWT và trả về cho người dùng thông qua response.
3.Người dùng nhận được JWT do Authentication Server vừa mới trả về làm "chìa khóa" để thực hiện các "lệnh" tiếp theo đối với Application Server.
4.Application Server trước khi thực hiện yêu cầu được gọi từ phía User, sẽ verify JWT gửi lên. Nếu OK, tiếp tục thực hiện yêu cầu được gọi.

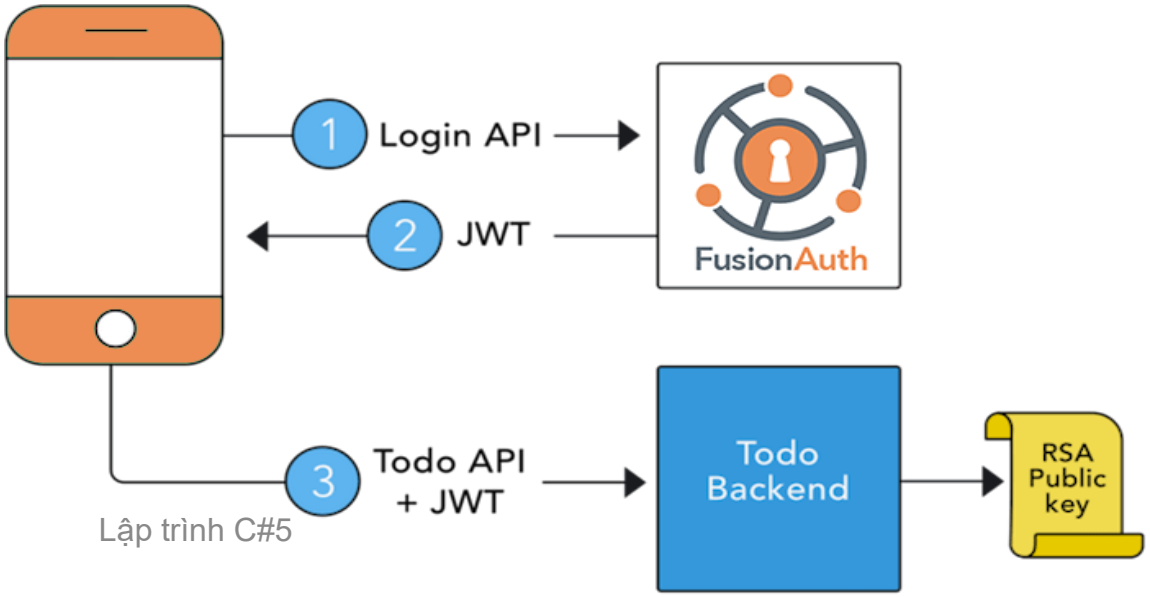


KHI NÀO DÙNG JSON WEB TOKEN?

- Authentication: Đây là trường hợp phổ biến nhất thường sử dụng JWT. Khi người dùng đã đăng nhập vào hệ thống thì những request tiếp theo từ phía người dùng sẽ chứa thêm mã JWT. Điều này cho phép người dùng được cấp quyền truy cập vào các url, service, và resource mà mã Token đó cho phép. Phương pháp này không bị ảnh hưởng bởi Cross-Origin Resource Sharing (CORS) do nó không sử dụng cookie

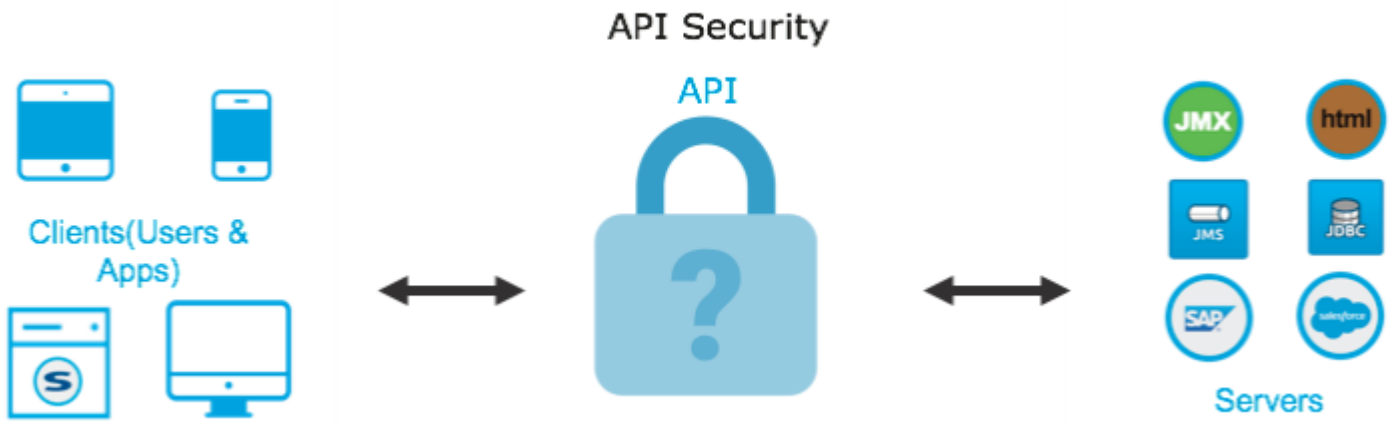
KHI NÀO DÙNG JSON WEB TOKEN?

- Trao đổi thông tin: JSON Web Token là 1 cách thức khá hay để truyền thông tin an toàn giữa các thành viên với nhau, nhờ vào phần signature của nó. Phía người nhận có thể biết được người gửi là ai thông qua phần signature. Và chữ ký được tạo ra bằng việc kết hợp cả phần header, payload lại nên thông qua đó ta có thể xác nhận được chữ ký có bị giả mạo hay không



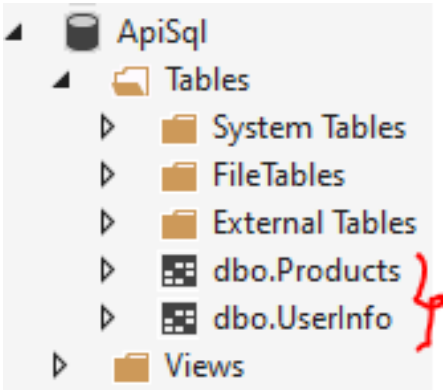
JWT – API AUTHENTICATION

- API hỗ trợ xử lý, chia sẻ nhiều tài nguyên với các platform bên ngoài do đó rất dễ bị tấn công hoặc lợi dụng cho mục đích không tốt
- Có nhiều kỹ thuật bảo mật Api: Authentication, Authorization, API Encryption và JSON WEB TOKEN



Ví dụ Jwt – Api Authentication

- Tạo REST APIs phục vụ các thao tác crud cho csdl sau



dbo.UserInfo [Data]

Max Rows: 1000

UserId	FirstName	LastName	UserName	Email	Password	CreatedDate
1	The	PV	Thepv	Thepv@fpoly.fpt	123	01/01/2021 12:0...
NULL	NULL	NULL	NULL	NULL	NULL	NULL

dbo.UserInfo [Data]

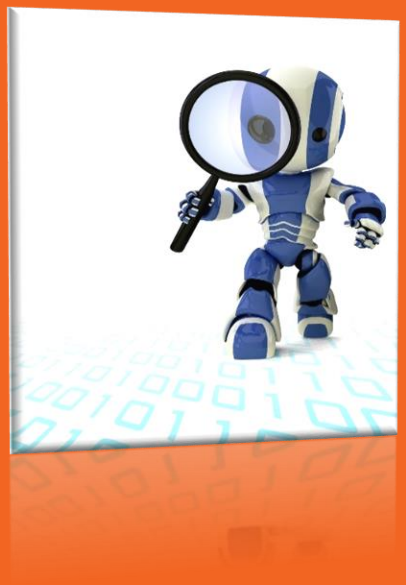
Max Rows: 1000

UserId	FirstName	LastName	UserName	Email	Password	CreatedDate
1	The	PV	Thepv	Thepv@fpoly.fpt	123	01/01/2021 12:0...
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Demo hiện thực các ví dụ

FPT Education

FPT POLYTECHNIC



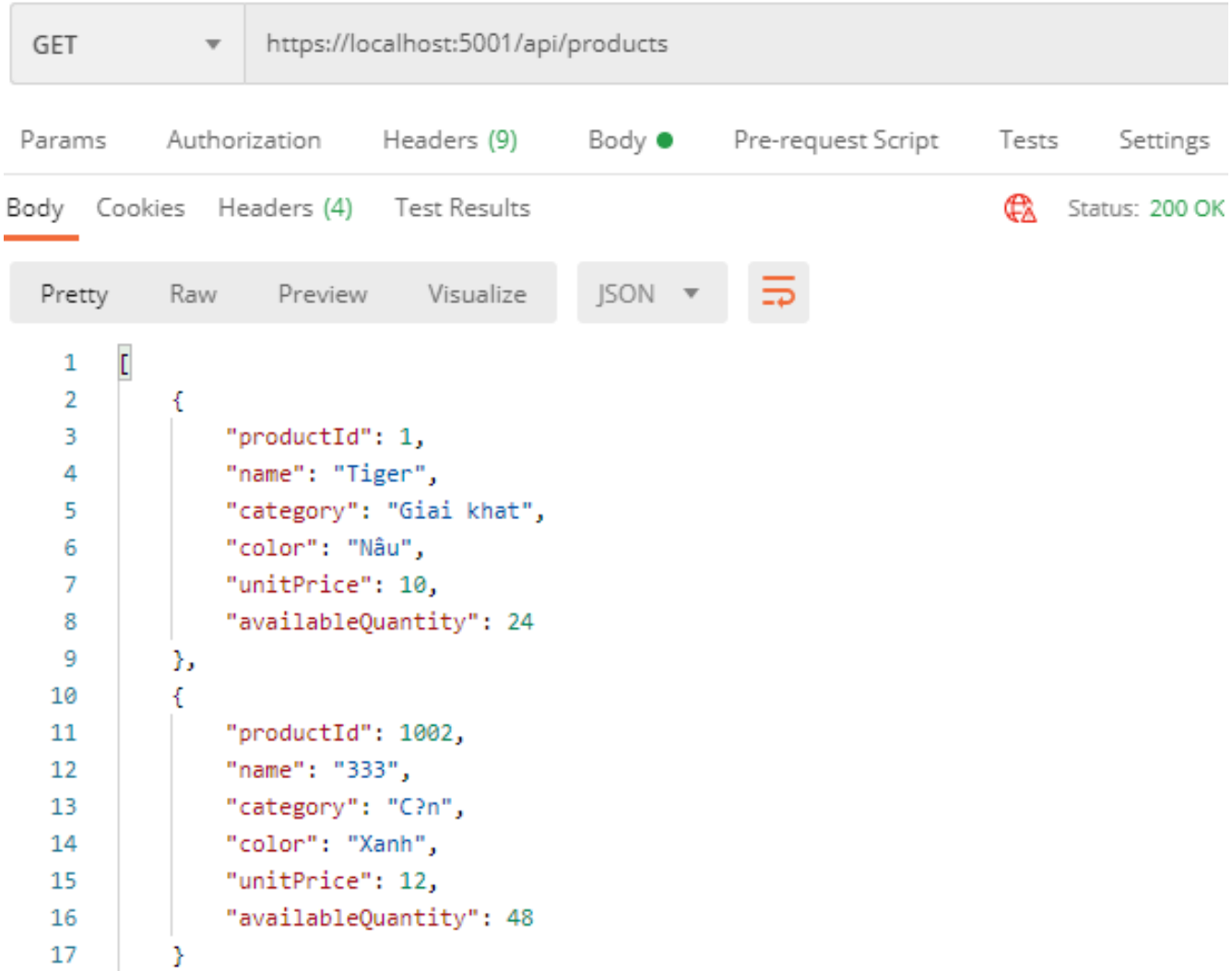
LẬP TRÌNH C# 5

BÀI 7: SECURE API ENDPOINT (P2)

www.poly.edu.vn

Ví dụ Jwt – Api Authentication

- RestApi test với postman theo url : https://localhost:5001/api/products



Ví dụ Jwt – Api Authentication

- Các bước bảo mật Jwt – Api Authentication
  1. Client requesting token: Client gửi passWord, nameUser tới server nhằm để xác thực việc đăng nhập
  2. Token creation: login thành công back-end sẽ tạo ra một generate a random String dạng json web token gửi về cho client
  3. Client sends token to resource server: Khi client muốn get data gì đó thì luôn gửi kèm token này lên cùng với http request
  4. Resource server verifies the token:Server nhận được http request từ client thì check token này available hay không? Rồi cho đi tiếp, còn không chặn lại, và có thể report ip này.

TẠO JWT

- Cấu hình JWT trong appsetting.json đăng ký một số thông tin liên quan jwt
  - Key – Super Secret Key dùng trong quá trình mã hóa
  - Issuer – Xác đinh nguồn tạo JWT
  - Audience – định danh đối tượng sử dụng jwt
  - DurationInMinutes – thời hạn hiệu lực jwt

```
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "InventoryDatabase": "Server=.;Database=ApiSql;Trusted_Connection=True;"
  },
  "Jwt": {
    "Key": "sdfsdfsjdbf78sdyfssdfsdfbuidfs98gdfsdbf",
    "Issuer": "SecureApiServer",
    "Audience": "SecureApiServicePostmanClient",
    "DurationInMinutes": 60,
    "Subject": "SecureApiServiceAccessToken"
  }
}
```

TẠO JWT

- Cài các package
  - System.IdentityModel.Tokens.Jwt : tạo và xác thực jwt token
  - Microsoft.AspNetCore.Authentication.JwtBearer: là middleware trong ASP.NET Core hỗ trợ nhận token trong request pipeline

TẠO JWT

- Tạo API controller tên 'TokenController'
  - Nhận vào username và pass
  - Kiểm tra so với dữ liệu trong database
    - Nếu hợp lệ thì trả về một access token
    - Nếu không hợp lệ thì trả về bad request error

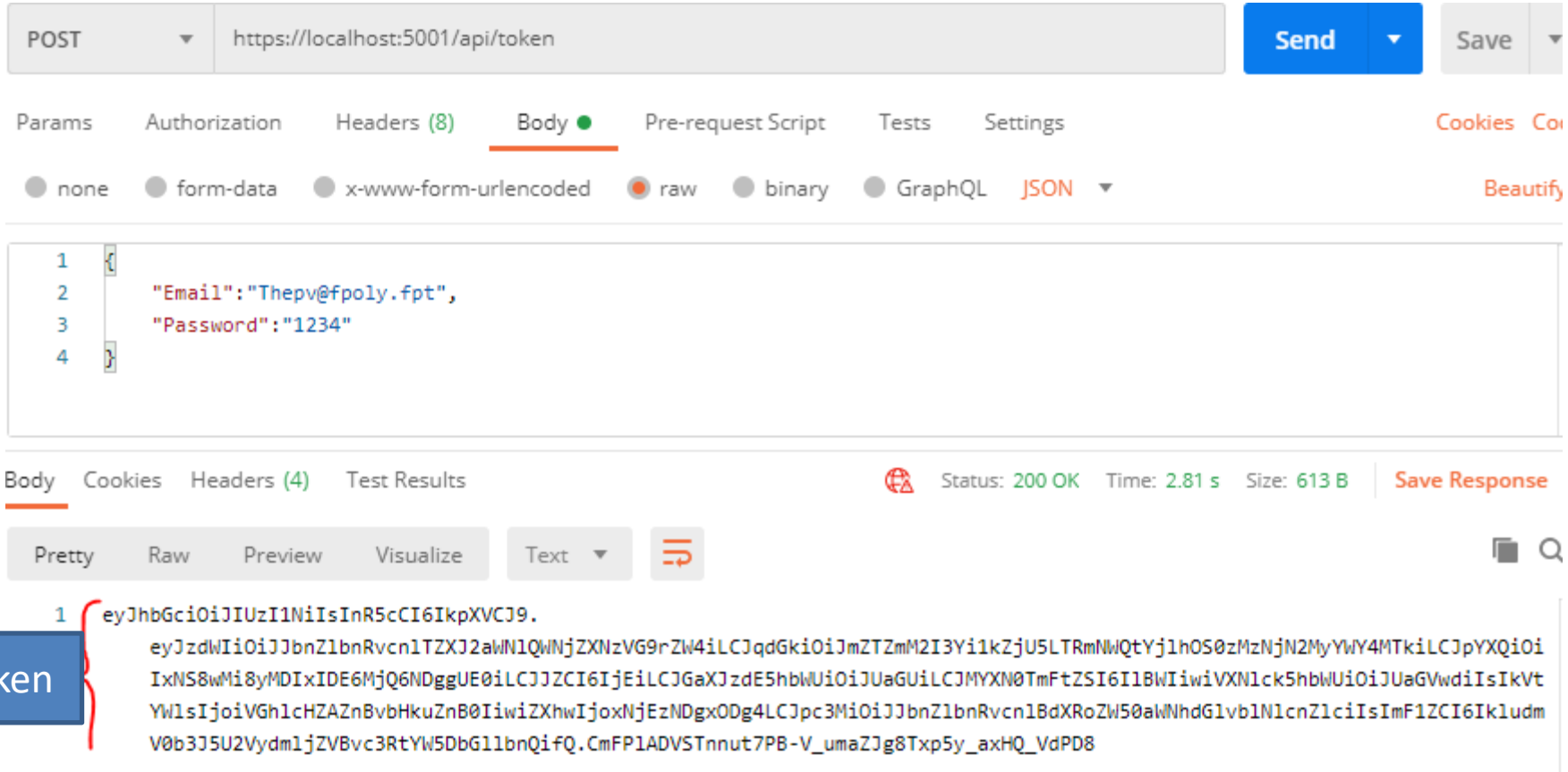
```
[Route("api/[controller]")]
[ApiController]
1 reference
public class TokenController : ControllerBase
{
    public IConfiguration _configuration;
    private readonly InventoryContext _context;
    0 references
    public TokenController(IConfiguration config, InventoryContext context)
    {
        _configuration = config;
        _context = context;
    }

private async Task<UserInfo> GetUser(string email, string password)
{
    return await _context.UserInfo.FirstOrDefaultAsync(u => u.Email
}
```

TẠO JWT

```
[HttpPost]
0 references
public async Task<IActionResult> Post(UserInfo _userData)
{
    if (_userData != null && _userData.Email != null && _userData.Password != null)
    {
        var user = await GetUser(_userData.Email, _userData.Password);
        if (user != null)
        {
            //create claims details based on the user information
            var claims = new[] {
            new Claim(JwtRegisteredClaimNames.Sub, _configuration["Jwt:Subject"]),
            new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),
            new Claim(JwtRegisteredClaimNames.Iat, DateTime.UtcNow.ToString()),
            new Claim("Id", user.UserId.ToString()),
            new Claim("FirstName", user.FirstName),
            new Claim("LastName", user.LastName),
            new Claim("UserName", user.UserName),
            new Claim("Email", user.Email)
           };
            var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_configuration["Jwt:Key"]));
            var signIn = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);
            var token = new JwtSecurityToken(_configuration["Jwt:Issuer"],
                _configuration["Jwt:Audience"], claims, expires: DateTime.UtcNow.AddDays(1),
                signingCredentials: signIn);
            return Ok(new JwtSecurityTokenHandler().WriteToken(token));
        }
        else
        {
            return BadRequest("Invalid credentials");
        }
    }
    else
    {
        return BadRequest();
    }
}
```

TẠO JWT

- Tạo Token với thông tin username và pass bằng Postman (lưu lại token này dùng về sau)
  - POST method
  - header to 'Content-Type': 'application/json'.



Token

SECURE API ENDPOINT

- Dùng JWT token để Secure API endpoint
  - Cần thêm vào startup file các namespaces

```
using Microsoft.AspNetCore.Authentication.JwtBearer;
using Microsoft.IdentityModel.Tokens;
using System.Text;
```

  - Cấu hình authorization middleware trong configureService method

```
public void ConfigureServices(IServiceCollection services)
{
    var connection = Configuration.GetConnectionString("ApiJWT");
    services.AddDbContextPool<InventoryContext>(options => options.UseSqlServer(connection));

    services.AddControllers();

    services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme).AddJwtBearer(options =>
    {
        options.RequireHttpsMetadata = false;
        options.SaveToken = true;
        options.TokenValidationParameters = new TokenValidationParameters()
        {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidAudience = Configuration["Jwt:Audience"],
            ValidIssuer = Configuration["Jwt:Issuer"],
            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configuration["Jwt:Key"]))
        };
    });
```

SECURE API ENDPOINT

- Sử dụng authorization attribute trong controller

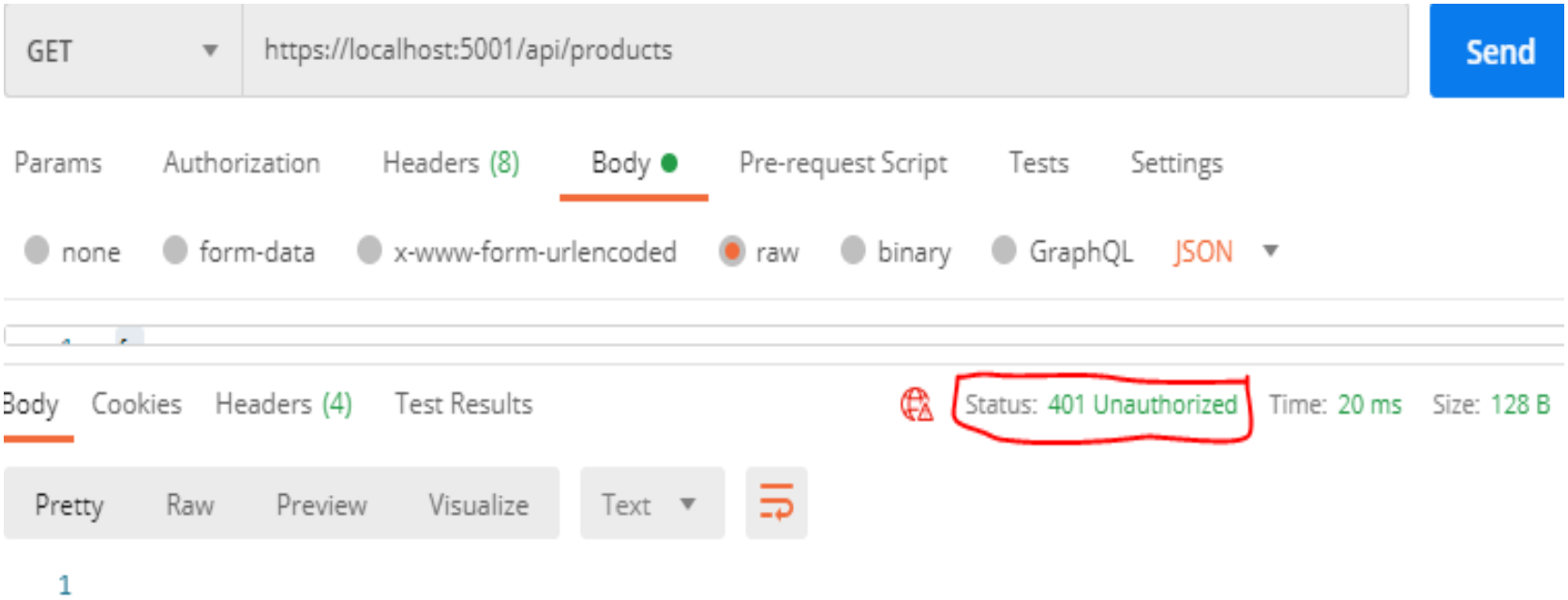
```
[Authorize]
[Route("api/[controller]")]
[ApiController]
1 reference
public class ProductsController : ControllerBase
{
    private readonly InventoryContext _context;
```

- Sử dụng authorization middleware trong Request pipeline

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }

    app.UseHttpsRedirection();

    app.UseRouting();

    app.UseAuthentication();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```

SECURE API ENDPOINT

- Cấu hình xong thì muốn truy cập endpoint https://localhost:5001/api/products thì client phải gởi kèm token đã tạo trong slide trước, nếu không sẽ bị chặn



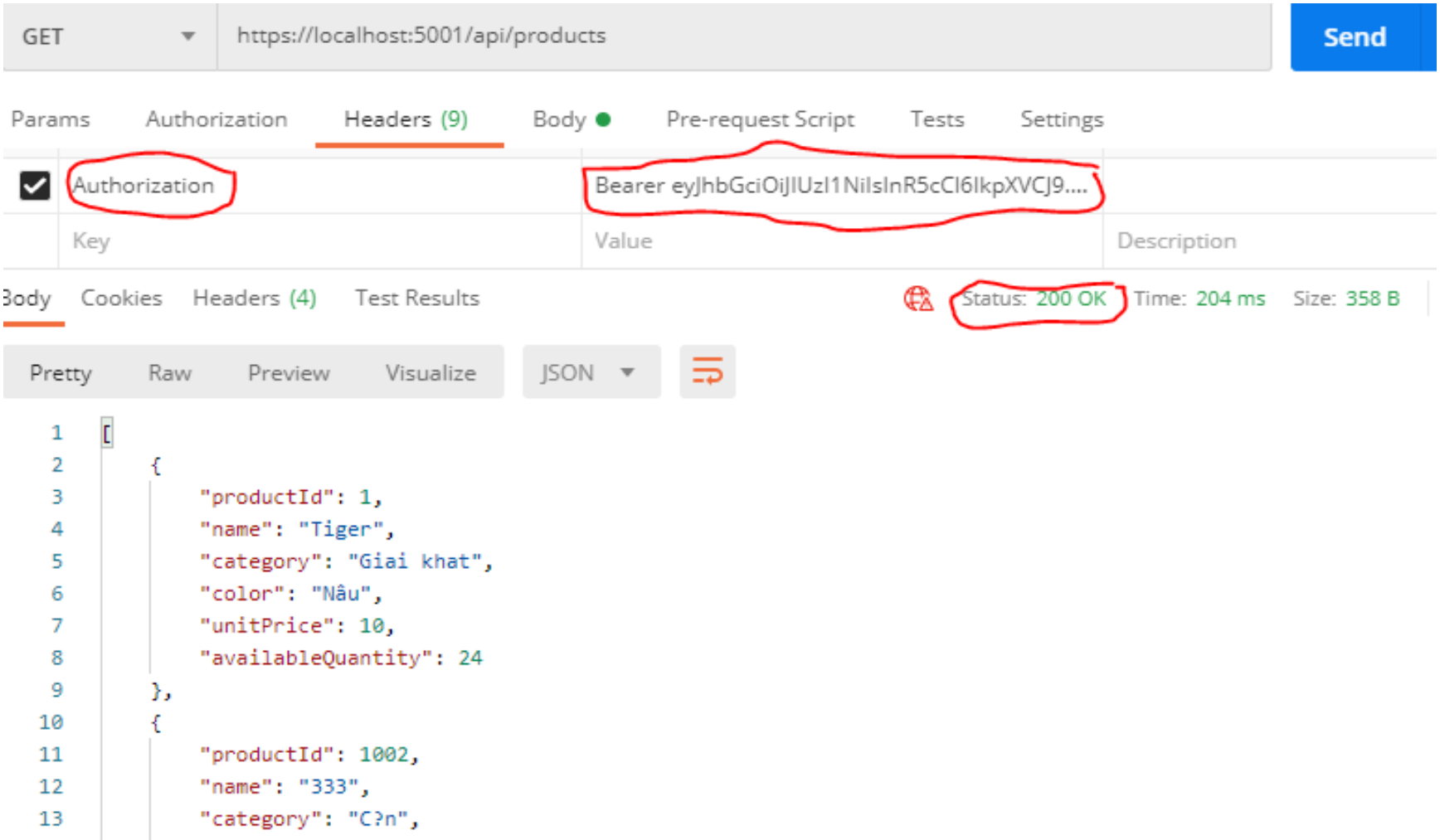
SECURE API ENDPOINT

- Muốn truy cập vào Route được bảo vệ (mà chỉ có User đã đăng nhập mới được phép), Browser sẽ gửi token JWT này trong Header Authorization, Bearer schema của request gửi đi.
- Thêm Bearer text vào token tạo trong slide trước

```
Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJJbnZlbnRvcnlT
2aWNlQWNjZXNzVG9rZW4iLCJqdGkiOiI3MjRiMjM1OS01NWI2LTQyMmUtYWVl
lMjdjMWNiNWNjNDkiLCJpYXQiOiIxNi8wMi8yMDIxIDE6NDc6MjcgQU0iLCJJ
6IjEiLCJGaXJzdE5hbWUiOiJUaGUiLCJMYXN0TmFtZSI6IlBWIiwiVXNlck5h
iOiJUaGVwdiIsIkVtYWlsIjoiVGhlcHZAZnBvbHkuZnB0IiwiZXhwIjoxNjEz
2NDQ3LCJpc3MiOiJJbnZlbnRvcnlBdXRoZW50aWNhdGlvblNlcnZlciIsImF1
6IkludmV0b3J5U2VydmljZVBvc3RtYW5DbGllbnQifQ.V_wS1uWstB8N4pnzi
mRlPWuyvgrLcn4BIQ6I7WMc
```

SECURE API ENDPOINT

- Gởi kèm token với Bearer text trong Authorization header để chứng thực, đúng thì nhận được kết quả



Demo hiện thực các ví dụ

Tổng kết bài học

- Token-based authentication
- JSON Web Token (JWT)
- Secure API endpoint - Jwt

KẾT THÚC