<https://www.youtube.com/watch?v=bWA-pZJrOFE&t=1200s>

**JWT (Json Web Token)**

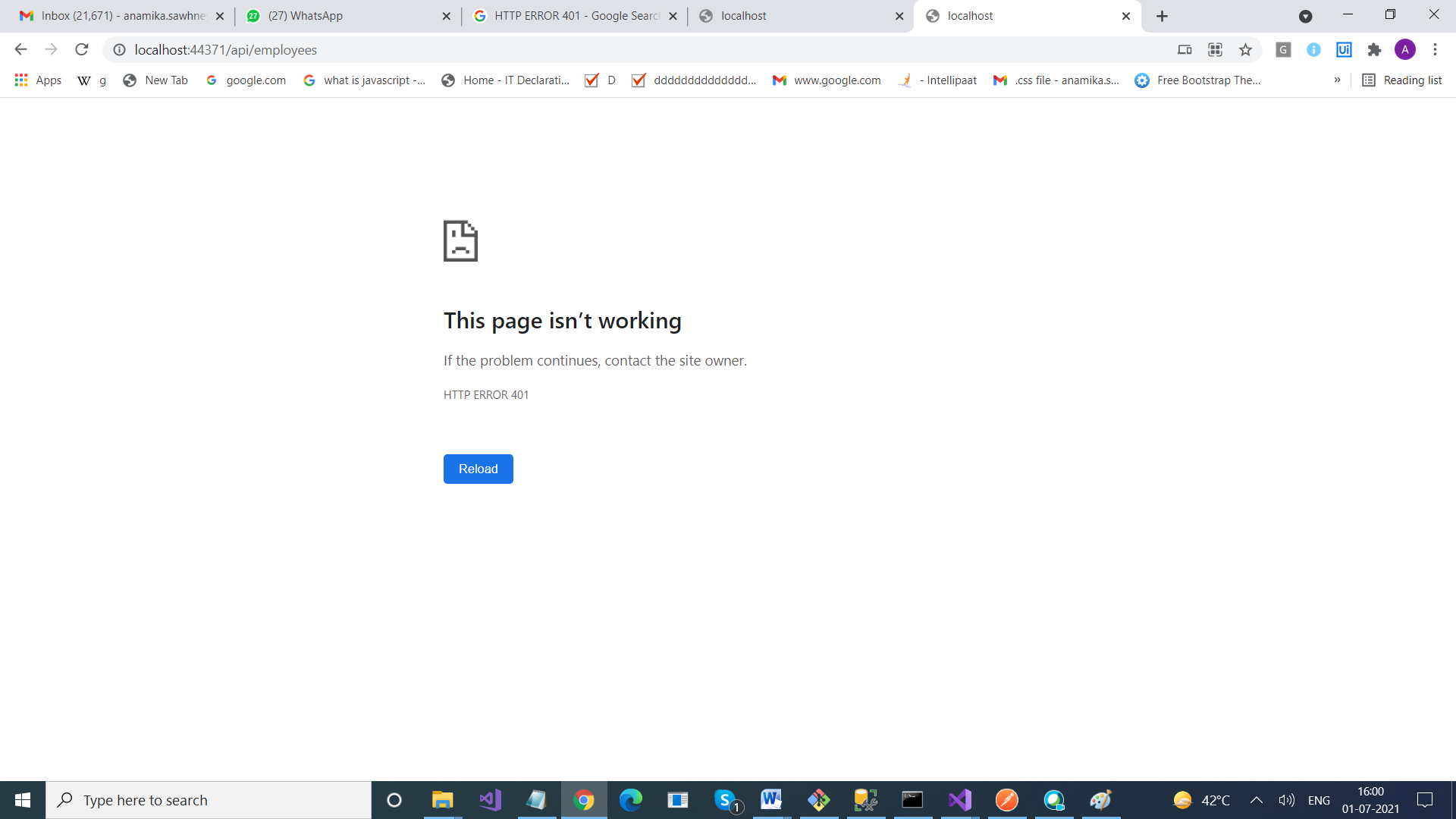
**With this, Our Web Api is secured**

**Step 1:**

Add [Authorize] at Action Method level OR at Controller level

It means controller OR Action Methods are secured, We can only access them , once we tell that that we are a valid user

When you add this Authorize keyword , and try to access the Employees Web Api , you get 401 Error



So, how do we know that we are going to use Authentication that too, JWT , for that we use In ASP.Net Core > Middleware , Pipeline

JWT

**Step 2:**

Install Microsoft.AspNetCore.Authentication.JwtBearer

3.1.16

**Step 3:**

AddSettings.json file

{

  "Jwt": {

    "Key": "ThisismySecretKey",

    "Issuer": "Test.com"

  }

}

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": {

"StudentDBContext": "server=LAPTOP-53S2KQS8;database=BookStore;integrated security=true"

},

**"Jwt": {**

**"Key": "ThisismySecretKey",**

**"Issuer": "Test.com"**

**}**

}

**Step 4: Add it to the Startup.cs file**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using DocumentFormat.OpenXml.EMMA;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using WebApplication16.Models;

namespace WebApplication16

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

**services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)**

**.AddJwtBearer(options =>**

**{**

**options.TokenValidationParameters = new TokenValidationParameters**

**{**

**ValidateIssuer = true,**

**ValidateAudience = true,**

**ValidateLifetime = true,**

**ValidateIssuerSigningKey = true,**

**ValidIssuer = Configuration["Jwt:Issuer"],**

**ValidAudience = Configuration["Jwt:Issuer"],**

**IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configuration["Jwt:Key"]))**

**};**

**});**

services.AddControllers();

services.AddSwaggerGen();

services.AddDbContext<StudentDbContext>(op => op.UseSqlServer(Configuration["ConnectionStrings:StudentDbContext"]));

//services.AddSwaggerGen();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseHttpsRedirection();

app.UseRouting();

**app.UseAuthentication();**

app.UseAuthorization();

app.UseSwagger();

// Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),

// specifying the Swagger JSON endpoint.

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API");

// c.RoutePrefix = string.Empty;

});

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

--------------------------------------------------------------

**public** **void** Configure(IApplicationBuilder app, IHostingEnvironment env)

{

app.UseAuthentication();

app.UseMvc();

}

What’s next??

Add a User class

public class User

{

public int Id { get; set; }

public string UserName { get; set; }

public string Password { get; set; }

}

I have provided some initial data for users : Where we can do that

**In OnModelCreating() event**

public class StudentDbContext : DbContext

{

public StudentDbContext(DbContextOptions<StudentDbContext> options)

: base(options) { }

public DbSet<Employee> Employees { get; set; }

public DbSet<User> Users { get; set; }

**protected override void OnModelCreating(ModelBuilder modelBuilder)**

**{**

**modelBuilder.Entity<User>()**

**.HasData(new User**

**{**

**Id = 1,**

**UserName = "user1",**

**Password = "user1"**

**},**

**new User**

**{**

**Id = 2,**

**UserName = "user2",**

**Password = "user2"**

**},**

**new User**

**{**

**Id = 3,**

**UserName = "user3",**

**Password = "user3"**

**}**

**);**

}

}

How do you pass these user details to database?

**Add-Migration AddUsers**

**Update-database**

Why are we using Class , so that we can validate users

For this we created one more Controller AuthenticationController

In this we will add some methods

1. Check whether the user is valid or not , means whether user exist or not
2. If user is valid, generate token foe that user

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Text;**

**using System.Threading.Tasks;**

**using Microsoft.AspNetCore.Http;**

**using Microsoft.AspNetCore.Mvc;**

**using Microsoft.EntityFrameworkCore;**

**using Microsoft.Extensions.Configuration;**

**using Microsoft.IdentityModel.Tokens;**

**using WebApplication16.Models;**

**using Microsoft.AspNetCore.Authentication.JwtBearer;**

**using System.IdentityModel.Tokens.Jwt;**

**namespace WebApplication16.Controllers**

**{**

**[Route("api/[controller]")]**

**[ApiController]**

**public class AuthenticationController : ControllerBase**

**{**

**private readonly StudentDbContext \_context;**

**private IConfiguration \_config;**

**public AuthenticationController(StudentDbContext context, IConfiguration config)**

**{**

**\_context = context;**

**\_config = config;**

**}**

**[AllowAnonymous]**

**1[HttpPost]**

**public IActionResult Login([FromBody] User User)**

**{**

**IActionResult response = Unauthorized();**

**var user = Authenticate(User);**

**if(user!=null)**

**{**

**var tokenString = GenerateJSONWebToken(user);**

**response = Ok(new { token = tokenString });**

**}**

**return response;**

**}**

**private string GenerateJSONWebToken(User user)**

**{**

**var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_config["Jwt:Key"]));**

**var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256);**

**var token = new JwtSecurityToken(\_config["Jwt:Issuer"],**

**\_config["Jwt:Issuer"],**

**null,**

**expires: DateTime.Now.AddMinutes(120),**

**signingCredentials: credentials);**

**return new JwtSecurityTokenHandler().WriteToken(token);**

**}**

**public User Authenticate(User user)**

**{**

**User obj = \_context.Users.FirstOrDefault(x => x.UserName == user.UserName && x.Password == user.Password);**

**return user;**

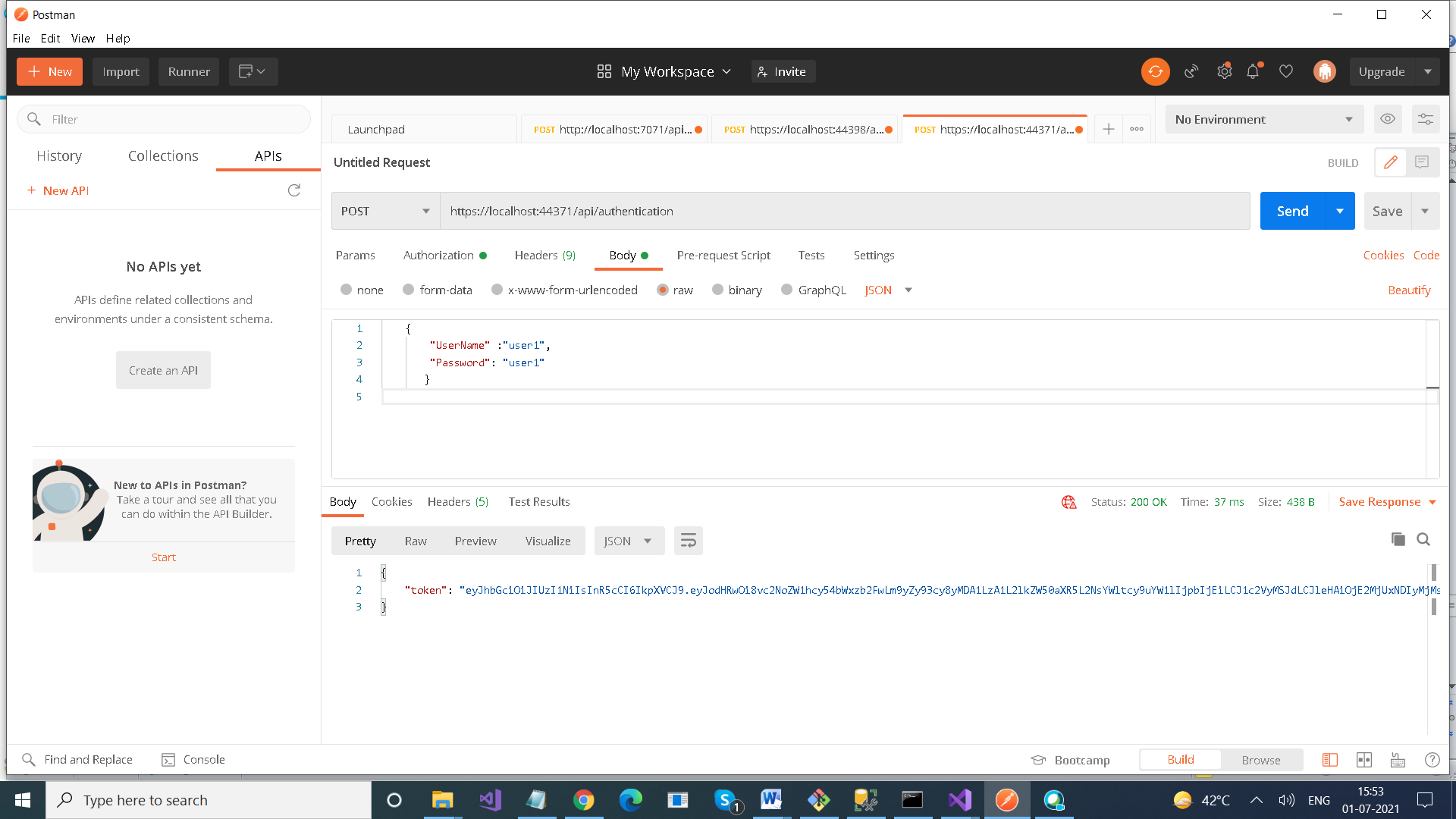
**}**

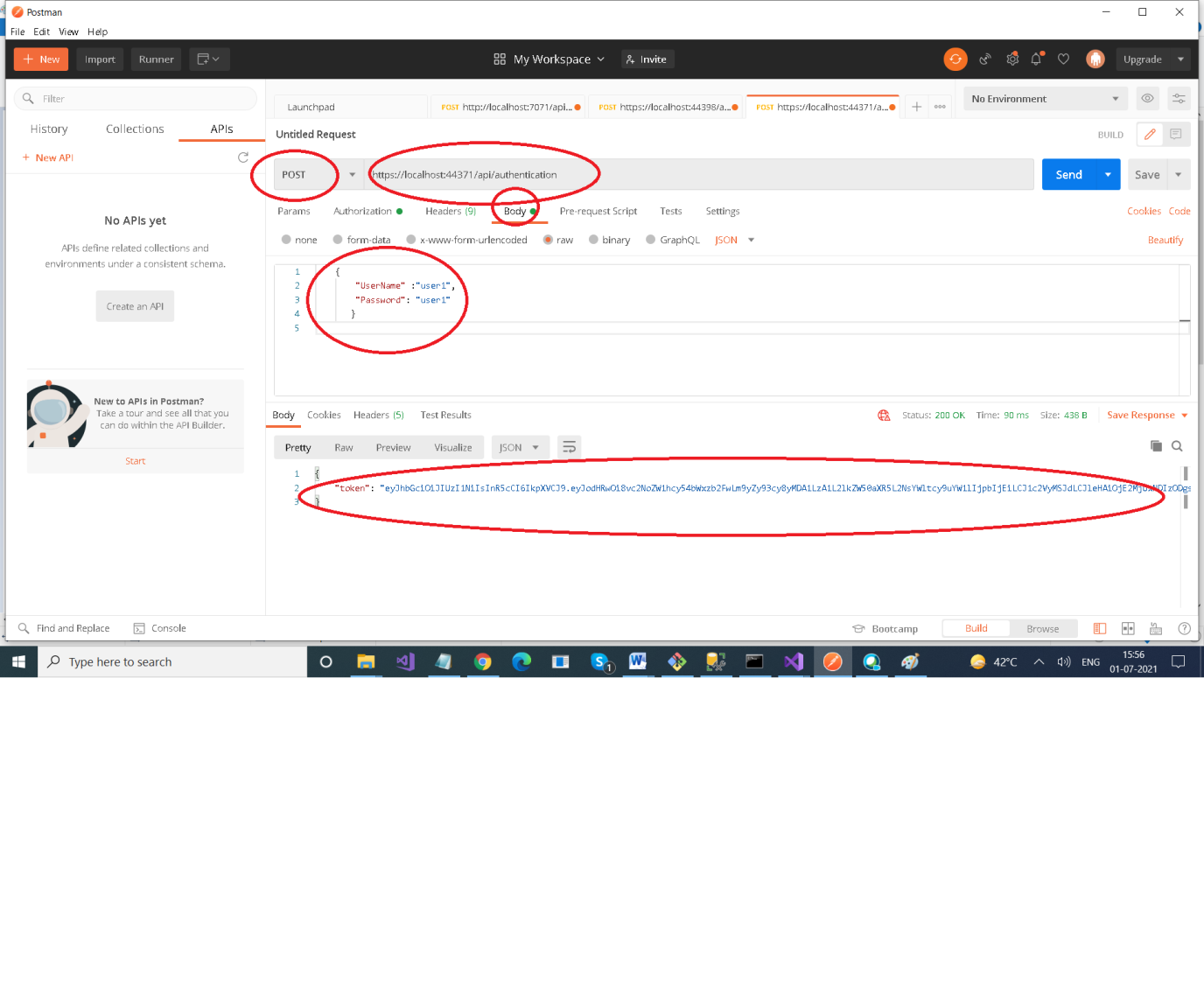
**}**

**}**

**After adding this. Now go to postman, run AuthenticationController**

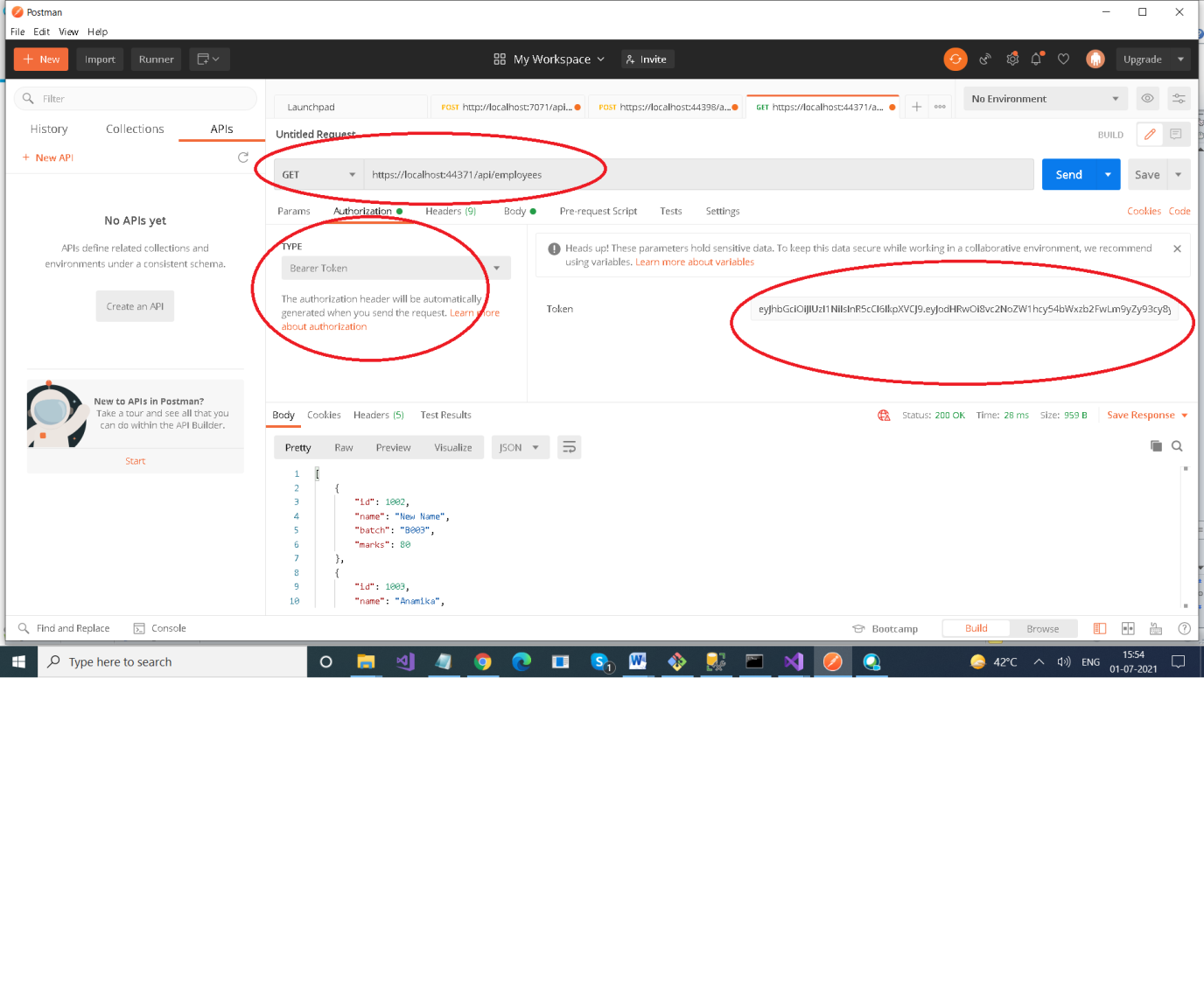
**Post Method > If user if found, you will see Token**



****

**Token is generated**

**Now with every request for Employee Controller , you need to pass this token**

****