ASP.NET

# Configure Services

1. Add NLog (Github for NET 6.0)
2. Configuration => builder.Configuration
3. Lambda => builder = policyBuilder
4. Services => builder.Services
5. this.GetType().Assembly => Assembly.GetExecutingAssembly()

# Configure

1. var app = builder.Build() : aby otrzymac obiekt app.
2. Seeder => var scope = app.Services.CreateScope();

var seeder = scope.ServiceProvider.GetRequiredService<RestaurantSeeder>();

1. Env =>app.Environment

# Swagger

Swashbuckle.AspNetCore.Swagger

Swashbuckle.AspNetCore.SwaggerUi

# Scopes

Transient: Mostly used, for querying database

Scoped: It doesn’t change entire time

Singleton: Stays the same entire time

# Creating Endpoint (as Controller)

Create controller + services + interface (for services)

For multiple endpoints in service add [Route(“nextDay”)] lub [HttpGet(“nextDay”)]

This, will add new routing, for example localhost:5001/weatherforcast/nextDay, localhost:5001/weatherforcast/currentDay etc…

To add parameters

1. Next([FromQuery]int take, [FromRoot]int max)

[HttpGet(“nextDay/{max}”)]

localhost:5001/weatherforcast/nextDay/123?take=50

[FromBody] we can pass json

1. We can set what code we want to return with ActionResult<string> Hello ([FromBody] string name) {

HttpContext.Response.StatusCode = 401;

return “Hello ” + name;

lub

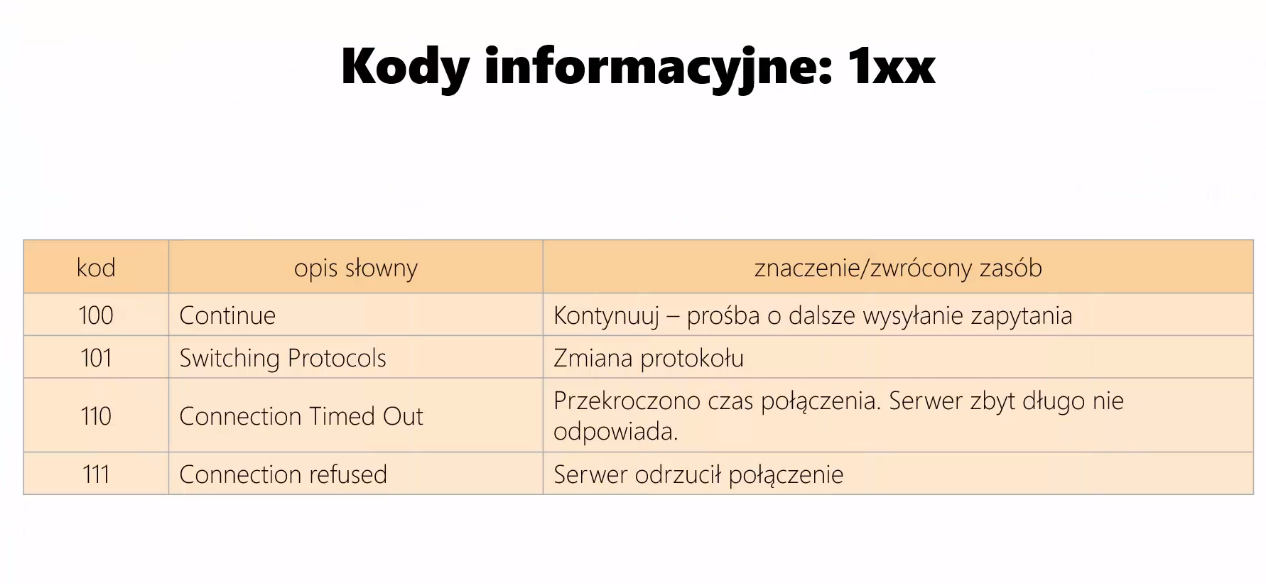
return StatusCode (401, “Hello “ + name);

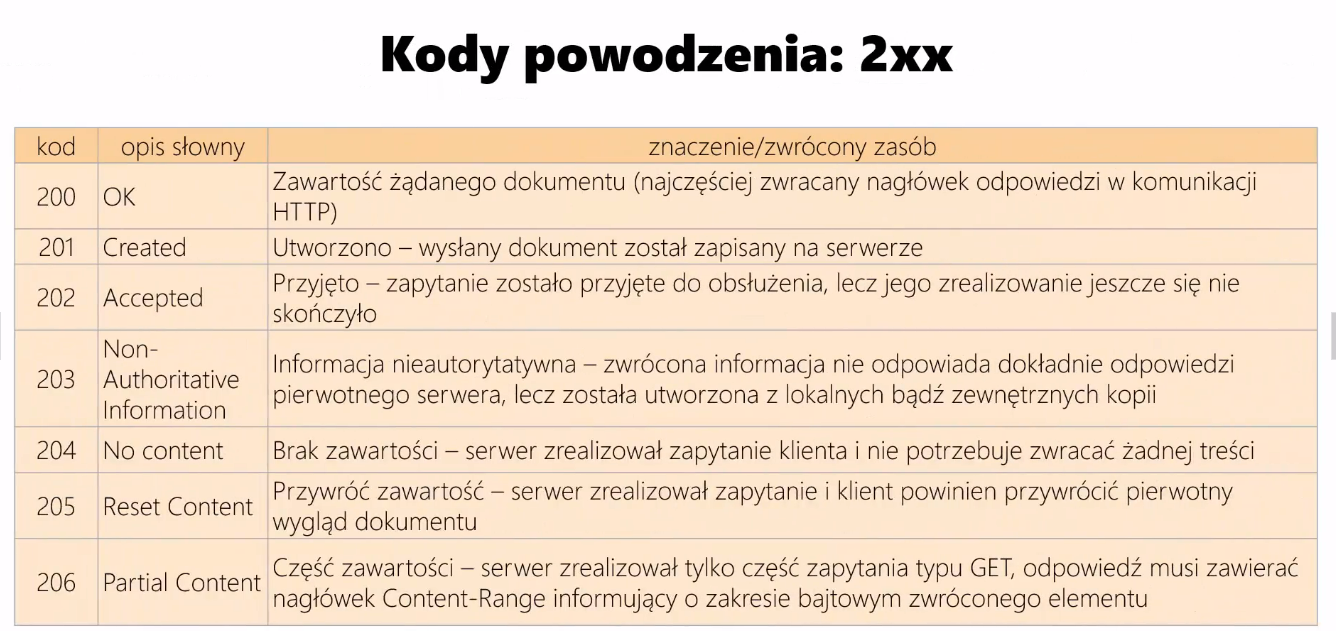
lub

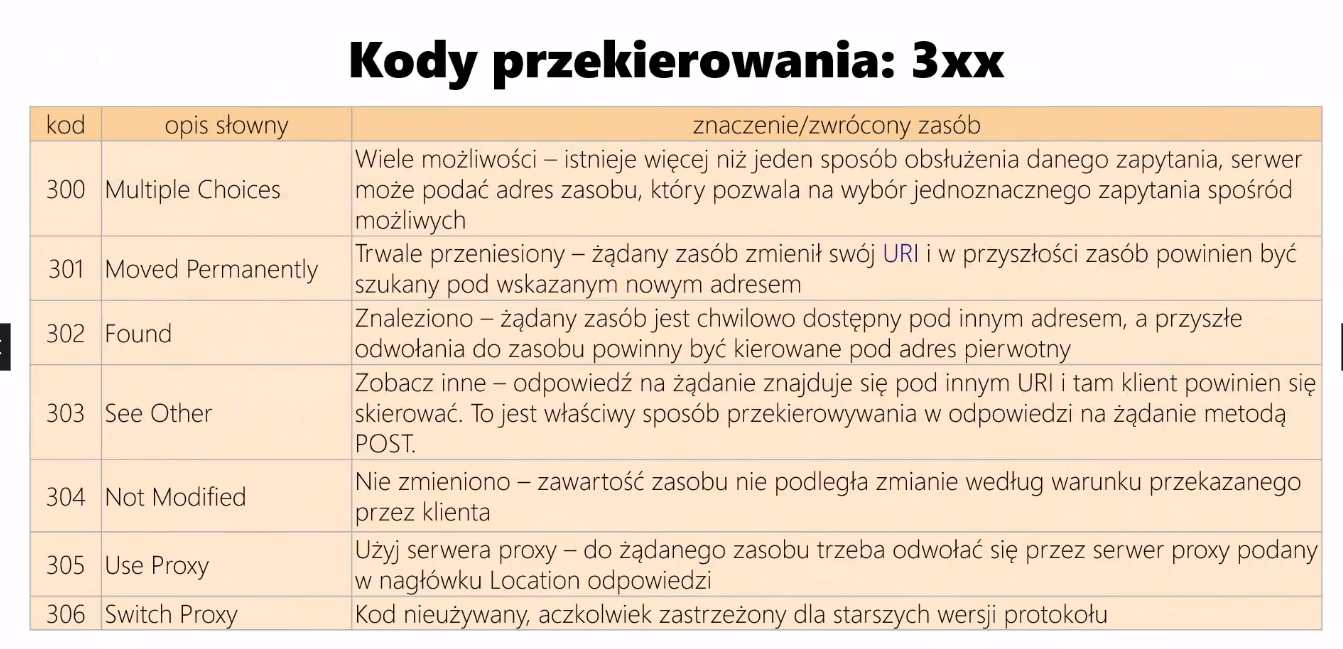
return NotFound(“Hello “ + name)

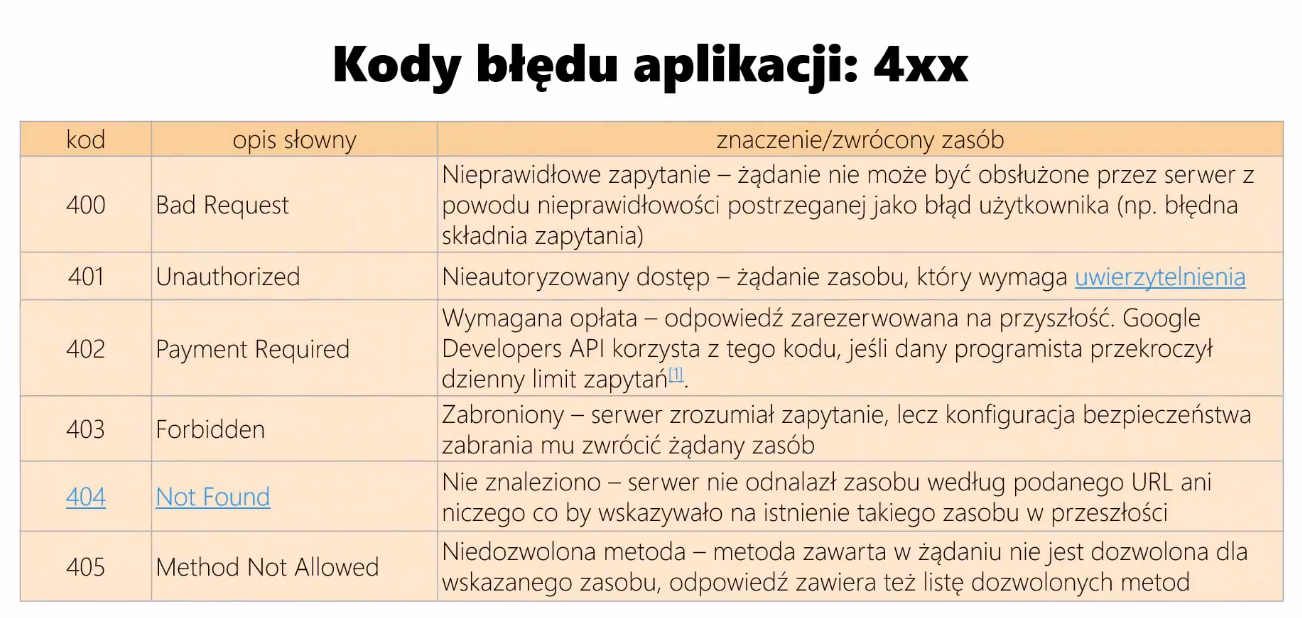
}

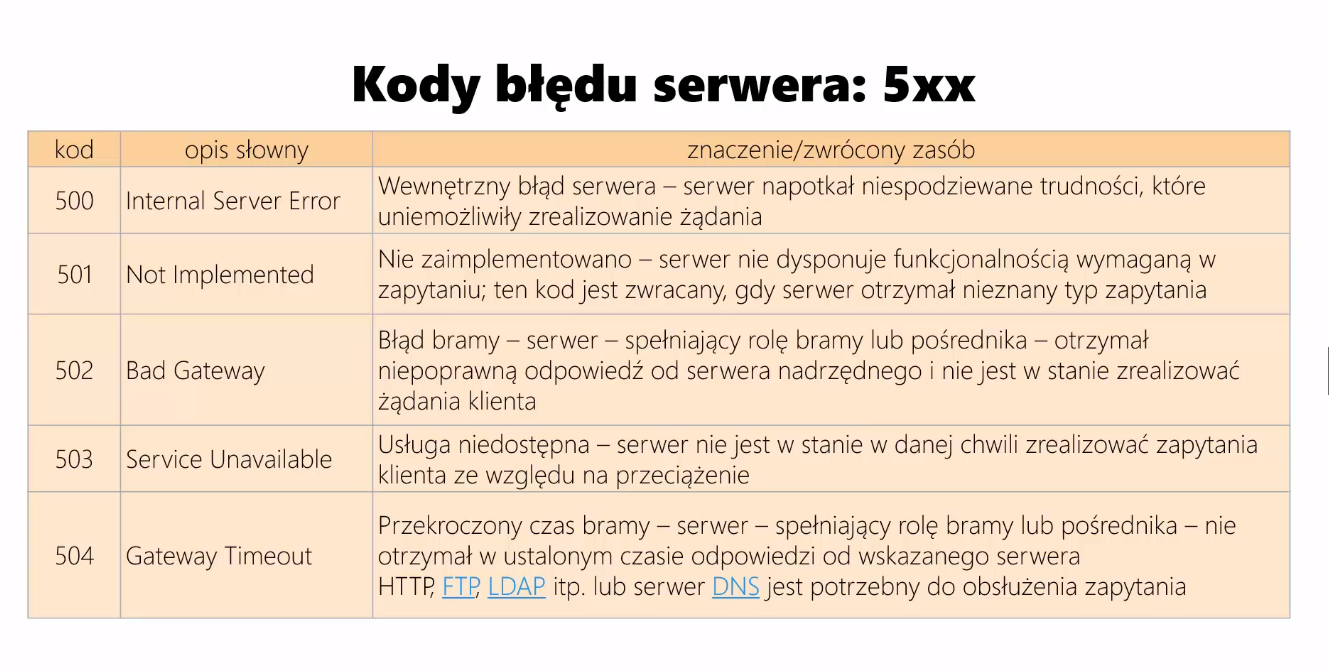
Kody:











# Entity Framework

Instalujemy z nuget EntityFremworkCore, EntityFremworkCore.SqlServer and EntityFremworkCore.Tools

Migracja -> Tools/Nuget Package Manager/Package Manager Console: add-migration Init

Po stworzeniu nowego projektu (Migrations). DO NOT MANUALLY MODIFY TABLES

update-database w package managerze, stworzy baze,

Local DB ->

private string \_connectionString = "Server=(localdb)\\mssqllocaldb;Database=RestaurantDb; Trusted\_Connection=True;";

AUTO MAPPER  
AutoMapper.Extensions.Microsoft.DependencyInjection

# ERROR LOG

Nie robić try{} catch{}

Zamiast tego: stworzyć middleware do logowania.

# MIGRATIONS

add-migration [nazwa]

remove-migration

Update-Database

Update-Database 0 – undo all

Update-Database [name] specific database

# VALIDATION

FluentValidation.AspNetCore

# JWT AUTENTICATION

Microsoft.AspNetCore.Authentication.JwtBearer

I can use Json.Serializer to insert list of objects as Claim in JWT token

# AUTHORIZATION

Prosta na podstawie roli,

Druga na podstawie Claima

I czy Claim ma wartosc na której chcemy autoryzować

Headers -> Authorization: Bearer [JWTtoken]

# FAKE DATA

Create bogus data with Bogus

# Publikacja (MIKR.US(?))

-Jak ustawić firewall, tak aby to naszych serwisow polaczenie było możliwe tylko poprzez aplikacje pod szczególnym adresem.

Features:

Product holds: Projects -> Bugs -> Roadmaps.

Project -> Project Child -> Project Child etc. (Track your project progress)

Bugs -> (Track your bug resolution progress)

Bug log endpoint -> endpoint that will be able to log bug details such as: component where issue happened, logs, stack, details about machine/OS/software and user (if needed)

In both projects and bugs we will have comments, backups, responsibility, deadlines, importance, tracking days overdue, client id, project type (enhancement, paied enhancement etc) owner of a bug, person who created it or was it automatically created.

We will group bugs, based on their source.

Roadmaps ->

Reports and insight->