Múltiplos Ambientes - Environments

O ASP.NET Core configura o comportamento do aplicativo com base no ambiente de runtime usando uma variável de ambiente.

- ASPNETCORE_ENVIRONMENT ou DOTNET_ENVIRONMENT
- São 3 valores possíveis: Development, Staging e Production
- Por padrão o valor é “Production”
- É possível criar seu próprio ambiente
- O arquivo lanchSettings.json configura o comportamento na ambiente do desenvolvedor
- Este arquivo não deve copiado no deploy da aplicação
- O ASP.NET confere as variáveis de ambiente do sistema operacional

Múltiplos Ambientes - Environments

```
// Configura a execução da aplicação no ambiente diferente de "Development"
if (!app.Environment.IsDevelopment())
{
    //Usa uma página de erro customizada
    app.UseExceptionHandler("/Error");
    // Valor padrão do HSTS é de 30 dias.
    app.UseHsts();
}
else
{
    app.UseDeveloperExceptionPage();
}
```

Code21

Autenticação no ASP.NET

Atenticação é o processo que determina se o usuário possui uma identidade de acesso ou não. O usuário é então registrado e segue a execução de suas ações.

- ASP.NET gerencia as autenticações pelo *IAuthenticationService*
- No serviço configuramos qual será o *schema* de autenticação
- Os Claims armazenam as informações do usuário
- É possível configurar múltiplos *schemas*
- Exemplo de configurações:
  - Autenticação baseada em Cookies que armazenam os dados
  - Atenticação baseada em Token JWT

AuthenticationService

```
builder.Services.AddAuthentication("appauth").AddCookie("appauth", options =>
{
    options.Cookie.Name = "appauth";
    options.LoginPath = "[URL]";
    options.AccessDeniedPath = "[URL]";
});
```

```
app.UseAuthentication();
app.UseAuthorization();
```

```
[Authorize]
0 references
public class HomeController : Controller{
    [HttpGet]
    0 references
    public IActionResult Index()
    {
```

Code21

Trabalhando com Autorização

Autorização é o processo que determina se um usuário tem ou não acesso a um recurso. Autorização é um processo independente da autenticação.

- No ASP.NET a autorização é baseadas nas *policies* de acesso
- Essas politicas podem ser diversos parâmetros
- *Claims, UserName, Cookie, Role(Perfil)*
- Uma *policy* é vinculada a um recurso do projeto
- A plataforma valida se o usuário possue acesso a essa *policy*
- Se não possuir acesso ele é então direcionado a uma url padrão

Trabalhando com Autorização

```
builder.Services.AddAuthorization(options =>
{
    options.AddPolicy("Administrador", policy =>
    {
        policy.RequireClaim("Perfil", "Administrador");
    });
});
```

```
[Area("Admin")]
[Authorize(Policy ="Administrador")]
0 references
public class HomeController : Controller
{
    0 references
    public IActionResult Index()
    {
        return View();
```

Code21

Exercício 14

Incluir Autenticação e Autorização