Armen Melkumyan • 1st
Technical / Solutions Architect
3mo • Edited •

.Net Technical Question

You have an API serving three user subscription tiers Standard, Plus, and Premium each entitled to a fixed number of requests per hour (20, 50, and 100, respectively). You need to ensure that once a user reaches their allotted limit within the hour, further requests are blocked (or returned with HTTP 429). The solution should:

Identify the user tier reliably (via claims, roles, or headers).

Enforce per-tier limits (20, 50, 100 requests/hour) in a maintainable way.

Scale in production, possibly using a distributed cache (e.g., Redis).

Follow best practices, such as returning helpful error codes/headers and integrating with logs/metrics.

How do you design and implement a rate-limiting solution in C#/.NET that enforces these per-tier request quotas, handles multi-instance deployments, and provides useful feedback (HTTP 429, retry headers, etc.) to clients?

Solution Using .NET Built-In RateLimiter

Below is a straightforward implementation using .NET s built-in RateLimiter middleware. This example enforces 20 requests/hour for Standard, 50 for Plus, and 100 for Premium users.

Any other ideas ?

#DotNet #Csharp #InterviewQuestions #RateLimiter

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddRateLimiter(options =>
{
    options.AddPolicy("PerUserTypePolicy", context =>
    {
        var userType = GetUserTier(context);

        return RateLimitPartition.GetFixedWindowLimiter(
            partitionKey: userType,
            partition => new FixedWindowRateLimiterOptions
            {
                PermitLimit = GetLimit(userType), // 20, 50, or 100
                Window = TimeSpan.FromHours(1),
                QueueLimit = 0,                   // No waiting queue
                AutoReplenishment = true
            }
        );
    });
});

var app = builder.Build();

app.UseRateLimiter("PerUserTypePolicy");

app.MapControllers();
app.Run();

string GetUserTier(HttpContext context)
{
    return context.User.FindFirst("subscription_type")?.Value?.ToLower() ?? "standard";
}

int GetLimit(string userType)
{
    return userType switch
    {
        "plus" => 50,
        "premium" => 100,
        _       => 20   // default to 20 for standard or unknown
    };
}
```