

ASP.NET Core

Minimal APIs

#10

{API.Rules}



Nabi Karampoor
@thisisnabi

Meaningful Endpoints

Align API endpoints with resources and use appropriate **HTTP verbs** (GET, POST, PUT, DELETE) for CRUD (Create, Read, Update, Delete) operations.

```
// Fetches all products
app.MapGet("/products", () => _productService.GetAll());

// Retrieves a specific product by ID
app.MapGet("/products/{id}", (int id) => _productService.GetById(id));

// Creates a new product
app.MapPost("/products", (Product product) => _productService.Add(product));

// Updates an existing product
app.MapPut("/products/{id}", (int id, Product product) => _productService.Update(id, product));

// Deletes a product by ID
app.MapDelete("/products/{id}", (int id) => _productService.Delete(id));

// Patch Updating specific parts of a resource
app.MapPatch("/products/{id}", (int id, bool status) => _productService.Active(id, status));
```



These verbs are fundamental to how client and servers communicate.



Nabi Karampoor
@thisisnabi

#1

Versioning

Implement API versioning to manage **changes** and **avoid breaking** existing integrations. Consider using URL segments, headers, or media types for versioning.

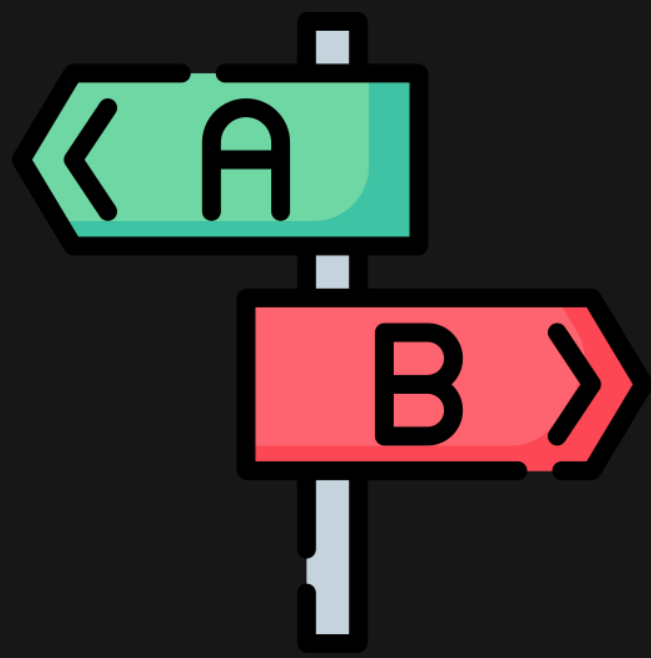
Versioning via URLs

```
app.MapGet("/api/v1/products", () => _productService.GetProducts());  
  
app.MapGet("/api/v2/products", () => _productService.GetProductsWithDetails());
```

Versioning via Headers

Versioning via QueryStrings

Versioning via URLs



We can...



Nabi Karampoor
@thisisnabi

#2

Meaningful Responses

Use appropriate HTTP **status codes** to communicate the outcome of API requests (2xx for success, 4xx for client errors, 5xx for server errors).

```
app.MapPost("/products", (Product product) =>
{
    if (_productService.AddProduct(product))
    {
        return Results.Created($"/products/{product.Id}", product);
    }
    return Results.BadRequest("Invalid product data");
});
```

HTTP

Status Code

1xx Informational

2xx Success

3xx Redirection

4xx Client Errors

5xx Server Errors



Nabi Karampoor
@thisisnabi

#3

Error Handling

Design a robust error handling mechanism to provide **informative error messages** and **avoid exposing sensitive information**.

```
app.MapPost("/products", (Product product) =>
{
    try
    {
        _productService.AddProduct(product);
        return Results.Created($"/products/{product.Id}", product);
    }
    catch (ProductAlreadyExistsException ex)
    {
        return Results.Problem(detail: ex.Message, statusCode: 409);
    }
});

public sealed class ProductAlreadyExistsException : Exception
{
    public ProductAlreadyExistsException(string message) : base(message) { }
}
```



Nabi Karampoor
@thisisnabi

#4

Data Validation

Validate user input to **ensure data integrity** and **prevent security vulnerabilities**. Consider using libraries like FluentValidation or data annotations.

```
app.MapPost("/products", (Product product) =>
{
    var validator = new ProductValidator();
    var validationResult = validator.Validate(product);
    if (!validationResult.IsValid)
    {
        return Results.BadRequest(validationResult.Errors);
    }
    // ... Add product logic
});

public class ProductValidator : AbstractValidator<Product>
{
    public ProductValidator()
    {
        RuleFor(p => p.Name).NotEmpty();
        RuleFor(p => p.Price).GreaterThan(0);
    }
}
```



Nabi Karampoor
@thisisnabi

#5

Security

Implement security measures like authentication and authorization to **protect your API** from unauthorized access.

```
services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        // Configure JWT settings (signing key, issuer, etc.)
    });

services.AddAuthorization(options =>
{
    options.AddPolicy("RequireAdminRole",
        policy => policy.RequireRole("Admin"));
});
```

```
app.MapGet("/products", () => _productService.GetProducts())
    .RequireAuthorization("RequireAdminRole");
```



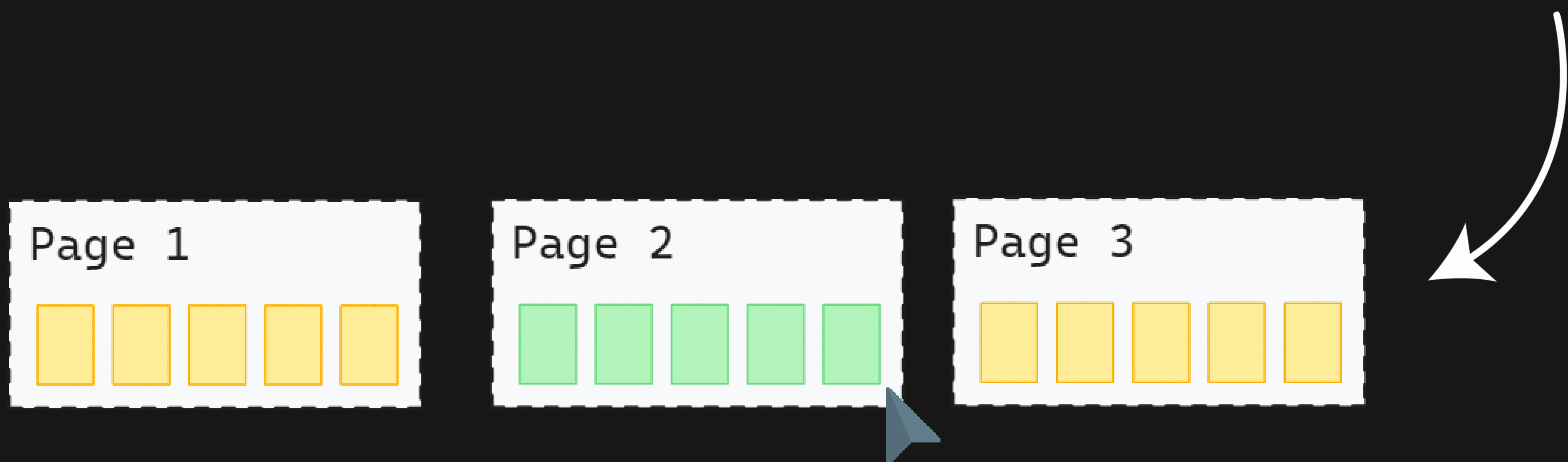
Nabi Karampoor
@thisisnabi

#6

Pagination

| Allow clients to request data in smaller chunks

```
app.MapGet("/products", (int pageIndex = 1, int pageSize = 5) =>
    _productService.GetProducts(pageIndex, pageSize));
```



Pagination is a valuable technique for enhancing both the **user experience** and **developer efficiency** when dealing with large datasets in web applications and APIs.




Nabi Karampoor
@thisisnabi

#7

Filtering

Filtering, in the context of data access and manipulation, refers to the process of selecting a subset of data based on specific criteria.

```
app.MapGet("/products", (ProductFilter filter) =>
{
    var products = _productService.GetProducts(filter);
    return Results.Ok(products);
});
```



```
public IEnumerable<Product> GetProducts(ProductFilter filter)
{
    var query = _products.AsQueryable();

    if (!string.IsNullOrEmpty(filter.Name))
    {
        query = query.Where(p => p.Name.Contains(filter.Name));
    }

    // other filters
    return query;
}
```



Nabi Karampoor
@thisisnabi

#8

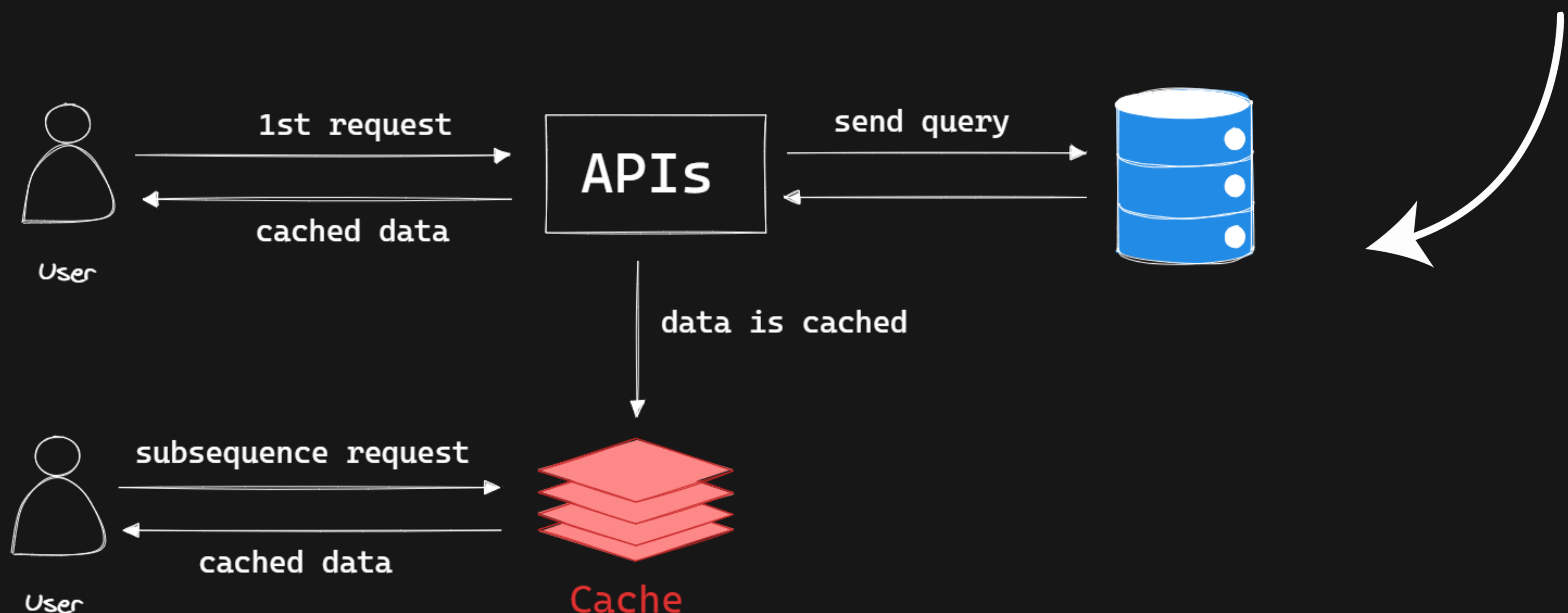
Caching

Implement caching mechanisms to improve API performance and **reduce server load** by storing frequently accessed data in memory.

```
app.MapGet("/products", async (ProductFilter filter) =>
{
    var cacheKey = filter.ToCacheKey();

    var products = await _cacheHandler.GetOrSetAsync(cacheKey,
                                                    () => _productService.GetProductsAsync(filter),
                                                    TimeSpan.FromMinutes(10));

    return Results.Ok(products);
});
```



Nabi Karampoor
@thisisnabi

#9

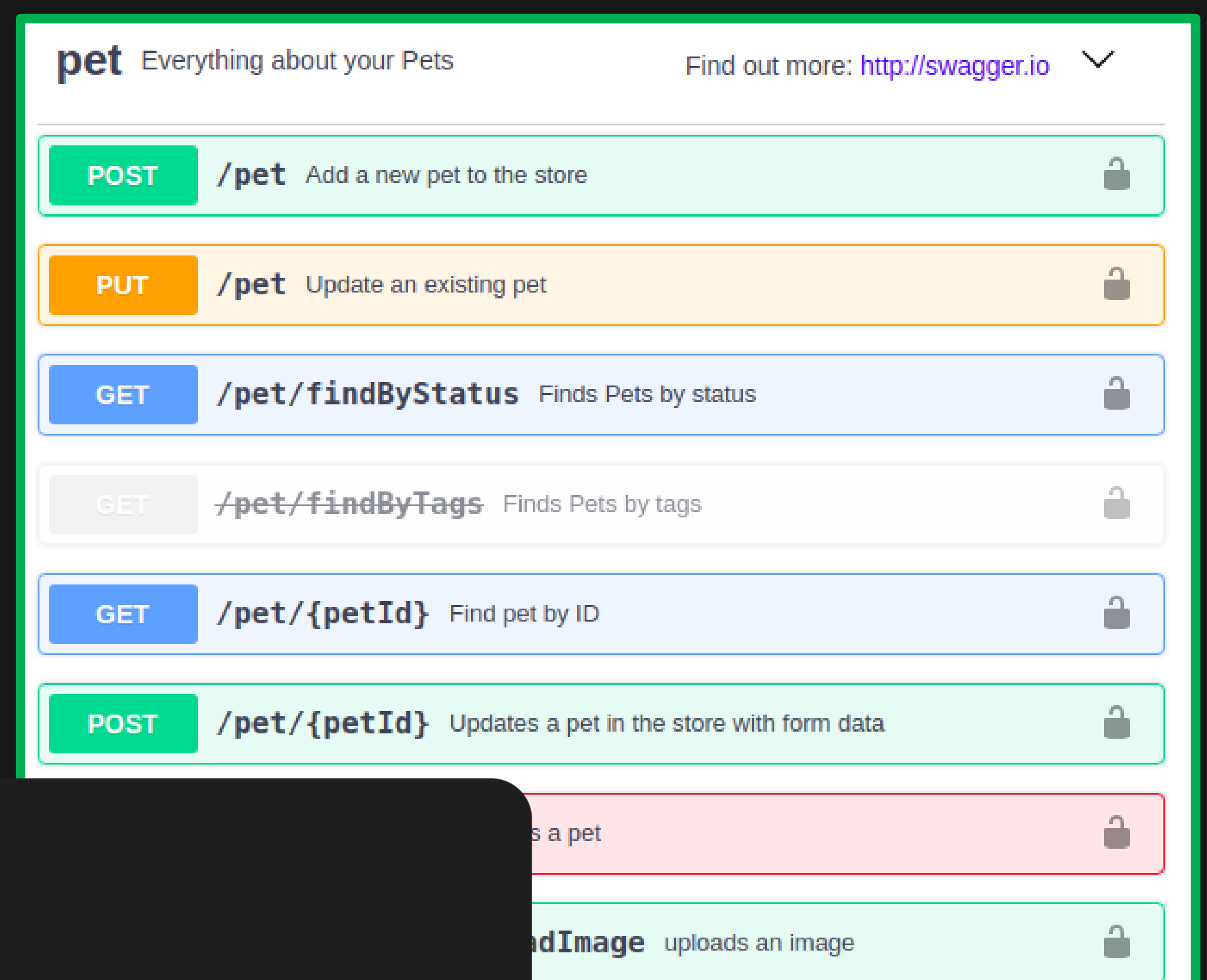
Documentation

Provide comprehensive API documentation to help developers understand endpoints, request and response formats, error handling, and other important details.

```
builder.Services.AddEndpointsApiExplorer();  
builder.Services.AddSwaggerGen();
```

Consider using OpenAPI
(Swagger) to generate
interactive documentation.

```
app.UseSwagger();  
app.UseSwaggerUI(c =>  
{  
    c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");  
});
```



Nabi Karampoor
@thisisnabi

#10

Repost, so your friends can **learn** too.

| Let's follow