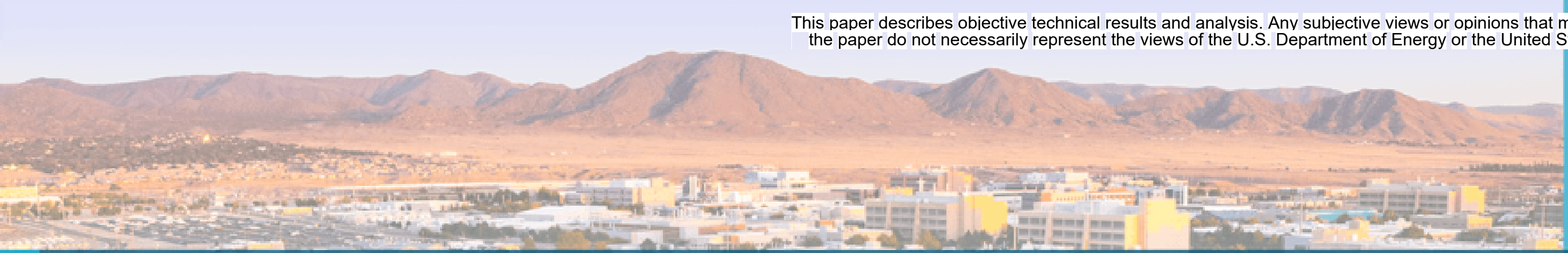
SAND2022-13350C



USEFUL THINGS WE LEARNED ALONG THE WAY

Commonly used components in one application

Greg Arnold – Sandia National Laboratories

Bob McCarthy – Sandia National Laboratories

NLIT Summit 2022

October 16 -19, 2022

Albuquerque, New Mexico

Introduction – Who are we?

Greg Arnold – Software System Engineer

Gregory Arnold is a software developer with over 25 years of experience. He has been working at Sandia National Laboratories for the last 13 years, developing innovative web applications for mission support. He has experience with many programming languages, but loves Python the most. He has BA and MA degrees in Linguistics. He enthusiastically advocates Agile methodologies. His favorite advice to new developers is “Why do it the easy way when you can do it the hard way.”

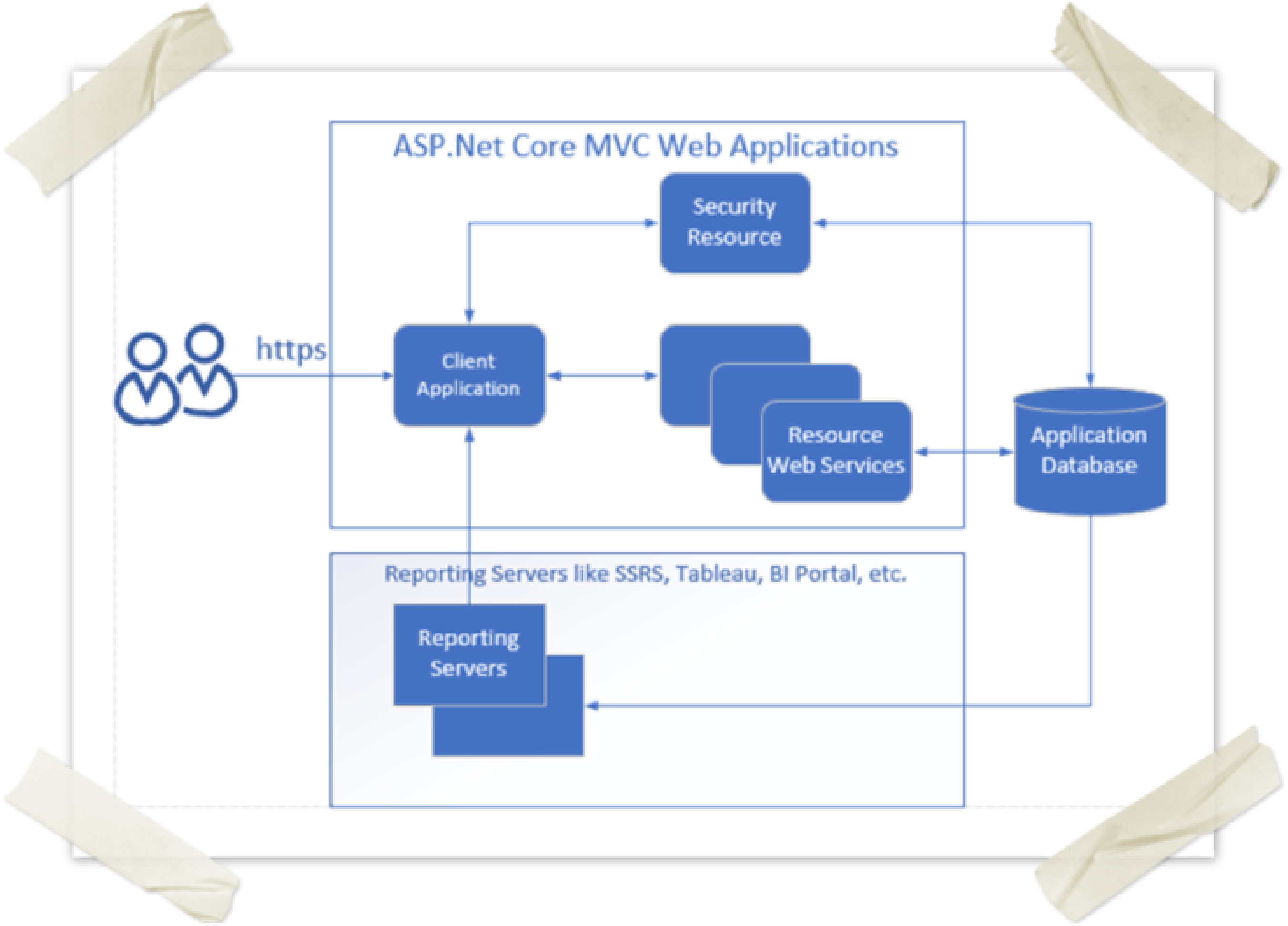
Bob McCarthy – Software System Engineer

- Currently been working at SNL for 6 years
- Over 35 years of development experience working for New Mexico State University, Westinghouse Corporation, Texas Utilities, Blue Cross/BlueShield, Computer Systems Development, Intel Corporation
- MBA – University of New Mexico
- BBA – New Mexico State University

Introduction – Things we learned along the way

- Architecture, Technology Stack and Data Flow
- Token-based security for communicating between web services
- Logging framework for tracking user actions and errors
- Workflow notifications and email for facilitating status-based business processes
- User security pipeline for managing online user presence and caching
- CI/CD Pipeline

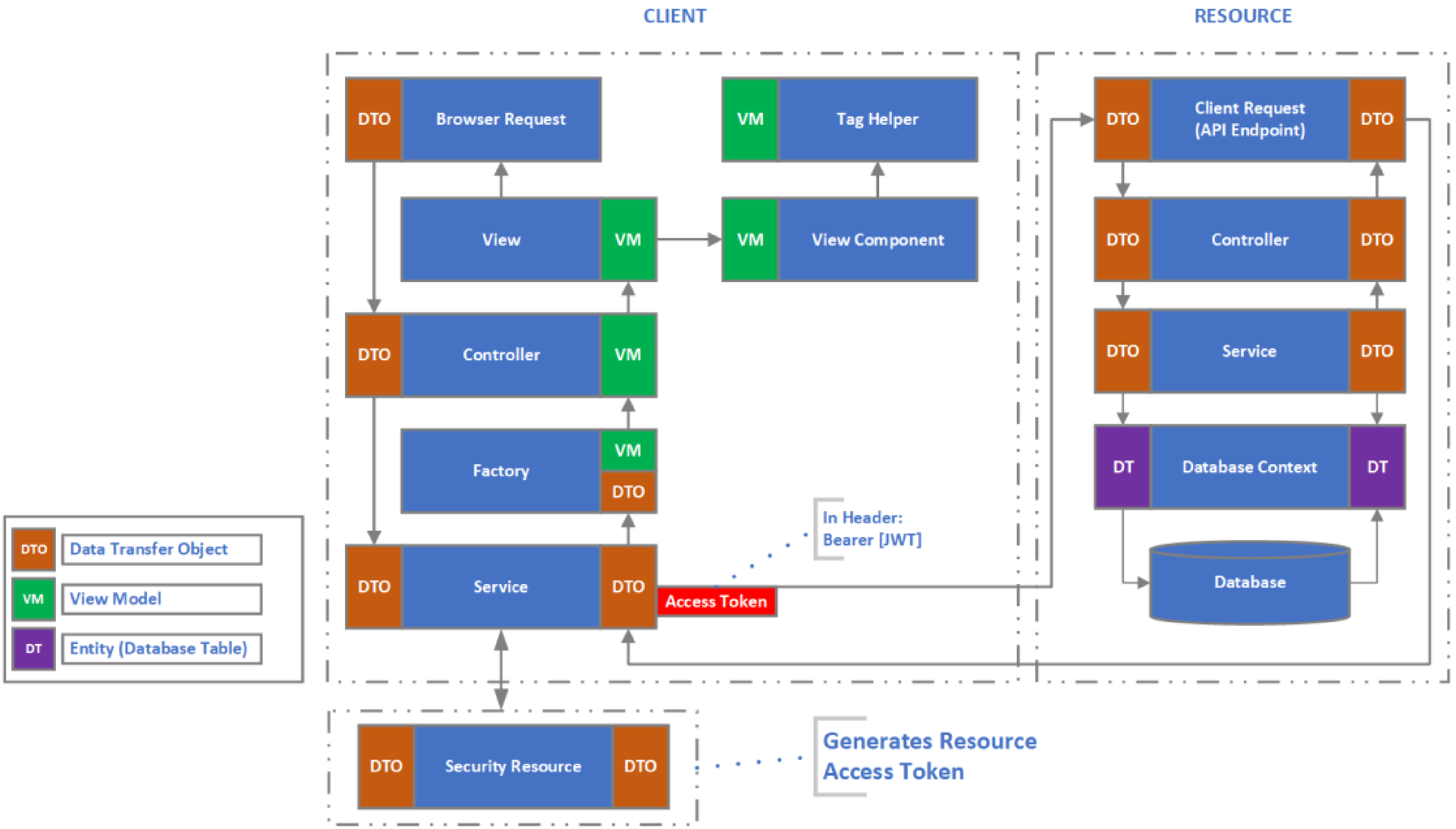
Architecture



Technology

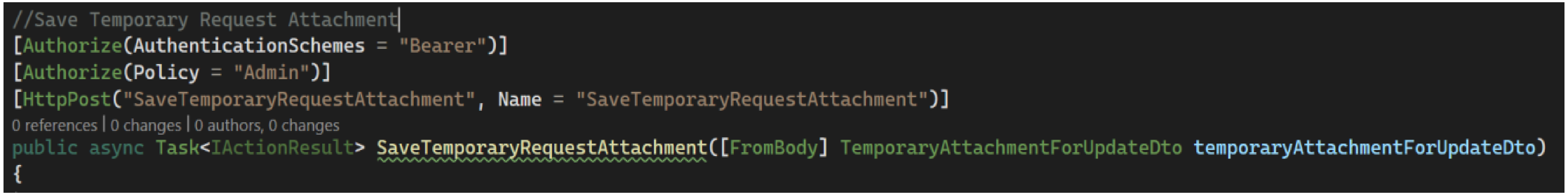
- Microsoft .NET Core Stack
  - Service Injection
  - Middleware Request Pipeline
  - Global “Usings”
  - Configuration (Application settings files)
- Entity Framework
  - Database Context
  - LINQ
- Microsoft SQL Server
- Model-View-Controller (MVC)
  - View Components
  - Tag Helpers
- wwwroot
  - jQuery/JavaScript
  - Bootstrap
  - Cascading Style Sheets (CSS)

Data Flow


CLIENT
RESOURCE
DTO
Browser Request
VM
Tag Helper
View
VM
VM
View Component
DTO
Controller
VM
Factory
VM
DTO
DTO
Service
DTO
Access Token
In Header:
Bearer [JWT]
DTO
Client Request
(API Endpoint)
DTO
DTO
Controller
DTO
DTO
Service
DTO
DT
Database Context
DT
Database
DTO
Security Resource
DTO
Generates Resource
Access Token
DTO
Data Transfer Object
VM
View Model
DT
Entity (Database Table)

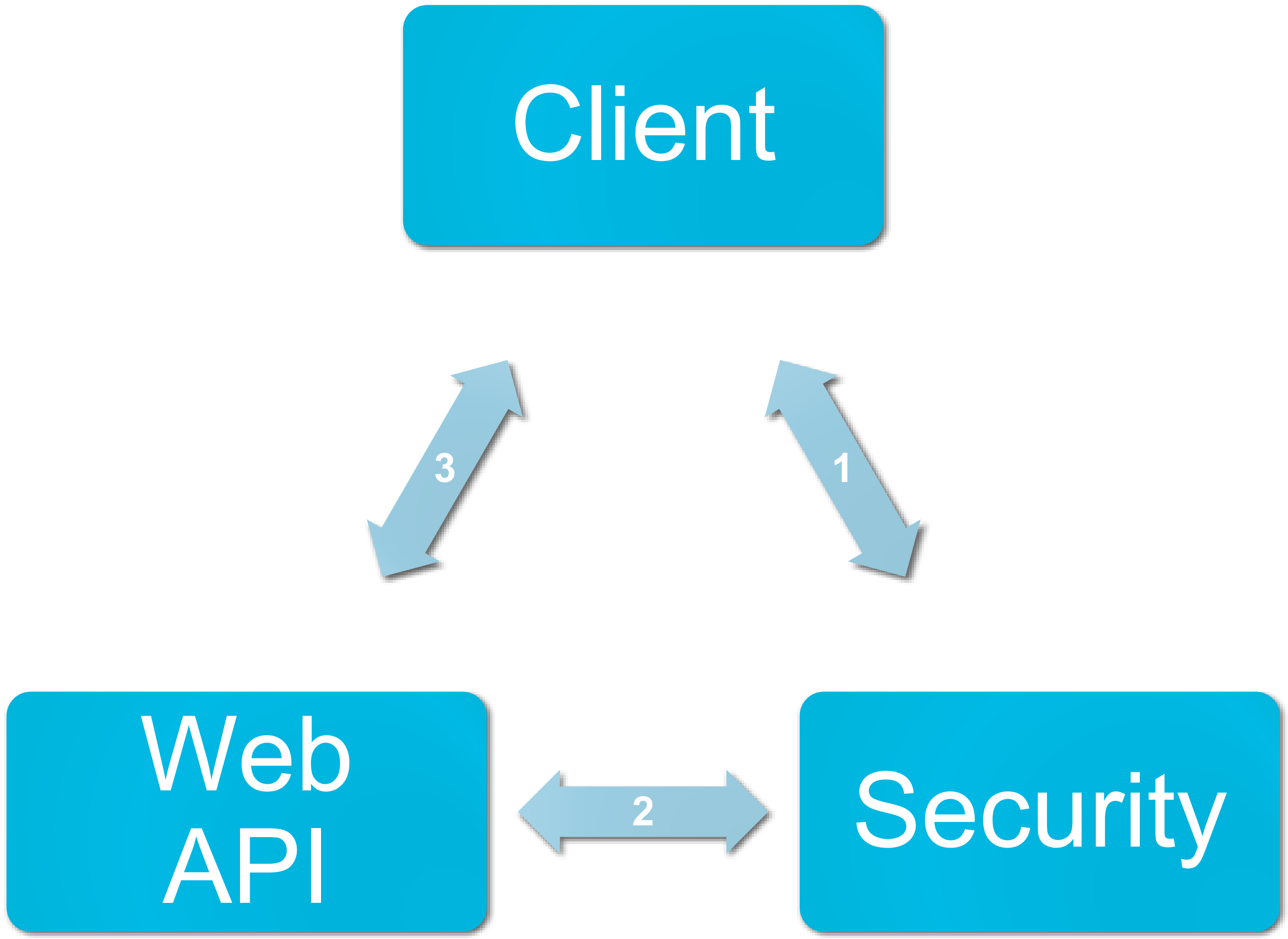
Token-based Security for Web Services - JWT

- JSON Web Tokens
- Request Header
- Security Web API Token Request
- Client Authorization Policies loaded at Resource
- Resource Web API Authorization using Bearer scheme
- Token unpacked and validated against calling Client



```
//Save Temporary Request Attachment
[Authorize(AuthenticationSchemes = "Bearer")]
[Authorize(Policy = "Admin")]
[HttpPost("SaveTemporaryRequestAttachment", Name = "SaveTemporaryRequestAttachment")]
0 references | 0 changes | 0 authors, 0 changes
public async Task<IActionResult> SaveTemporaryRequestAttachment([FromBody] TemporaryAttachmentForUpdateDto temporaryAttachmentForUpdateDto)
{
```

Token-based Security for Web Services - JWT



Logging Framework

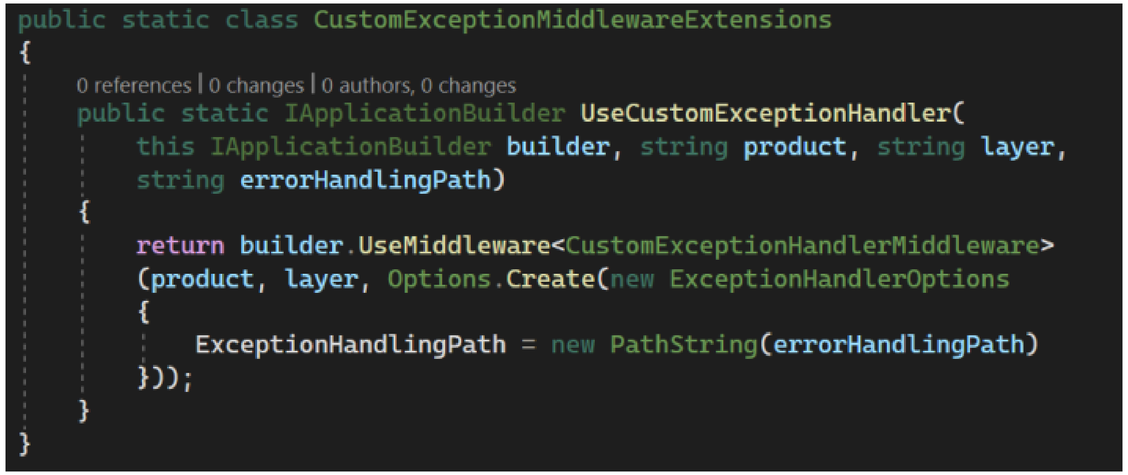
- Two uses of logging
  - Errors
  - Audits
- Resource throws error to client
- Use Middleware Request Pipeline to catch error
- Custom Exception logged from Client application
- User events logged directly for auditing
- Log details are inserted into database

Logging Framework

Startup Class:

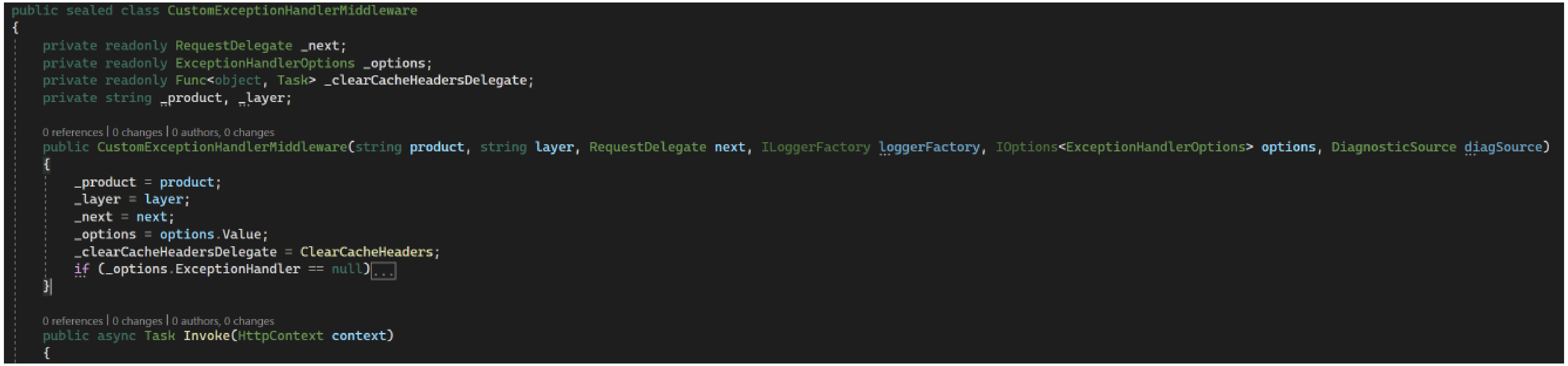
app.UseCustomExceptionHandler("EBIPortal", "Core MVC", "/Home/Error");

Add CustomExceptionMiddlewareExtension class:



```
public static class CustomExceptionMiddlewareExtensions
{
    0 references | 0 changes | 0 authors, 0 changes
    public static IApplicationBuilder UseCustomExceptionHandler(
        this IApplicationBuilder builder, string product, string layer,
        string errorHandlingPath)
    {
        return builder.UseMiddleware<CustomExceptionHandlerMiddleware>
        (product, layer, Options.Create(new ExceptionHandlerOptions
        {
            ExceptionHandlingPath = new PathString(errorHandlingPath)
        }));
    }
}
```

Implement CustomExceptionMiddleware class:

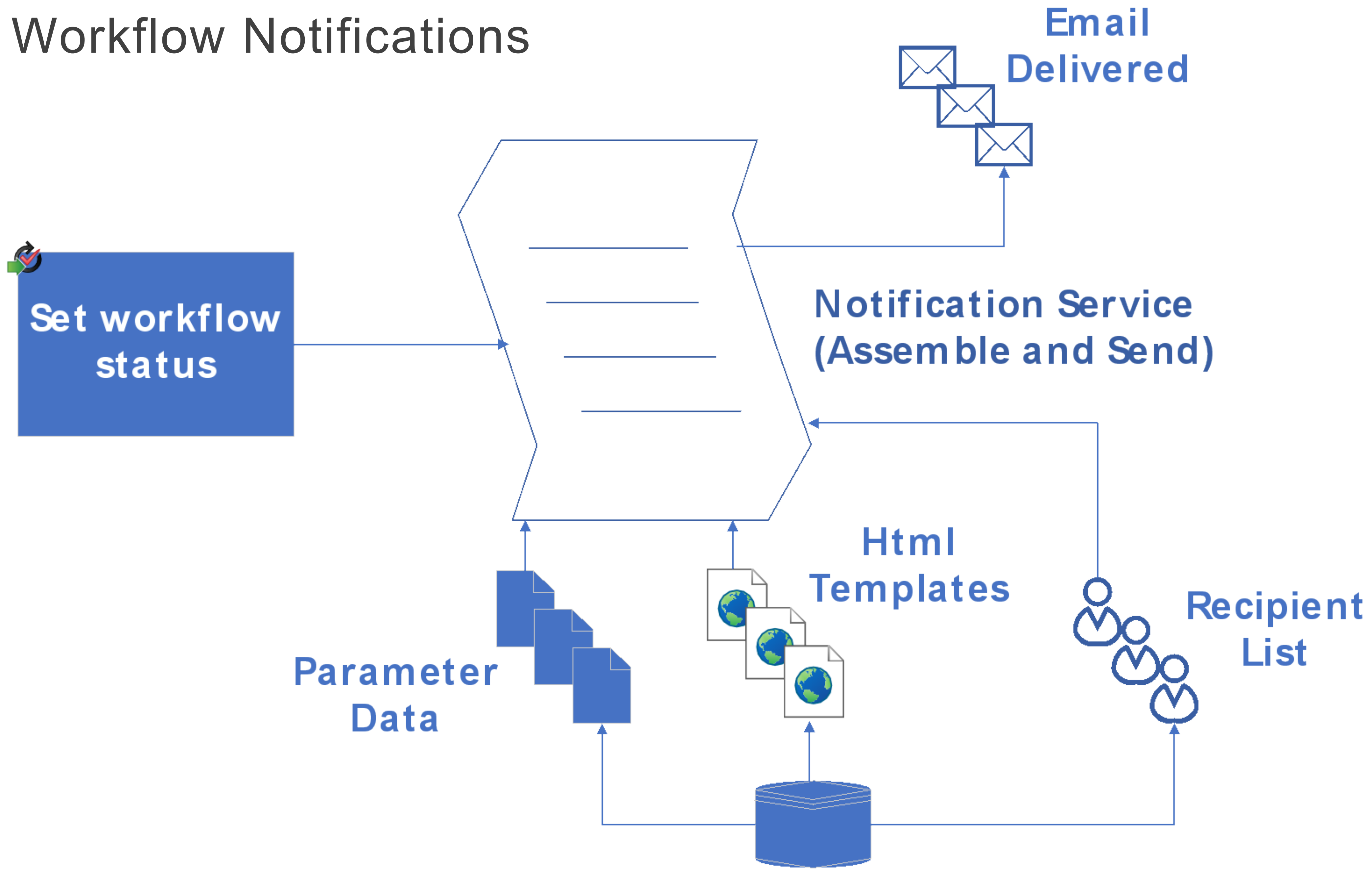


```
public sealed class CustomExceptionHandlerMiddleware
{
    private readonly RequestDelegate _next;
    private readonly ExceptionHandlerOptions _options;
    private readonly Func<object, Task> _clearCacheHeadersDelegate;
    private string _product, _layer;

    0 references | 0 changes | 0 authors, 0 changes
    public CustomExceptionHandlerMiddleware(string product, string layer, RequestDelegate next, ILoggerFactory loggerFactory, IOptions<ExceptionHandlerOptions> options, DiagnosticSource diagSource)
    {
        _product = product;
        _layer = layer;
        _next = next;
        _options = options.Value;
        _clearCacheHeadersDelegate = ClearCacheHeaders;
        if (_options.ExceptionHandler == null)...
    }

    0 references | 0 changes | 0 authors, 0 changes
    public async Task Invoke(HttpContext context)
    {
```

Workflow Notifications

- Workflow Step Initiated (New Status)
- Notification Data Retrieved
  - Notification Template (html)
  - Recipient List
  - Parameter Data
- Parameters Merged with Template
- Email Constructed for Recipients
- Notification Sent

Workflow Notifications



User Security Pipeline

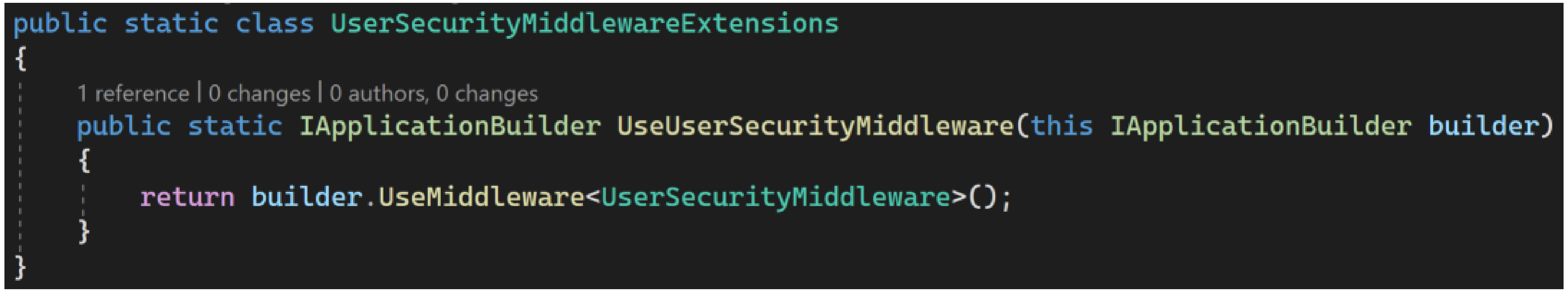
- Use Middleware Request Pipeline to Authorize User
    - Get User Information
    - Get User Role Information
    - Cache Online User in memory
    - Create Timestamp (update when user is active)
    - Invoke Timeout when inactive for configured time period (e.g. 20 minutes)

User Security Pipeline

Startup class:

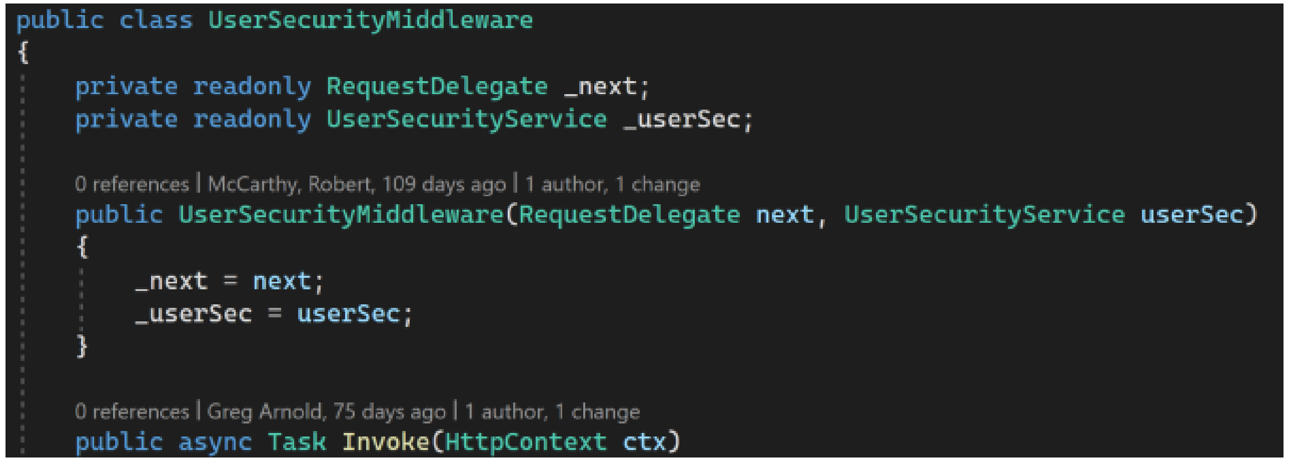
app.UseWhen(context => !context.Request.Path.StartsWithSegments("/Timeout"), appBuilder => { appBuilder.UseUserSecurityMiddleware(); });

Add UserSecurityMiddlewareExtensions class:



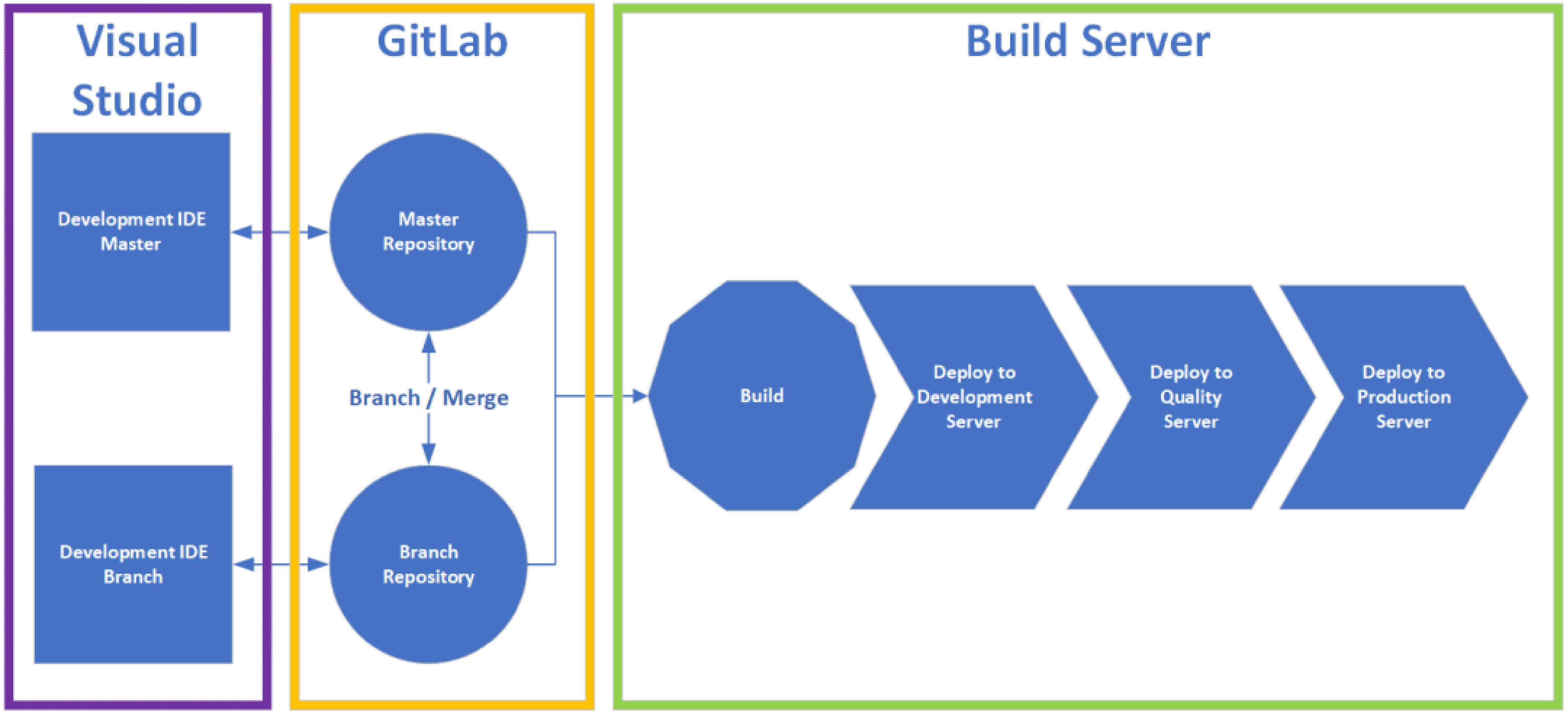
```
public static class UserSecurityMiddlewareExtensions
{
    1 reference | 0 changes | 0 authors, 0 changes
    public static IApplicationBuilder UseUserSecurityMiddleware(this IApplicationBuilder builder)
    {
        return builder.UseMiddleware<UserSecurityMiddleware>();
    }
}
```

Implement UserSecurityMiddleware class:



```
public class UserSecurityMiddleware
{
    private readonly RequestDelegate _next;
    private readonly UserSecurityService _userSec;

    0 references | McCarthy, Robert, 109 days ago | 1 author, 1 change
    public UserSecurityMiddleware(RequestDelegate next, UserSecurityService userSec)
    {
        _next = next;
        _userSec = userSec;
    }

    0 references | Greg Arnold, 75 days ago | 1 author, 1 change
    public async Task Invoke(HttpContext ctx)
```

CI/CD Pipeline



Summary

- Reusable architecture and structures
- Faster project startup time
- Previously tested and reliable components
- Manageable framework that uses simplified development patterns
- Code structures facilitates debugging complete stack
- Simplifies maintenance effort
- Distributed code base reduces memory footprint
- Use of middleware pipeline allows consistent user security, logging and workflow implementations
- Puts the fun back into development!

Q & A

If you ask me anything I don’t know, I’m not going to answer. – Yogi Berra

Contact Us

Bob McCarthy – robmcca@sandia.gov

Greg Arnold – garnold@sandia.gov