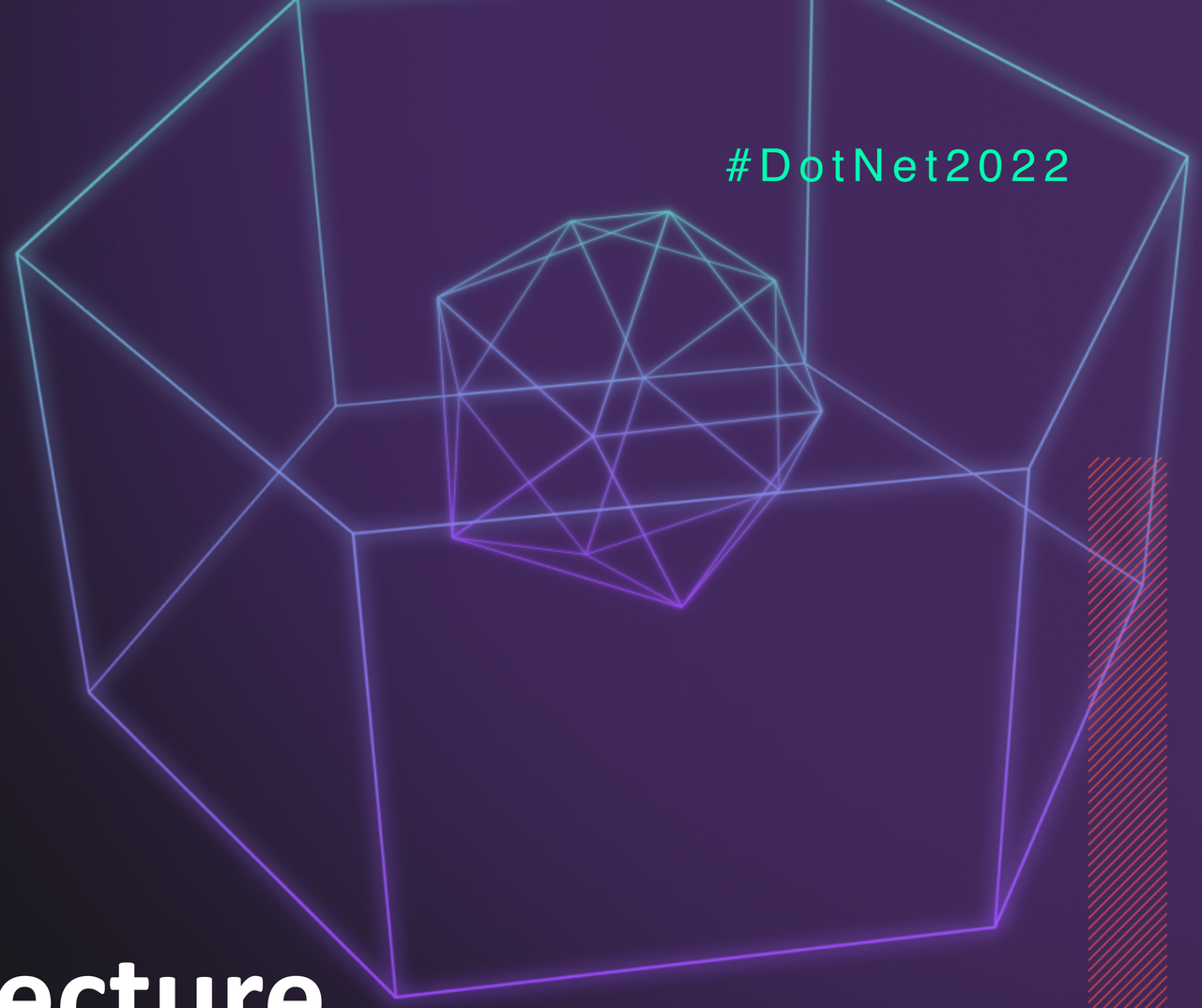
DotNet 2022

22nd June 2022

TECH CONFERENCE

#DotNet2022



Vertical Slice Architecture

Powered by

plain concepts

DotNet 2022

SPONSORS

Microsoft

intel

intelequia

LEMON CODE

DevsDNA

COLLABORATORS

<XantarDev/>
/* Technical Community */

My Public Inbox

SEVILLADOTNET
COMUNIDAD .NET

Power Platform Valencia

W4TT

#DotNet2022

DotNet2022

TECH CONFERENCE

#DotNet2022

Andoni Santamaría

Developer en Plainconcepts

Desarrollador de software y padre a tiempo completo.

Me gusta afrontar cualquier reto y encontrar una solución con la que me pueda sentir satisfecho, la zona de confort no es mi lugar habitual.

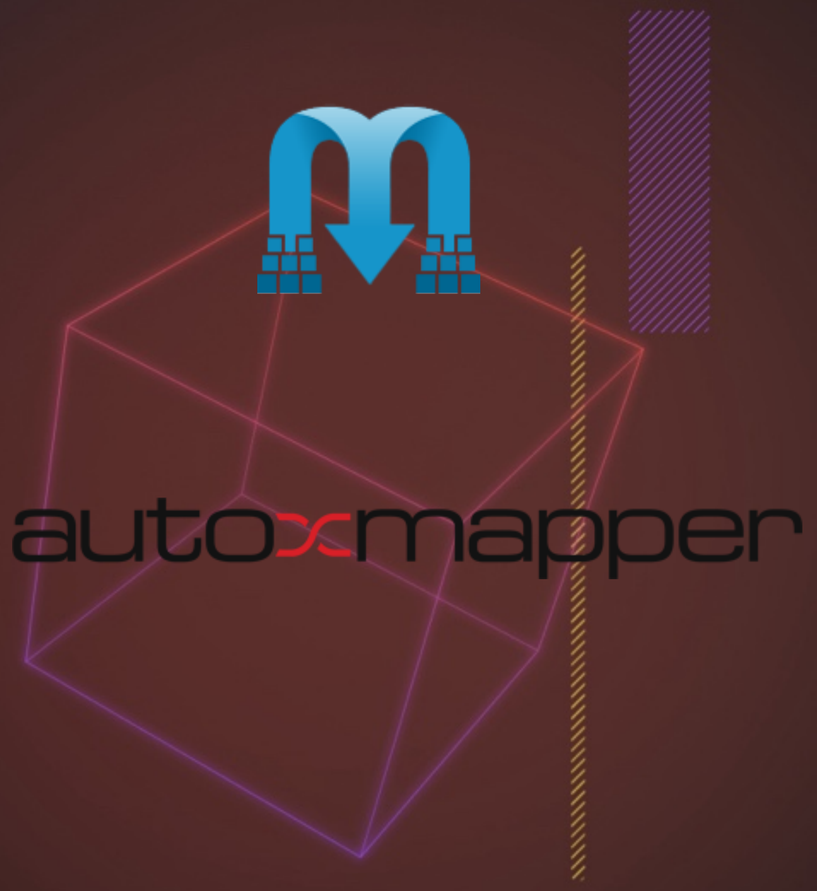
Me considero más backender pero desarrollo cosas decentes en front. Eso sí, no me pidas que combine colores.

#tardeo #cervezeo #colegueo #ps5 #athletic

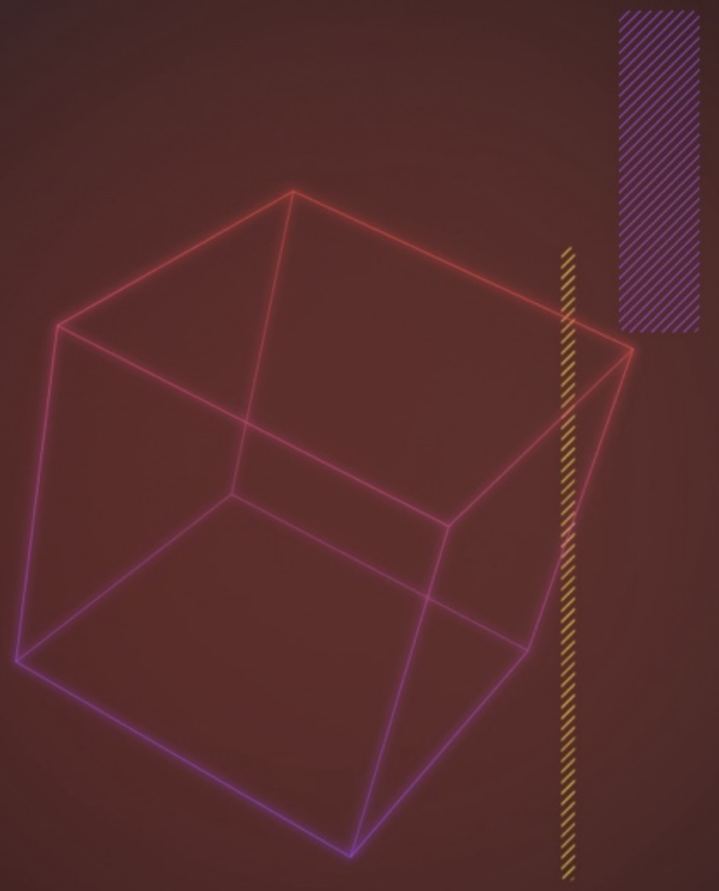
@AndoniSantamari

asantamaria@plainconcepts.com

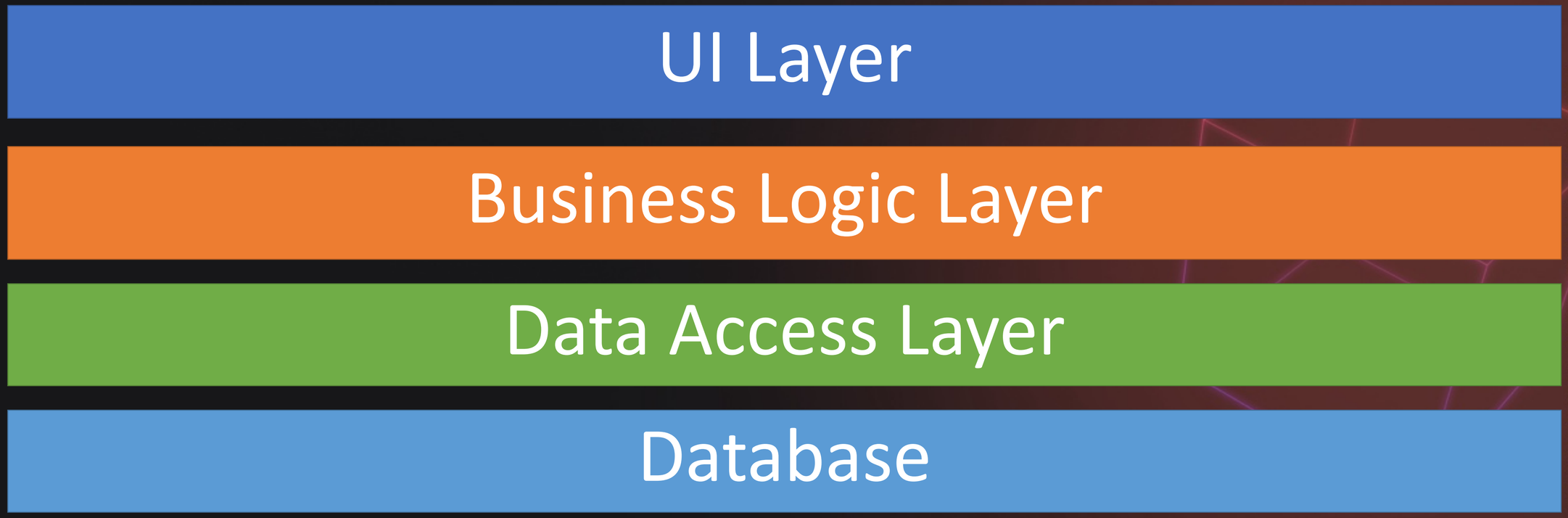
Special thanks to **Jimmy Bogard**



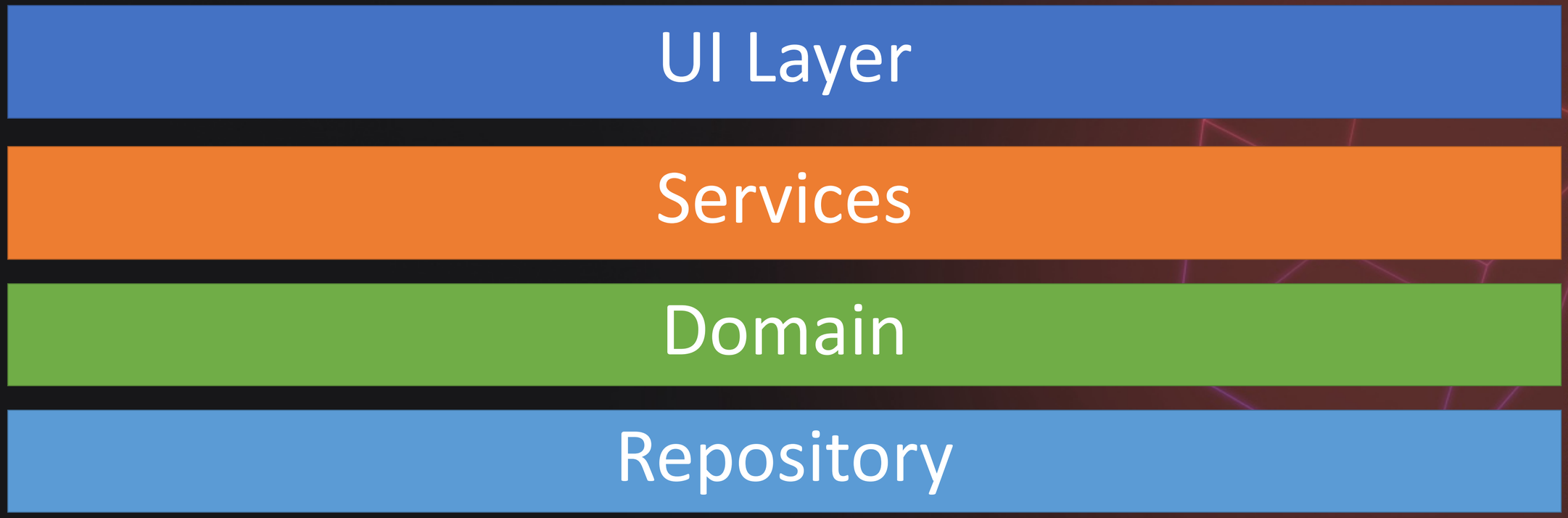
Introduction



Traditional n-tier



“DDD”-style n-tier



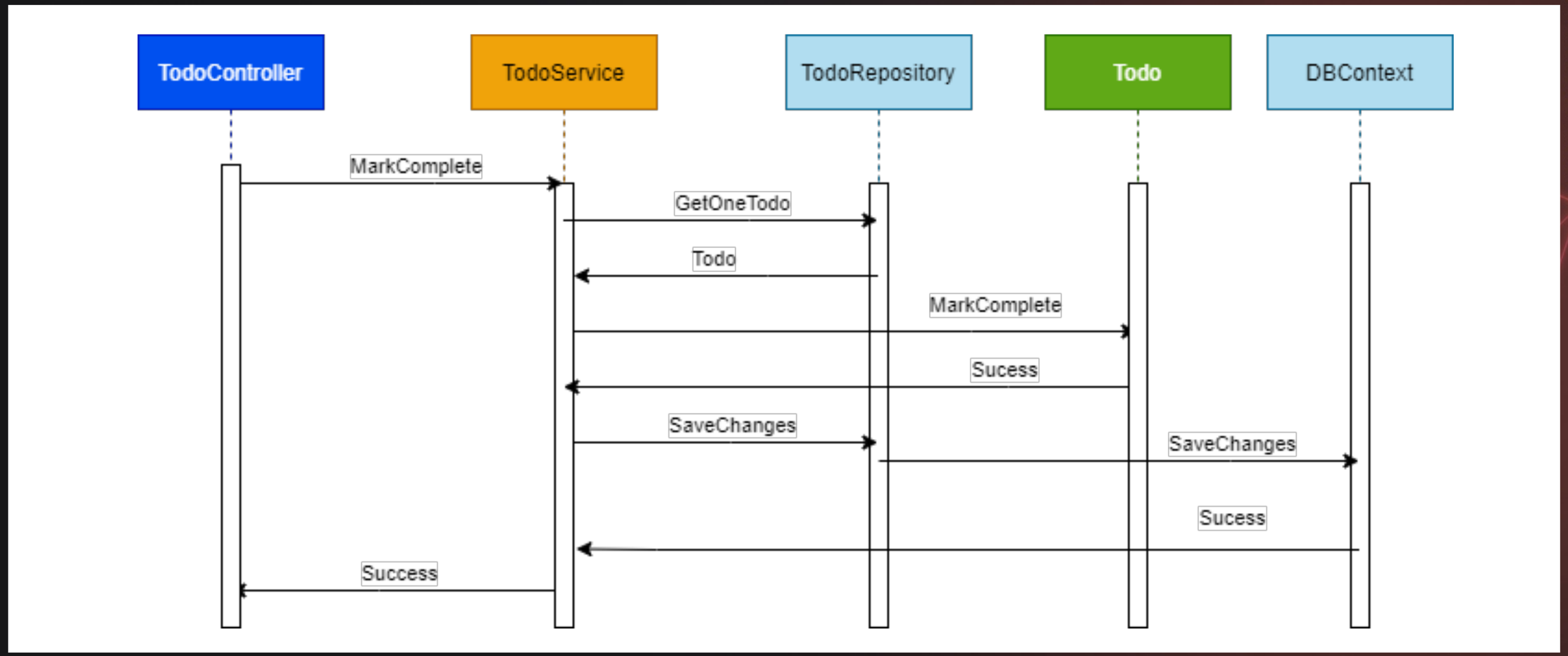
Uncle Bob's Clean Architecture

The primary purpose of architecture is to support the life cycle of the system.

Good architecture makes the system easy to understand, easy to develop, easy to maintain, and easy to deploy.

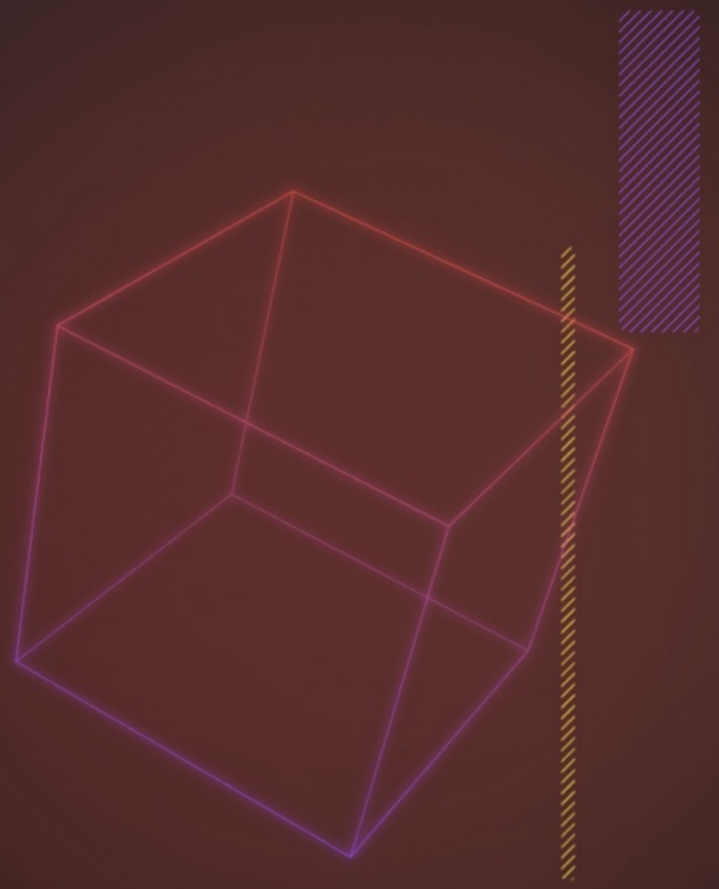
The ultimate goal is to minimize the lifetime cost of the system and to maximize programmer productivity.

Sequence diagram example

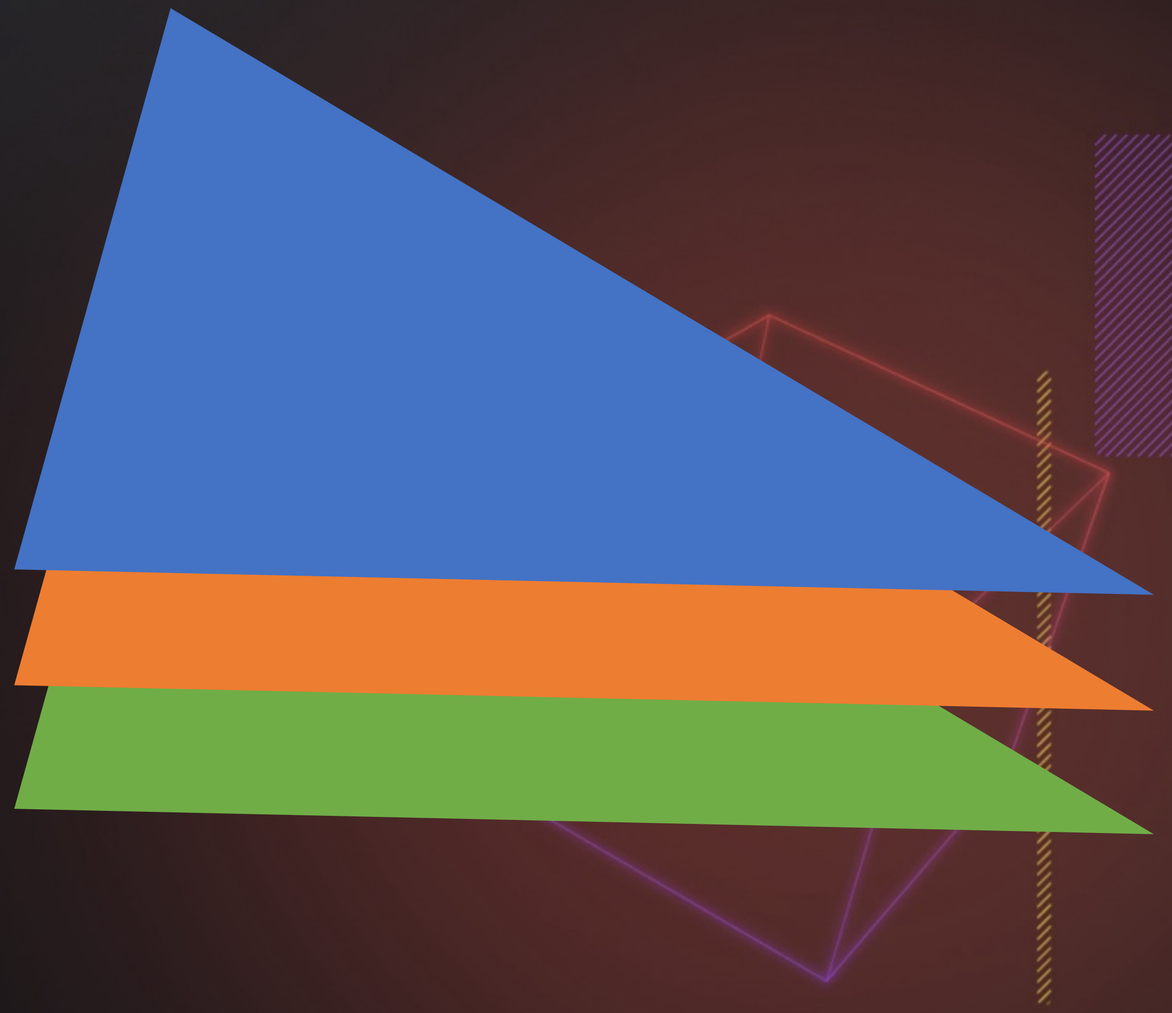


Problems

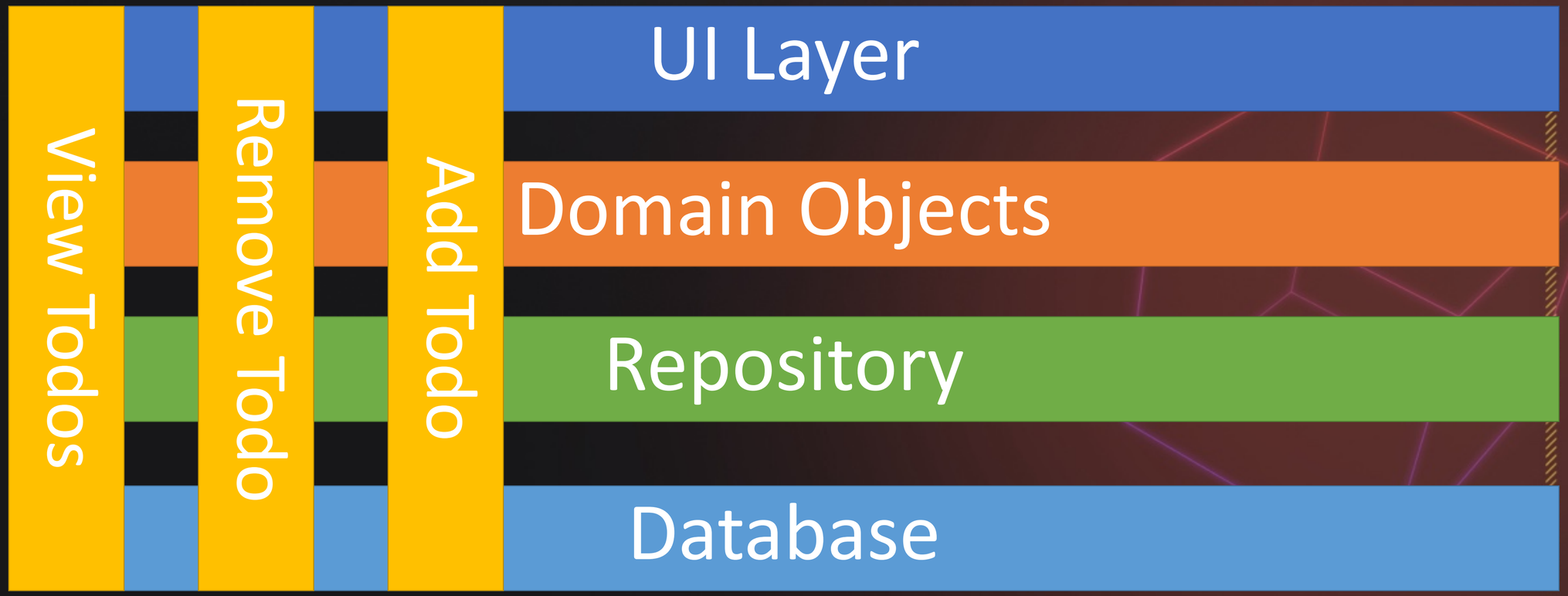
- Difficult for new developers to enter
- Difficult to move to other solution
- Side effects / coupling
- Difficult to maintain
- Heavy testing behaviours / mocks everywhere
- ¿**S**OLID?



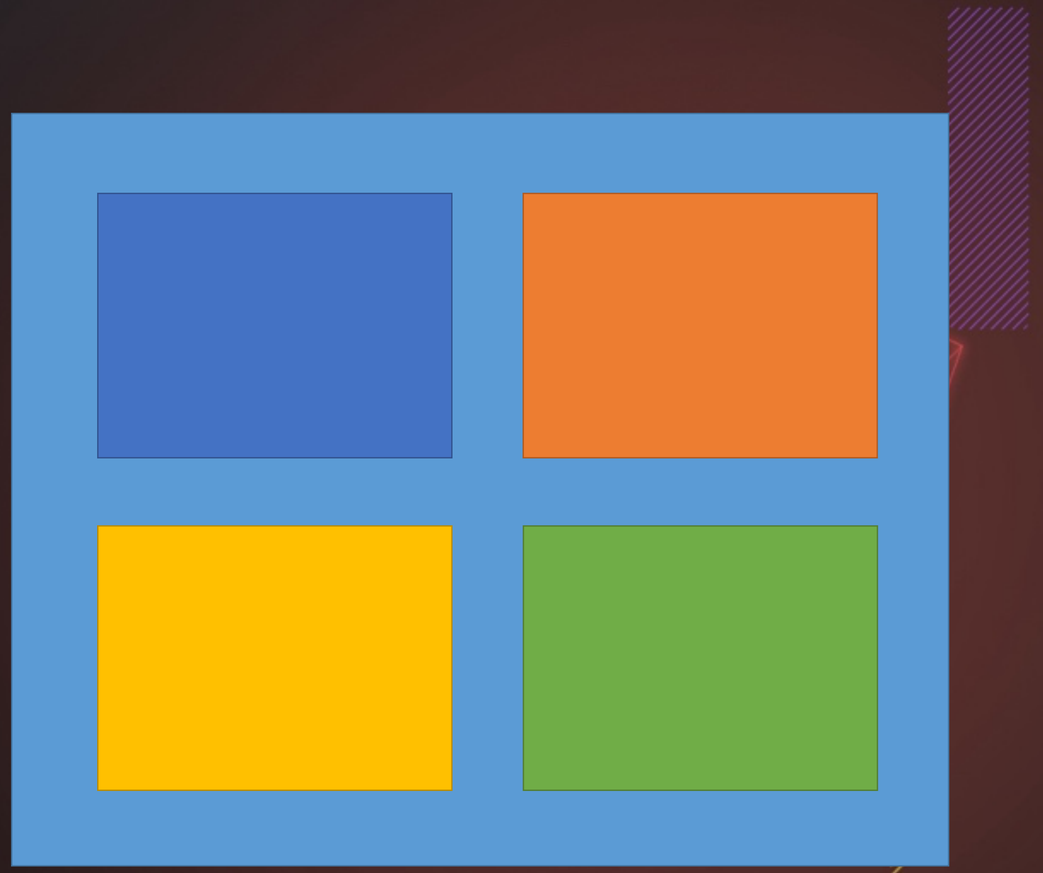
Vertical Slices



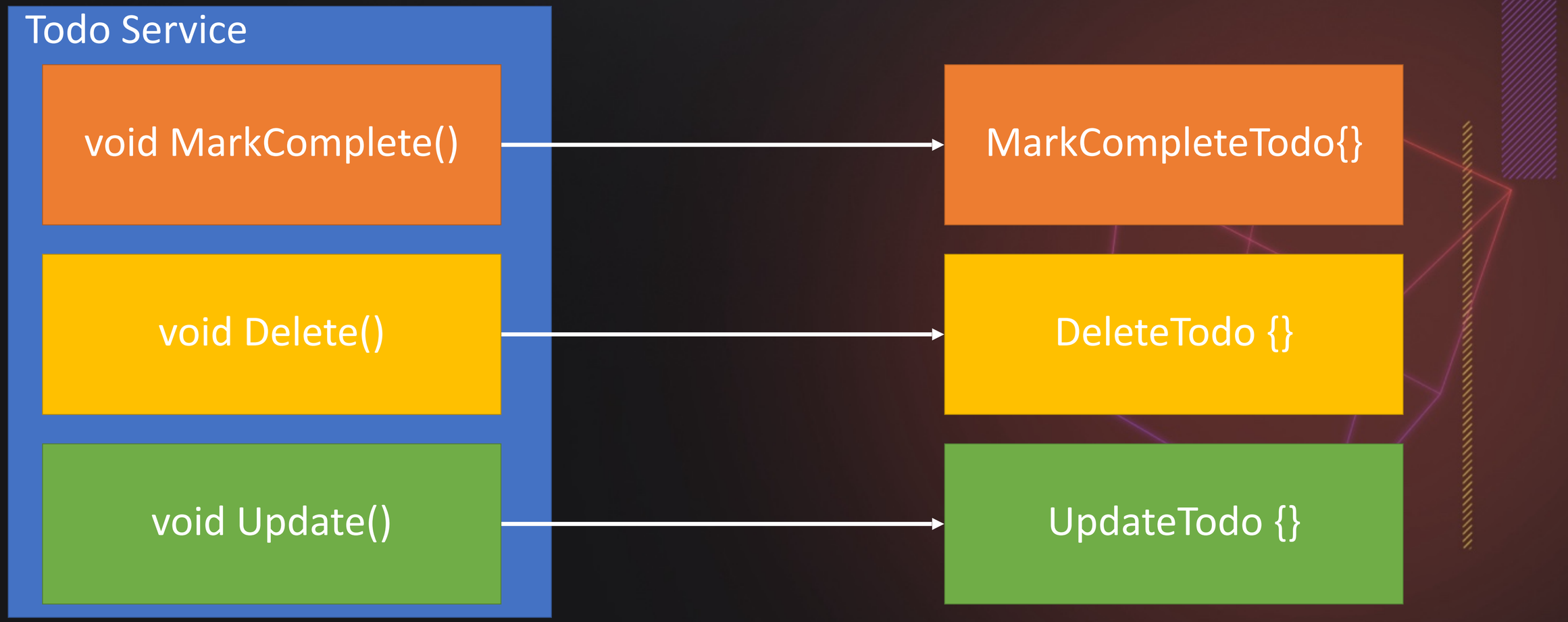
Axes of change



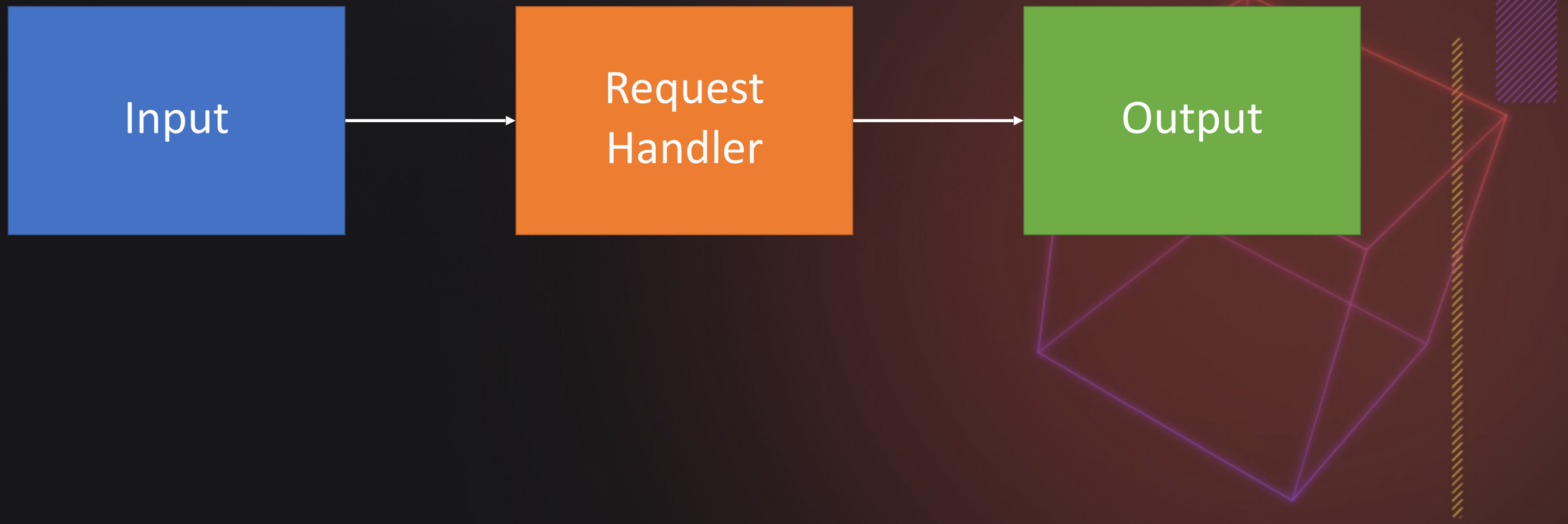
Move code to single place



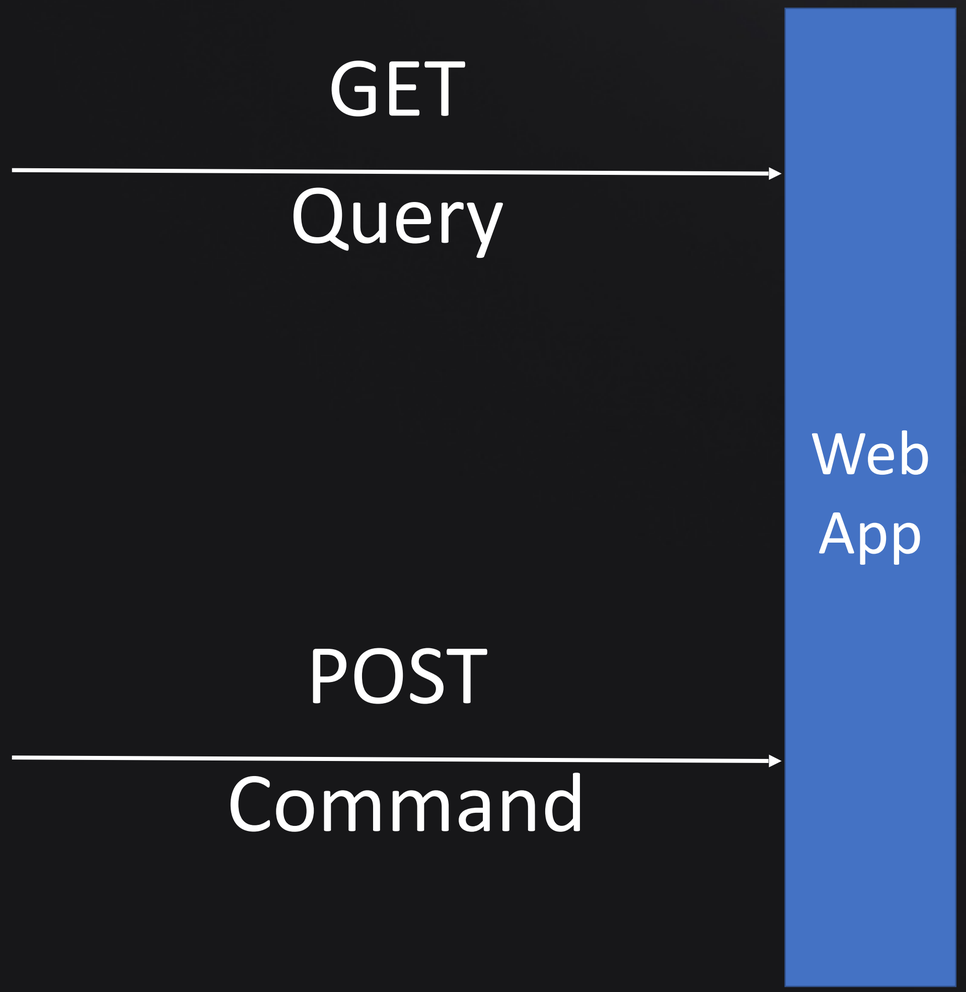
Move service methods to classes



Modeling requests



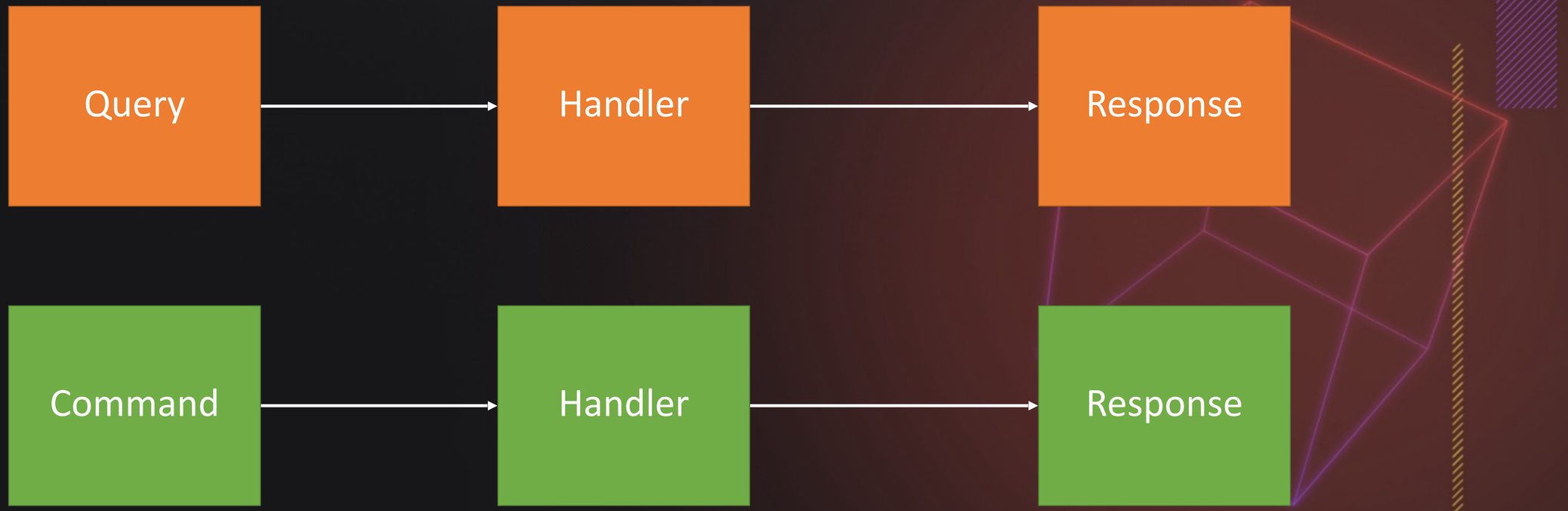
Commands and Queries



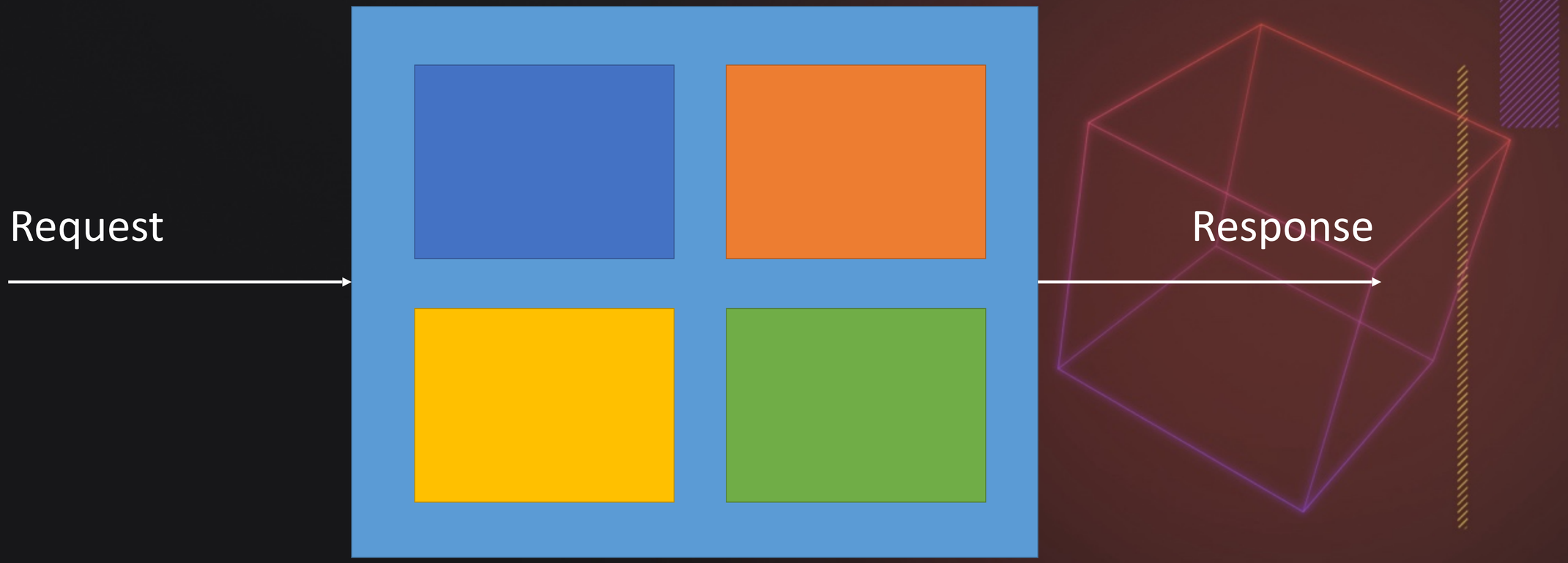
- Safe
- Idempotent

- Unsafe
- Not Idempotent

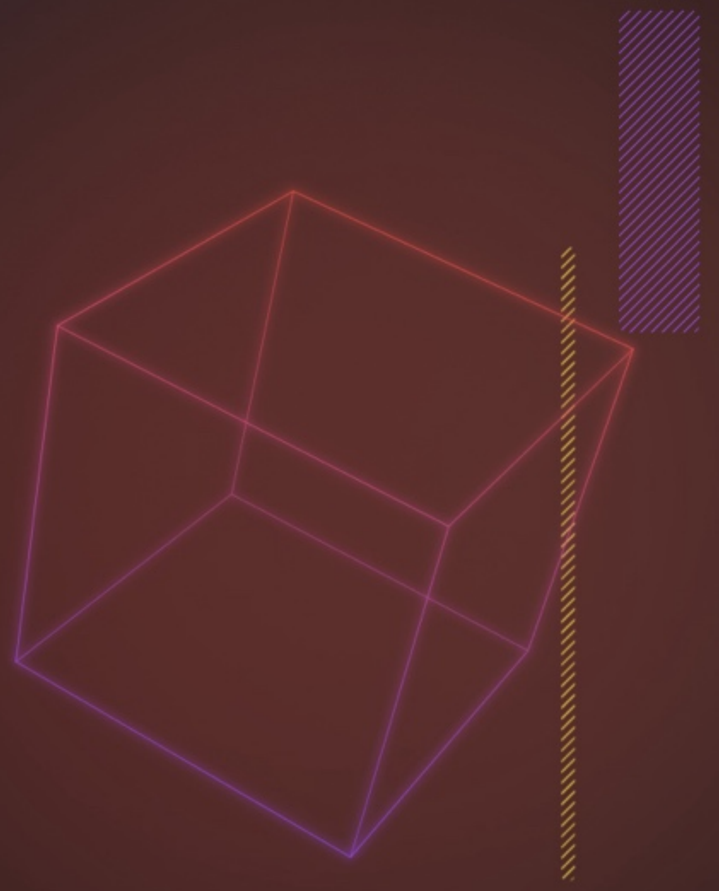
One model in, one model out



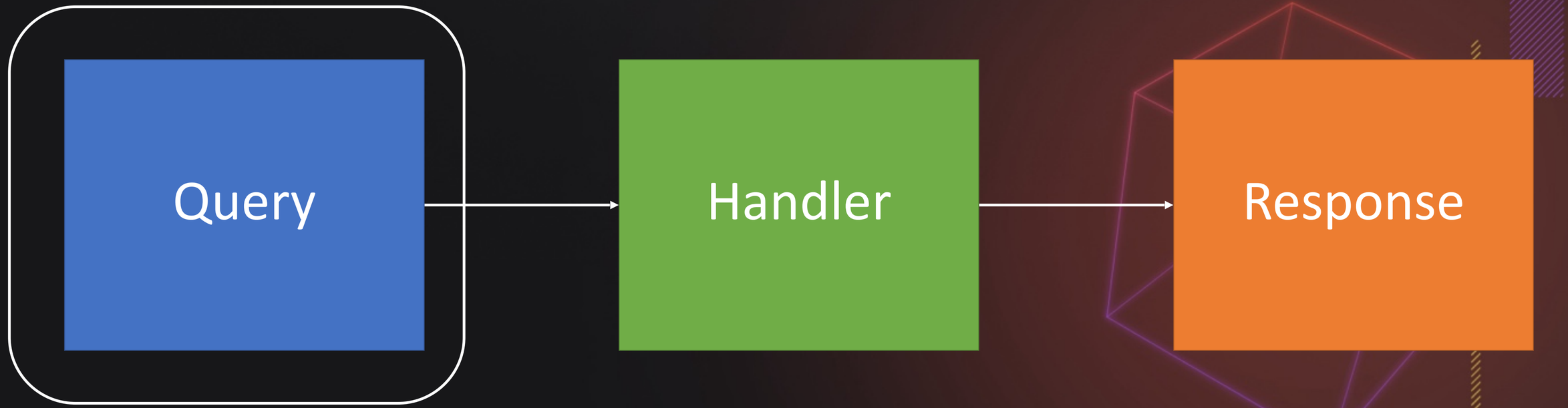
Complete encapsulation



Queries



Modeling Queries



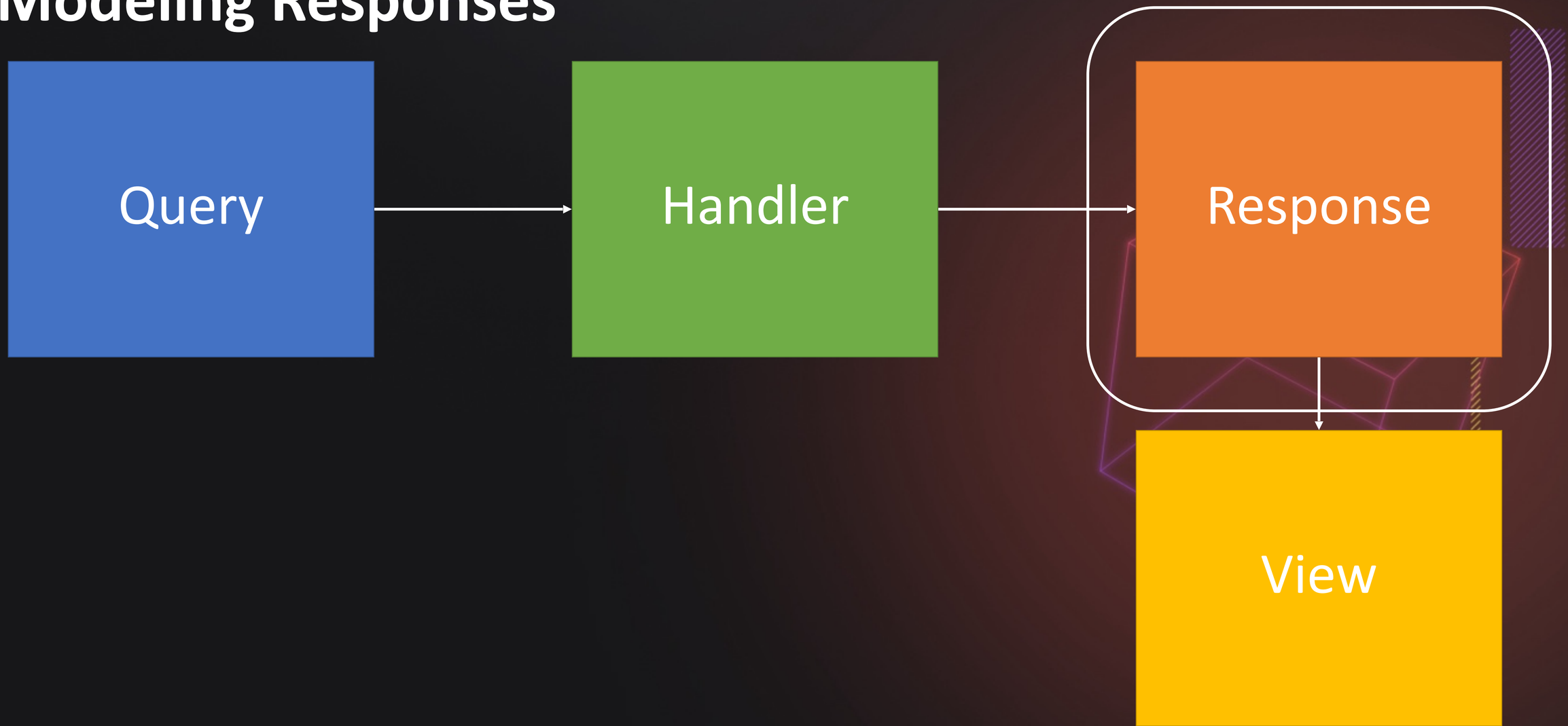
Queries

```
public class GetTodo
{
    public record Query(string TodoId) : IQuery<Response>;
}
```

Pagination Data

```
public record Query() : IQuery<Response>
{
    public int Offset { get; set; } = 0;

    public int Limit { get; set; } = 10;

    public string OrderBy { get; set; } = string.Empty;

    public OrderType OrderType { get; set; } = OrderType.Asc;
}
```

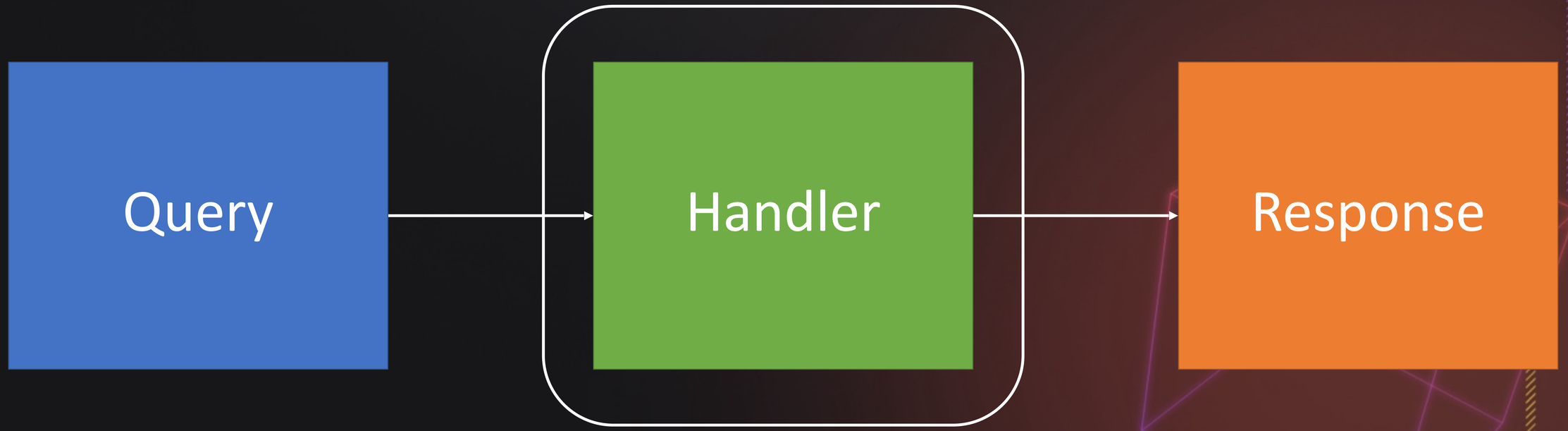
Modeling Responses



Response

```
public class GetTodos
{
    public record Query() : IQuery<PaginatedResponse>;

    public record PaginatedResponse()
    {
        public int Pages { get; set; }
        public Response[] Elements { get; set; } = Array.Empty<Response>();
    }

    public record Response(string Id, string Caption, bool IsCompleted);
}
```

Modeling Query Handlers

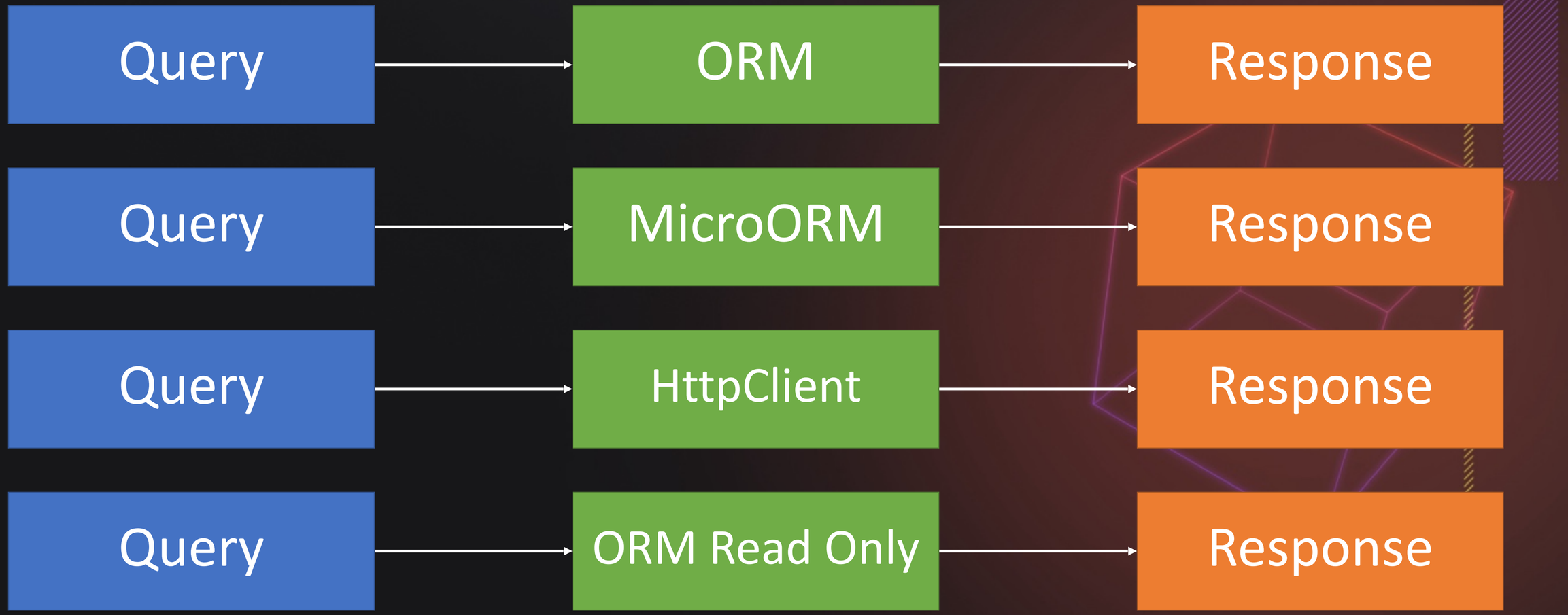


~~NO~~ MINIMAL ABSTRACTIONS ARCHITECTURE

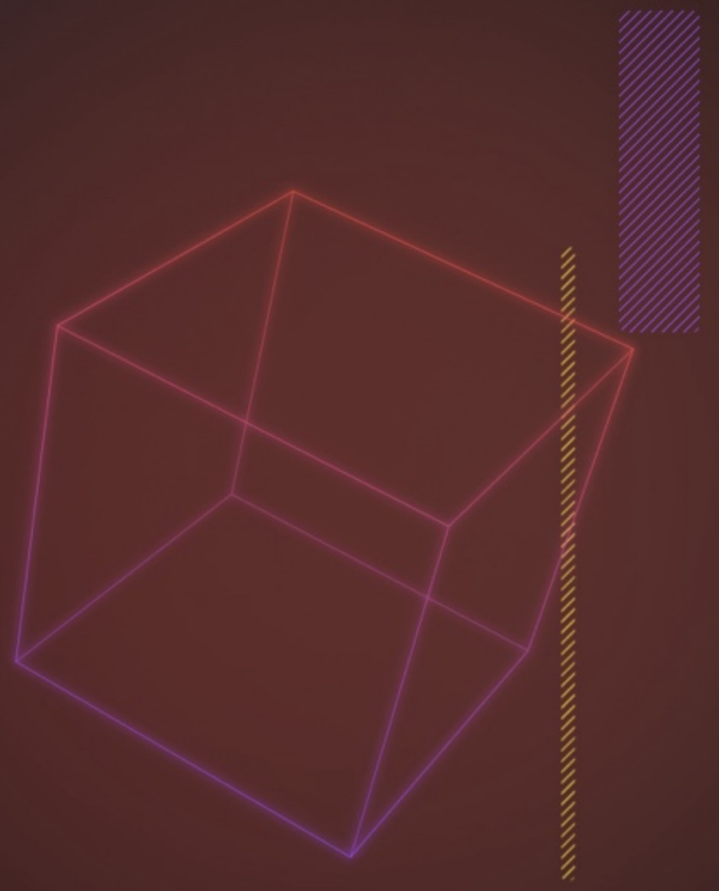
Handler

```
public class RequestHandler : IRequestHandler<Query, Response>
{
    …

    public async Task<Response> Handle(Query request, CancellationToken cancellationToken =
default)
    {
            return await _context.Todos
                .AsNoTracking()
                .Where(t => t.Id == request.TodoId)
                .Select(td => new Response(td.Id, td.Title, td.Completed))
                .SingleOrDefaultAsync(cancellationToken);
        }
    }
}
```

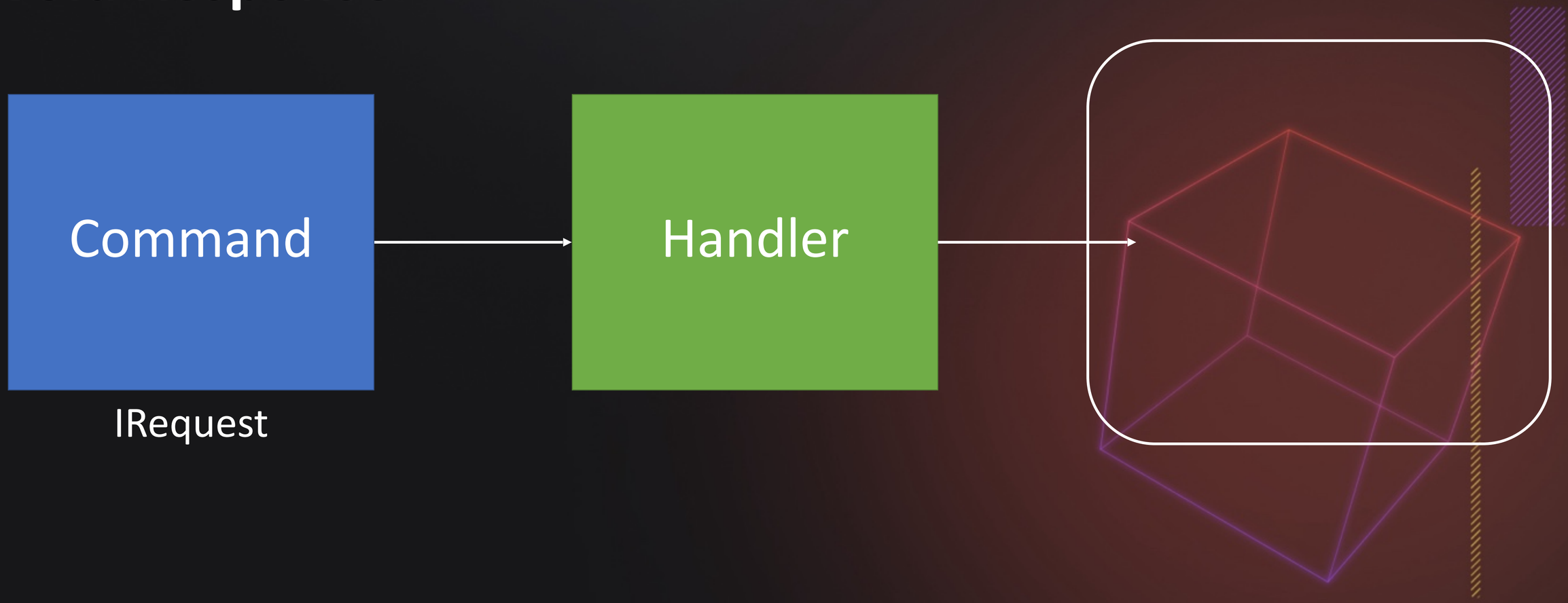
Encapsulated Logic



Commands



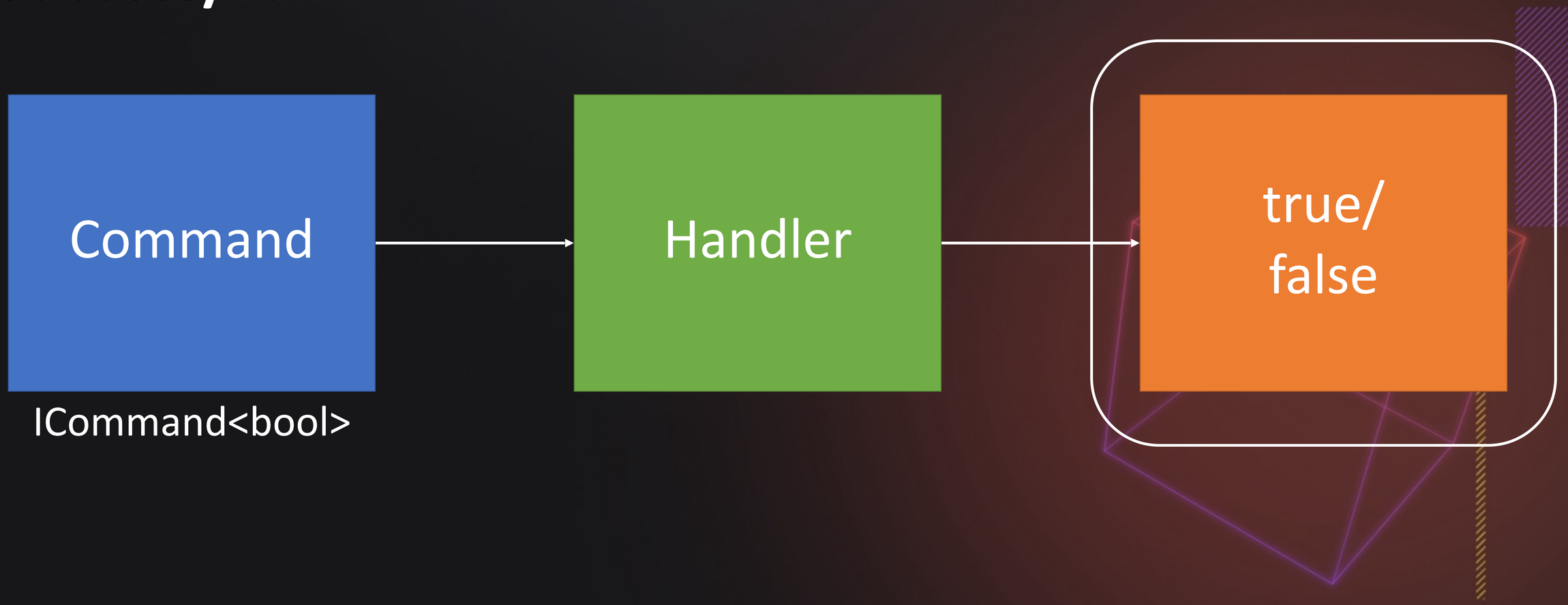
Void Response



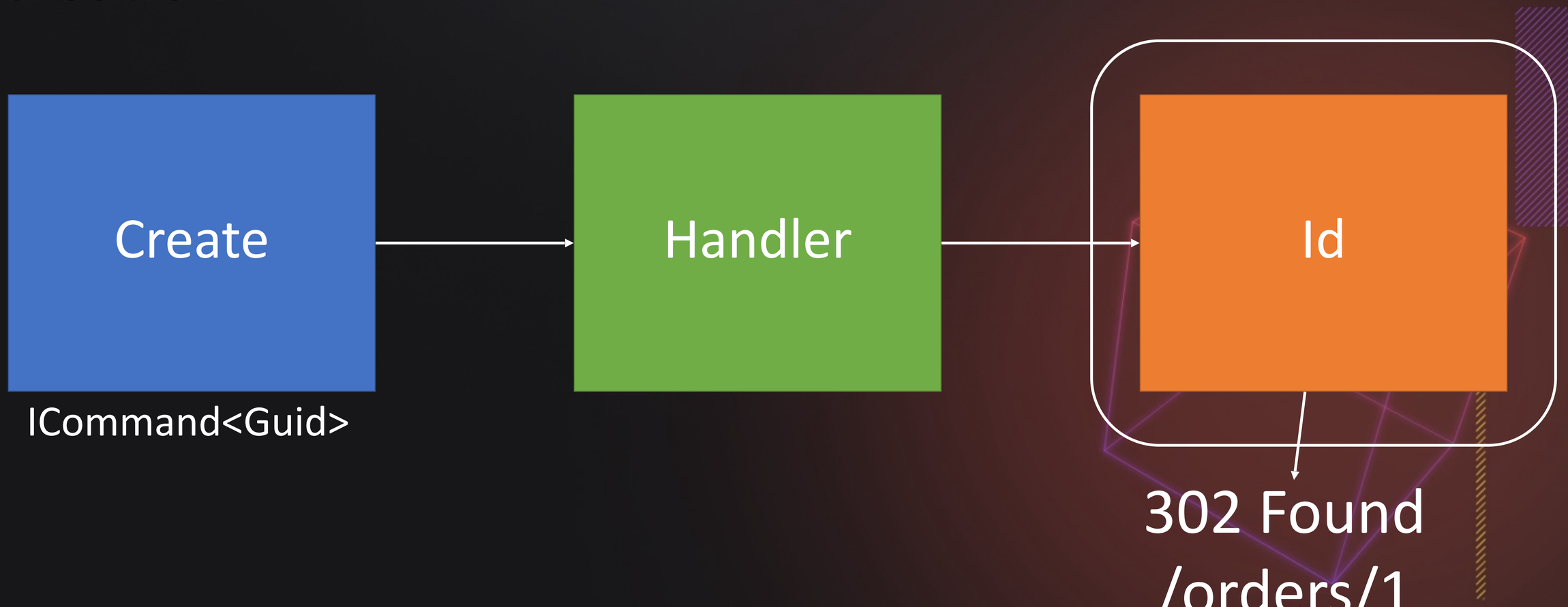
Commands

```
public class CreateTodo
{
    public record Command(string Title) : ICommand;
}
```

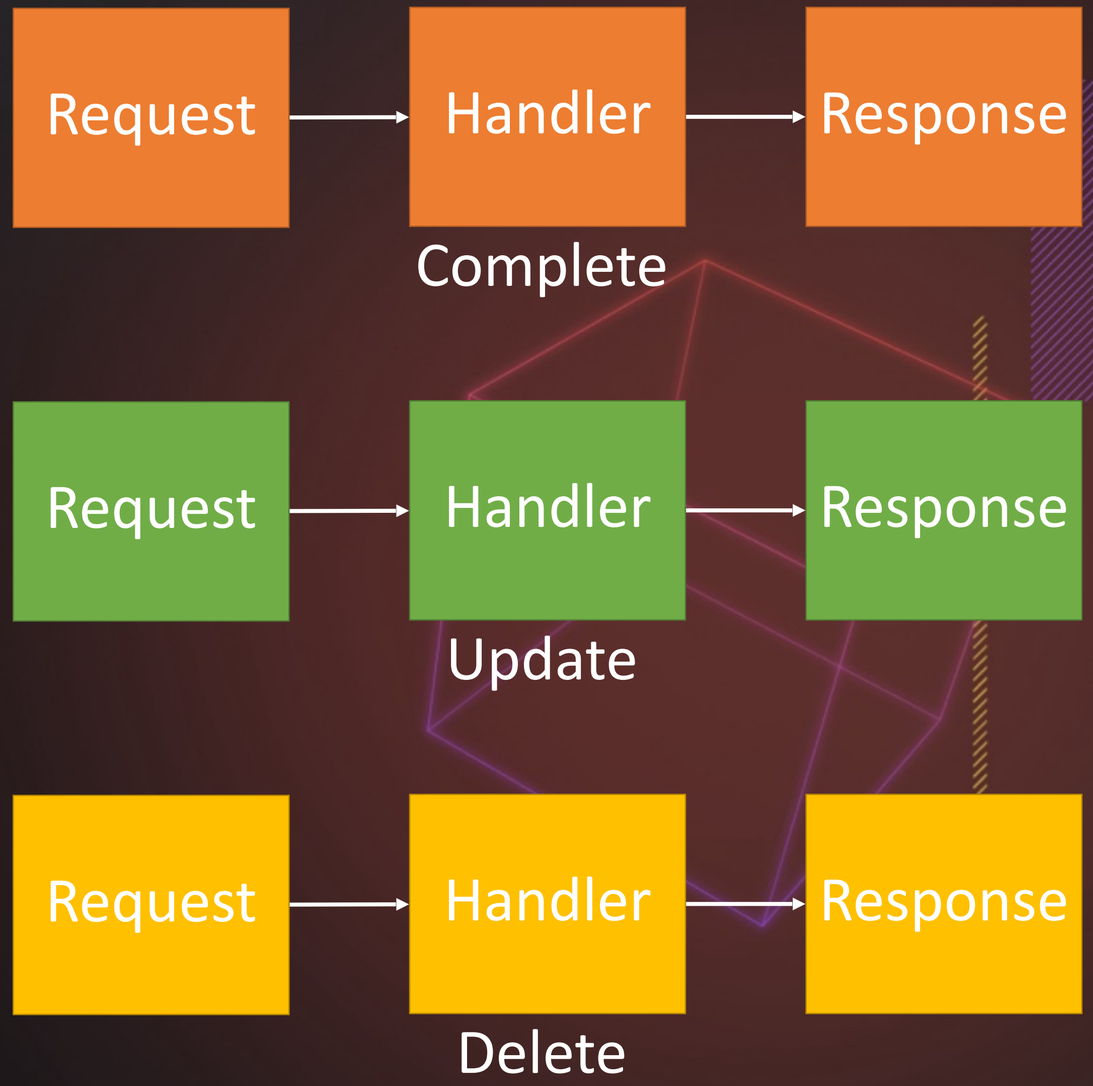
Success/Fail



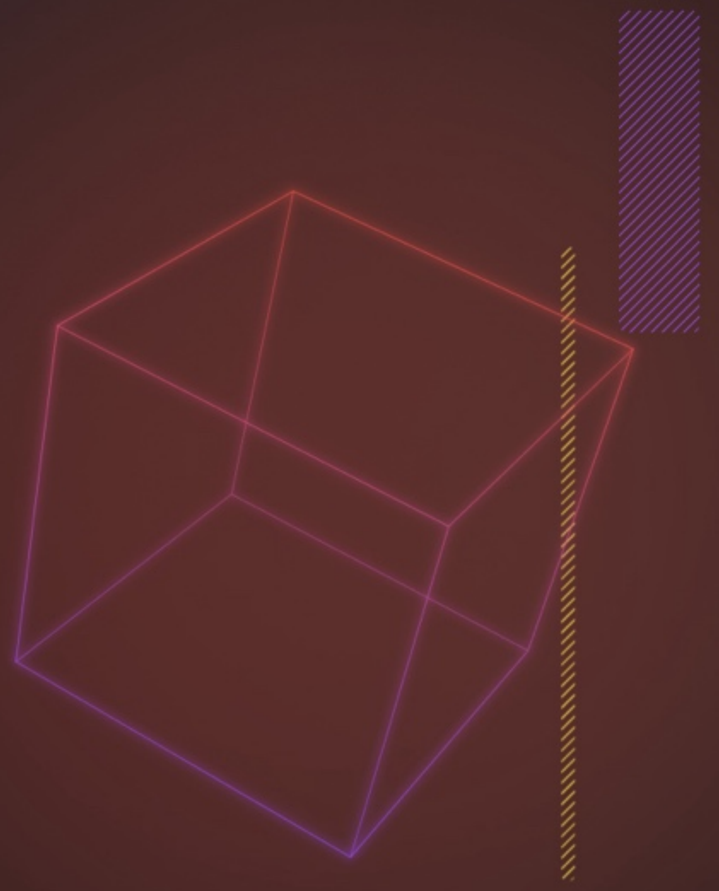
Creation



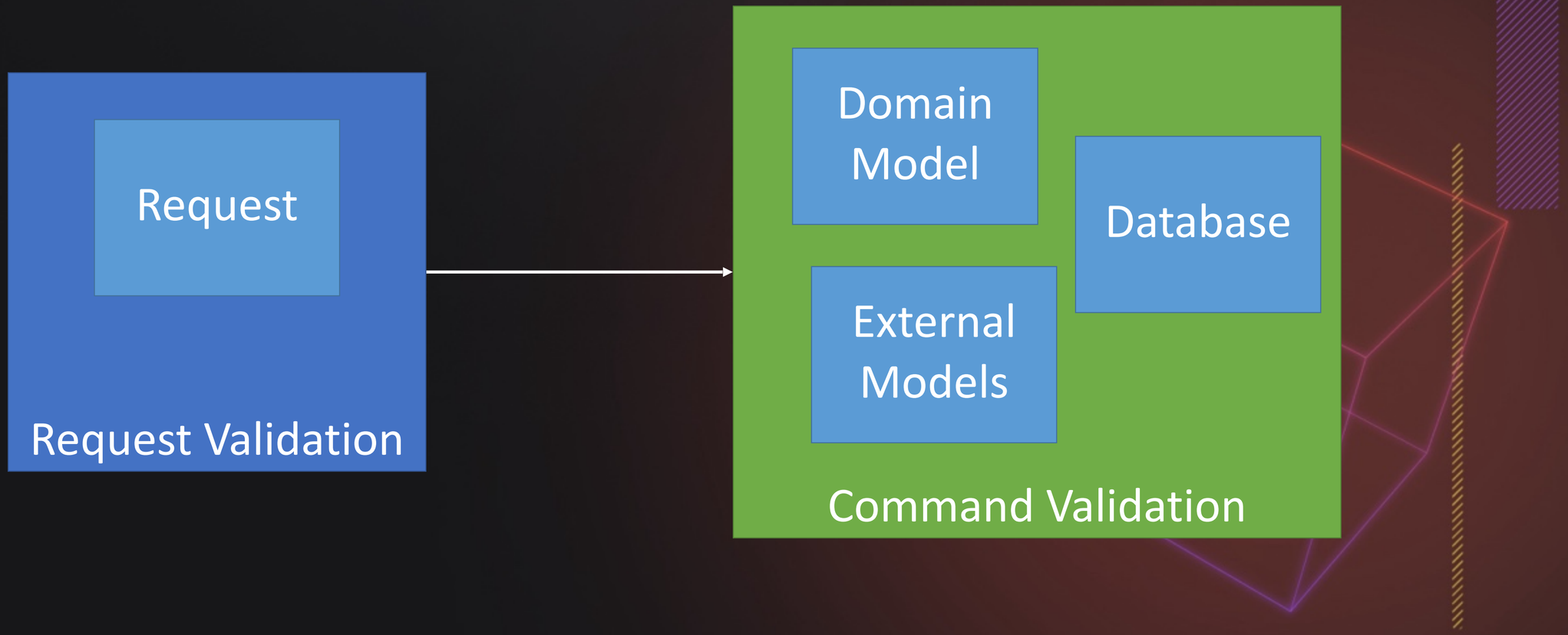
Task-Based UIs



Validation



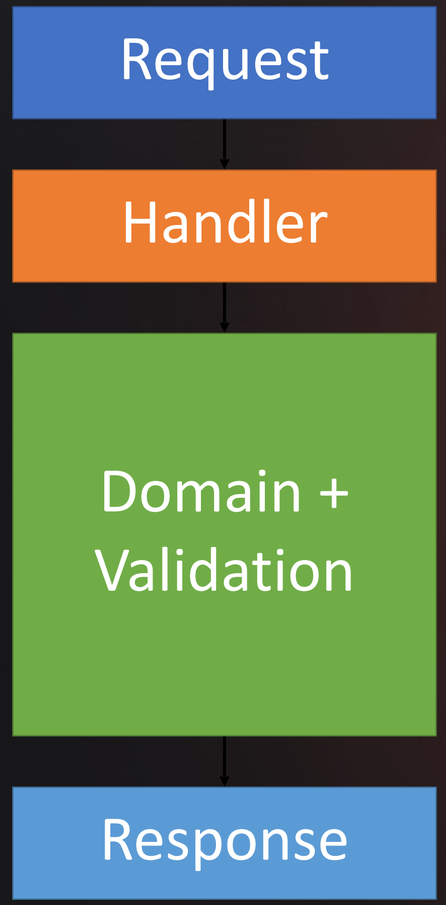
Validation Scopes

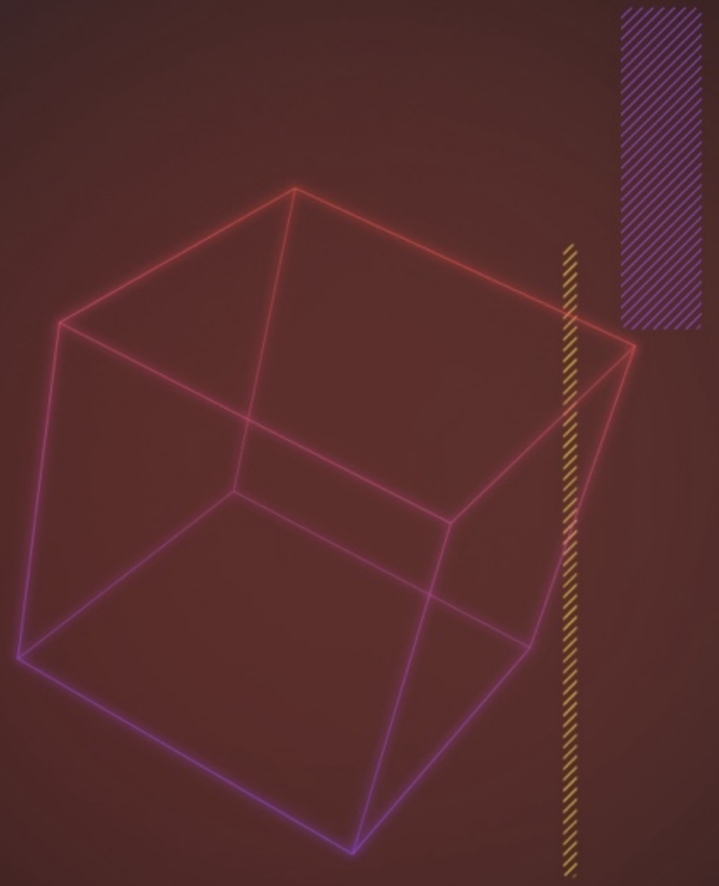


Request Validation

```
public class Validator : AbstractValidator<Command>
{
    public Validator()
    {
         RuleFor(r => r.Id).NotEmpty();
        RuleFor(r => r.Title).NotEmpty();
    }
}
```

Domain Validation



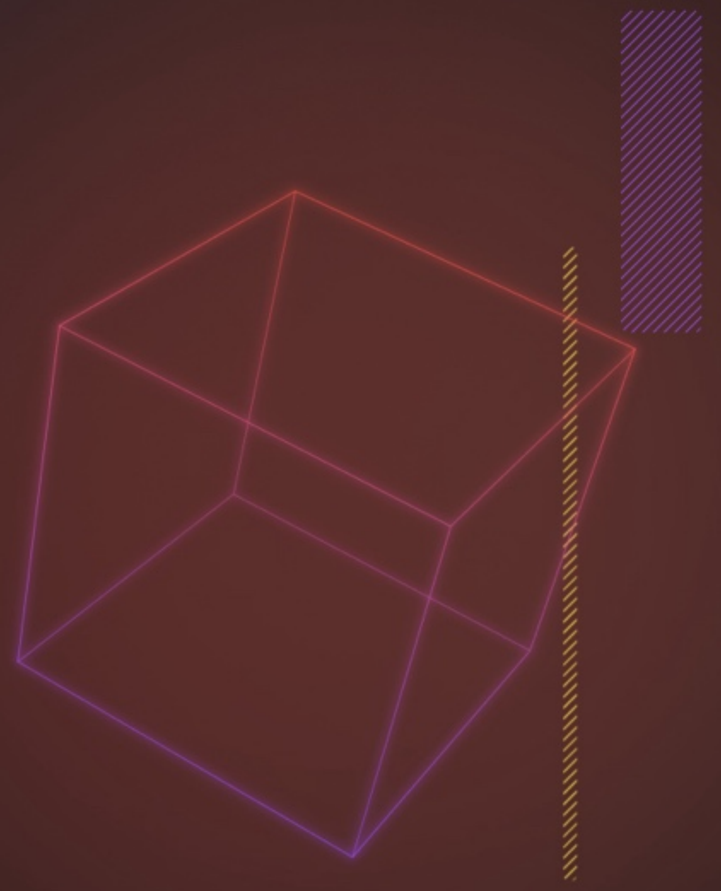


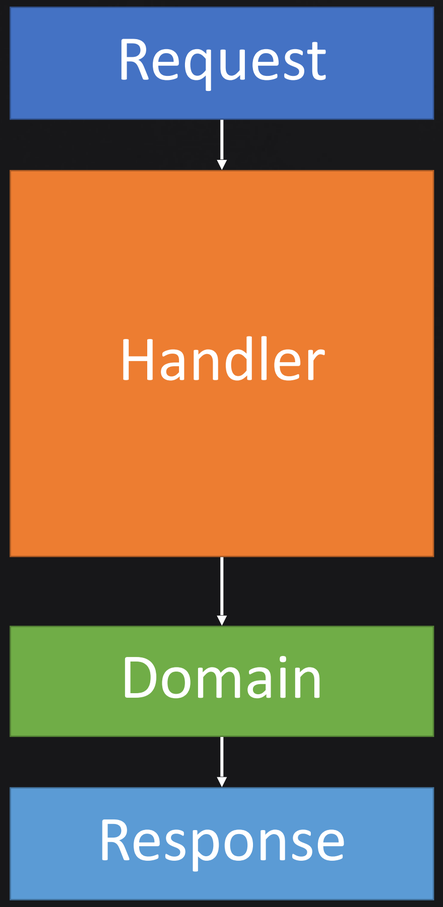
Feature Module

```
public class GetTodo : IFeatureModule
{
    public void AddRoutes(IEndpointRouteBuilder app){…}

    public record Query(string TodoId) : IQuery<Response>;

    public record Response(string Id, string Caption, bool IsCompleted);

    public class RequestHandler : IRequestHandler<Query, Response>){…}

    public class RequestValidator : AbstractValidator<Query>){…}
}
```

SHOW ME THE CODE

Complexity





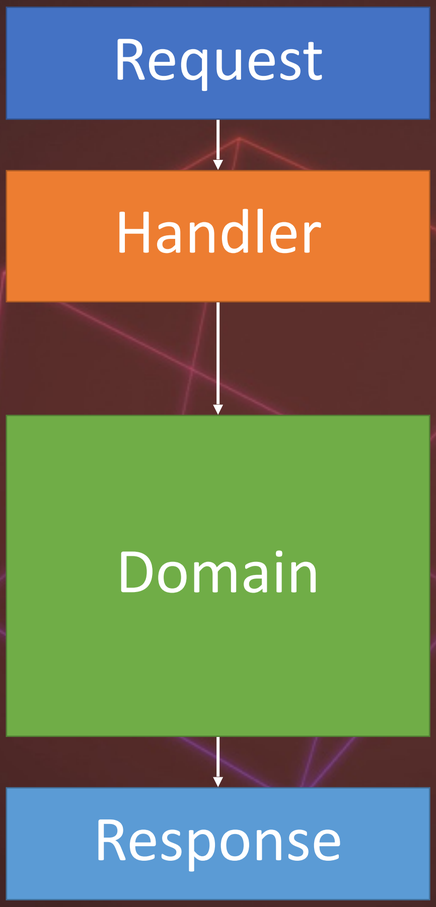
Push Behavior Down

Extracting Domain Services

Encapsulating Data

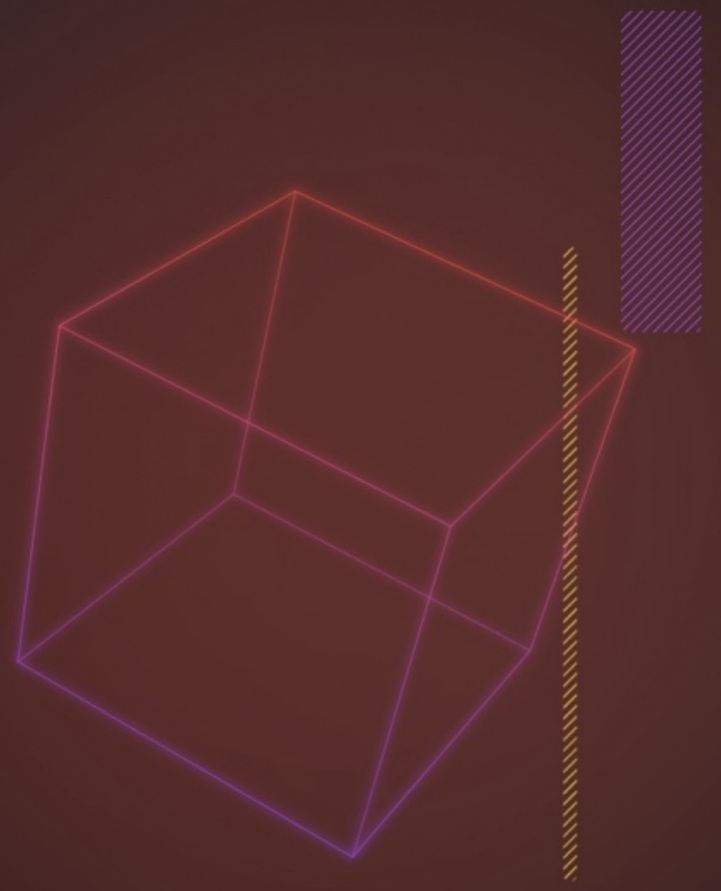
Encapsulating Collections

REFACTOR

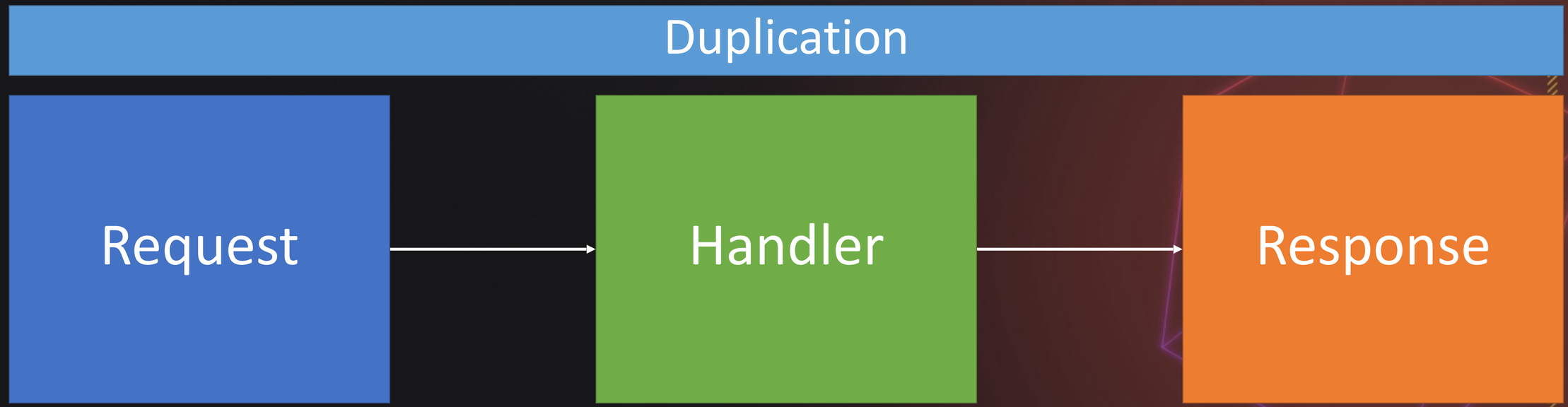


REFACTOR TIME ‘DDD’

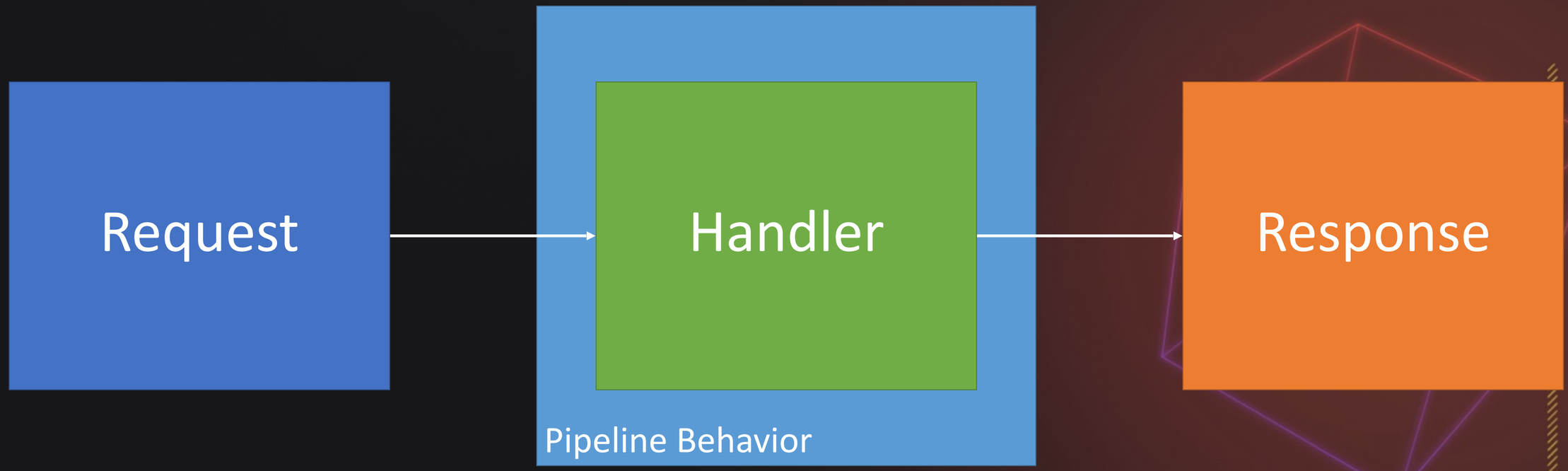
Duplication



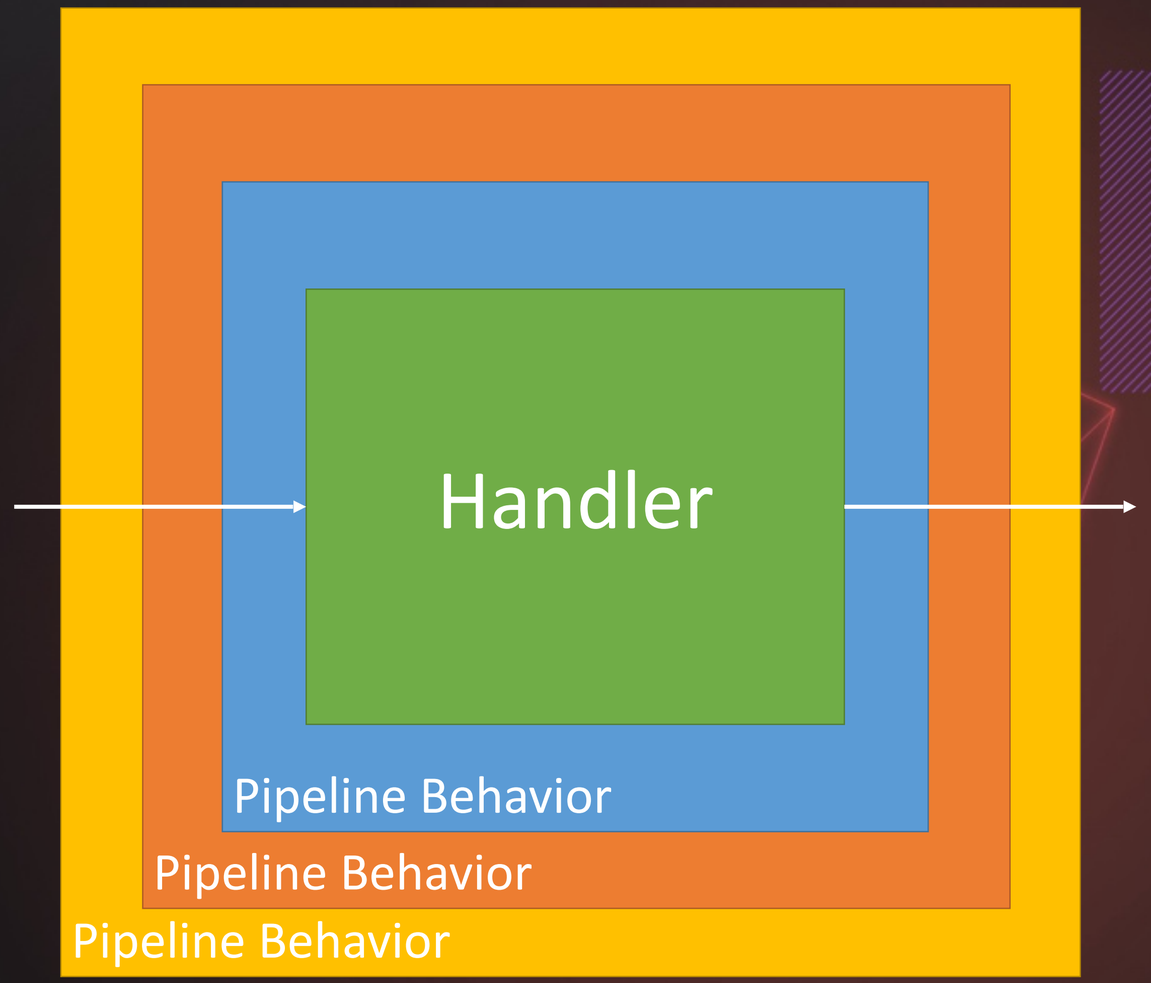
Cross-Cutting Concerns



Pipeline Behavior (Decorator)



Stackable Decorators



Logging

```
public class LoggingBehavior<TRequest, TResponse> : IPipelineBehavior<TRequest, TResponse>
    where TRequest : IRequest<TResponse>
{
    private readonly ILogger<TRequest> _logger;

    public LoggingBehavior(ILogger<TRequest> logger)
    {
        _logger = logger;
    }

    public async Task<TResponse> Handle(TRequest request, CancellationToken cancellationToken,
        RequestHandlerDelegate<TResponse> next)
    {
        _logger.LogInformation($"Handling {typeof(TRequest).Name}");
        var response = await next();
        _logger.LogInformation($"Handled {typeof(TResponse).Name}");

        return response;
    }
}
```

Transactions

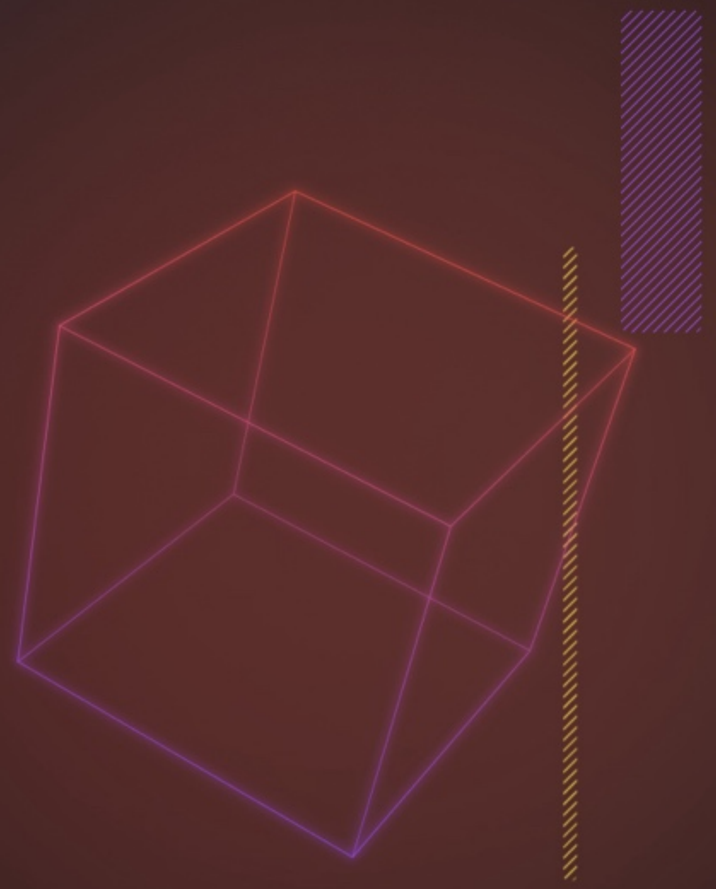
Concurrency and Retries

Time Elapsed Logging

Validation

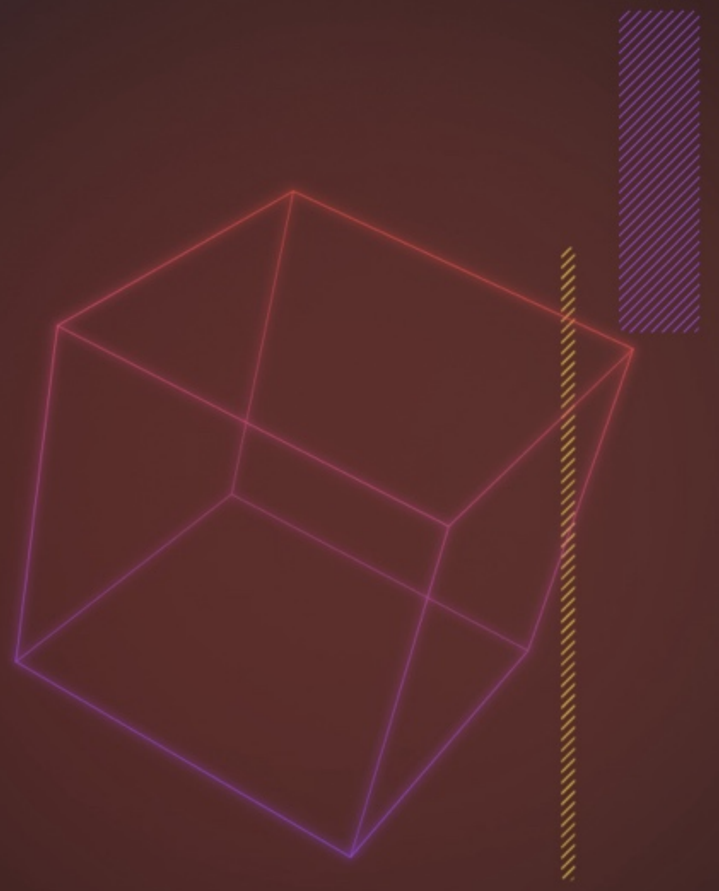
Unit of Work

....

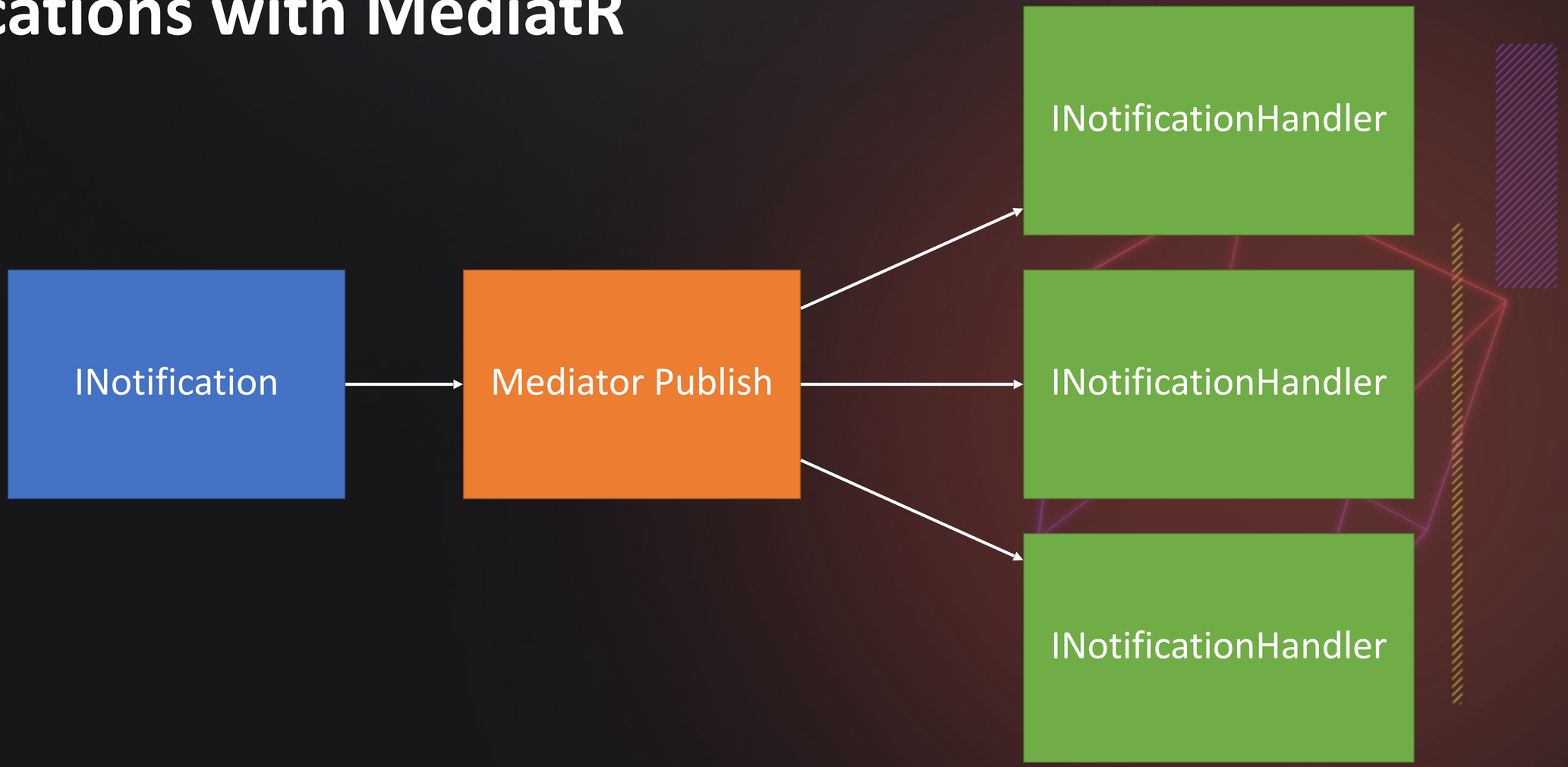


Caching

Events & Integrations



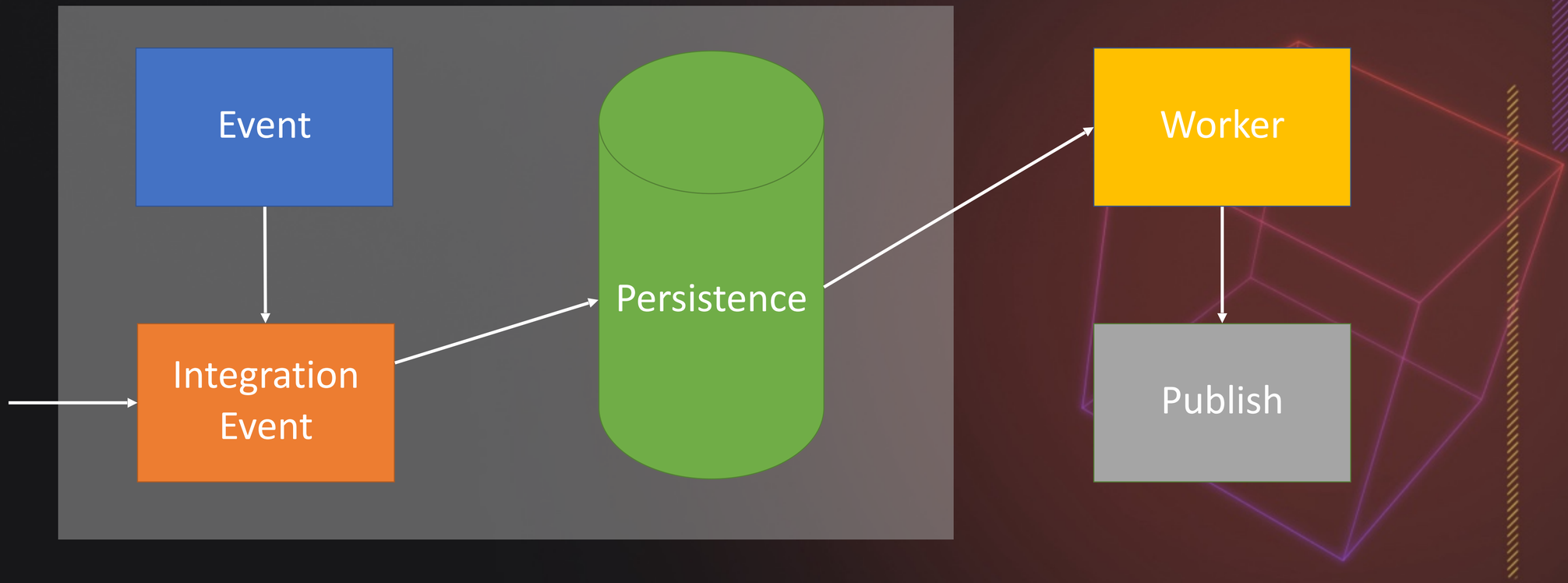
Notifications with MediatR



More decoupling

```
public class TodoCompletedEvent : INotification
{
    public Todo Todo { get; }

    public TodoCompletedEvent(Todo todo)
    {
        Todo = todo;
    }
}

await _mediator.Publish(new TodoCompletedEvent(todo));


public class TodoCompletedEventHandler : INotificationHandler<TodoCompletedEvent> {…}
```

Domain events vs integration events
Outbox pattern -> no orchestration

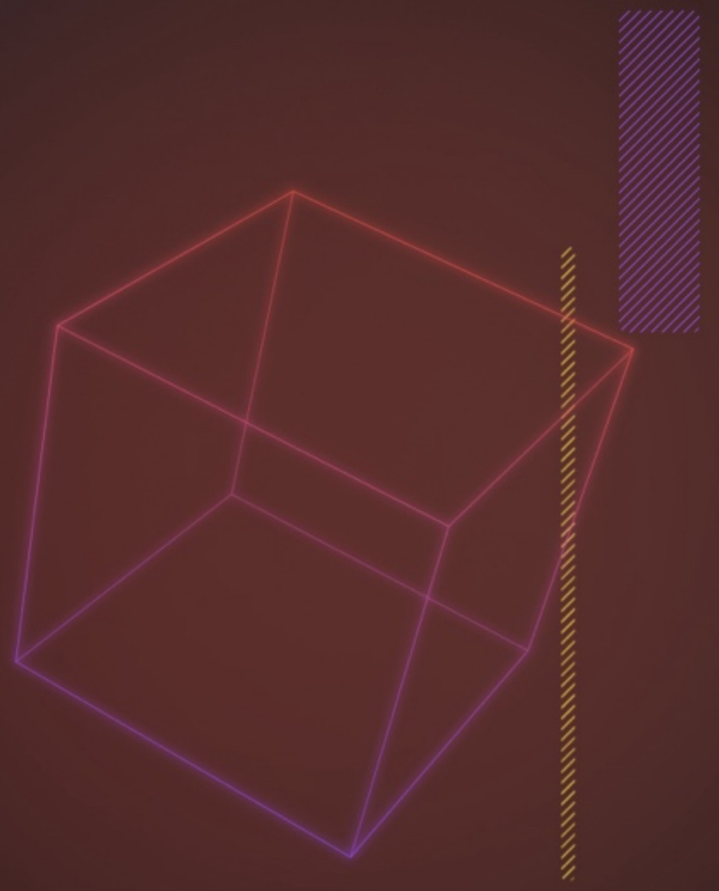


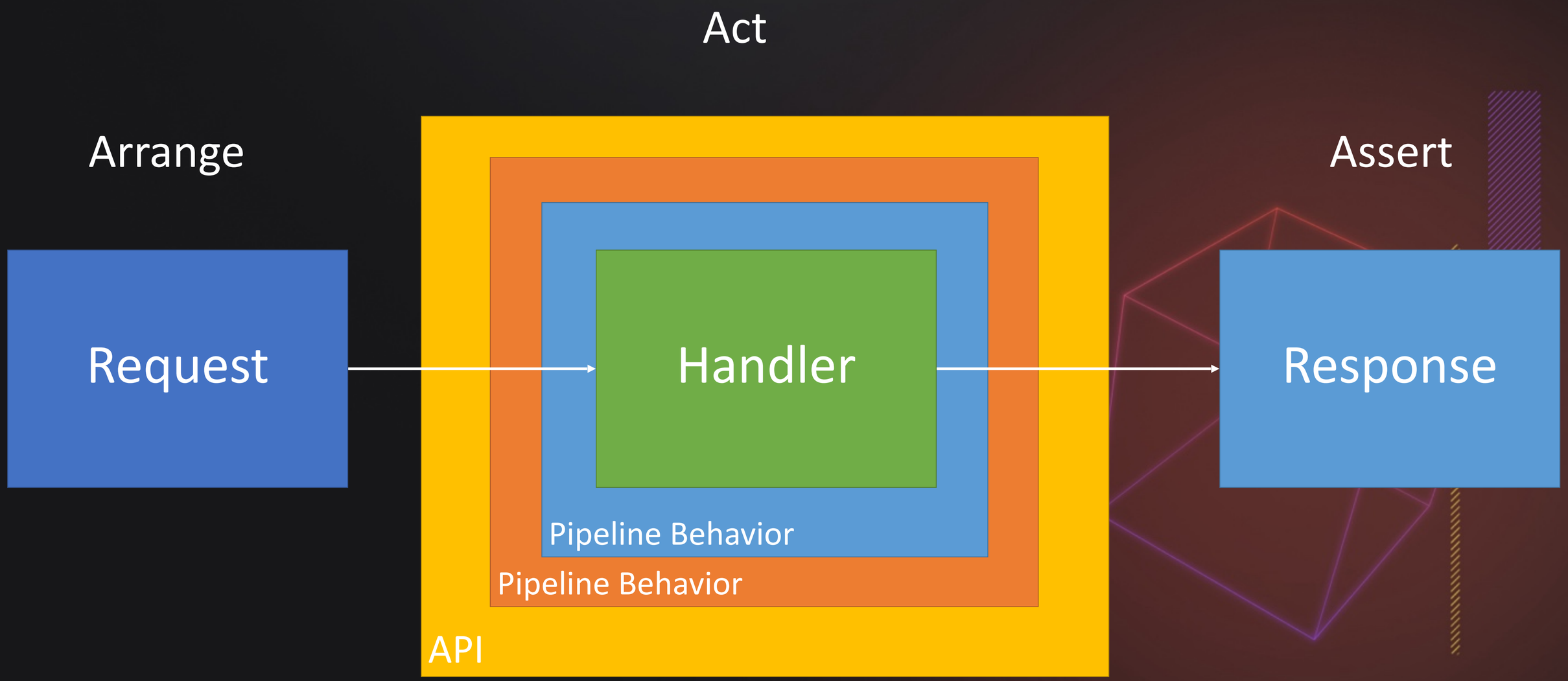
SAME TRANSACTION – SAME LIFETIME

COMMAND

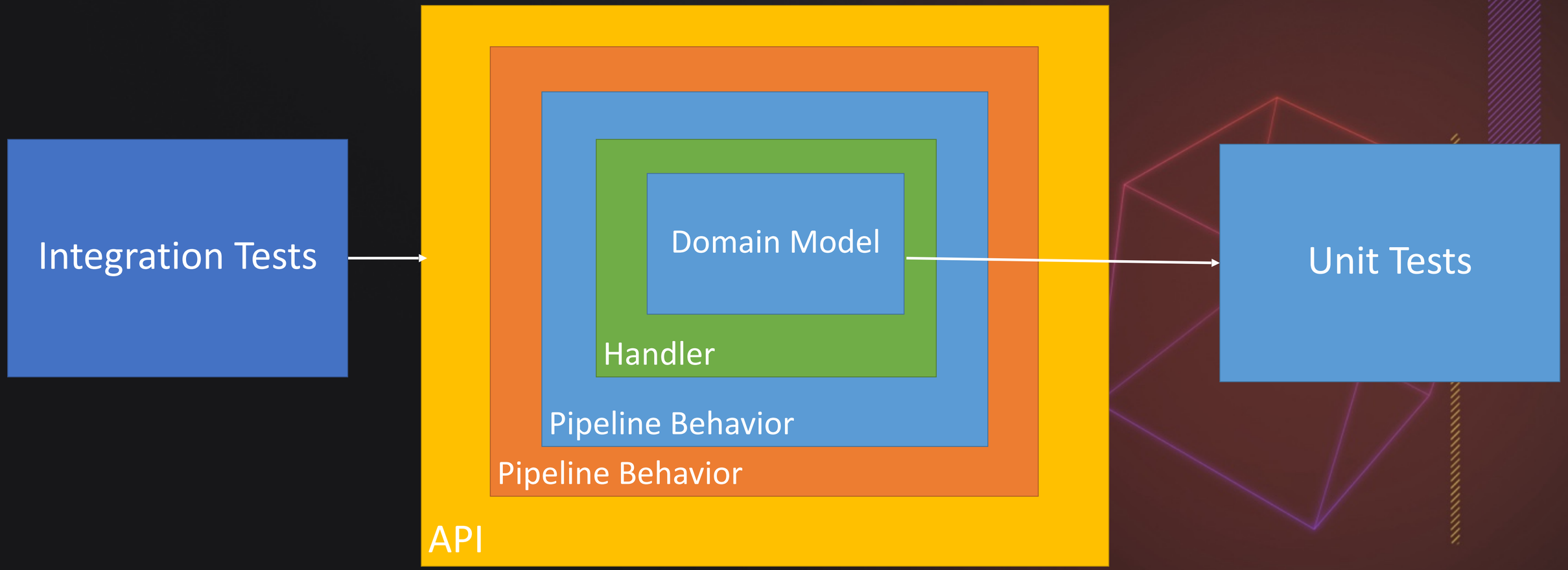
SHOW ME THE CODE

Testing




Act
Arrange
Assert
Request
Handler
Response
Pipeline Behavior
Pipeline Behavior
API

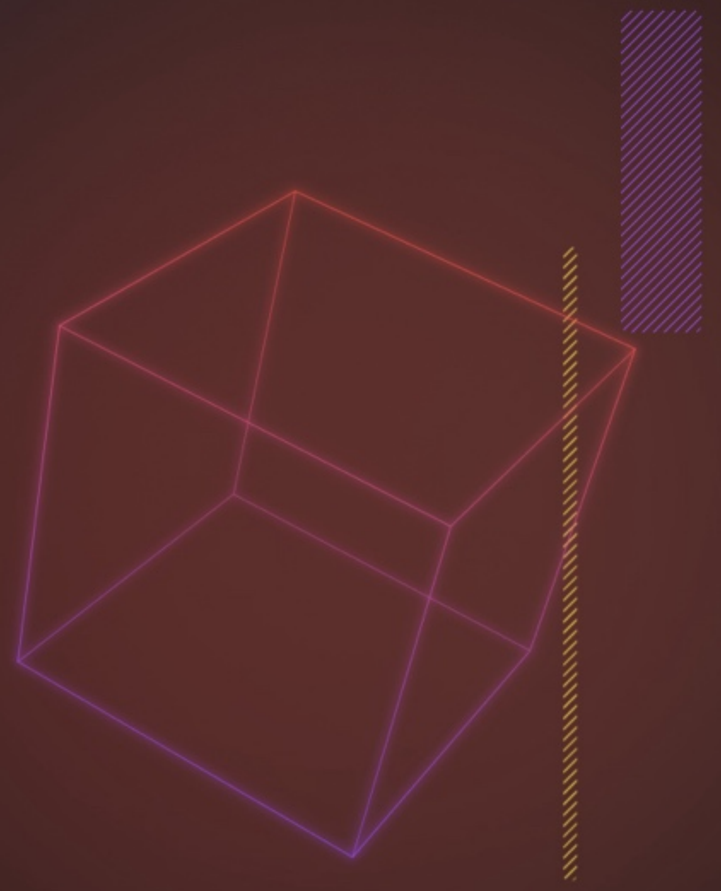
Unit and Integration Tests (Reproduce Reality)




ENOUGH BLAH BLAH
SHOW ME SOME CODE!
makeameme.org

Summary

- Vertical slices makes code easy to add/change/delete
- Easy to move to another solution/host
- No side effects
- Do not skip refactor! , push behavior down
- Integration test handlers/api, unit test domain
- Make any architecture YOURS

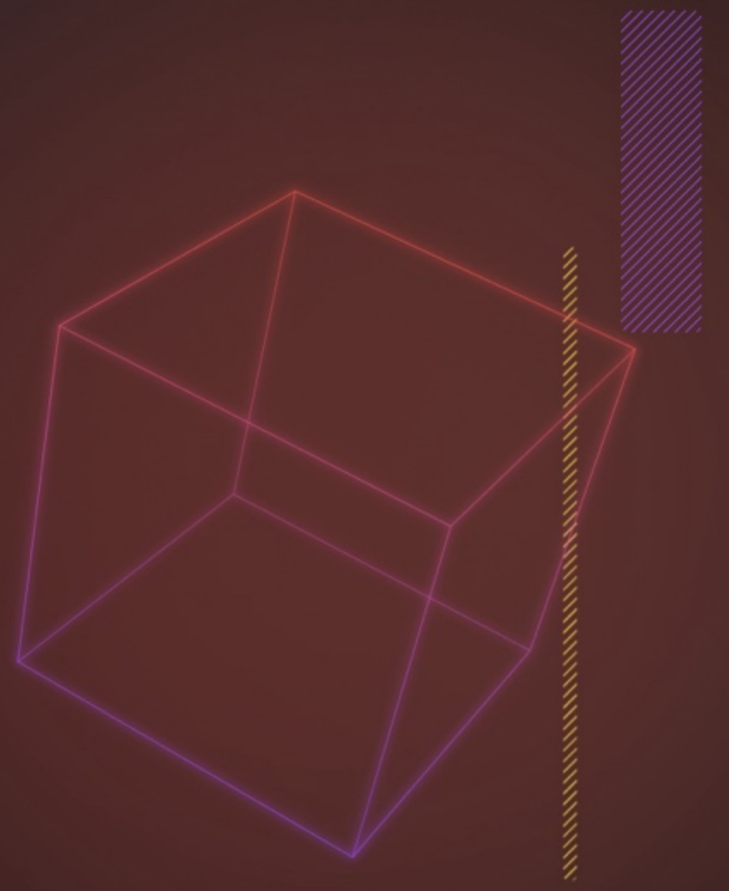


DotNet2022

TECH CONFERENCE

#DotNet2022

Questions & Answers



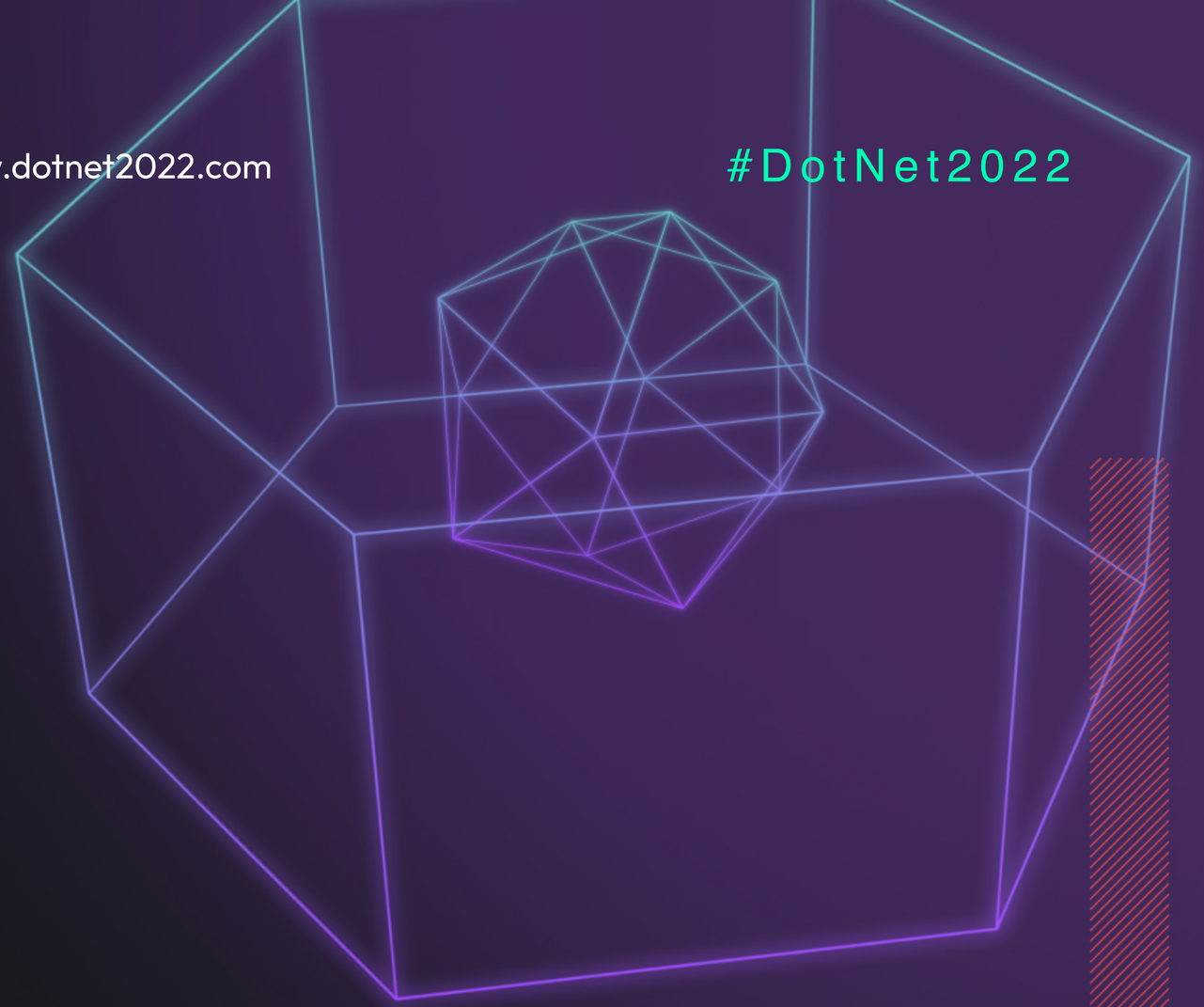

T. HANKS

DotNet 2022

TECH CONFERENCE

www.dotnet2022.com

#DotNet2022



Thanks and ...
See you soon!

Thanks also to the sponsors. Without whom this would not have been posible.

plain concepts

Microsoft

intel

LEMON CODE

intelequia

Devs DNA