



РОСАТОМ

DOTNEXT



Пешехонов Денис
Team Lead
управление Multi-D
Атомстройэкспорт



Химушкин Александр
Архитектор
управление Multi-D
Атомстройэкспорт

Укрощаем DDD на практике

Цифровизация в атомной промышленности

Что думают соискатели на
собеседовании:



В чём на самом деле наша
предметная область:



И какая проблема то?

Сложная предметная область с особыми требованиями.
Слишком разношёрстная кодовая база.

Domain Driven Design



Domain Driven Design



«Value-Object»
Annotation



Policy



Specification

«Entity»
Book

Entity



```
public class Book
{
    public Book(string title, List<Page> pages)
    {
        if (string.IsNullOrEmpty(title))
        {
            throw new Exception("Название книги не может быть пустым");
        }
        // ...
    }

    public string Title { get; }
    public Annotation Annotation { get; }

    // ...
}
```

Value Object

```
public record Annotation
{
    public string Content { get; init; }

    public bool NoAnnotation()
    {
        return string.IsNullOrEmpty(Content);
    }
}
```

Repository

```
interface IBookRepository
{
    Book FindBook(string bookTitle);

    Book DeleteBook(string bookTitle);
}
```


Policy

```
public static class BookPolicy
{
    public static void ThrowIfBookNotExistsWhenOpened(Book book)
    {
        if (book is null)
        {
            throw new Exception("Книга не найдена");
        }
    }
}
```

Specification

```
public class BookSpecifications
{
    public Expression<Func<Book, bool>> IsHorrorBook(Book book)
    {
        return b => b.Title.Contains("Франкенштейн");
    }
}
```

Проблема решена

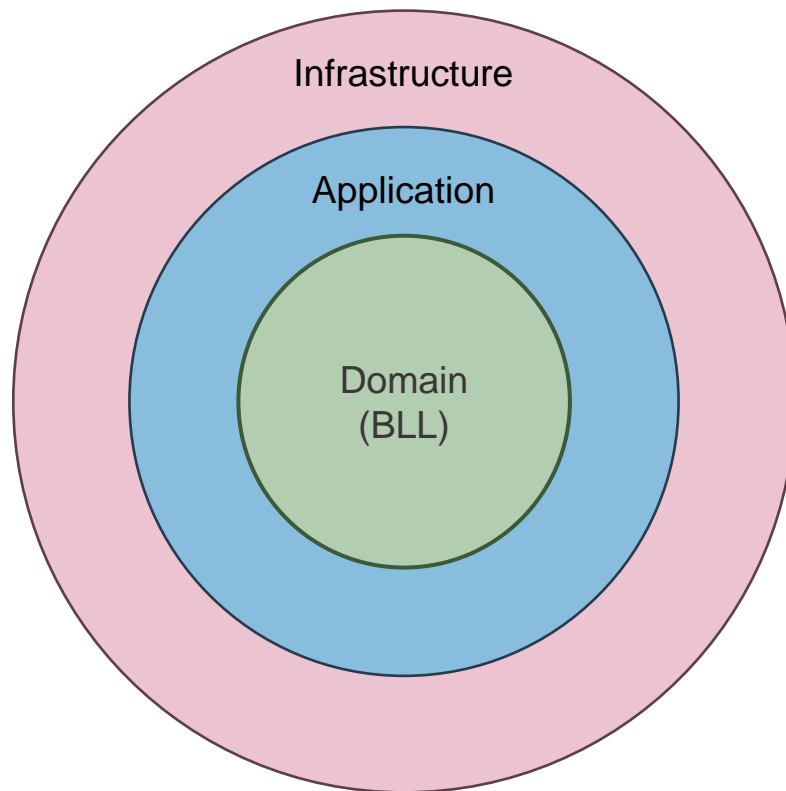
Сложная предметная область → **DDD**

Да, но...

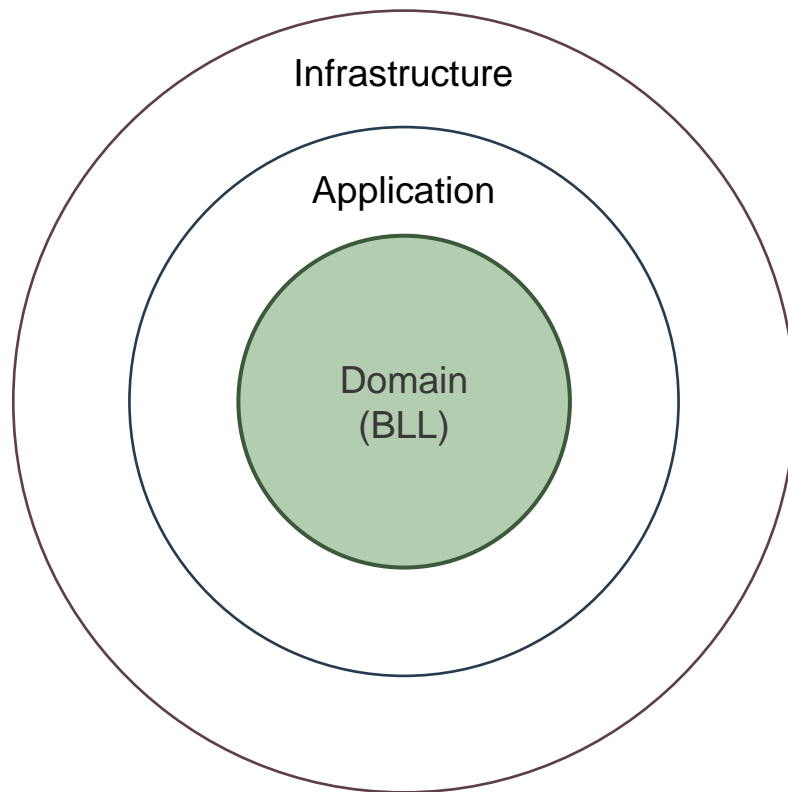
Сложная предметная область → **DDD**

Теперь у нас сложные модели,
поддерживать которые могут не только лишь все

Clean Architecture или Луковая



Clean Architecture или Луковая



BLL — защита от ошибок



```
public class Book
{
    // ...
    public void Open(int pageNumber)
    {
        if (pageNumber > this.TotalPages)
        {
            throw new Exception(/* ... */);
        }
        // ...
    }
}
```

BLL — узнавание предметной области

```
public class Book
{
    private Book(BookData data, string isbn, BookType type)
    {
        // ...
    }

    public static Book CreateSelfBook(BookData data)
    {
        return new Book(data, null, BookType.Self);
    }

    public static Book CreatePublishedBook(BookData data, string isbn)
    {
        return new Book(data, isbn, BookType.Published);
    }
}
```


BLL — узнавание предметной области

```
public class Book
{
    public static Book CreatePublishedBook(BookData data, string isbn)
    {
        return new Book(data, isbn, BookType.Published);
    }
}
```

BLL — Primitive Obsession

```
public static Book CreatePublishedBook(BookData data, string isbn)
```

```
public static Book CreatePublishedBook(BookData data, ISBN isbn)
```

```
public class ISBN
{
    private string _value;

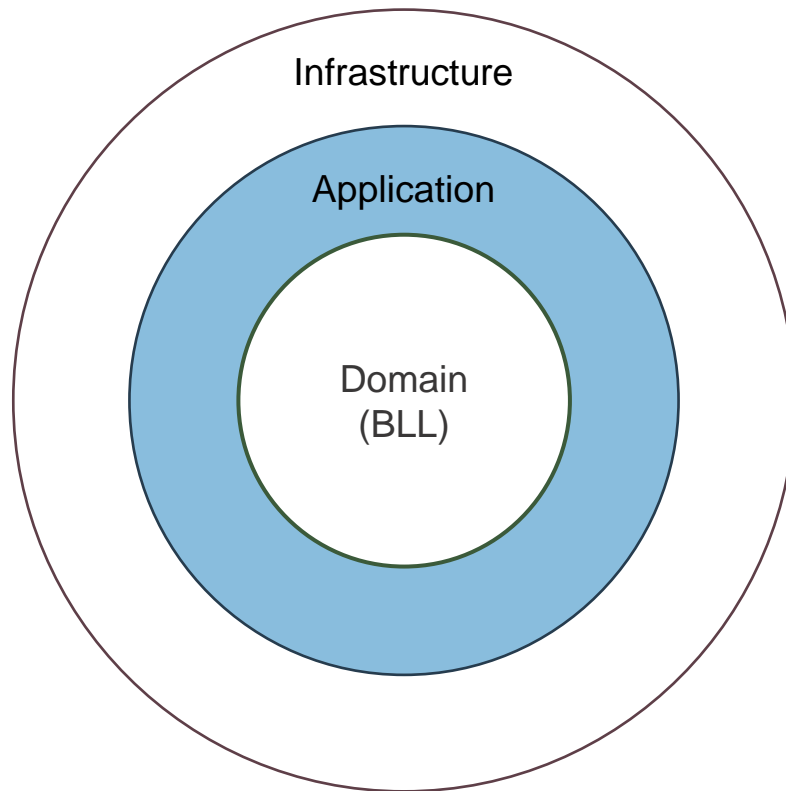
    public ISBN(string value)
    {
        IsbnValidator.ThrowIfNotValid(value);
        _value = value;
    }
}
```

BLL — records

```
public class ISBN
{
    public static bool operator ==(ISBN one, ISBN two)
    {
        return one._value == two._value;
    }
}
```

```
public record ISBN
{
    // ...
}
```

Clean Architecture или Луковая



Domain vs Application

Domain

Что *МОЖНО* сделать

```
public class Book
{
    // ...
    public void Open(int pageNumber)
    public void TearPageOut(int pageNumber)
    // ...
}
```

Application

Что *будем* делать

```
public class ReadBookApplicationService
{
    // ...
    public void Handle(int bookId, string userId)
    {
        var user = _userRepository.Get(userId);
        int page = user.PageStoppedOn;

        var book = _bookRepository.Get(bookId);
        book.Open(page);

        user.ReadPage(book.CurrentPageText);

        book.Close();
    }
}
```

Проблема решена

Сложная предметная область → **DDD**

Сложные модели → **Clean Architecture**

Да, но...

Сложная предметная область → **DDD**

Сложные модели → **Clean Architecture**

Бизнес-сценарии постоянно меняются и обрастают требованиями, вся разработка идёт в Application.



Decorator

```
public interface IReadBookApplicationService
{
    void Handle(int bookId, string userId);
}

public class ReadBookAgeDecorator : IReadBookApplicationService
{
    public ReadBookAgeDecorator(IReadBookApplicationService service) { /* ... */}

    public void Handle(int bookId, string userId)
    {
        var user = _userRepository.Get(userId);
        if (user.Age < 12)
        {
            throw new Exception("Нельзя читать книги лицам до 12 лет");
        }

        _service.Handle(bookId, userId);
    }
}
```

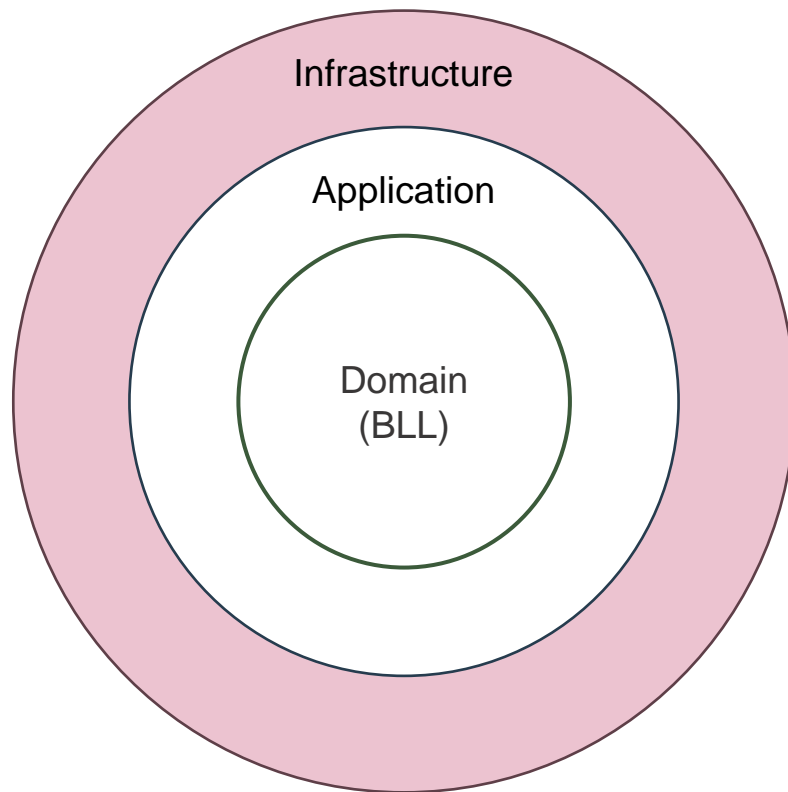

Decorator — подключение



Библиотека Scrutor

```
services
    .AddScoped<IReadBookApplicationService, ReadBookApplicationService>()
    .Decorate<IReadBookApplicationService, ReadBookAgeDecorator>();
```

Clean Architecture или Луковая



Да, но...

Сложная предметная область → **DDD**

Сложные модели → **Clean Architecture**

Толстый Application → **Decorators**

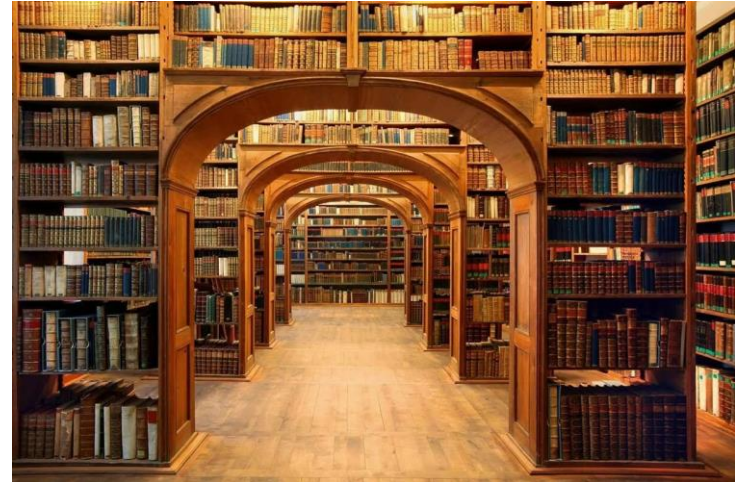
Application-слой не зависит от инфраструктуры,
но должен как-то ей пользоваться.

Application <- Infrastructure

Книжный магазин



Библиотека



```
public class Book
{
    // ...
}
```

```
public class Book
{
    // ...
}
```

Application <- Infrastructure

Книжный магазин

```
public interface IBookStorePaymentProvider
{
    PaymentResult TryWithdraw(
        UserId userId,
        decimal price
    );
}
```

Библиотека

```
public interface ILibraryRegister
{
    bool CheckBookAvailability(
        BookId bookId,
        DateTime fromDate,
        DateTime toDate
    );
}
```

Application <- Infrastructure



Книжный магазин

Библиотека

Application

```
public interface IBookStorePaymentProvider
{
    PaymentResult TryWithdraw(
        UserId userId,
        decimal price
    );
}
```

```
public interface ILibraryRegistry
{
    bool CheckBookAvailability(
        BookId bookId,
        DateTime fromDate,
        DateTime toDate
    );
}
```



Infrastructure

```
public class SomeBankPaymentProvider :
    IBookStorePaymentProvider
{
    // ...
}
```

```
public class PostgreSQLLibraryRegistry :
    ILibraryRegistry
{
    // ...
}
```

Проблема решена

Сложная предметная область → **DDD**

Сложные модели → **Clean Architecture**

Толстый Application → **Decorators**

Application не зависит от инфраструктуры → **Inversion of Control**

Связи между сущностями

```
public class Book
{
    // ...

    public List<User> Users { get; } // ???
}
```


Связи между сущностями

```
public class Book  
{  
    // ...  
  
    public List<User> Users { get; } // ????  
}
```

```
public class BookCard // ???  
{  
    // ...  
  
    public Book Book { get; }  
    public List<User> Users { get; }  
}
```



Связи между сущностями

```
public class Book  
{  
    // ...  
  
    public List<User> Users { get; } // ????  
}
```

```
public class BookCard // ????  
{  
    // ...  
  
    public Book Book { get; }  
    public List<User> Users { get; }  
}
```

```
const book = await fetch('/GetBookById', /*...*/);  
const users = await fetch('/GetUsersByBook', /*...*/);
```

Да, но...

Сложная предметная область → **DDD**

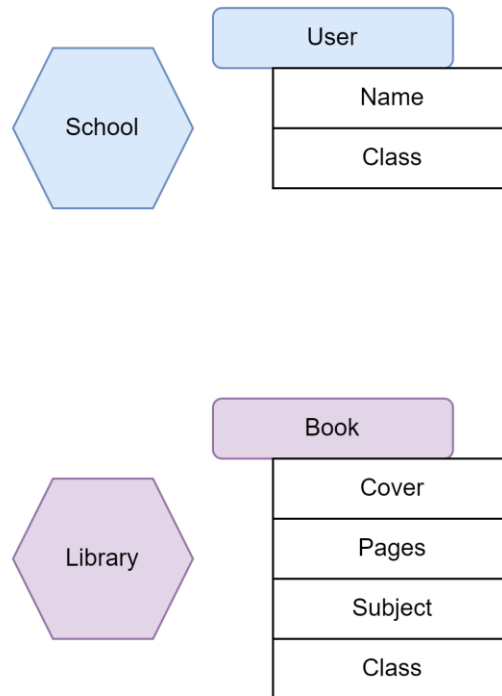
Сложные модели → **Clean Architecture**

Толстый Application → **Decorators**

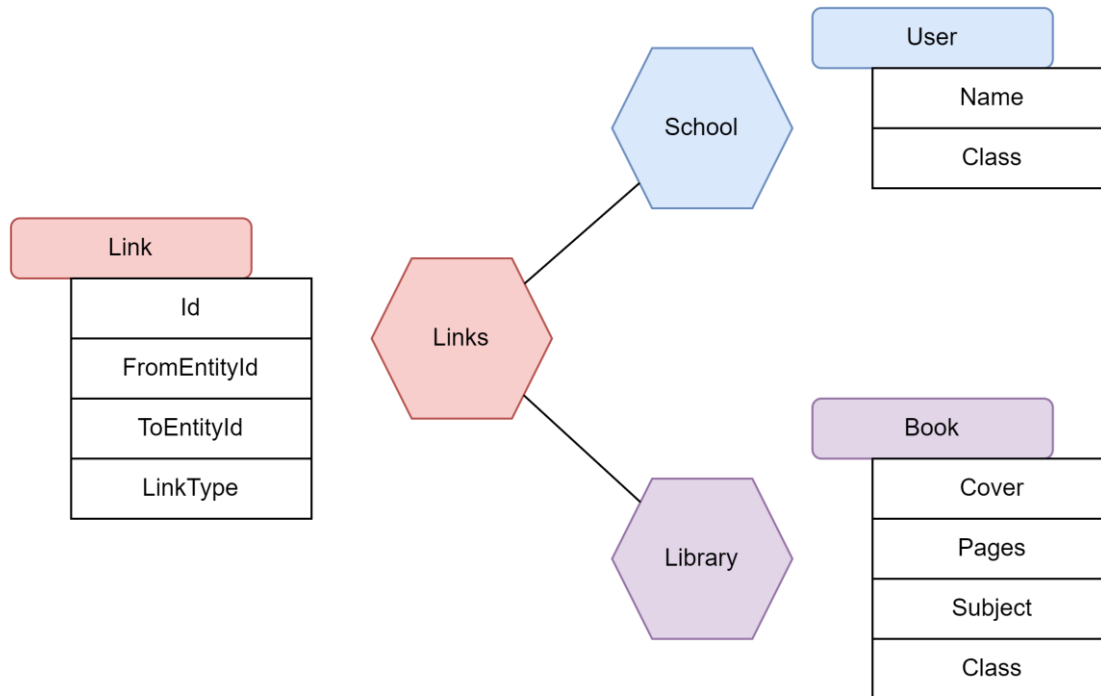
Application не зависит от инфраструктуры → **Inversion of Control**

Как организовать связи между сущностями?

Связь как сущность



Связь как сущность



Проблема решена

Сложная предметная область → **DDD**

Сложные модели → **Clean Architecture**

Толстый Application → **Decorators**

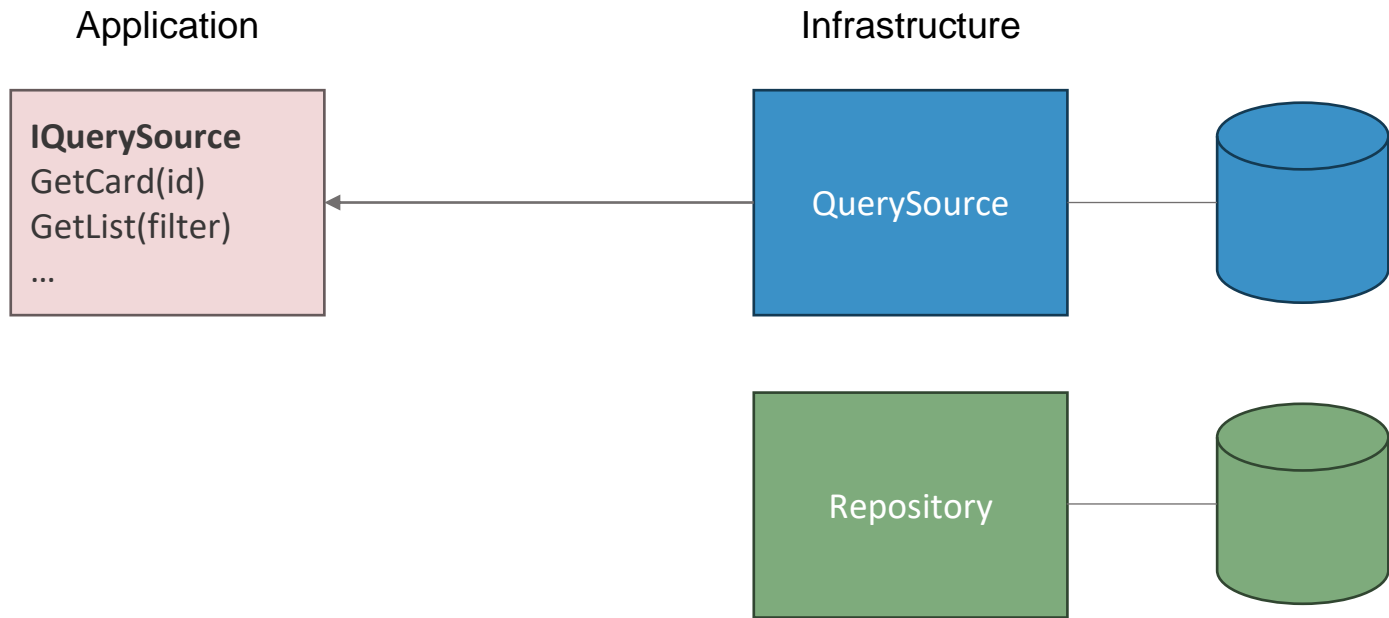
Application не зависит от инфраструктуры → **Inversion of Control**

Связи между сущностями → **Link entity**

Мальчик: не встречается с девушками, у которых кто-то был

Мужчина: не читает из таблицы, в которую записывают данные

CQRS — QuerySource

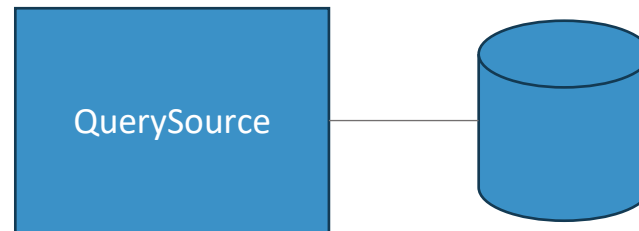




CQRS — QuerySource

```
public class BookViewModel
{
    public string Title { get; set; }
    public AuthorMetaViewModel Author { get; set; }
    public List<UserMetaViewModel> Users { get; set; }
}
```

```
public class UserMetaViewModel
{
    public string Id { get; set; }
    public string FullName { get; set; }
}
```



Проблема решена

Сложная предметная область → **DDD**

Сложные модели → **Clean Architecture**

Толстый Application → **Decorators**

Application не зависит от инфраструктуры → **Inversion of Control**

Связи между сущностями и агрегация → **Link entity + CQRS**

Да, но...

Сложная предметная область → **DDD**

Сложные модели → **Clean Architecture**

Толстый Application → **Decorators**

Application не зависит от инфраструктуры → **Inversion of Control**

Связи между сущностями и агрегация → **Link entity + CQRS**

Как наполнить данными QuerySource?

Domain Events

```
public sealed record BookCreatedDomainEvent : IDomainEvent
{
    public string Title { get; set; }
    public List<Page> Pages { get; set; }

    // ...
}
```

```
public sealed record BookOpenedDomainEvent : IDomainEvent
{
    public string Title { get; set; }
    public int CurrentPage { get; set; }

    // ...
}
```



Domain Events

```
public sealed record BookCreatedDomainEvent : IDomainEvent
{
    public string Title { get; set; }
    public List<Page> Pages { get; set; }

    // ...
}
```

```
public sealed record BookOpenedDomainEvent : IDomainEvent
{
    public string Title { get; set; }
    public int CurrentPage { get; set; }

    // ...
}
```

```
public abstract class Entity
{
    public List<IDomainEvent> DomainEvents = new();
}
```

Domain Events — Visitor

```
public interface IDomainEvent
{
    public long OccuredOn { get; }

    public void Accept(IDomainEventVisitor visitor);
}
```

```
public interface IDomainEventVisitor
{
}
```

```
public interface IDomainEventVisitor<in T> : IDomainEventVisitor where T : IDomainEvent
{
    public void Visit(T domainEvent);
}
```

Domain Events — Visitor

```
public sealed record BookOpenedDomainEvent : IDomainEvent
{
    public void Accept(IDomainEventVisitor visitor)
    {
        (visitor as IDomainEventVisitor<BookOpenedDomainEvent>)?.Visit(this);
    }
}

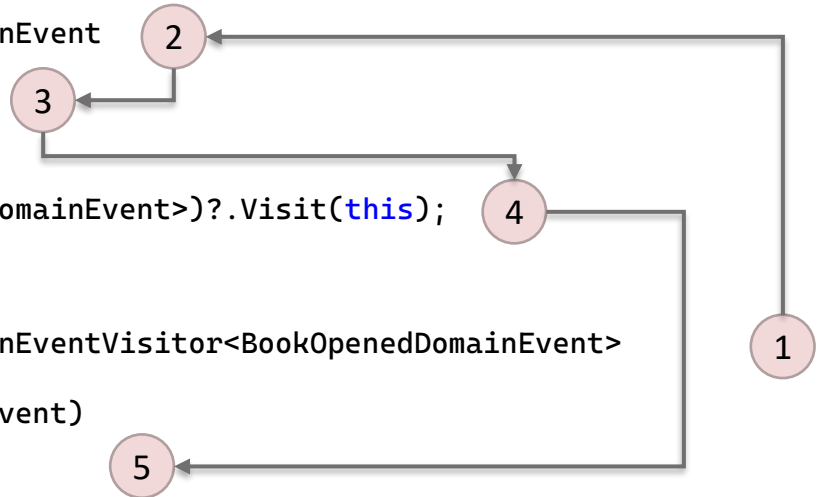
public sealed class BookDomainEventVisitor : IDomainEventVisitor<BookOpenedDomainEvent>
{
    public void Visit(BookOpenedDomainEvent domainEvent)
    {
        ...
    }
}
```



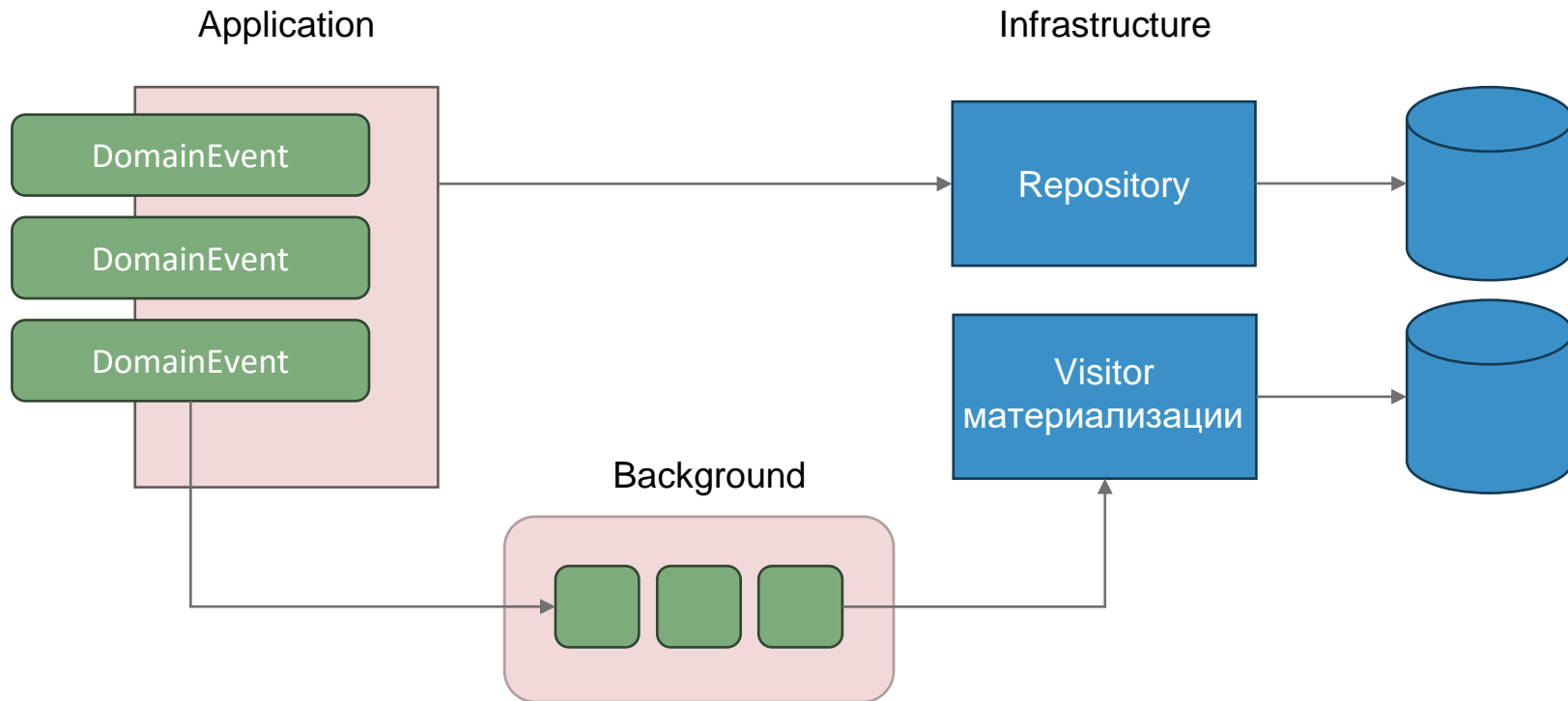
Domain Events — Visitor

```
public sealed record BookOpenedDomainEvent : IDomainEvent
{
    public void Accept(IDomainEventVisitor visitor)
    {
        (visitor as IDomainEventVisitor<BookOpenedDomainEvent>)?.Visit(this);
    }
}

public sealed class BookDomainEventVisitor : IDomainEventVisitor<BookOpenedDomainEvent>
{
    public void Visit(BookOpenedDomainEvent domainEvent)
    {
        ...
    }
}
```



CQRS — агрегация



Domain Events — Visitor



Dmitri Nesteruk

@DmitriNesteruk

A random assortment of screencasts related to software development (C#... >



Проблема решена

Сложная предметная область → **DDD**

Сложные модели → **Clean Architecture**

Толстый Application → **Decorators**

Application не зависит от инфраструктуры → **Inversion of Control**

Связи между сущностями и агрегация → **Link entity + CQRS**

Наполнить данными QuerySource → **Domain Events + Visitor**

Да, но...

Сложная предметная область → **DDD**

Сложные модели → **Clean Architecture**

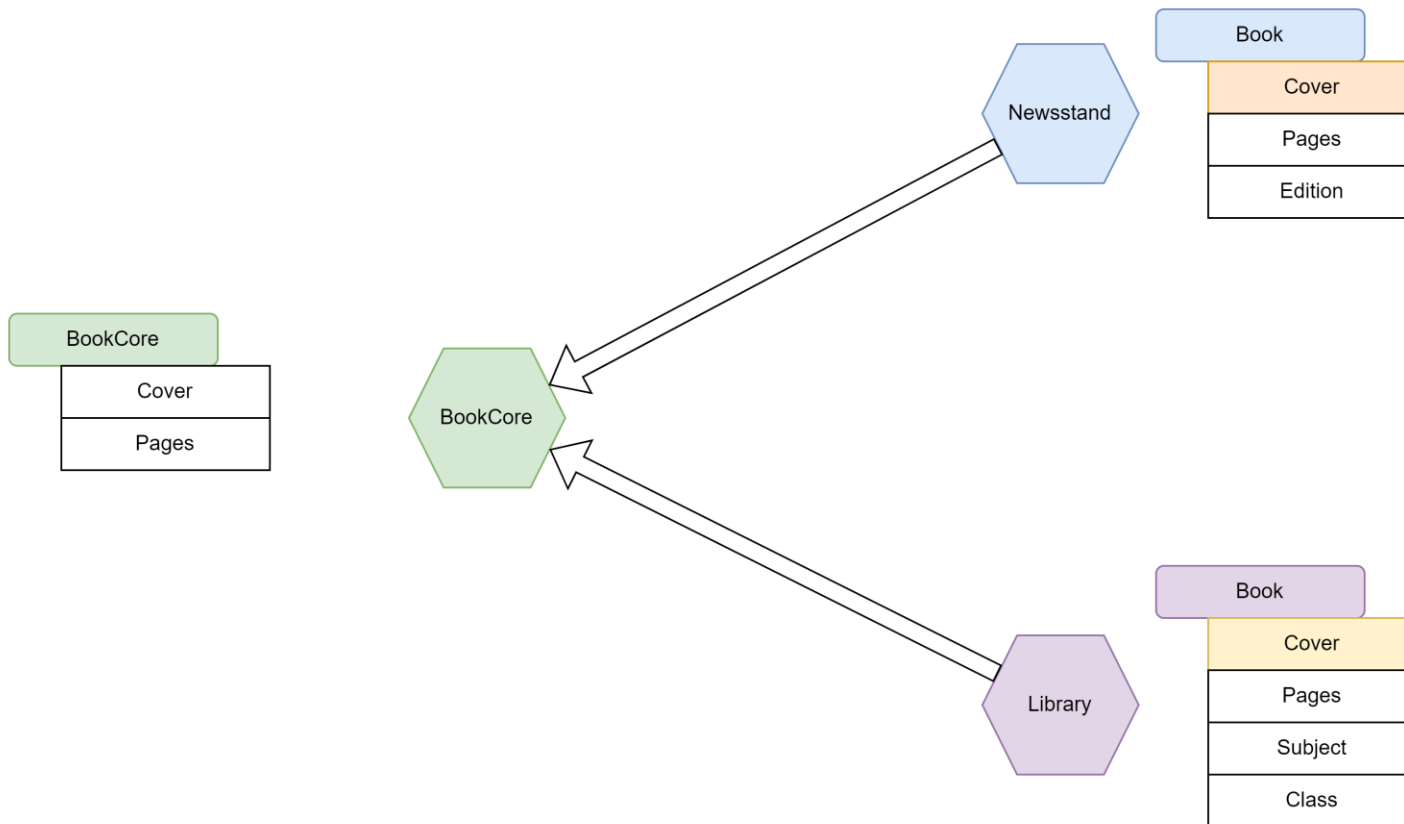
Толстый Application → **Decorators**

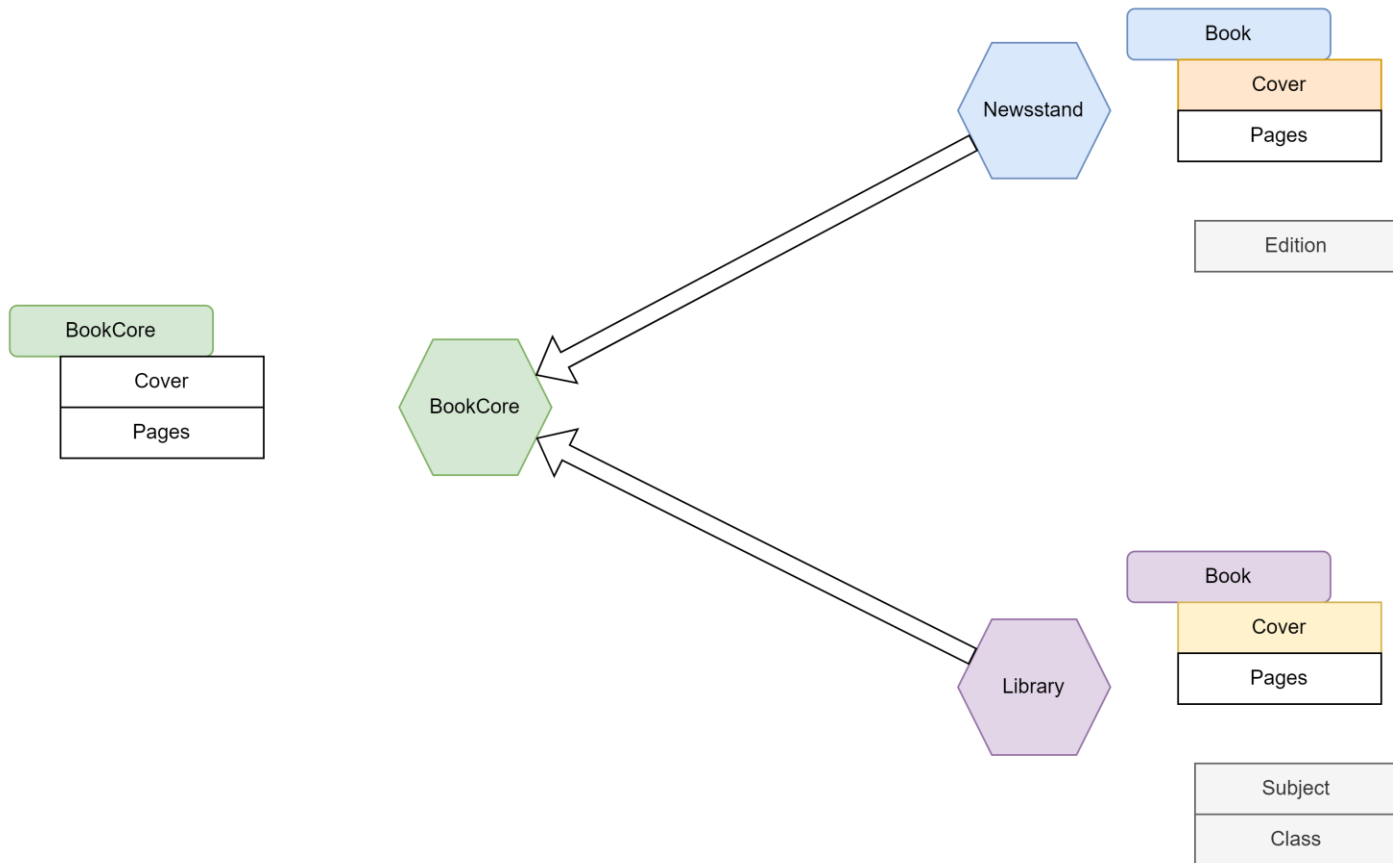
Application не зависит от инфраструктуры → **Inversion of Control**

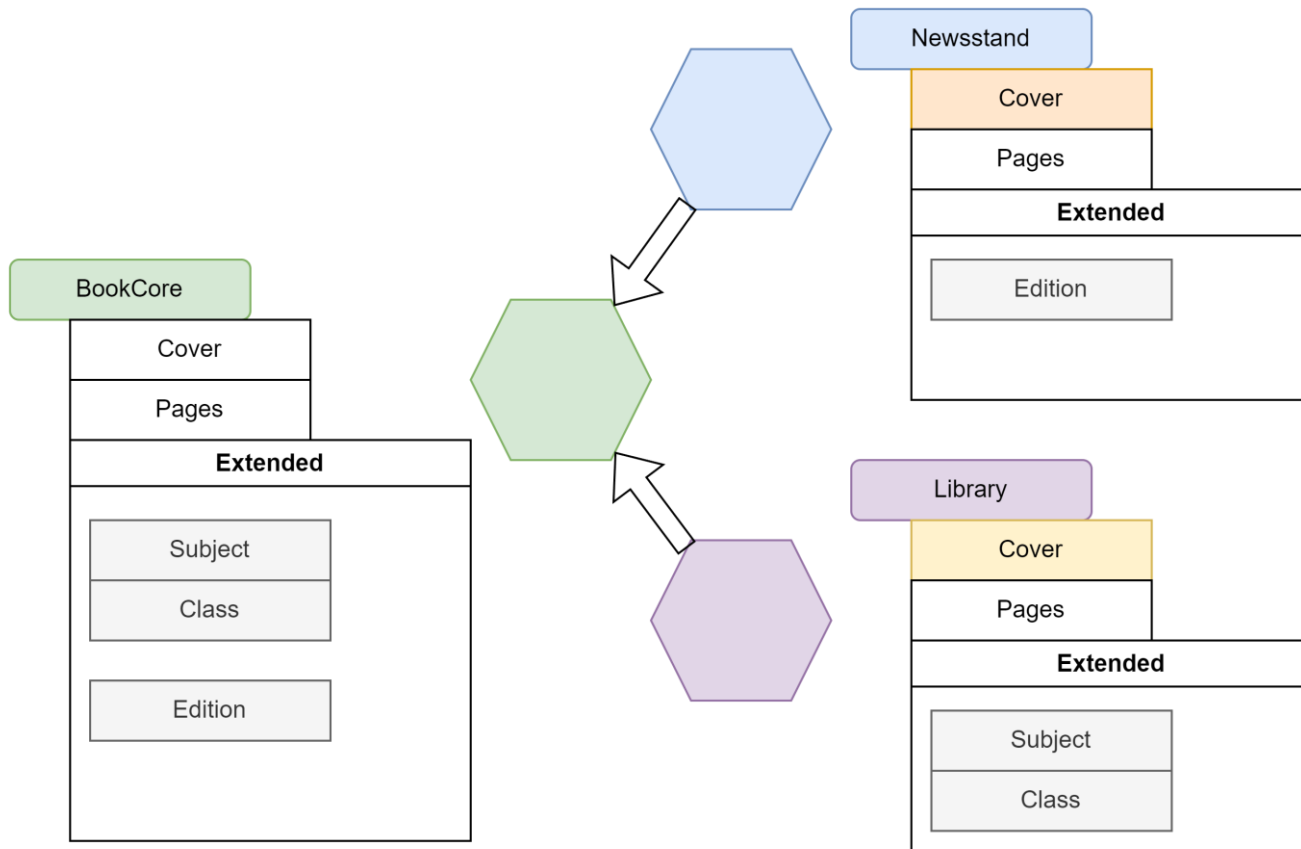
Связи между сущностями и агрегация → **Link entity + CQRS**

Наполнить данными QuerySource → **Domain Events + Visitor**

Систему нужно поставлять разным заказчикам
с немного разными сетапами







```
public class Book
{
    public string Title { get; }
    public List<Page> Pages { get; }

    // ...

    public Dictionary<string, ExtendedAttribute> Extended { get; } = new();
}
```

```
public record ExtendedAttribute
{
    public string Name { get; }
    public string Value { get; }
}
```


Всё решено (?)

Сложная предметная область → **DDD**

Сложные модели → **Clean Architecture**

Толстый Application → **Decorators**

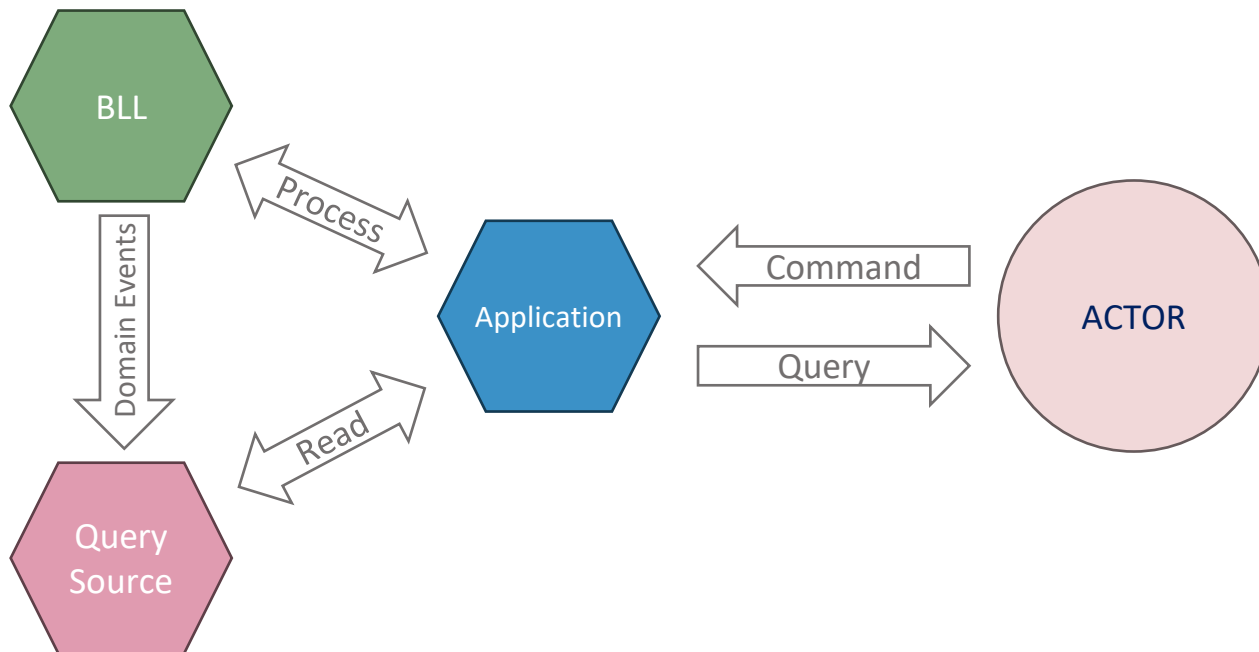
Application не зависит от инфраструктуры → **Inversion of Control**

Связи между сущностями и агрегация → **Link entity + CQRS**

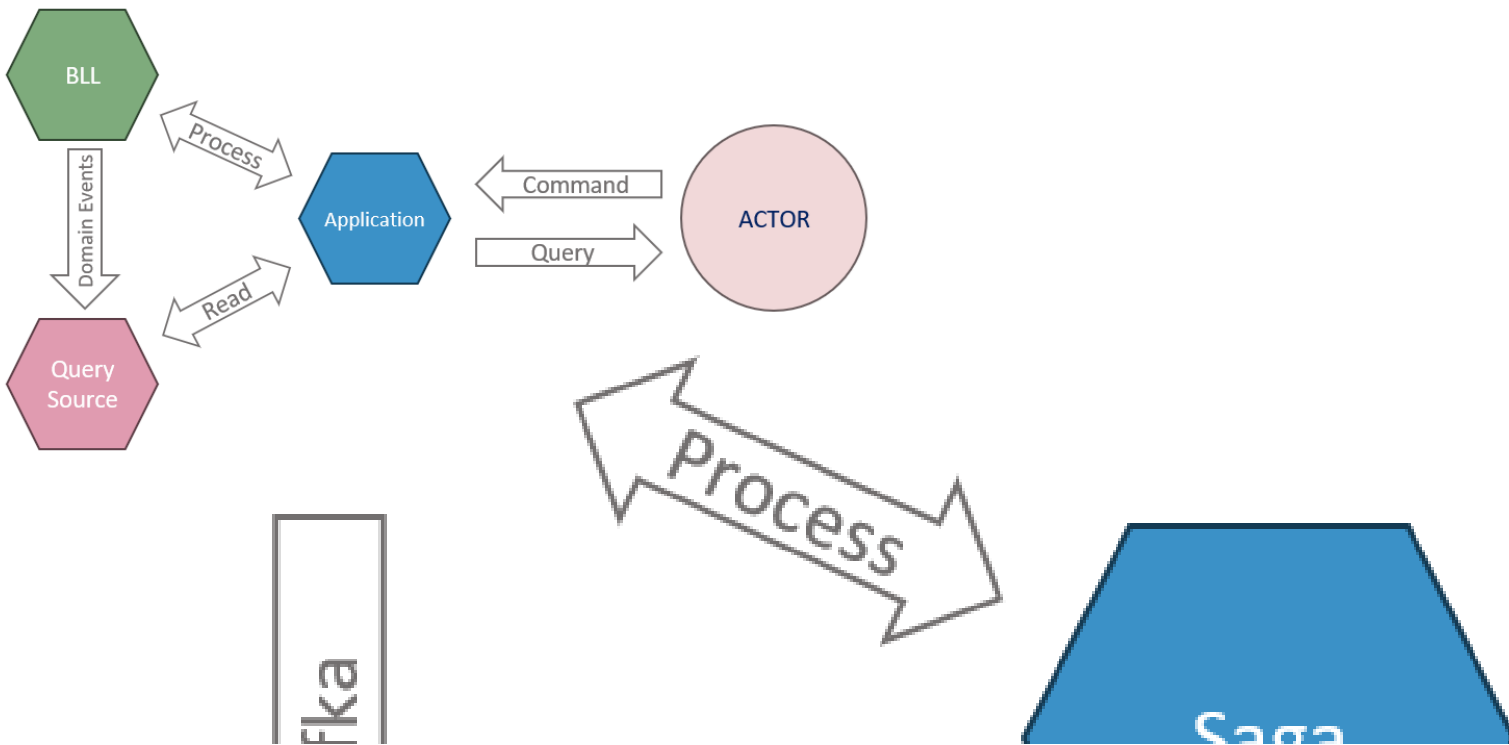
Наполнить данными QuerySource → **Domain Events + Visitor**

Модульность → **Core + Extended**

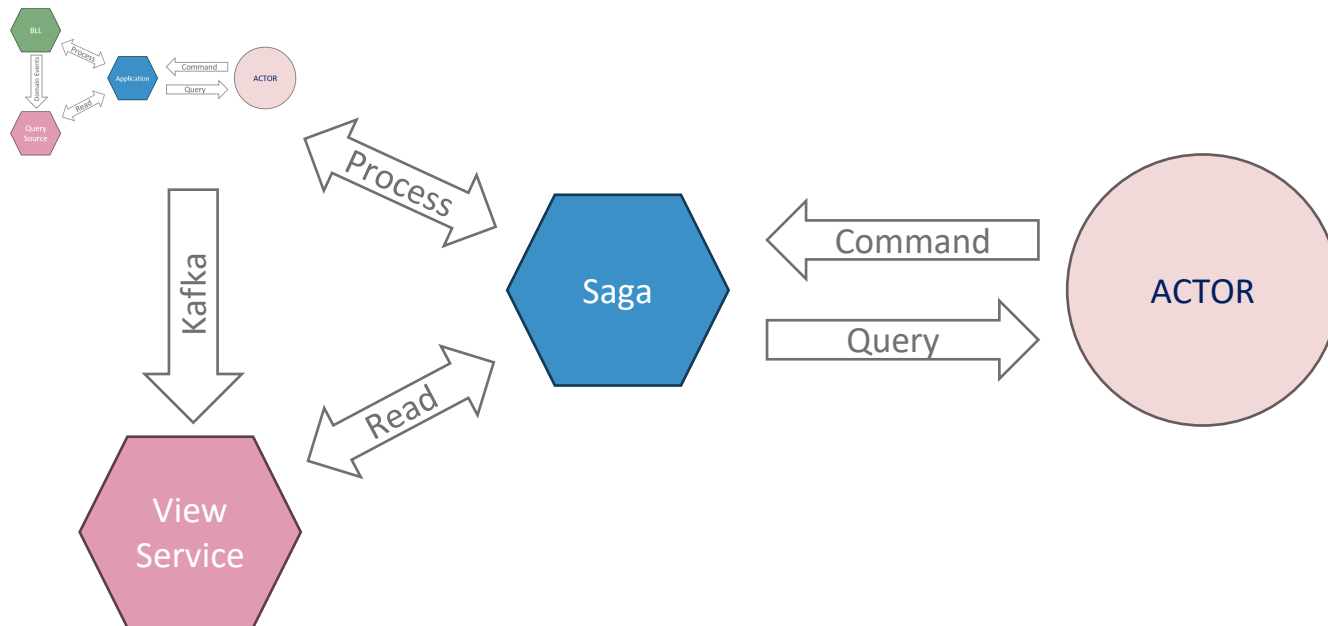
Подводя итог



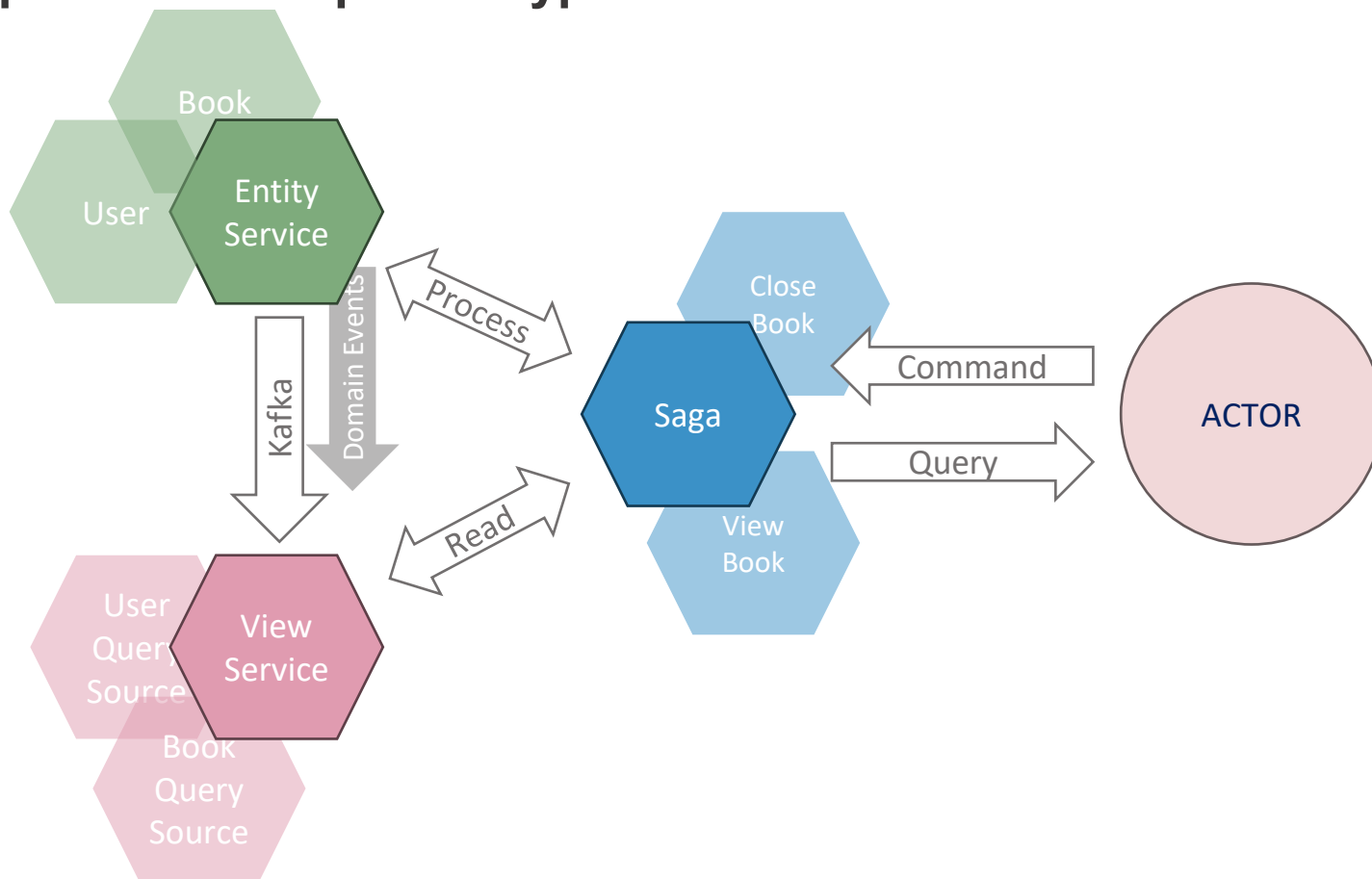
Подводя итог



Фрактальная архитектура



Фрактальная архитектура





АСЭ
РОСАТОМ

Спасибо за внимание

