Микросервисы на C#

C# на сервере и архитектура

Марк Шевченко

Backend разработчик

http://markshevchenko.pro

@markshevchenko

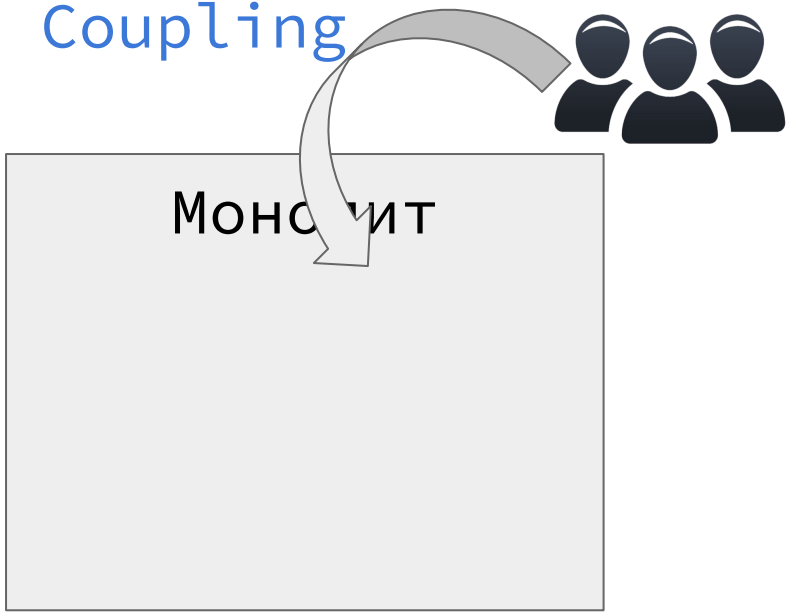
http://prog.msk.ru

Микросервисы – это...

Не монолит

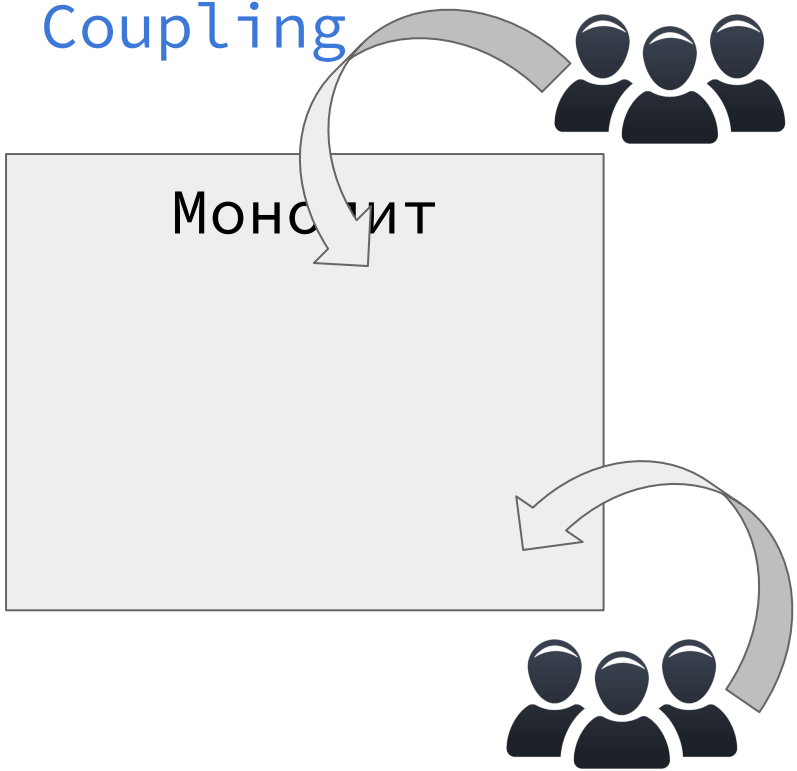
Не монолит

Coupling



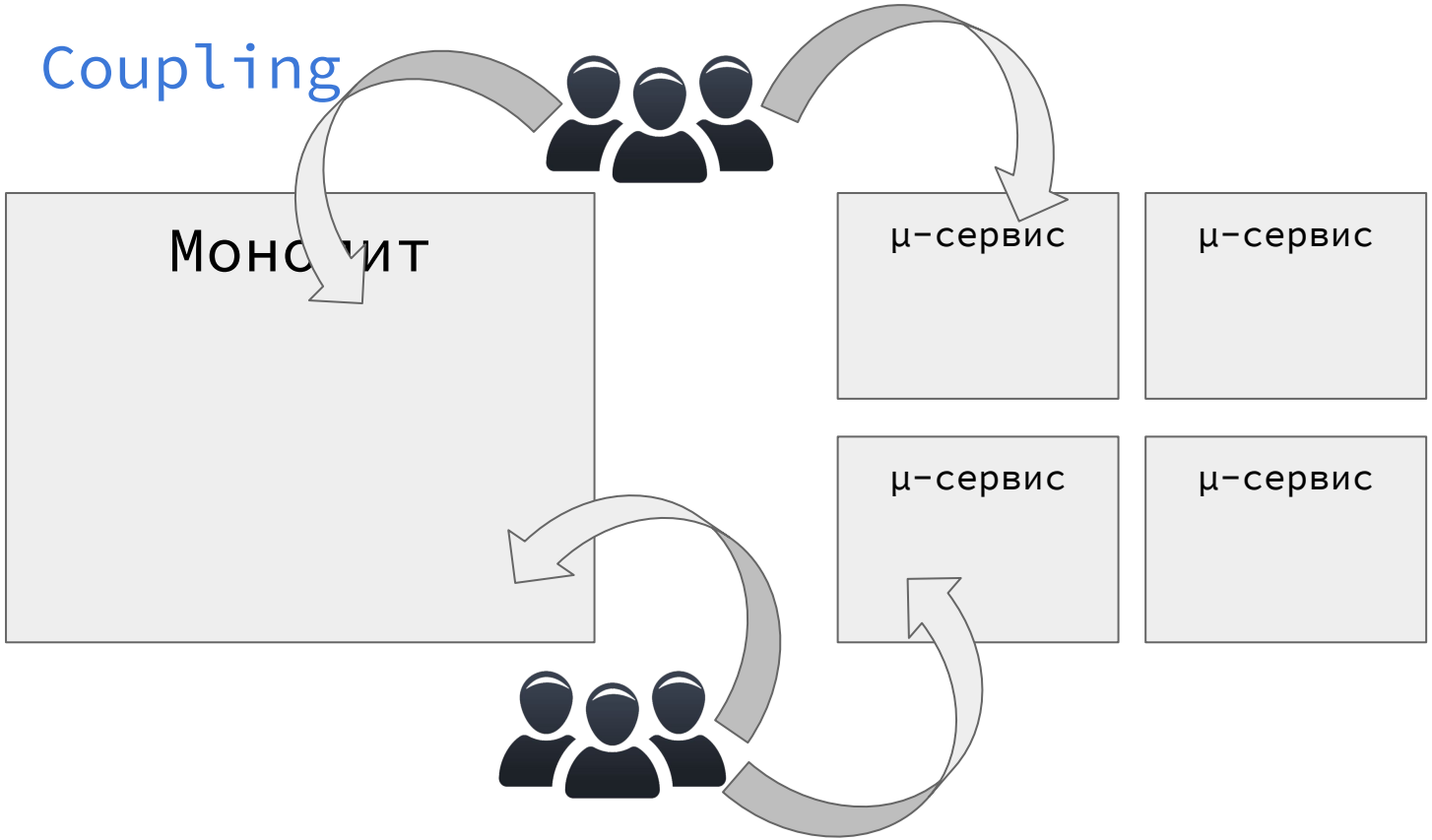
Не монолит

Coupling



Не монолит

Coupling



UNIX WAY

Make each program do one thing well

UNIX WAY

Make each program do one thing well

```
ps aux | tail -n +2 | wc -l
```

Масштабирование

Масштабирование

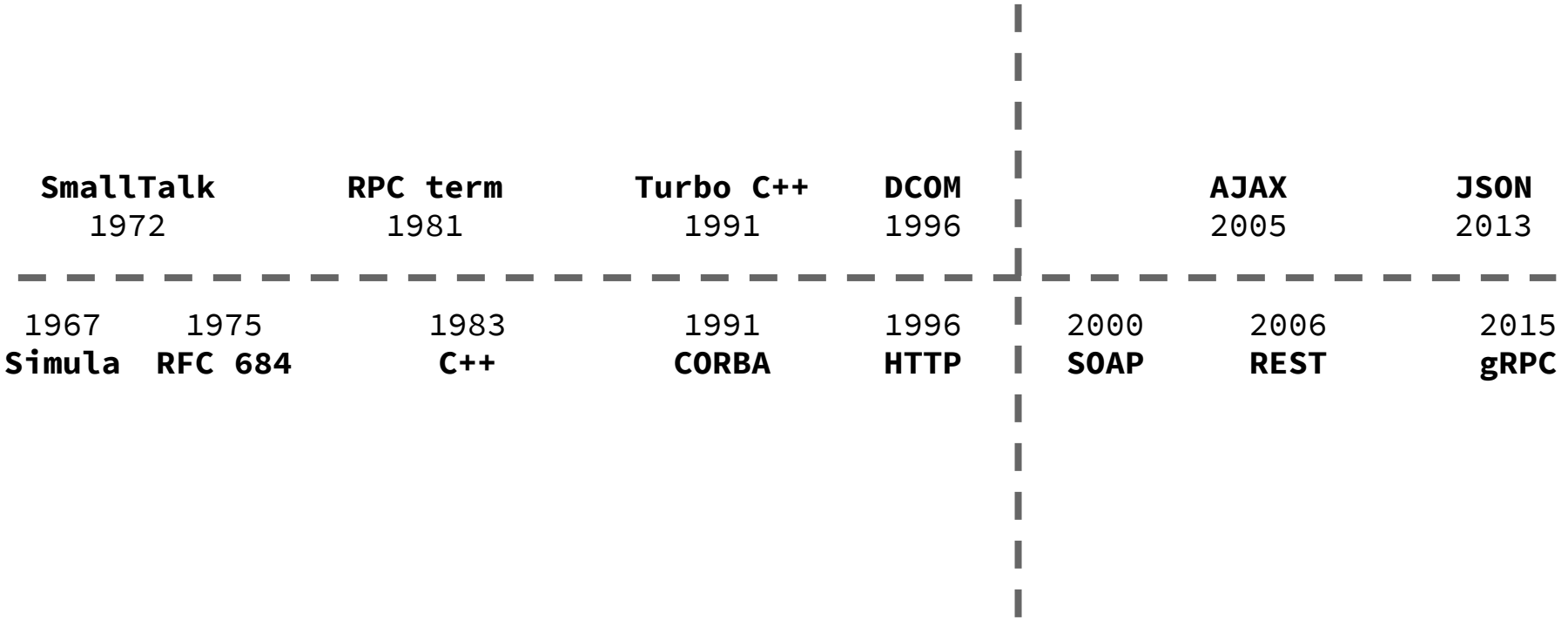
Масштабирование

Remote Procedure Call

SmallTalk	**RPC term**	**Turbo C++**	**DCOM**	**AJAX**	**JSON**
1972	1981	1991	1996	2005	2013

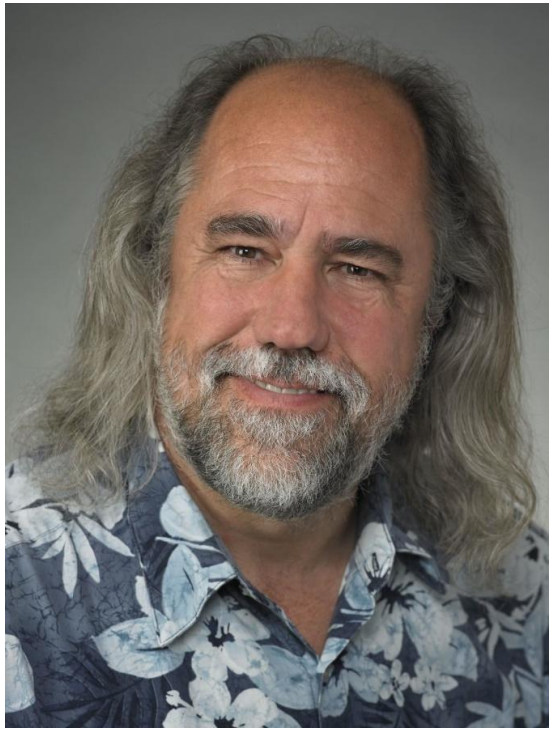
1967	1975	1983	1991	1996	2000	2006	2015
Simula	**RFC 684**	**C++**	**CORBA**	**HTTP**	**SOAP**	**REST**	**gRPC**

Remote Procedure Call

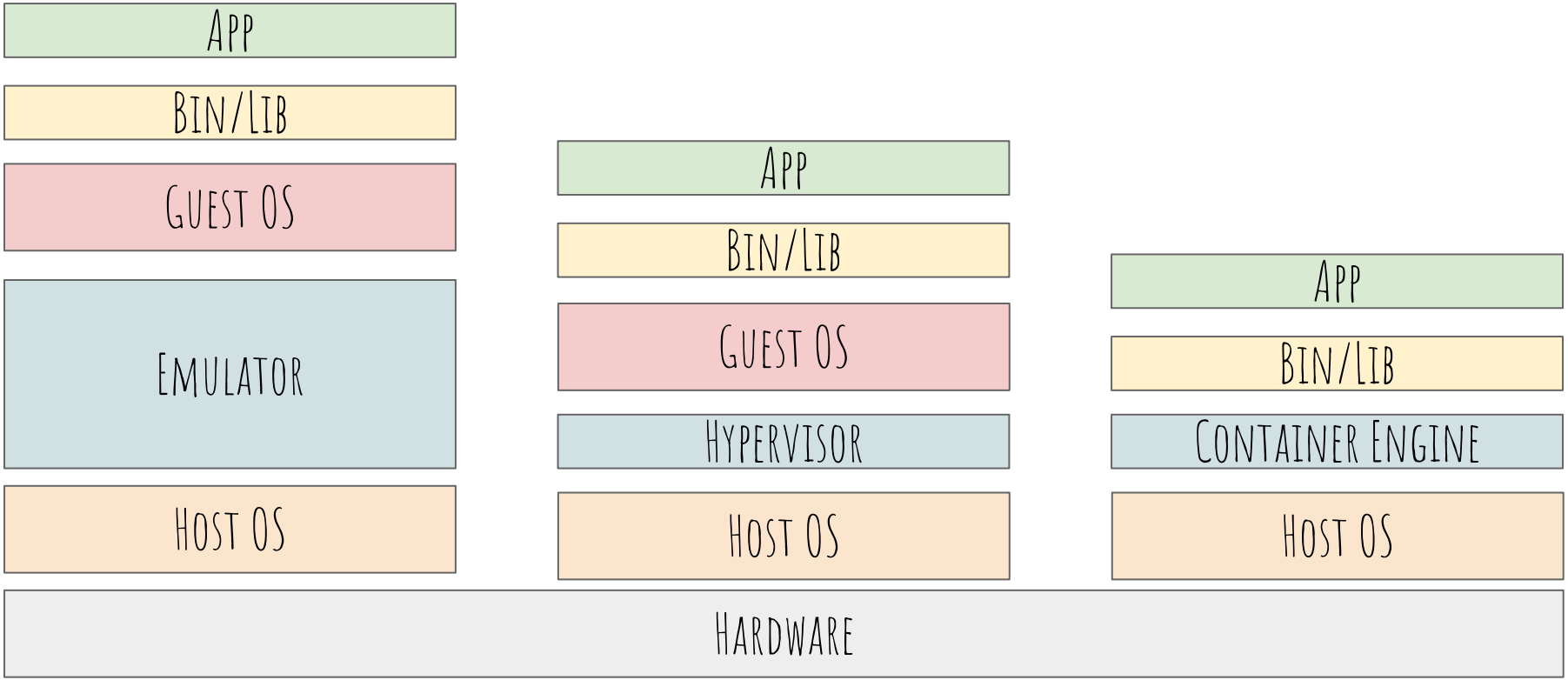


Микросервисы — это веб-приложения

Continuous Integration

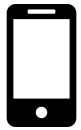


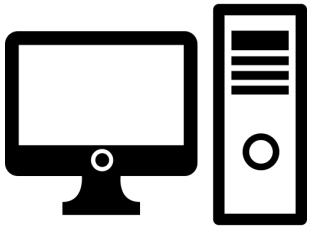
Виртуализация



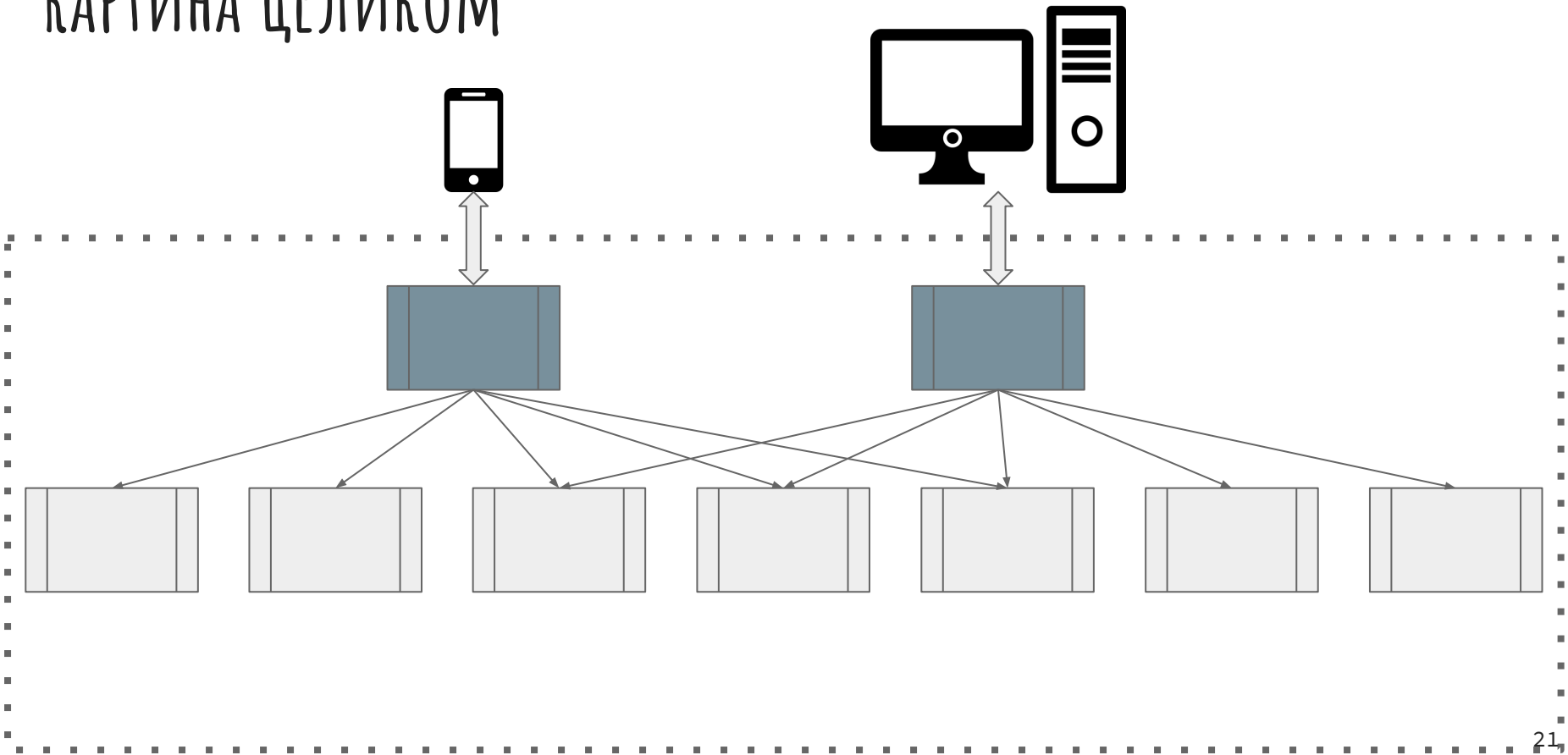
Картина целиком

Картина целиком

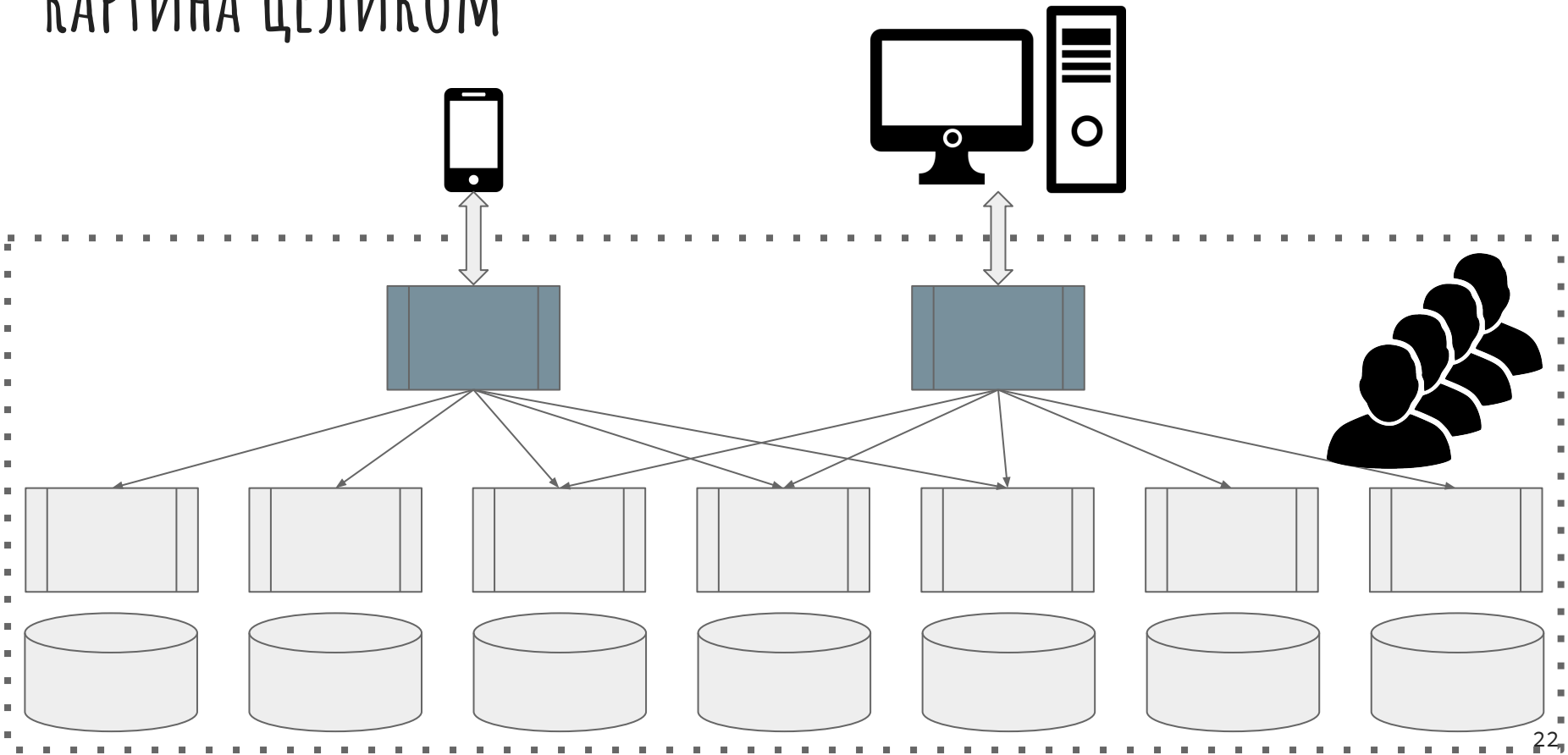




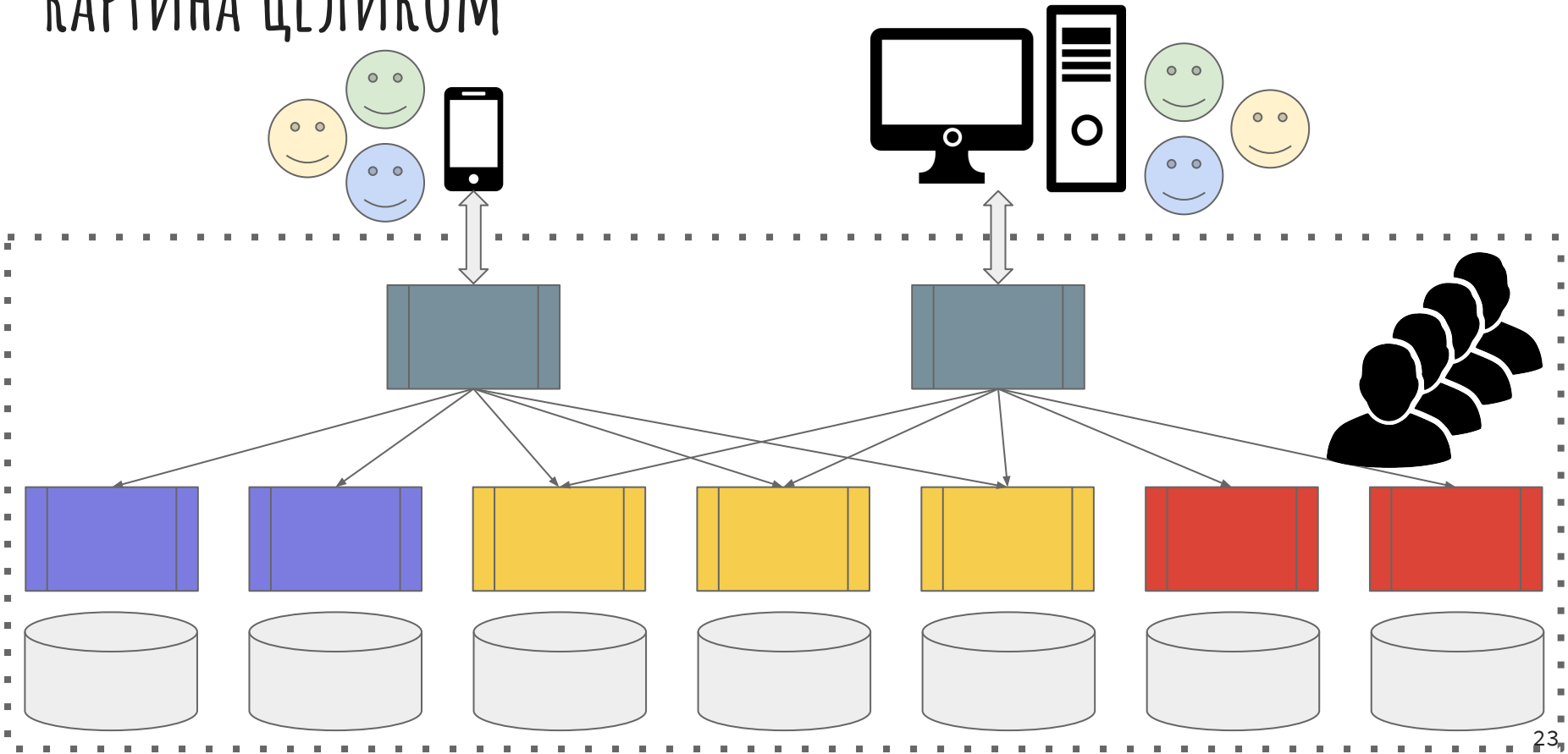
Картина целиком



Картина целиком

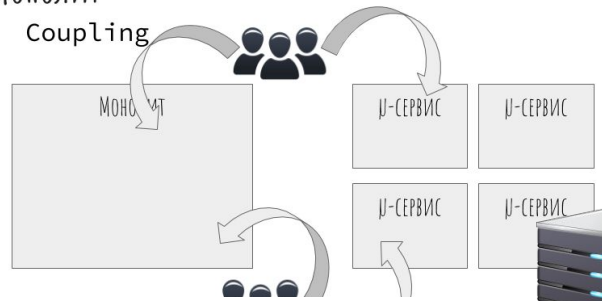


Картина целиком



Микросервисы – это...

Не монолит

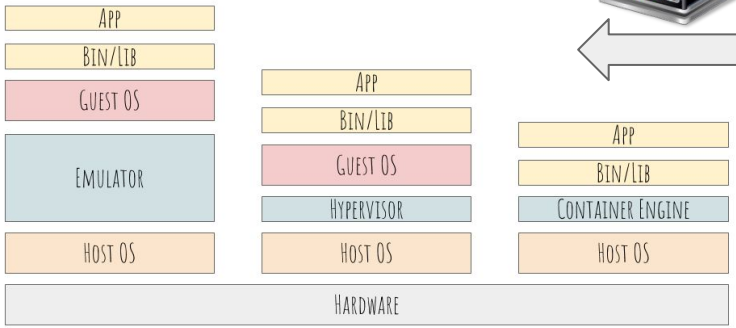


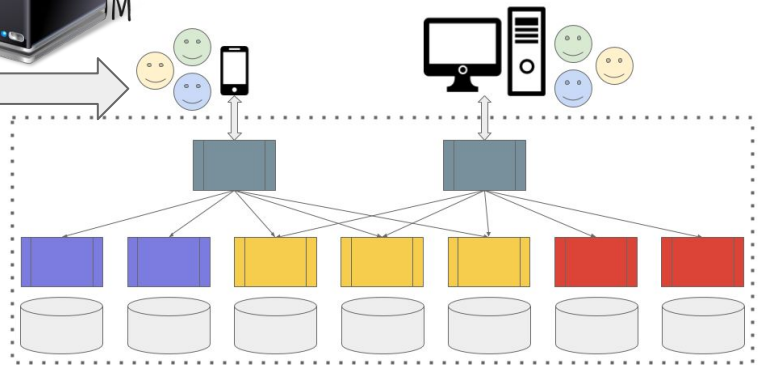
Remote Procedure Call

SmallTalk	RPC term	Turbo C++	DCOM	AJAX	JSON
1972	1981	1991	1996	2005	2013

1967	1975	1983	1991	1996	2000	2006	2015
Simula	RFC 684	C++	CORBA	HTTP	SOAP	REST	gRPC

Виртуализация





Ссылки про микросервисы

- https://auth0.com/blog/an-introduction-to-microservices-part-1/
  https://auth0.com/blog/an-introduction-to-microservices-part-2-API-gateway/
  https://auth0.com/blog/an-introduction-to-microservices-part-3-the-service-registry/
  https://auth0.com/blog/introduction-to-microservices-part-4-dependencies/

- https://12factor.net/ru/
  https://microservices.io/patterns/apigateway.html
  https://martinfowler.com/articles/microservices.html

Web в .NET

Web в .NET

WCF
2006

Web API
2009

Katana
2013

.NET Core 1
2016

2003
WebForms

2009
MVC

2013
OWIN

2016
Kestrel

2017
.NET Core 2

Пример

Пример

```
[Route("api/v1/todo-items")]
public class TodoItemsController : ControllerBase
{
    private readonly TodoDbContext dbContext;

    [HttpGet(“{id}”)]
    public async Task<TodoItem> GetByIdAsync(int id)
    {
        return await dbContext.TodoItems
                                .SingleAsync(x => x.Id == id);
    }
}
```

Пример

```
[Route("api/v1/todo-items")]
public class TodoItemsController : ControllerBase
{
    private readonly TodoDbContext dbContext;

    [HttpGet(“{id}”)]
    public async Task<TodoItem> GetByIdAsync(int id)
    {
        return await dbContext.TodoItems
                                .SingleAsync(x => x.Id == id);
    }
}
```

Пример

```
[Route("api/v1/todo-items")]
public class TodoItemsController : ControllerBase
{
    private readonly TodoDbContext dbContext;

    [HttpGet(“{id}”)]
    public async Task<TodoItem> GetByIdAsync(int id)
    {
        return await dbContext.TodoItems
                                .SingleAsync(x => x.Id == id);
    }
}
```

GET /api/v1/todo-items/123

Пример

GET /api/v1/todo-items/123

```
[Route("api/v1/todo-items")]
public class TodoItemsController : ControllerBase
{
    private readonly TodoDbContext dbContext;

    [HttpGet("{id}")]
    public async Task<TodoItem> GetByIdAsync(int id)
    {
        return await dbContext.TodoItems
                              .SingleAsync(x => x.Id == id);
    }
}
```

Пример

```
[Route("api/v1/todo-items")]
public class TodoItemsController : ControllerBase
{
    private readonly TodoDbContext dbContext;

    [HttpGet(“{id}”)]
    public async Task<TodoItem> GetByIdAsync(int id)
    {
        return await dbContext.TodoItems
                                .SingleAsync(x => x.Id == id);
    }
}
```

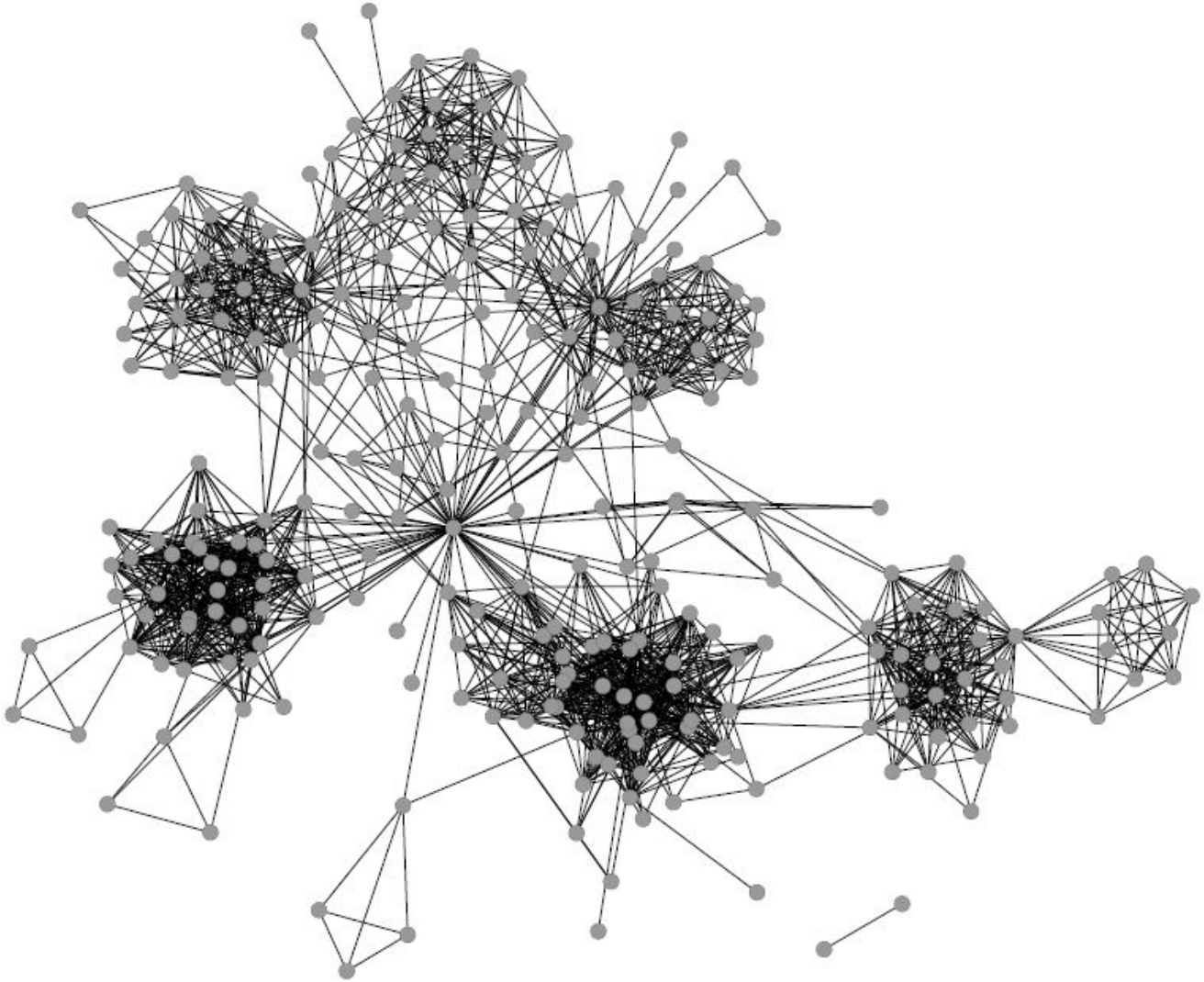
Пример

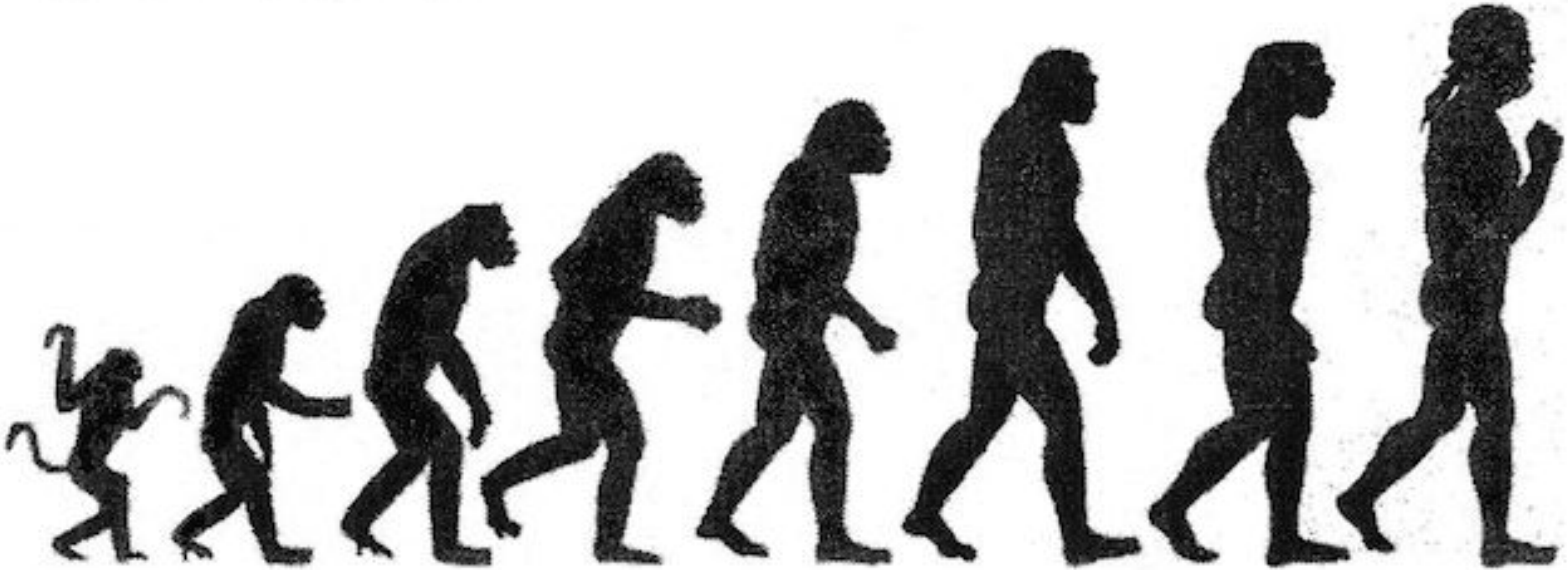
```
[Route("api/v1/todo-items")]
public class TodoItemsController : ControllerBase
{
    private readonly TodoDbContext dbContext;

    [HttpGet(“{id}”)]
    public async Task<TodoItem> GetByIdAsync(int id)
    {
        return await dbContext.TodoItems
                              .SingleAsync(x => x.Id == id);
    }
}
```

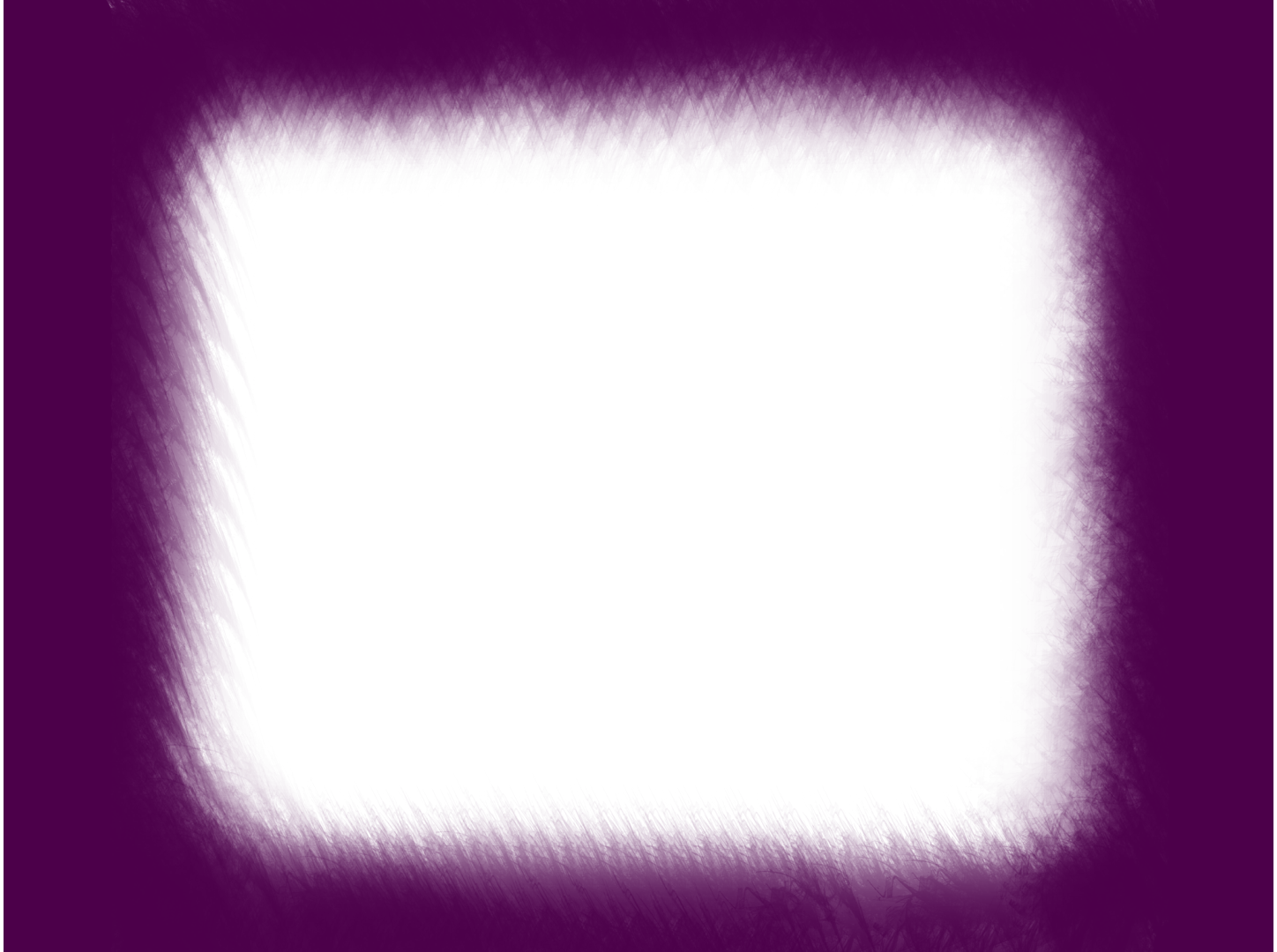
Ссылки про веб-приложения

- https://dotnetcore.show/episode-1-a-brief-history-of-net-core/
- https://docs.microsoft.com/ru-ru/aspnet/core/tutorials/first-web-api
- https://github.com/dotnet-architecture/eShopOnContainers

Проблемы







Ссылки про проблемы

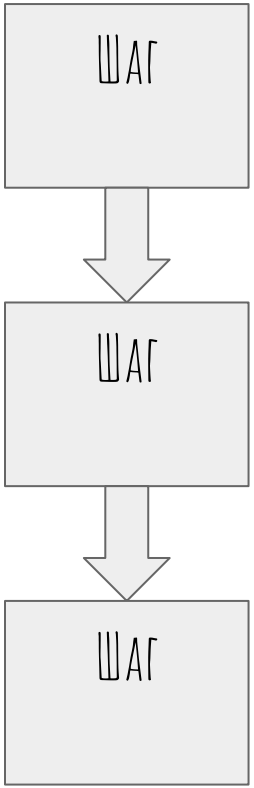
- https://dwmkerr.com/the-death-of-microservice-madness-in-2018/

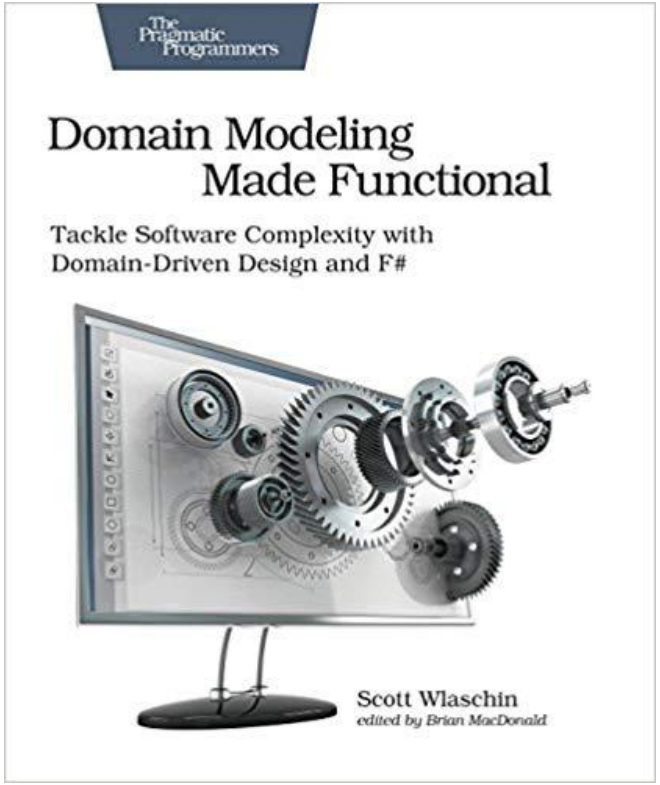
Решения

Транзакции

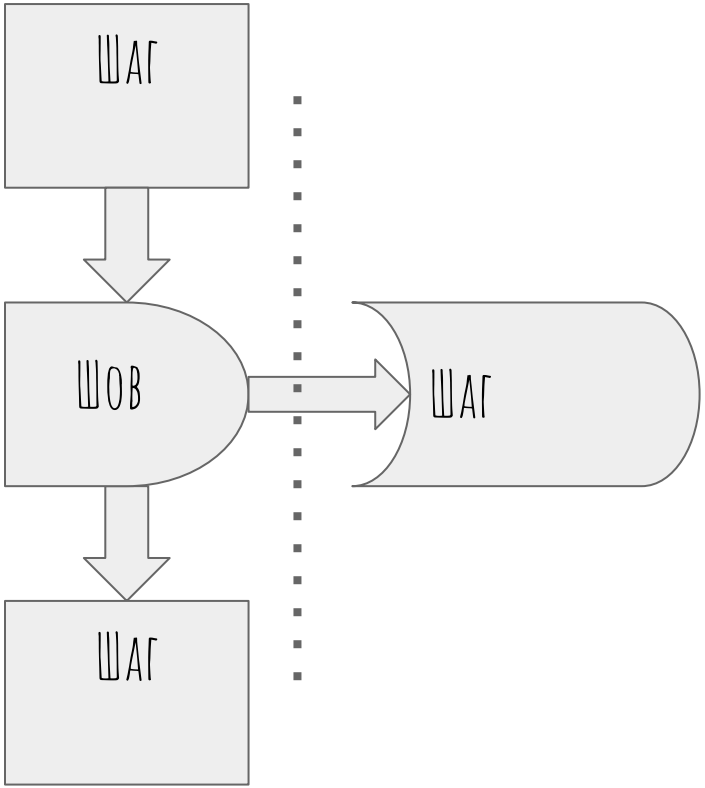
- отказ
- партиции, шардирование
- эмуляция
- уборка
- в виде исключения

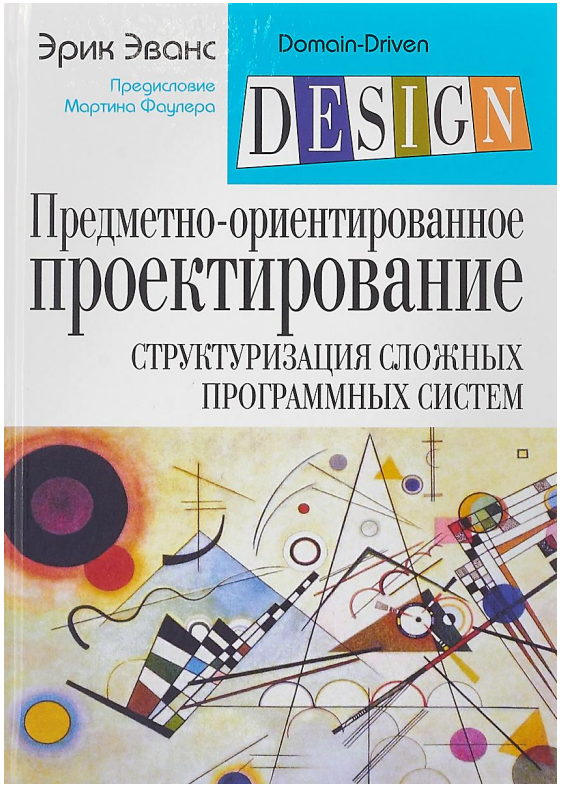
Сцепленность



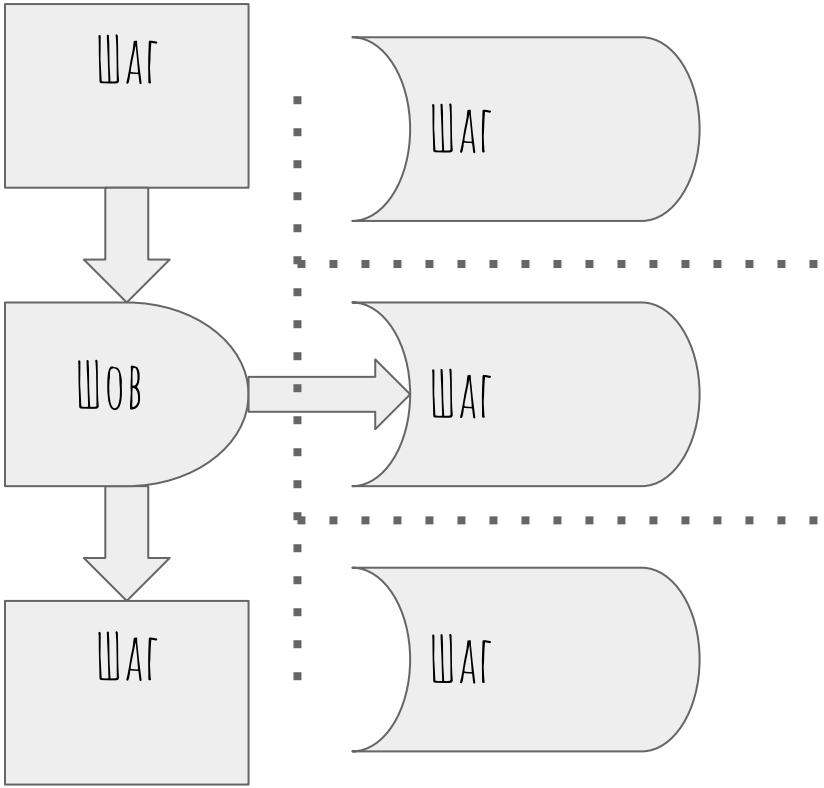


Сцепленность



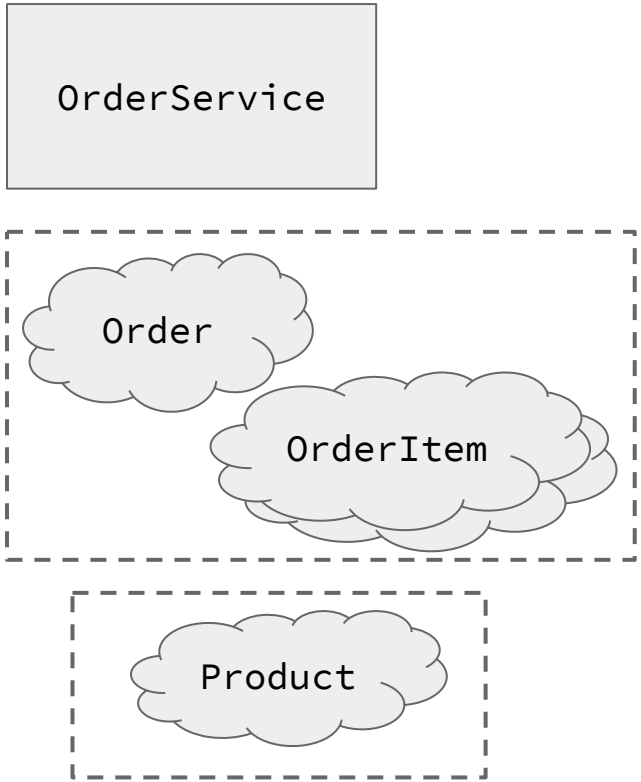


Сцепленность

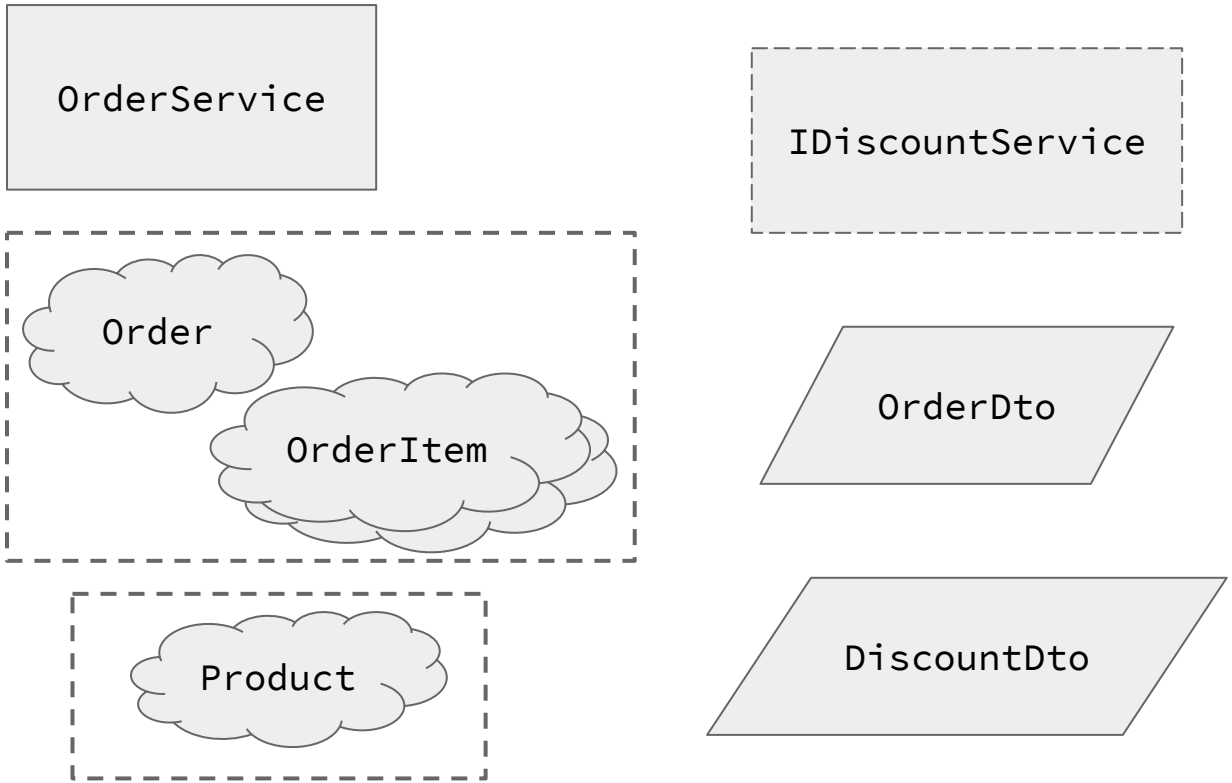


Микросервисы — это деталь

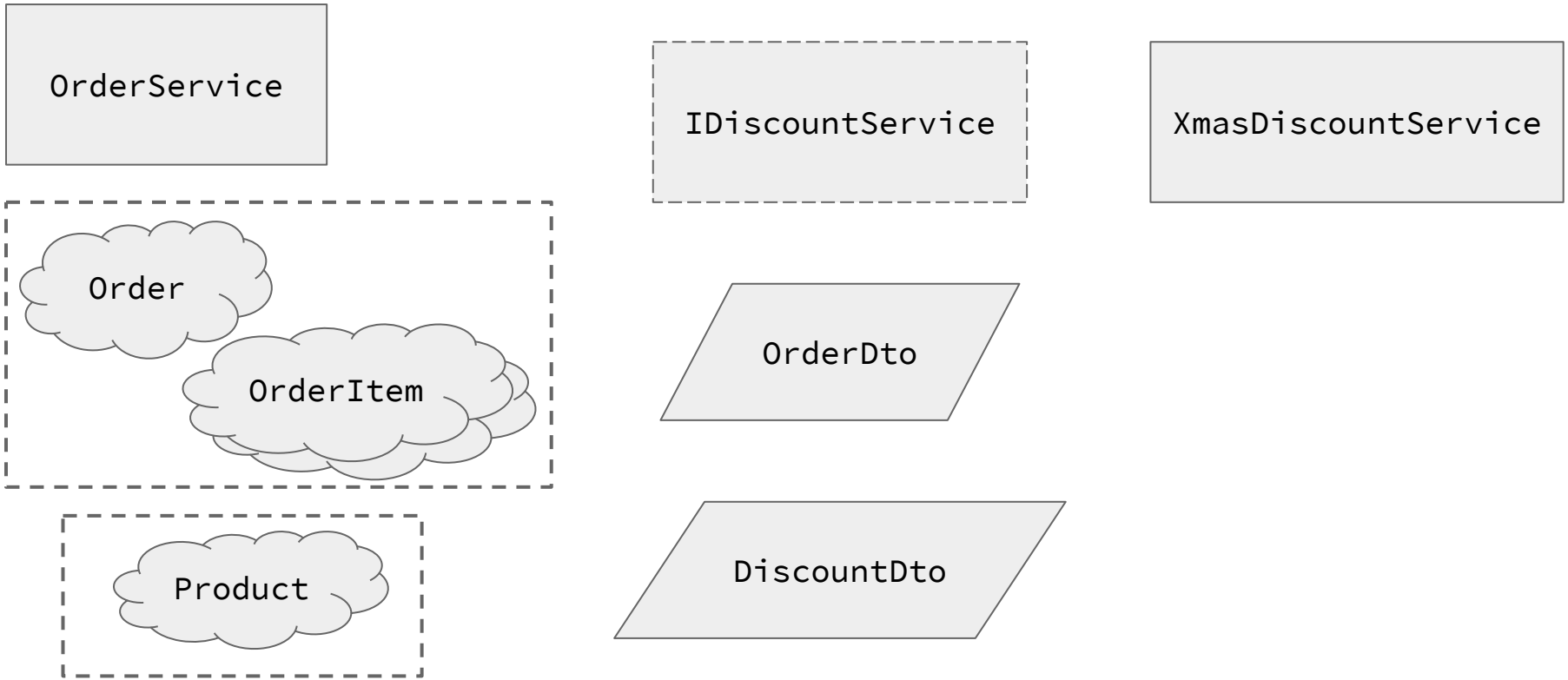
Деталь реализации



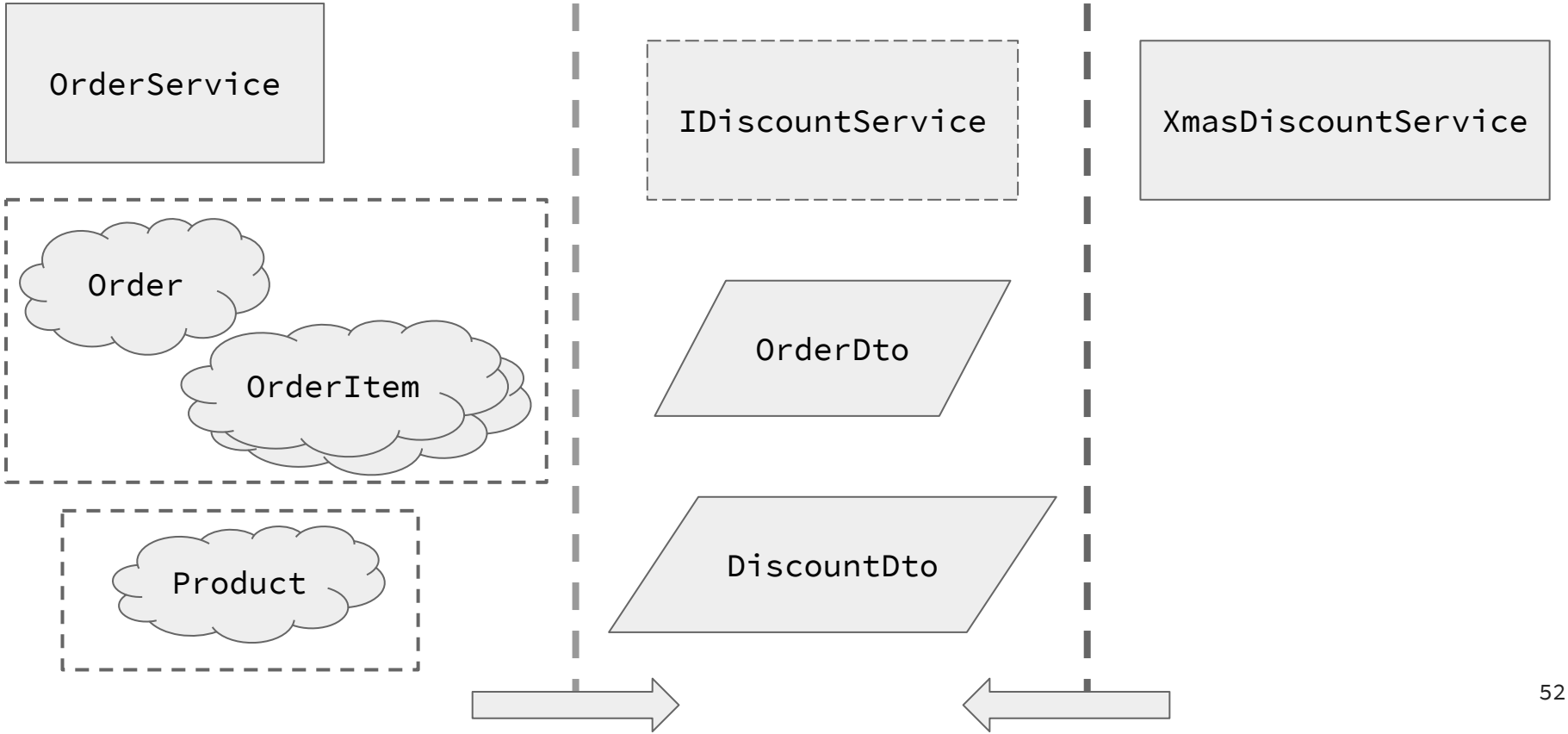
Деталь реализации



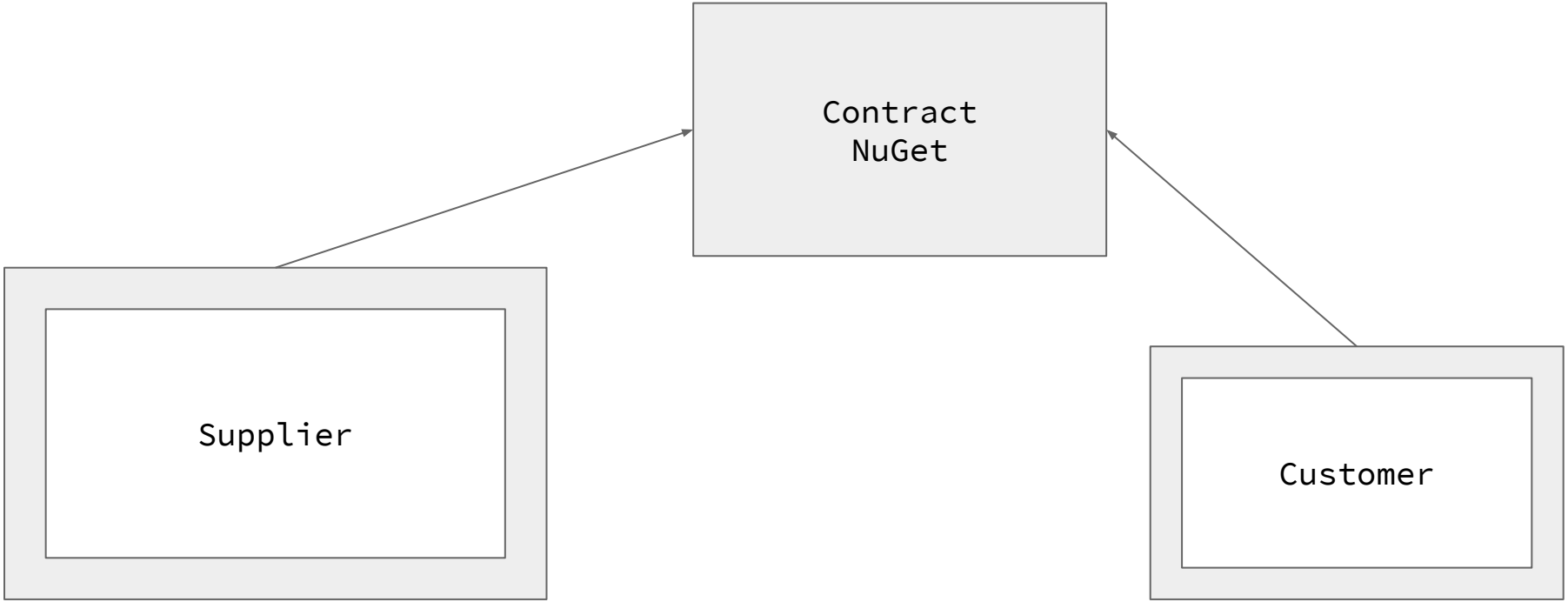
Деталь реализации



Деталь реализации



Деталь реализации



Contract NuGet

```
interface IDiscountService
{
    DiscountDto[] Calculate(OrderDto orderDto);
}


class DiscountDto
{
    public int ProductId { get; set; }
    public decimal Amount { get; set; }
    public string Description { get; set; }
}


class OrderDto
{
    . . .
}
```

Customer

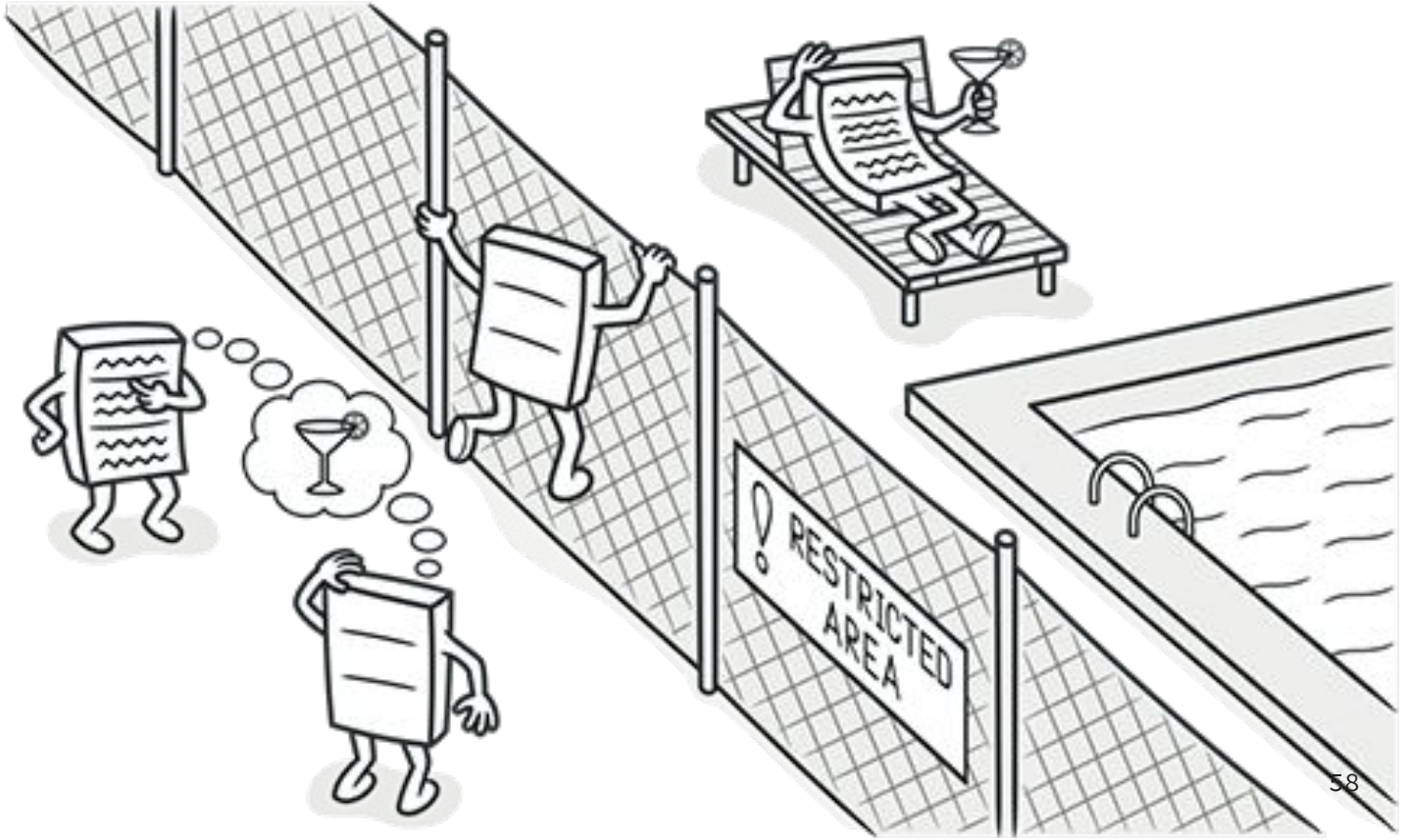
```
[Route(“api/v1/discounts”)]
class DiscountController : ControllerBase, IDiscountService
{
    [HttpPost]
    public DiscountDto[] Calculate(OrderDto orderDto)
    {
        var discounts = from item in orderDto.Items
                        select new DiscountDto
                        {
                            ProductId = item.ProductId,
                            Amount = item.Sum * 0.03m,
                            Description = “Merry Christmas”
                        };

        return discounts.ToArray();
    }
}
```

Supplier

```
class DiscountProxy : IDiscountService
{
    private readonly IHttpClientFactory clientFactory;

    public DiscountDto[] Calculate(OrderDto orderDto)
    {
        return clientFactory.Create()
                            .Put<DiscountDto[]>(“api/v1/discount”, orderDto);
    }
}
```

```
paths:
  /:
    get:
      operationId: listVersionsv2
      summary: List API versions
      responses:
        '200':
          description: |-
            200 response
          content:
            application/json:
              examples:
                foo:
                  value: {
                    "versions": [
                      {
                        "status": "CURRENT",
                        "updated": "2011-01-21T11:33:21Z",
                        "id": "v2.0",
                        "links": [
```

Feature Envy



Решения

- Supplier/Customer или Shared Core
- Интерфейсы служб
- Косвенный вызовы
- DTO, а не POCO
- NuGet
- Message Queue
- Open-Closed Principle

Ссылки про решения

- https://github.com/ThreeMammals/Ocelot
- https://www.envoyproxy.io/
- https://www.nuget.org/packages/Binateq.JsonRestClient
- https://github.com/domaindrivendev/Swashbuckle
- https://github.com/RicoSuter/NSwag

Содержание

- не монолит
- UNIX way
- масштабирование
- SOAP, REST, gRPC
- CI/CD
- контейнеры, оркестрация
- шлюзы
- manage its own data
- web в .NET
- MVC, маршрутизация
- асинхронность
- проблема: транзакции
- проблема: сеть
- проблема: версии
- решения для транзакций
- решения для сцепленности
- деталь реализации, Open API
- feature envy, MQ