

DDD в унаследованном коде –
способы борьбы со
сложностью ПО

Райффайзен



Обо мне



Константин Густов

архитектор,
Райффайзенбанк

10+ лет опыта в разработке

konst.gustov@gmail.com

Райффайзен



Темы

I. Унаследованный код.

Темы

I. Унаследованный код.

II. It's DDD Time!

Райффайзен



Темы

- I. Унаследованный код.
- II. It's DDD Time!
- III. Как перейти от монолита к микросервисам.

Унаследованный код





Большой ком грязи

Беспорядочно
структурированные,
растянутые, неряшливые,
напоминающие спутанную
пленку джунгли спагетти
кода (Brian Foote and Joseph
Yoder, *Big Ball of Mud*)

Райффайзен



Проблемы кода

- Повторная реализация функционала.

Райффайзен



Проблемы кода

- Повторная реализация функционала.
- Классы содержат свойства и методы, не относящиеся к ним.

Райффайзен



Проблемы кода

- Повторная реализация функционала.
- Классы содержат свойства и методы, не относящиеся к ним.
- Бизнес-логика сосредоточена в сервисах (менеджерах, процессорах).

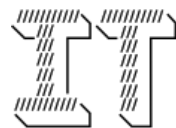
Райффайзен



Проблемы кода

- Повторная реализация функционала.
- Классы содержат свойства и методы, не относящиеся к ним.
- Бизнес-логика сосредоточена в сервисах (менеджерах, процессорах).
- Ответственность распределена между классами, глядя на один, трудно понять, что он делает.

Райффайзен



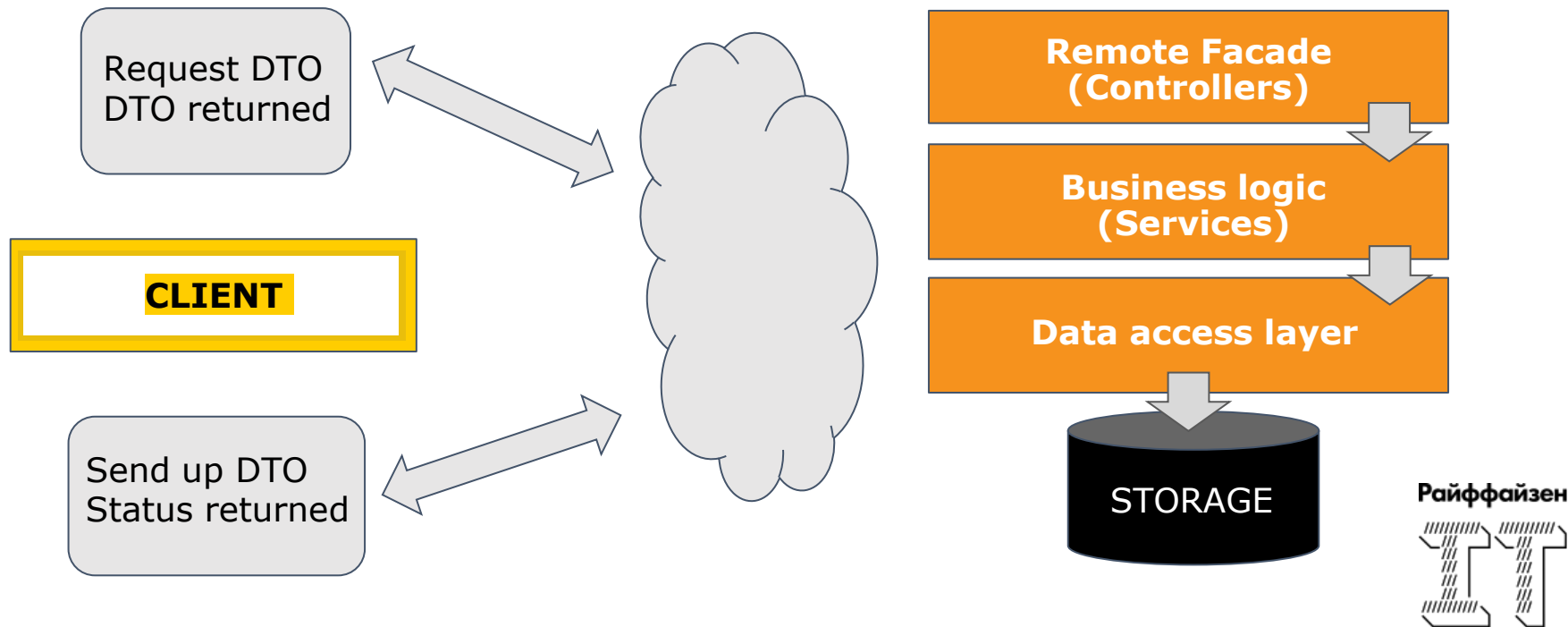
Обычное приложение

- Администратор добавляет новые заявки на доставку.
- Администратор просматривает состояние заявок.
- Курьеры назначают заявки на себя.
- Курьеры работают с заездами.

Райффайзен

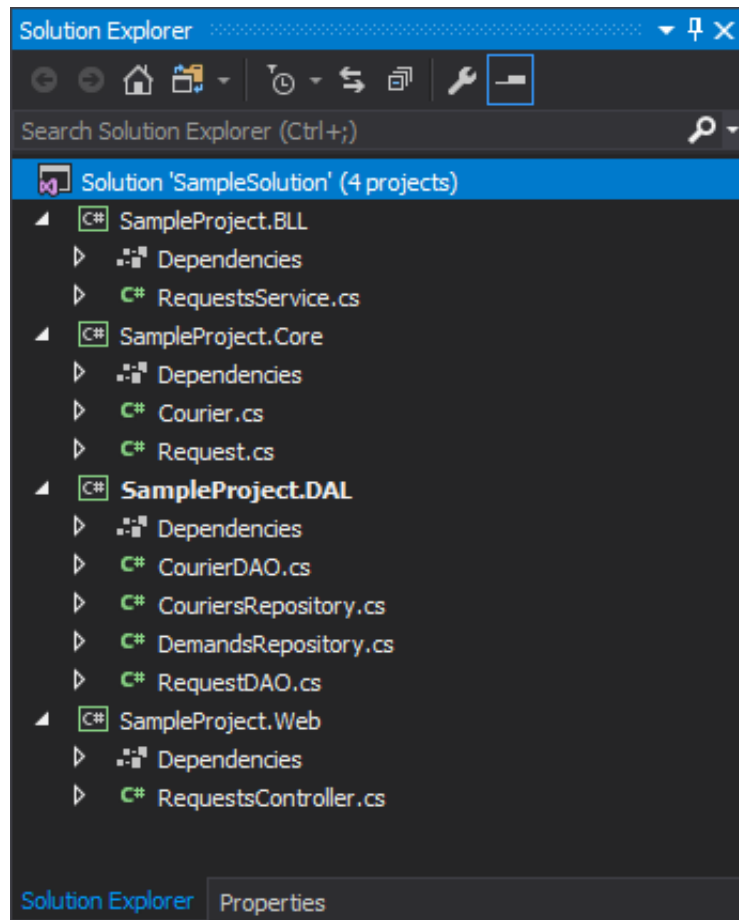


Архитектура обычного приложения



Структура решения

- Трехслойная архитектура.
- Слой контроллеров для реализации шаблона MVC.
- Слой бизнес-логики.
- Слой доступа к данным.



Райффайзен

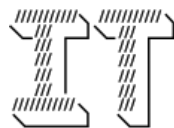


Контроллер заявок

```
[RoutePrefix("api/requests")]
public class RequestsController : ApiController
{
    ...

    [HttpPost]
    [Route("/{extId}/courier/{courierId}")]
    public IHttpActionResult MarkRequest(int requestId, int courierId)
    {
        _requestsService.BindRequestToCourier(requestId, courierId);
        ...
    }
}
```

Райффайзен



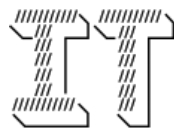
Фрагмент бизнес-логики

```
public class RequestsService
{
    public void BindRequestToCourier(int requestId, int courierId)
    {
        var demand = _demandsRepository.GetDemand(requestId);
        var courier = _couriersRepository.GetCourier(courierId);
        ValidateCanConnect(demand, courier);

        demand.Courier = courier;
        demand.Status = RequestStatus.Bound;
        demand.HasError = false;
        demand.Error = string.Empty;
        demand.ErrorCode = ErrorCode.NoError;
        courier.Request = request;

        if(demand.Appropriatable && demand.RequestType == RequestType.VIP)
            _emailNotifier.SendRequestHasBound(demand);
        ...
    }
}
```

Райффайзен



Классы «Заявка» и «Курьер»

```
public class Request
{
    public int Id { get; set; }

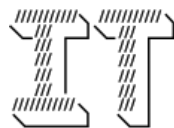
    public Courier Courier { get; set; }
    public RequestStatus Status { get; set; }
    public bool HasError { get; set; }
    public string Error { get; set; }
    public ErrorCode ErrorCode { get; set; }

    ...
}
```

```
public class Courier
{
    public int Id { get; set; }
    public Request Request { get; set; }

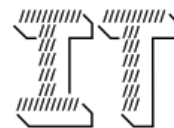
    ...
}
```

Райффайзен



It's DDD Time!

Райффайзен





Эрик Эванс

Специалист в проектировании ПО, автор книги «Предметно-ориентированное проектирование. Структуризация сложных программных систем».

Райффайзен



Domain-driven design

Набор принципов и схем, направленных на создание оптимальных систем объектов.

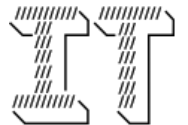
При правильном применении оно приводит к созданию программных абстракций, которые называются моделями предметных областей.



Заблуждения

- DDD – это решение всех проблем.

Райффайзен



Заблуждения

- DDD – это решение всех проблем.
- DDD – это фреймворк.

Райффайзен



Заблуждения

- DDD – это решение всех проблем.
- DDD – это фреймворк.
- Самое важное в DDD – это тактические шаблоны.

Райффайзен

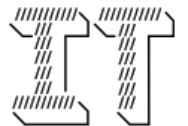


Модель предметной области

Это система абстракций, описывающая некоторые аспекты предметной области.

Модель упрощенно представляет реальность и помогает более или менее хорошо решать задачи.

Райффайзен



Хорошая модель

- Модель описывает не только структуру данных, но и поведение.

Райффайзен



Хорошая модель

- Модель описывает не только структуру данных, но и поведение.
- Удобна для реализации программистами, понятна доменным специалистам.

Хорошая модель

- Модель описывает не только структуру данных, но и поведение.
- Удобна для реализации программистами, понятна доменным специалистам.
- Модель одинаково представляется как в коде, так и в документации.



Хорошая модель

- Модель описывает не только структуру данных, но и поведение.
- Удобна для реализации программистами, понятна доменным специалистам.
- Модель одинаково представляется как в коде, так и в документации.
- Сущности и бизнес-правила одинаково важно влияют на модель.



Проблемы модели

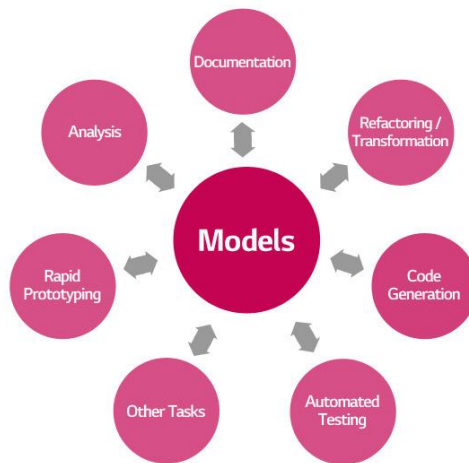
- Анемия.
- Семантический разрыв.

Райффайзен



Как снизить сложность с DDD

1. Используйте разработку на основе модели (model-driven development).



Райффайзен



Единый язык

Это язык, разработанный командой, состоящей из доменных экспертов и разработчиков, в процессе решения конкретных задач.

Удобен для реализации программистам, понятен доменным специалистам.



Хорошие практики единого языка

- Для каждого понятия должен быть один термин (+перевод).

Райффайзен



Хорошие практики единого языка

- Для каждого понятия должен быть один термин (+перевод).
- Термины должны быть согласованы с доменным экспертом.



Хорошие практики единого языка

- Для каждого понятия должен быть один термин (+перевод).
- Термины должны быть согласованы с доменным экспертом.
- Не используйте многозначные (политики) или специфические (транзакции) термины.



Хорошие практики единого языка

- Для каждого понятия должен быть один термин (+перевод).
- Термины должны быть согласованы с доменным экспертом.
- Не используйте многозначные (политики) или специфические (транзакции) термины.
- Не используйте имена шаблонов проектирования.



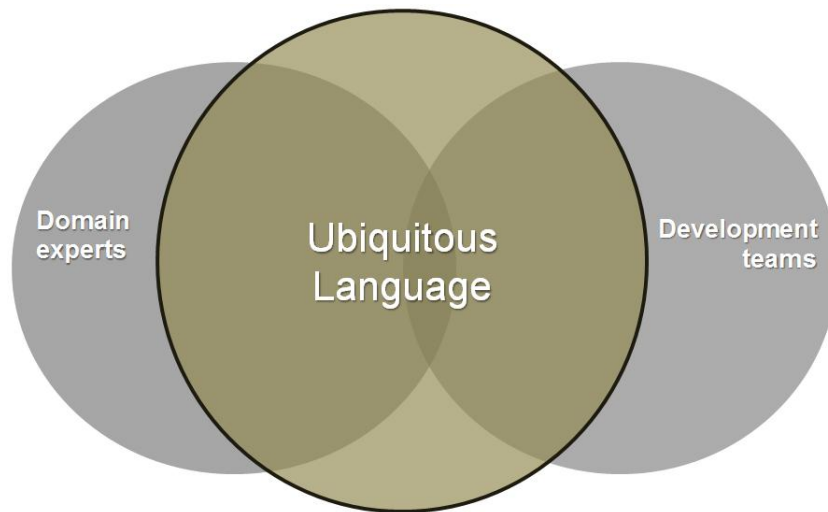
Хорошие практики единого языка

- Для каждого понятия должен быть один термин (+перевод).
- Термины должны быть согласованы с доменным экспертом.
- Не используйте многозначные (политики) или специфические (транзакции) термины.
- Не используйте имена шаблонов проектирования.
- Тестируйте модель на слух.



Как снизить сложность с DDD

2. Разработайте единый язык.



Райффайзен



DDD на практике

Райффайзен



Новые требования

- Администратор добавляет новые заявки на доставку.
- Администратор просматривает состояние заявок.
- Курьеры назначают заявки на себя.
- Курьеры работают с заездами.
- Курьеры могут брать заявки в зависимости от собственного статуса

Райффайзен

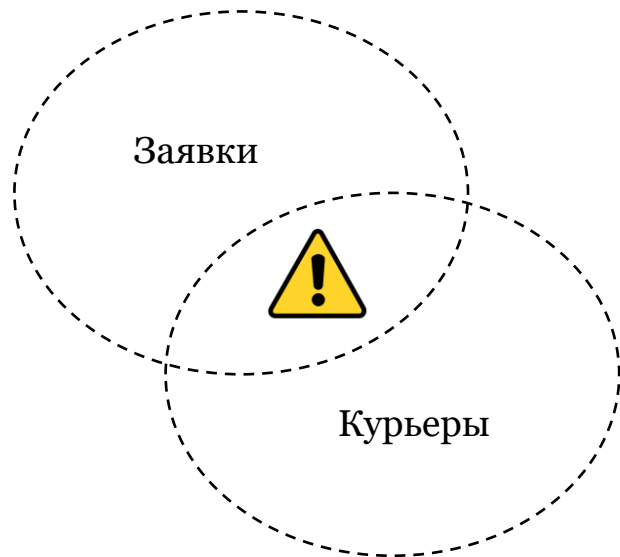


Ограниченный контекст

Граница, в пределах которой понятия единого языка имеют вполне конкретное контекстное значение (Bounded Context).



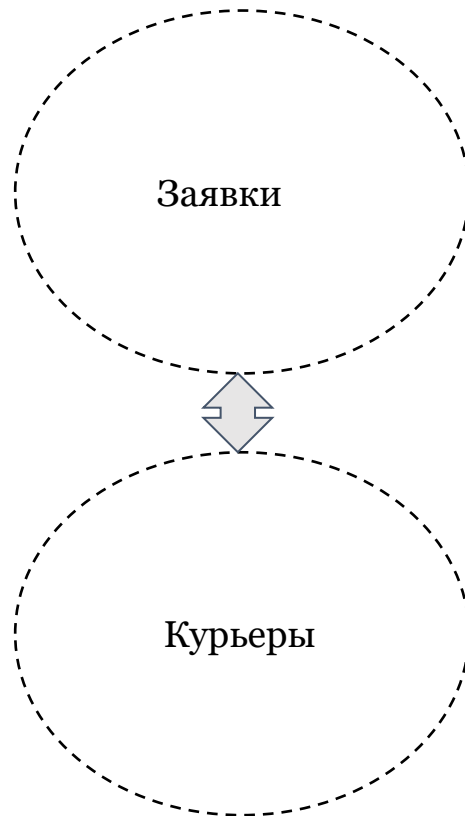
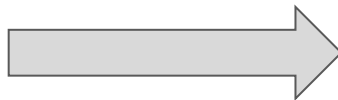
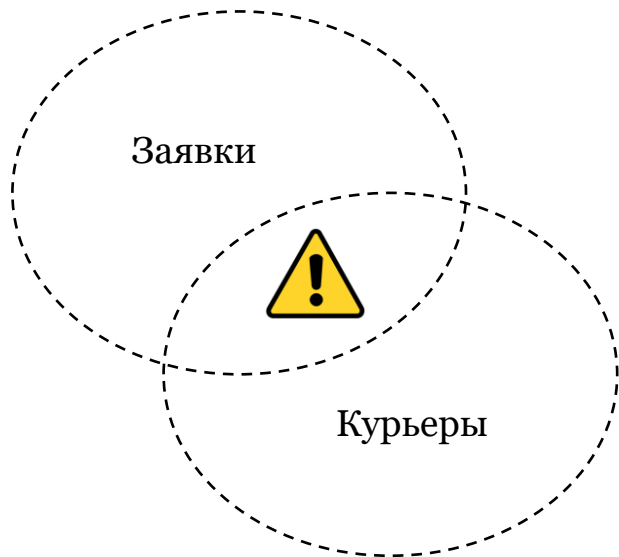
Контексты в приложении



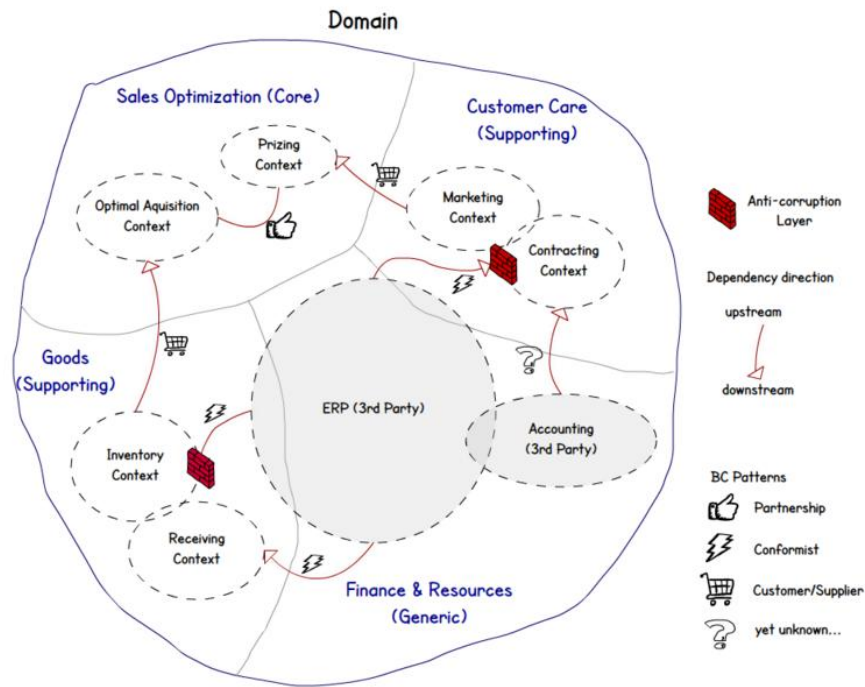
Райффайзен



Контексты в приложении



Карта контекстов



Райффайзен



Выделение микросервиса

- Микросервис решает одну бизнес-задачу.

Райффайзен



Выделение микросервиса

- Микросервис решает одну бизнес-задачу.
- Микросервис принадлежит одному ограниченному контексту.

Райффайзен

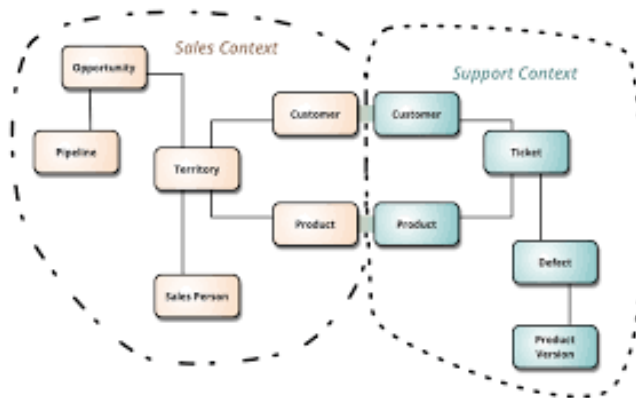


Выделение микросервиса

- Микросервис решает одну задачу.
- Микросервис принадлежит одному ограниченному контексту.
- Микросервис взаимодействует с другими микросервисами посредством «глупых» каналов связи.

Как снизить сложность с DDD

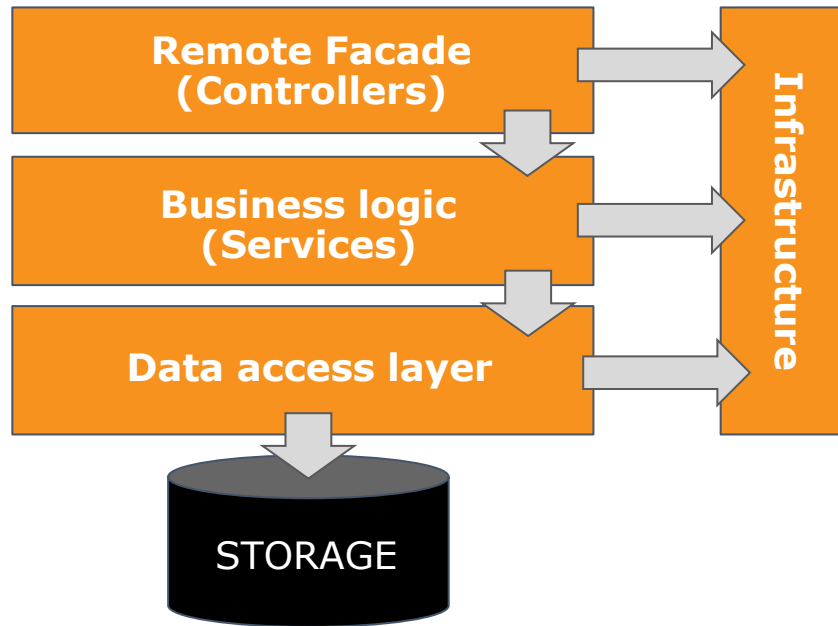
3. Разделяйте доменную логику приложения по ограниченным контекстам.



Райффайзен



Традиционная слоёная архитектура

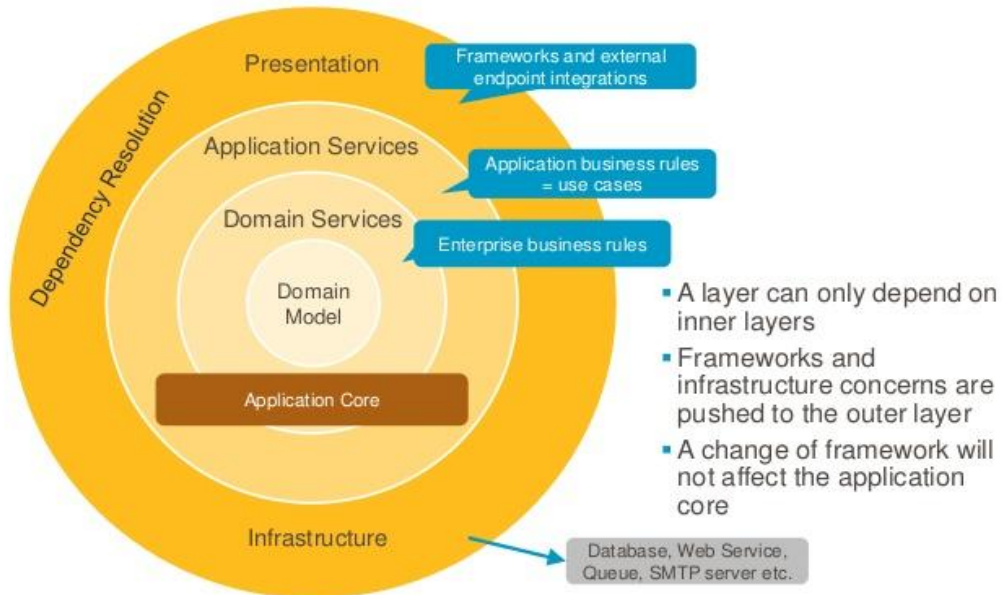


Райффайзен



Луковичная архитектура

Onion Architecture

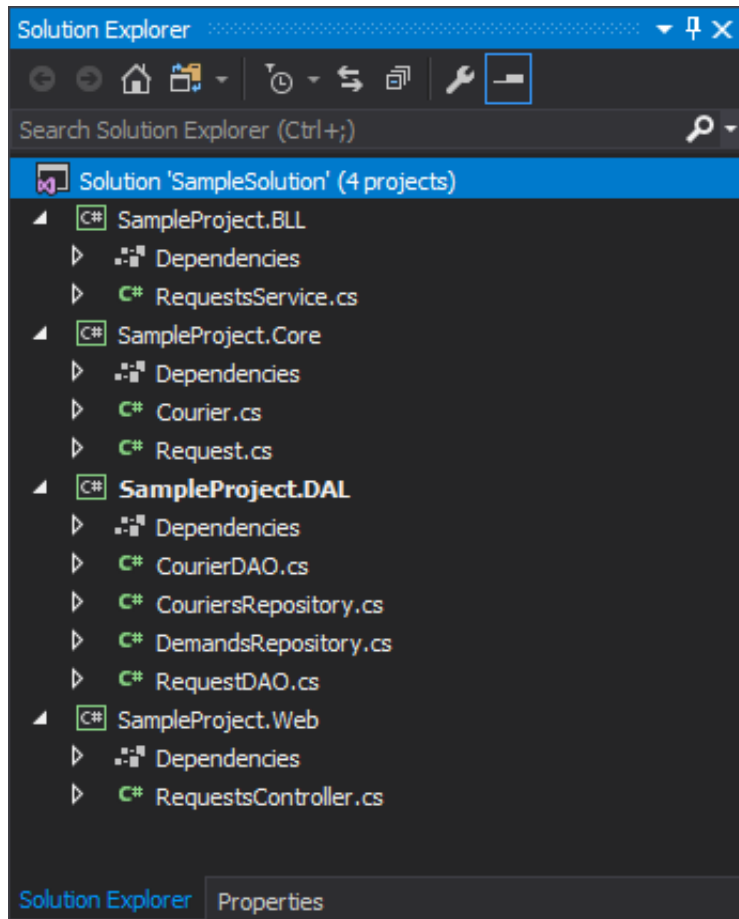


Райффайзен



Структура решения

- Трехслойная архитектура.
- Слой контроллеров для реализации шаблона MVC.
- Слой бизнес-логики.
- Слой доступа к данным.

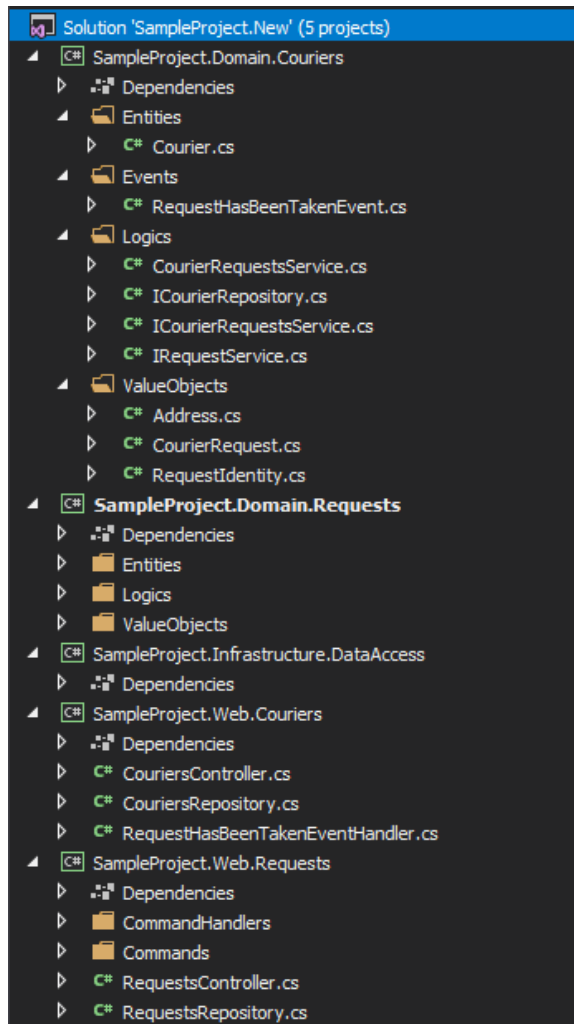


Райффайзен



Структура решения

- Луковичная архитектура.
- Доменная модель не зависит от инфраструктуры напрямую.
- Вся бизнес-логика сосредоточена в доменной модели.
- Точка сборки приложения – аппликационные сервисы.

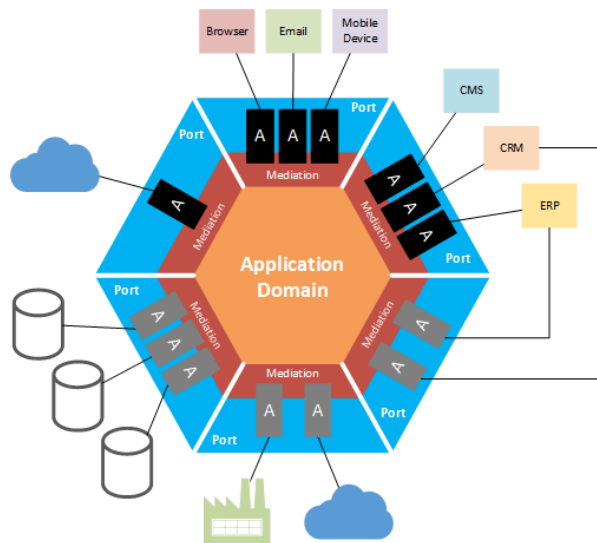


Райффайзен



Как снизить сложность с DDD

4. Используйте архитектуру, где основой является модель.



Райффайзен



Составные части доменной модели

- Сущность.
- Объект-значение.
- Агрегат.
- Доменное событие.
- Доменный сервис.
- Репозиторий.
- Фабрика.



Сущность

- Индивидуально существующие логические единицы.

Райффайзен



Сущность

- Индивидуально существующие логические единицы.
- Имеет уникальный идентификатор.

Райффайзен



Сущность

- Индивидуально существующие логические единицы.
- Имеет уникальный идентификатор.
- Идентификатор не изменяется после создания сущности.



Сущность

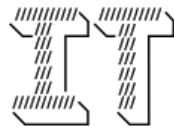
- Индивидуально существующие логические единицы.
- Имеет уникальный идентификатор.
- Идентификатор не изменяется после создания сущности.
- Не зависит от совокупности своих атрибутов, может изменяться с течением времени.



Сущность Courier

```
public class Courier
{
    public int Id { get; set; }
    public Request Request { get; set; }
    ...
}
```

Райффайзен

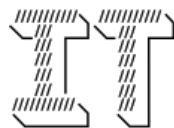


Сущность Courier

```
public class Courier : IAggregateRoot, IEventProvider
{
    private IEventSender _eventSender;
    public int Id { get; private set; }
    public CourierRequest Request { get; private set; }
    ...
    public void TakeRequest(CourierRequest request, IRequestService requestService)
    {
        ValidateCanTake(request, requestService);
        Request = request;
        _eventSender.RaiseEvent(new RequestHasBeenTakenEvent(Id, request));
    }

    public static Courier Create(int id, CourierRequest request, IEventSender sender)
    {
        var courier = new Courier { Id = id, Request = request };
        courier.SetSender(sender);
        return courier;
    }
}
```

Райффайзен



Объект-значение

- Объект без собственной идентичности.

Объект-значение

- Объект без собственной идентичности.
- Полностью определяется набором своих атрибутов.

Райффайзен



Объект-значение

- Объект без собственной идентичности.
- Полностью определяется набором своих атрибутов.
- Неизменяем (когда позволяют ресурсы).

Объект-значение

- Объект без собственной идентичности.
- Полностью определяется набором своих атрибутов.
- Неизменяем (когда позволяют ресурсы).
- Заменяем.

Объекты-значения

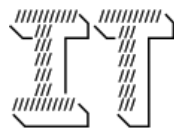
```
public class CourierRequest
{
    public RequestIdentity ExternalId { get; private set; }
    public Address To { get; private set; }
    public decimal Price { get; private set; }

    public static CourierRequest CreateRequest(int extId,
                                              Address addr,
                                              decimal price)
    {
        return new CourierRequest
        {
            ExternalId = new RequestIdentity(extId),
            To = address,
            Price = price
        };
    }
}
```

```
public class RequestIdentity
{
    public RequestIdentity(int id)
    {
        Identity = id;
    }

    public int Identity { get; }
}
```

Райффайзен



Доменные сервисы

- Представляют не предметы, а процессы.

Доменные сервисы

- Представляют не предметы, а процессы.
- Операции в ней неудобно помещать в сущности или объекты-значения.



Доменные сервисы

- Представляют не предметы, а процессы.
- Операции в ней неудобно помещать в сущности или объекты-значения.
- Интерфейс службы определен через другие объекты предметной области.

Доменные сервисы

- Представляют не предметы, а процессы.
- Операции в ней неудобно помещать в сущности или объекты-значения.
- Интерфейс службы определен через другие объекты предметной области.
- Службы не имеют состояния.

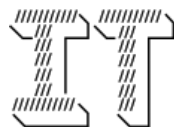
Вовсе не доменный сервис

```
public class RequestsService
{
    public void BindRequestToCourier(int requestId, int courierId)
    {
        var demand = _demandsRepository.GetDemand(requestId);
        var courier = _couriersRepository.GetCourier(courierId);
        ValidateCanConnect(demand, courier);

        demand.Courier = courier;
        demand.Status = RequestStatus.Bound;
        demand.HasError = false;
        demand.Error = string.Empty;
        demand.ErrorCode = ErrorCode.NoError;

        courier.Request = request;
        if(demand.Appropriatable && demand.RequestType == RequestType.VIP)
            _emailNotifier.SendRequestHasBound(demand);
    }
}
```

Райффайзен



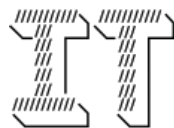
Хороший доменный сервис

```
internal class CourierRequestsService : ICourierRequestsService
{
    private readonly IRequestService _requestService;
    private readonly ICourierRepository _courierRepository;

    public void TakeRequest(int requestId, int courierId)
    {
        var request = _requestService.GetRequest(requestId);
        var courier = _courierRepository.Get(courierId);
        ValidateCanTake(request, courier);
        courier.TakeRequest(request, _requestService);
    }

    private void ValidateCanTake(CourierRequest request, Courier courier)
    {
        var couriers = _courierRepository.Matches(CourierSpecifications.Request(request));
        if (couriers.Any(x => x.Id != courier.Id))
            throw new ValidationException("Request Id is already used");
    }
}
```

Райффайзен



Фабрики

- Порождение сложных сущностей и агрегатов следует поручать фабрикам

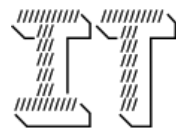
Фабрики

- Порождение сложных сущностей и агрегатов следует поручать фабрикам
- Фабрики скрывают подробности создания сущностей

Фабрики

- Порождение сложных сущностей и агрегатов следует поручать фабрикам
- Фабрики скрывают подробности создания сущностей
- В качестве технической реализации можно взять любой порождающий паттерн GoF.

Райффайзен



Фабрики

```
public class Courier : IAggregateRoot, IEventProvider
{
    ...
    public static Courier Create(int id,
                                CourierRequest request,
                                IEventSender sender)
    {
        var courier = new Courier { Id = id, Request = request };
        courier.SetSender(sender);
        return courier;
    }
}
```

Райффайзен



Репозитории

- Репозиторий представляет все объекты определенного типа в виде концептуального множества.

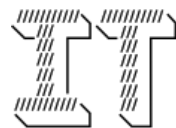
Райффайзен



Репозитории

- Репозиторий представляет все объекты определенного типа в виде концептуального множества.
- Репозиторий обеспечивает запись и восстановление сущностей и агрегатов.

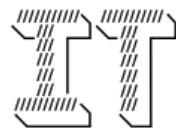
Райффайзен



Репозитории

- Репозиторий представляет все объекты определенного типа в виде концептуального множества.
- Репозиторий обеспечивает запись и восстановление сущностей и агрегатов.
- Клиенты получают нужные данные из репозитория, используя запросы.

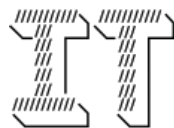
Райффайзен



Репозитории

```
var courier = _courierRepository.Get(courierId);  
...  
var couriers =  
_courierRepository.Matches(CourierSpecifications.Request(request));
```

Райффайзен



События

- События являются реакцией системы на изменения.

События

- События являются реакцией системы на изменения.
- События позволяют запускать несколько процессов обработки.

События

- События являются реакцией системы на изменения.
- События позволяют запускать несколько процессов обработки.
- События дают возможность строить системы из независимых компонентов.



События

- События являются реакцией системы на изменения.
- События позволяют запускать несколько процессов обработки.
- События дают возможность строить системы из независимых компонентов.
- Идеально подходят в случае асинхронного взаимодействия.



Производство событий

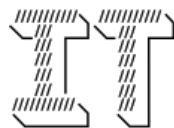
```
public class Courier : IAggregateRoot, IEventProvider
{
    private IEventSender _eventSender;

    ...

    public void TakeRequest(CourierRequest request, IRequestService requestService)
    {
        ...
        _eventSender.RaiseEvent(new RequestHasBeenTakenEvent(Id, request));
    }

    public static Courier Create(int id, CourierRequest request, IEventSender sender)
    {
        var courier = new Courier { Id = id, Request = request };
        courier.SetSender(sender);
        return courier;
    }
}
```

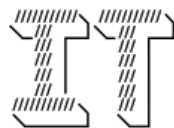
Райффайзен



Обработка событий

```
internal class RequestHasBeenTakenEventHandler :  
    ICanHandleEvent<RequestHasBeenTakenEvent>  
{  
    ...  
    public async void Handle(RequestHasBeenTakenEvent message)  
    {  
        var response = await _client.PostAsync($"/api/requests/"+  
            $"{message.Request.ExternalId}/courier/{message.CourierId}");  
  
        if (response.IsNotSuccess())  
        {  
            Logger.LogError($"Request with id {message.Request.ExternalId}"+  
                "didn't processed correctly");  
        }  
    }  
}
```

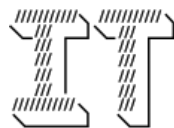
Райффайзен



Простая шина сообщений

```
public class SimpleEventBus : IEventBus
{
    ...
    public void RaiseEvent<TEvent>(TEvent message) where TEvent : class, IEvent
    {
        var list = _listeners;
        foreach (var handler in list.OfType<ICanHandleEvent<TEvent>>())
        {
            if (message.IsAsync)
            {
                var h = handler;
                _asyncEventStrategy.ProcessAsync(h, message);
            }
            else
            {
                handler.Handle(message);
            }
        }
    }
}
```

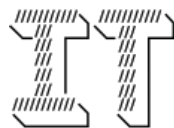
Райффайзен



Контроллер заявок

```
[RoutePrefix("api/requests")]
public class RequestsController : ApiController
{
    private readonly IEventBus _eventBus;
    ...
    [HttpPost]
    [Route("/{extId}/courier/{courierId}")]
    public IHttpActionResult MarkRequestAsTakenByCourier(int requestId,
                                                         int courierId)
    {
        _eventBus.SendCommand(
            new MarkRequestAsTakenByCourierCommand(requestId, courierId));
    }
}
```

Райффайзен



Команда

```
internal class MarkRequestAsTakenByCourierCommand : ICommand
{
    public MarkRequestAsTakenByCourierCommand(int requestId,
                                                int courierId)
    {
        RequestId = requestId;
        CourierId = courierId;
    }

    public int RequestId { get; }
    public int CourierId { get; }
}
```

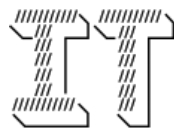
Райффайзен



Обработчик команды

```
internal class MarkRequestAsTakenByCourierCommandHandler :  
    ICanHandleCommand<MarkRequestAsTakenByCourierCommand>  
{  
    private readonly IRequestRepository _requestRepository;  
    private readonly IEmailNotifier _emailNotifier;  
  
    public void Handle(MarkRequestAsTakenByCourierCommand message)  
    {  
        var request = _requestRepository.Get(message.RequestId);  
        request.TakenByCourier(message.CourierId, _emailNotifier);  
    }  
}
```

Райффайзен



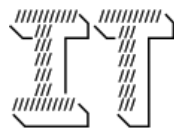
Сущность «Заявка»

```
public class Request : IAggregateRoot
{
    public int Id { get; set; }
    public CourierId Courier { get; set; }
    public RequestStatus Status { get; set; }
    public Error Error { get; set; }
    ...
    public void TakenByCourier(int courierId, IClientNotifier notifier)
    {
        Courier = new CourierId(courierId);
        Status = RequestStatus.Bound;

        Error = ErrorClass.CreateEmpty();

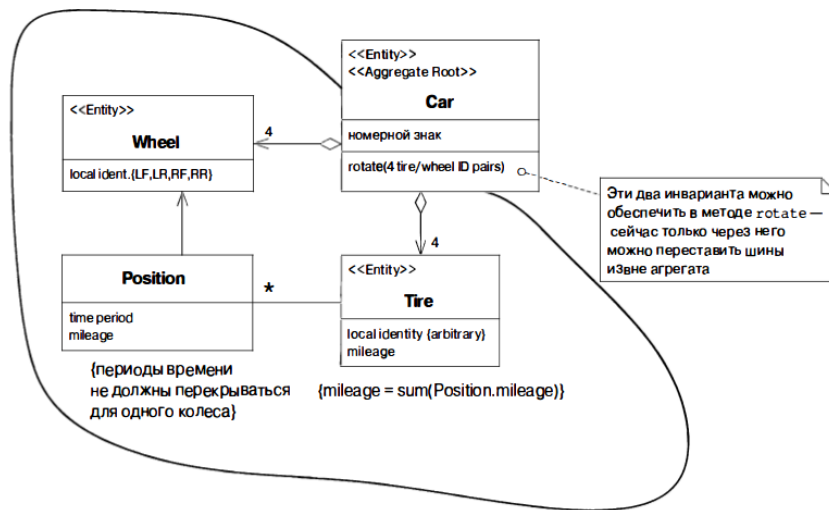
        if (ShouldNotify())
        {
            notifier.NotifyRequestHasTaken(this);
        }
    }
}
```

Райффайзен



Как снизить сложность с DDD


5. Стройте доменную логику на основе тактических шаблонов DDD.



Райффайзен



Резюме

- Унаследованный код может серьезно мешать развитию приложения.
- DDD хорошо подходит для уменьшения сложности в приложениях с запутанной бизнес-логикой.
- $MDD + UL =$ 
- Архитектура должна иметь в основе доменную модель.
- Тактические шаблоны DDD позволяют строить понятные модели в коде.

Райффайзен



Спасибо!

<http://domaindrivendesign.org>

<http://udidahan.com/>

<https://goodenoughsoftware.net/>

<https://lostechies.com/jimmybogard/>

<https://msdn.microsoft.com/ru-ru/magazine/dd419654.aspx>

Райффайзен

