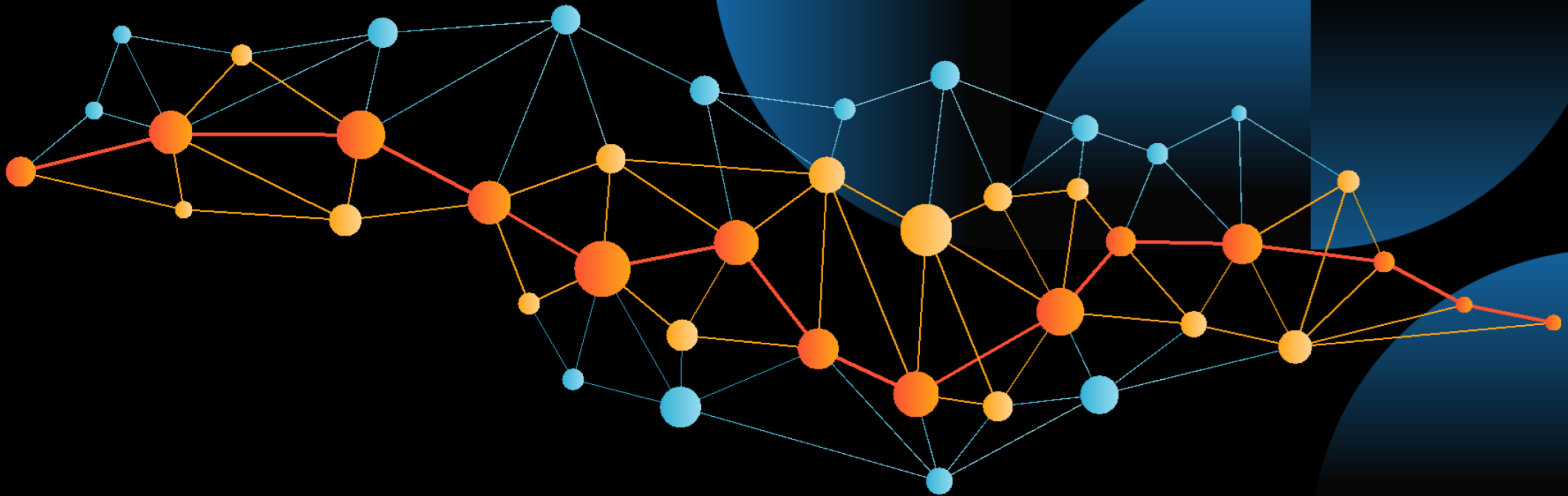
.NET DEVELOPER DAYS

CodeDesign



Florin Coroș

Designing a Distributed System for Long-Term Development

CodeDesign

Florin Coroș

Software Architect Consultant

Technical Trainer

Founder of Code Design

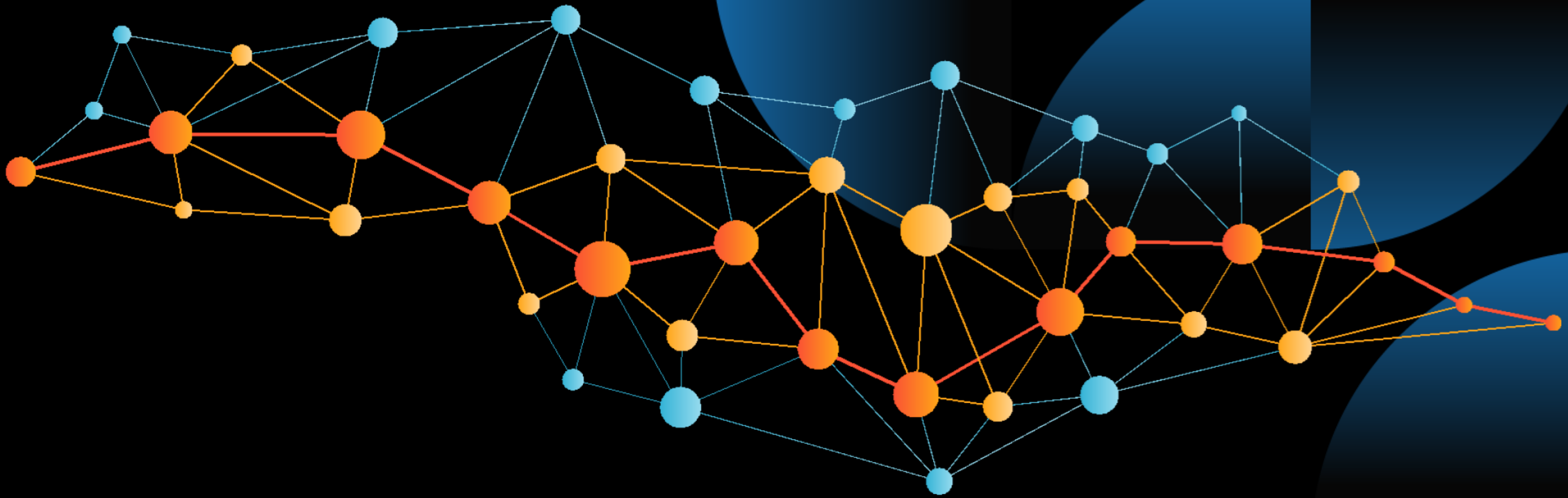
enjoing playing GO

enjoing traveling

onCodeDesign.com/DevDaysPL

.NET DEVELOPER DAYS

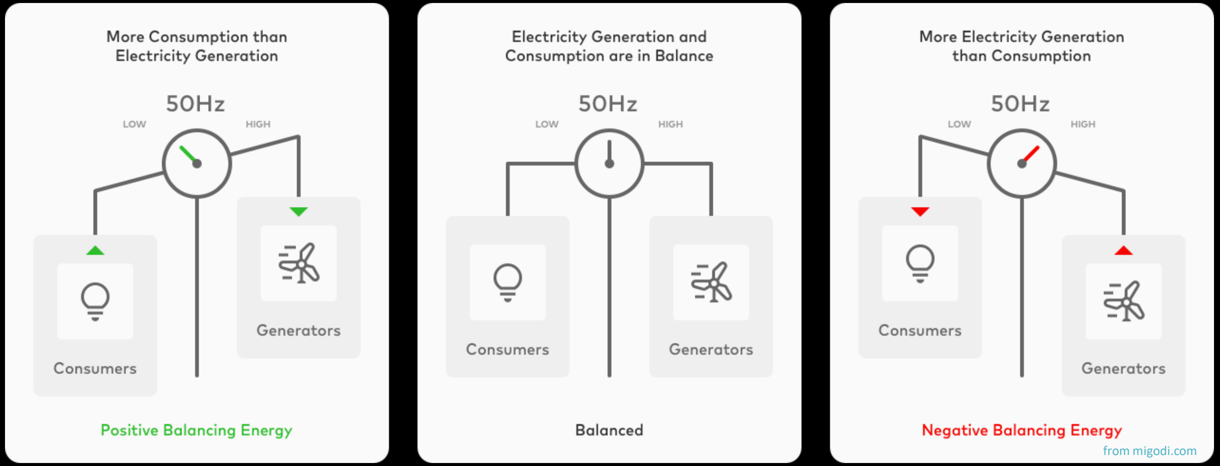
CODEDESIGN



Florin Coroș

Designing a Distributed System for Long-Term Development

Context: Grid Balancing and Energy Trading


More Consumption than Electricity Generation
50Hz
LOW
HIGH
Consumers
Generators
Positive Balancing Energy
Electricity Generation and Consumption are in Balance
50Hz
LOW
HIGH
Consumers
Generators
Balanced
More Electricity Generation than Consumption
50Hz
LOW
HIGH
Consumers
Generators
Negative Balancing Energy
from migodi.com

Balancing the Grid

Transmission System Operators (TSOs) and Balance Responsible Partners have the critical task of maintaining balance in the power grid. This means balancing supply and demand every second of every day. Measured in Hertz (50hz in Europe), maintaining balance is crucial as significant deviations can lead to power outages and resulting damages to society and infrastructure

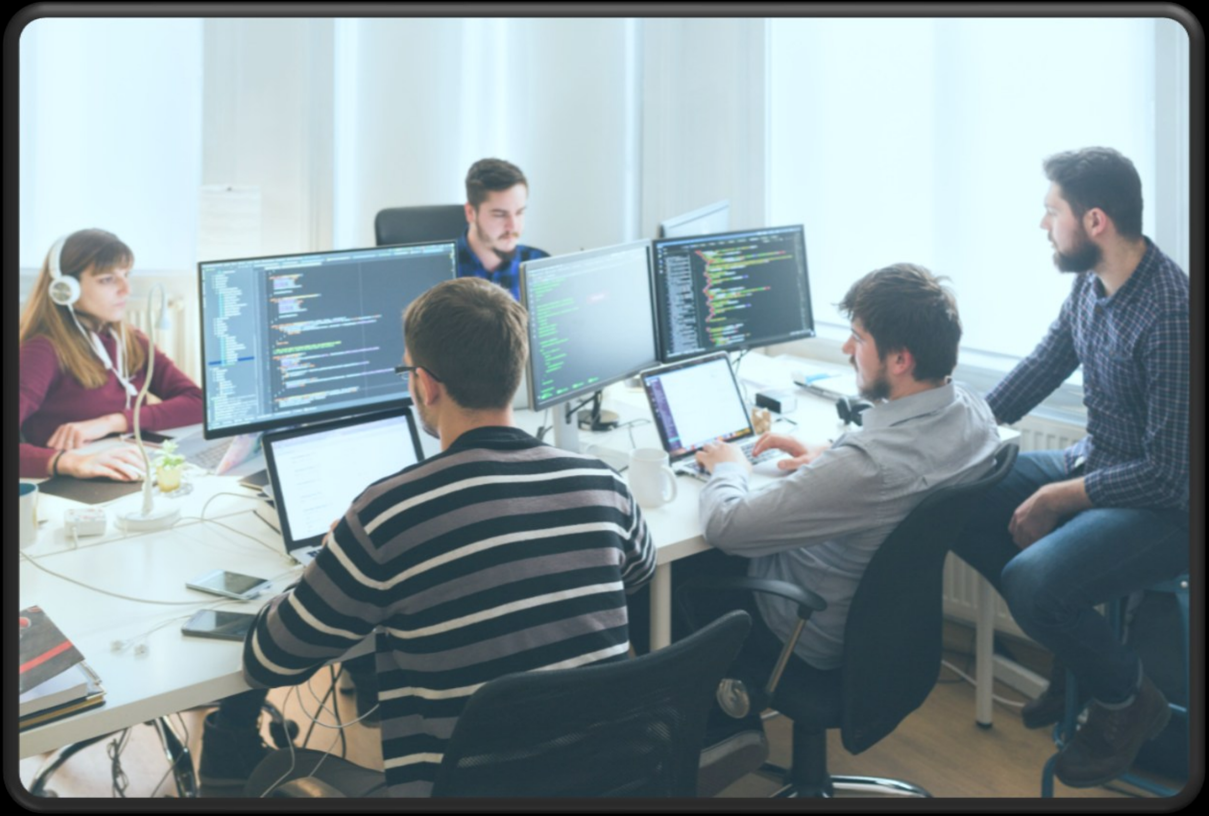
- → Be Resilient, Reliability, High Availability, No Data Loss
- → Security
- → Deploy in any Public Cloud and in On-Prem Data Centres
- → Granular Deployments
- → …. ….. …. …..

Long Term Development

- ➢ 10 to 18 months to release the 1st version in Prod
- ➢ > 3 years of actively development to "feature complete"

- ❖ invest in foundation vs deliver features
- ❖ team volatility & team scale-up
- ❖ adapt to changes in external systems APIs

Team Scaleup + Volatility



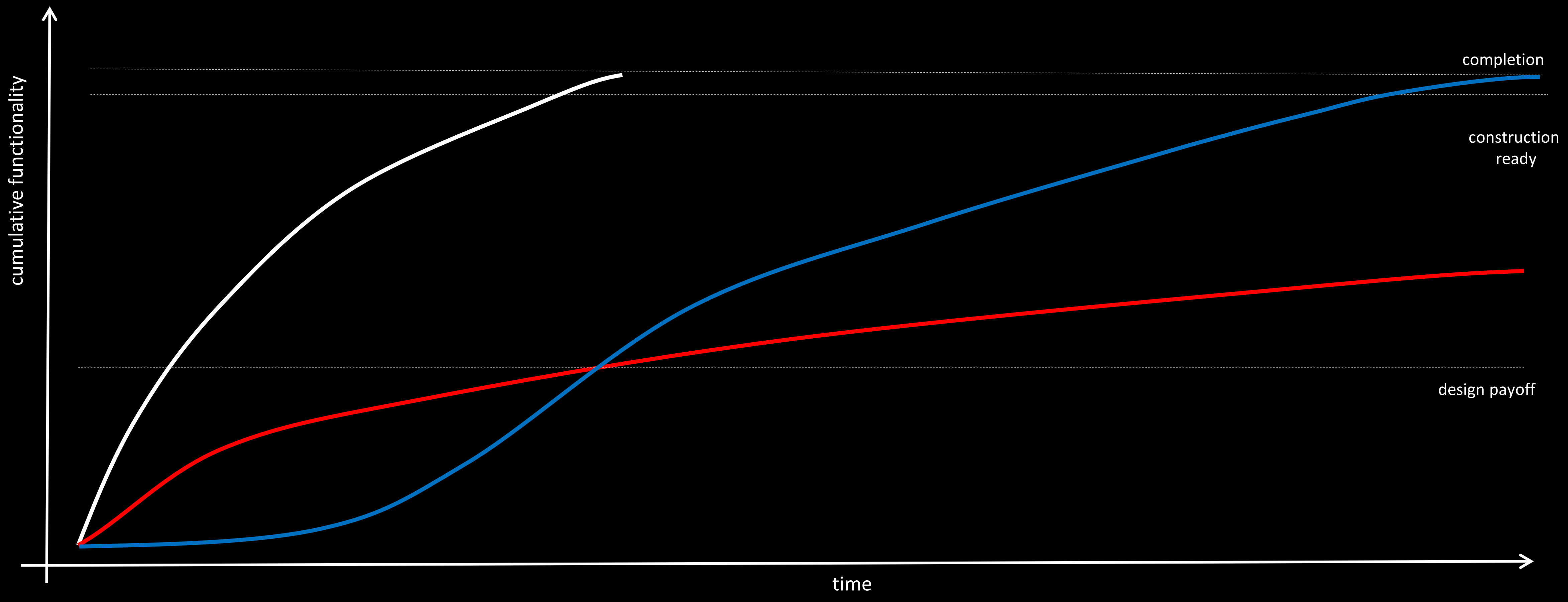
❖Scale up the Team

- grow from ~2 – 3 developers to 12+

❖Team Volatility

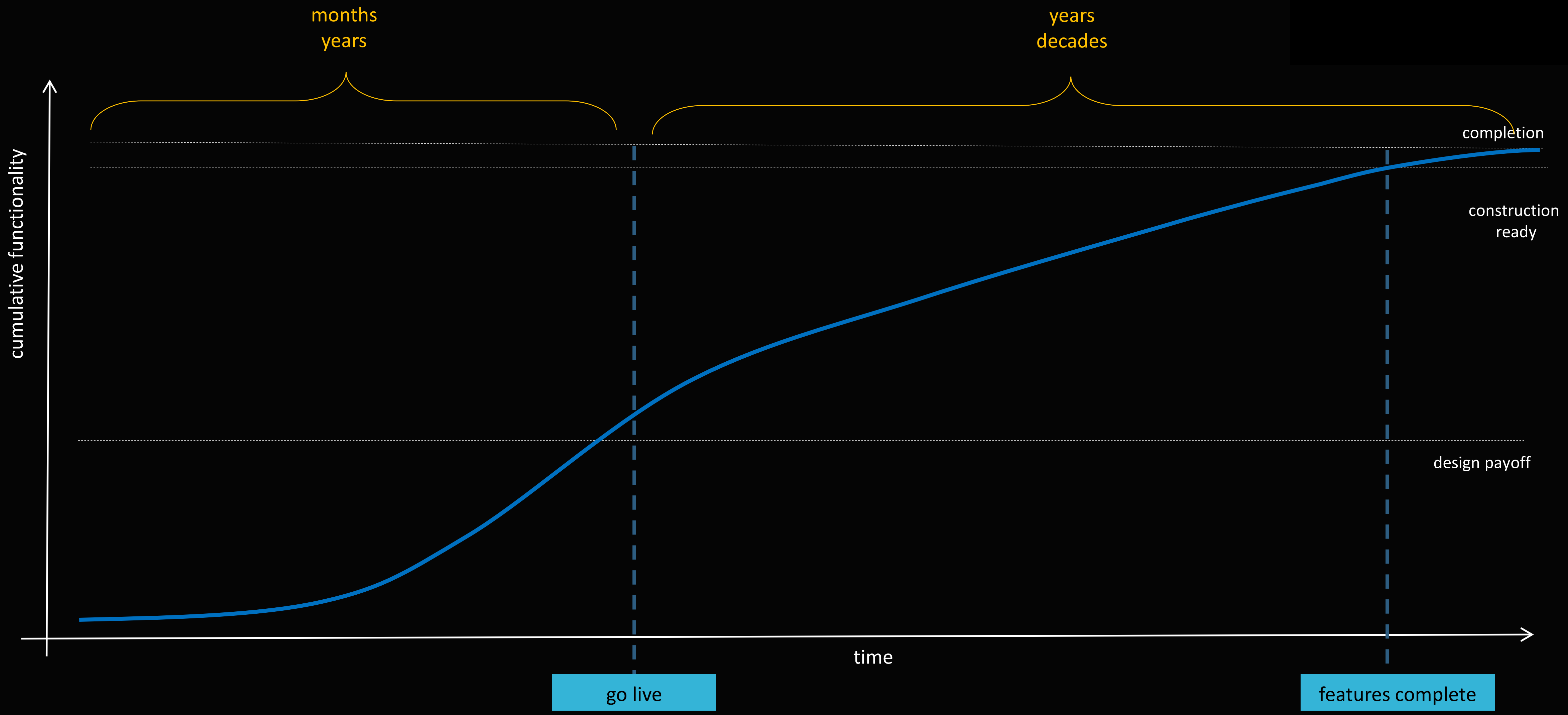
- people leaving and joining the team

Invest in Design. Build a Foundation, a Framework

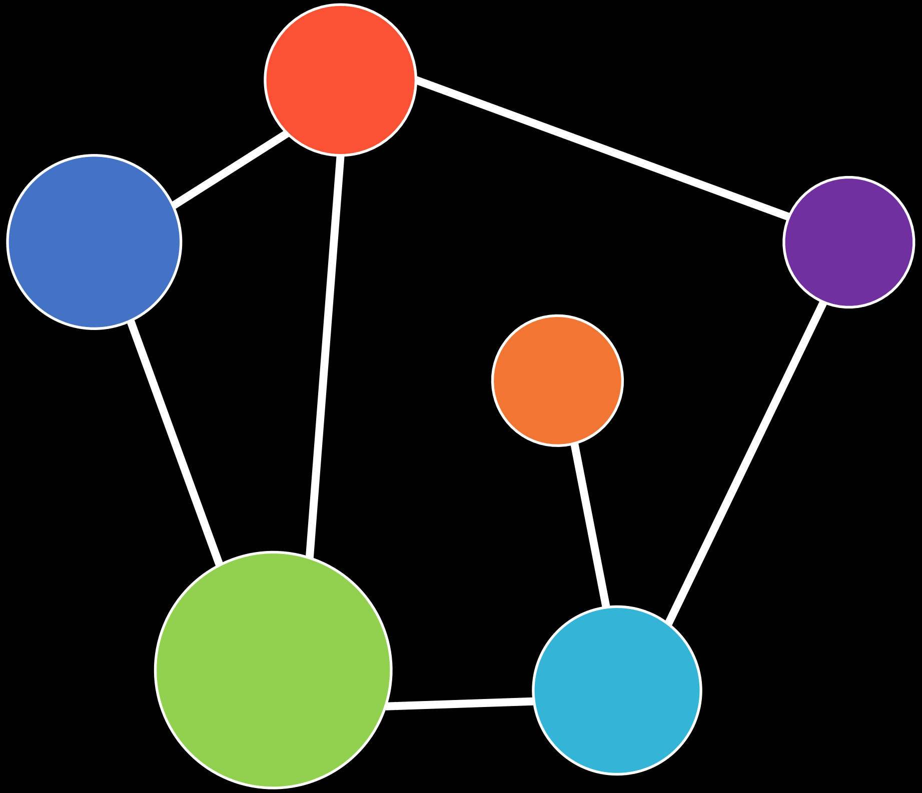


http://martinfowler.com/bliki/DesignStaminaHypothesis.html
http://tinyurl.com/DesignPayoff

Invest in Design. Build a Foundation, a Framework



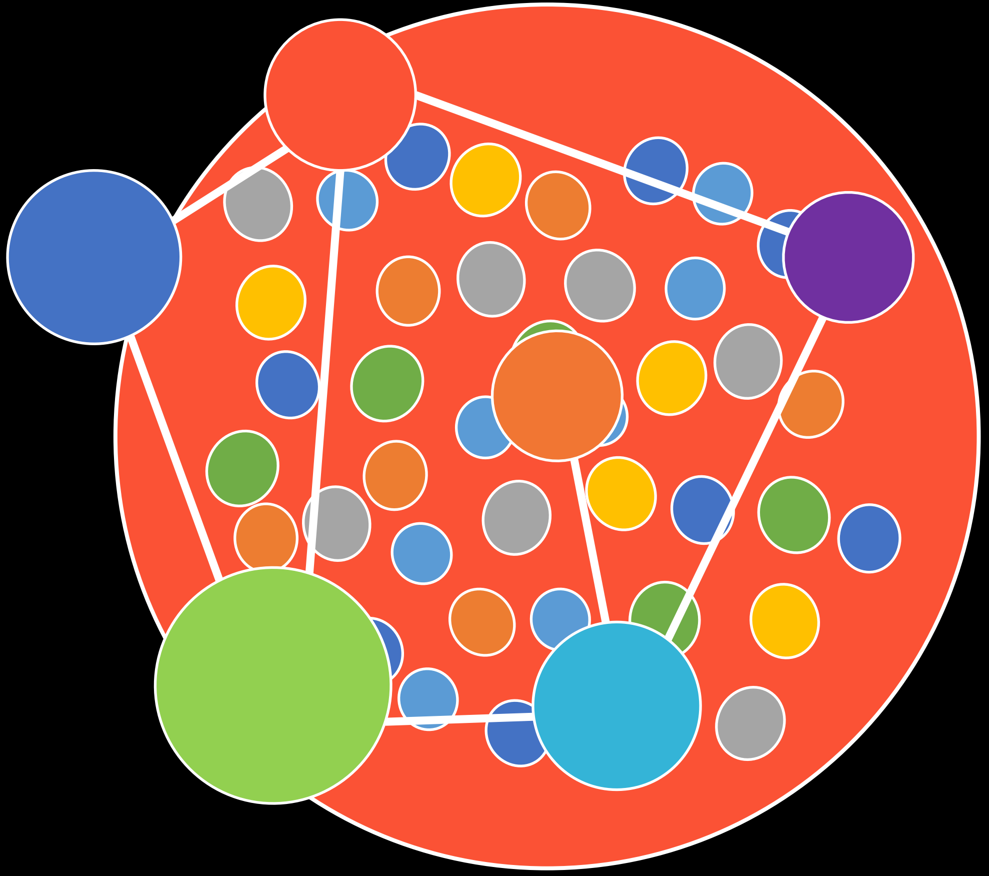
Modular System - Concept

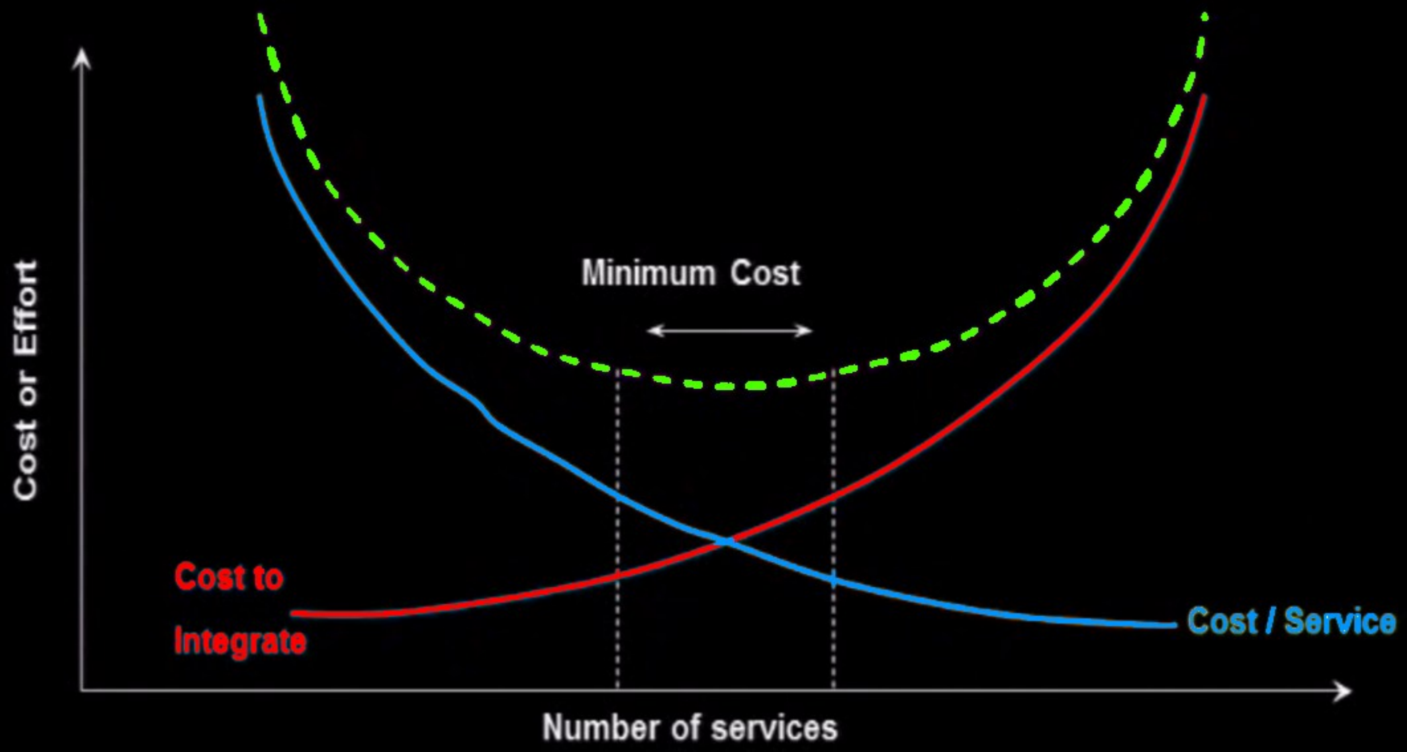


- ➢Maintainability
- ➢Extensibility
- ➢Reusability

CHANGE PREDICTABILITY

How many services?





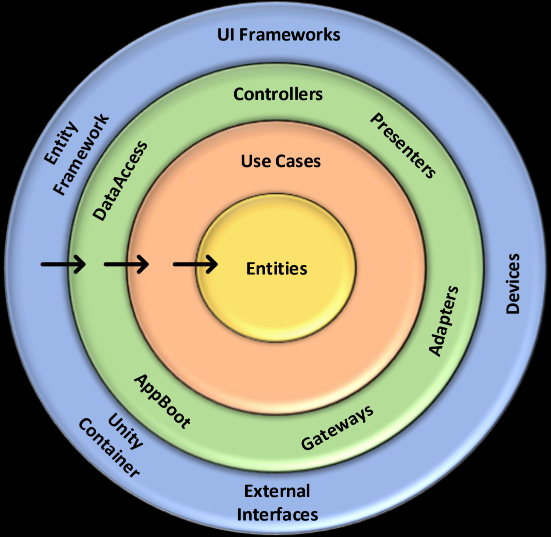
from *Righting Software* by Juval Lowy

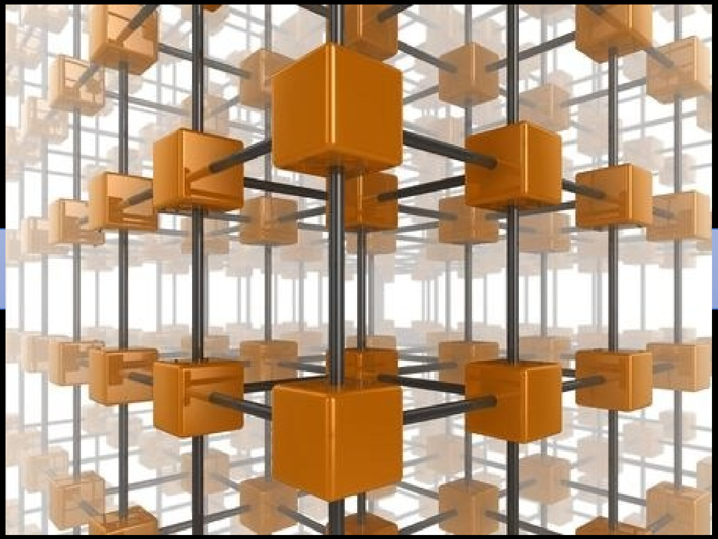
Contracts – Are Key in Modular Systems

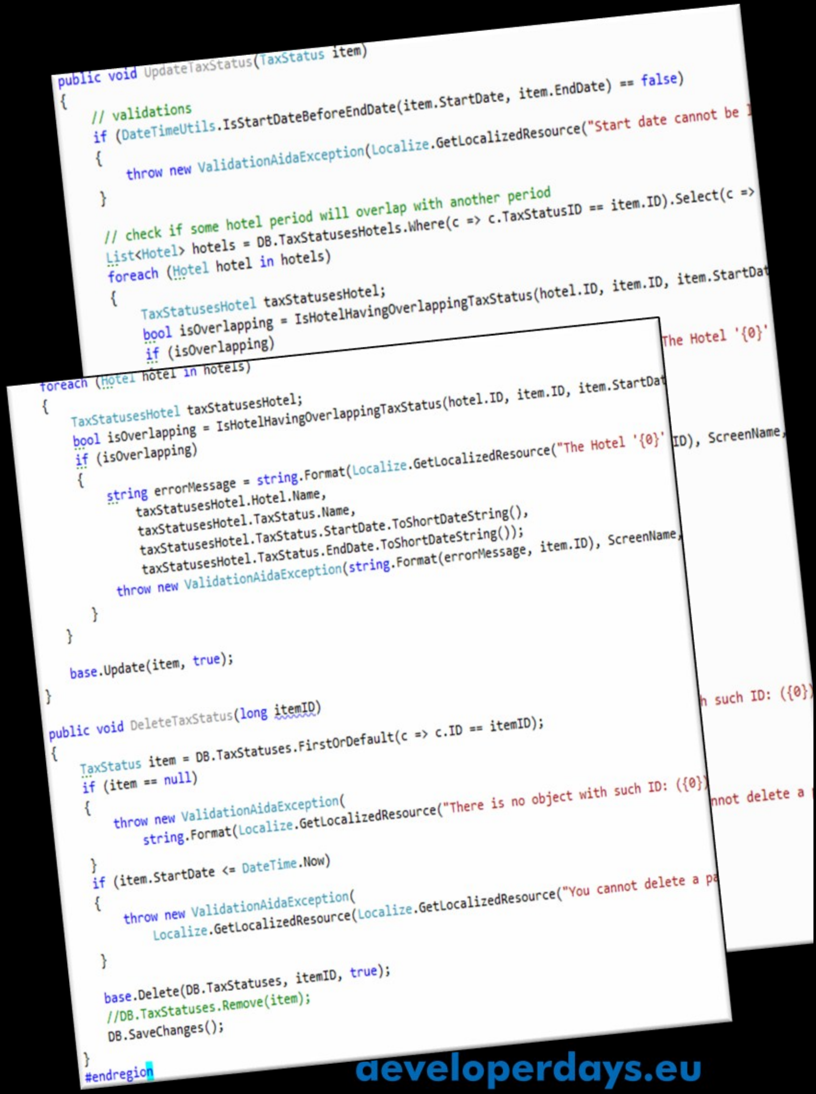
- Services communicate through **Explicit Contracts**
  - **Abstract** the functions it provides
  - **Encapsulate** (hide) the implementation details

- Contracts described with language constructs:
  - Operation Contracts – functions the interfaces
  - Data Contracts – DTOs (the in/out params)
  - Fault Contracts - Exceptions

Structure that Supports the Architecture



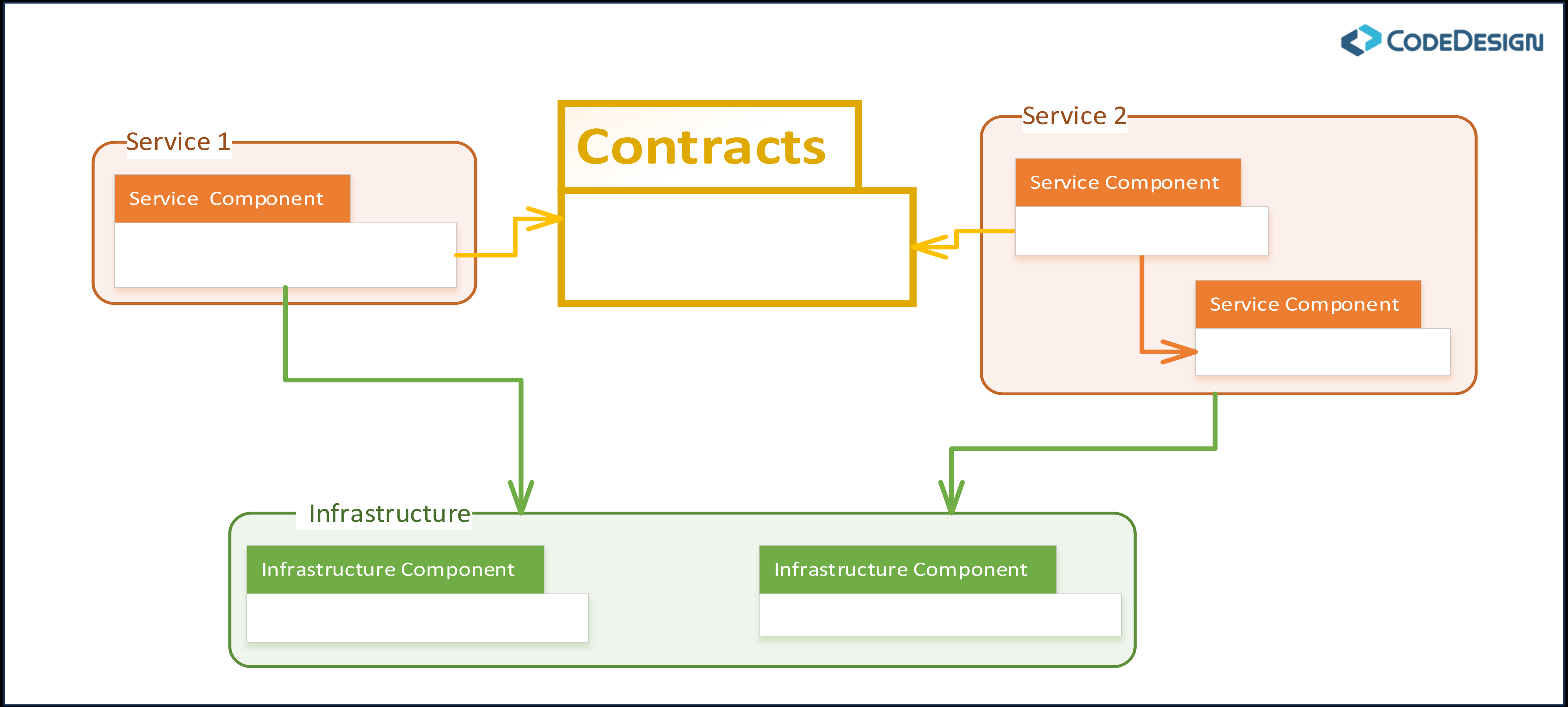




```
public void UpdateTaxStatus(TaxStatus item)
{
    // validations
    if (DateTimeUtils.IsStartDateBeforeEndDate(item.StartDate, item.EndDate) == false)
    {
        throw new ValidationAidaException(Localize.GetLocalizedResource("Start date cannot be 
    }

    // check if some hotel period will overlap with another period
    List<Hotel> hotels = DB.TaxStatusesHotels.Where(c => c.TaxStatusID == item.ID).Select(c =>
    foreach (Hotel hotel in hotels)
    {
        TaxStatusesHotel taxStatusesHotel;
        bool isOverlapping = IsHotelHavingOverlappingTaxStatus(hotel.ID, item.ID, item.StartDat
        if (isOverlapping)
```

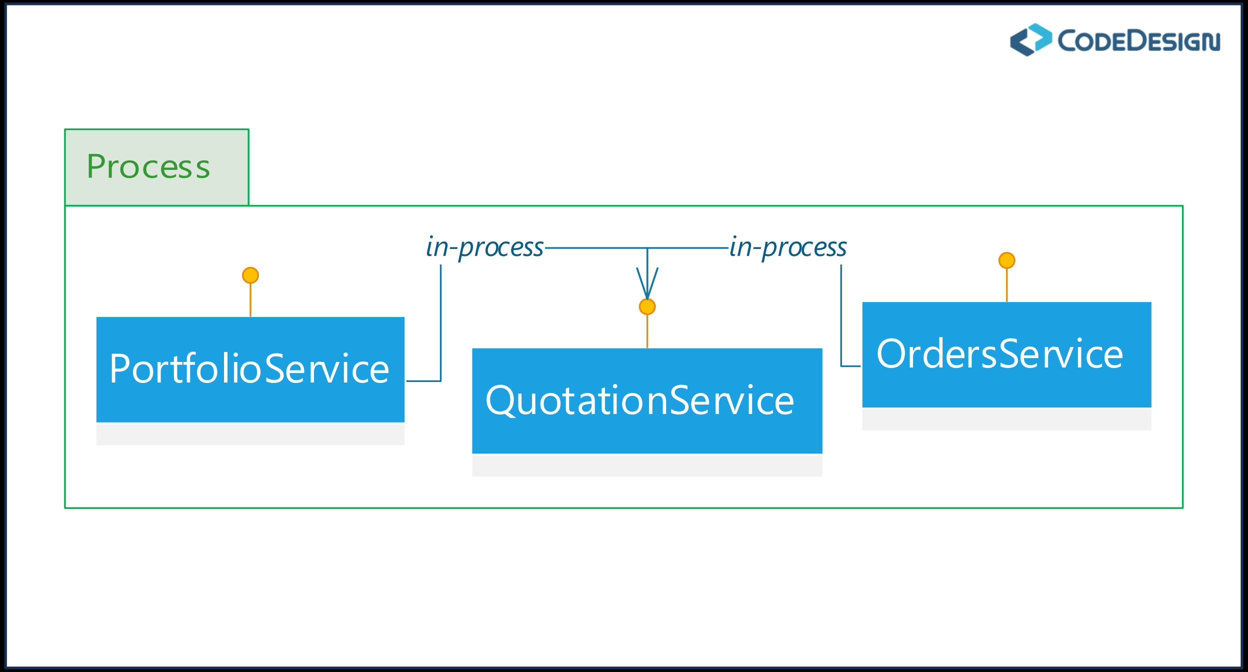
```
foreach (Hotel hotel in hotels)
{
    TaxStatusesHotel taxStatusesHotel;
    bool isOverlapping = IsHotelHavingOverlappingTaxStatus(hotel.ID, item.ID, item.StartDat
    if (isOverlapping)
    {
        string errorMessage = string.Format(Localize.GetLocalizedResource("The Hotel '{0}'
            taxStatusesHotel.Hotel.Name,
            taxStatusesHotel.TaxStatus.Name,
            taxStatusesHotel.TaxStatus.StartDate.ToShortDateString(),
            taxStatusesHotel.TaxStatus.EndDate.ToShortDateString());
        throw new ValidationAidaException(string.Format(errorMessage, item.ID), ScreenName,
    }
}

base.Update(item, true);
}

public void DeleteTaxStatus(long itemID)
{
    TaxStatus item = DB.TaxStatuses.FirstOrDefault(c => c.ID == itemID);
    if (item == null)
    {
        throw new ValidationAidaException(
            string.Format(Localize.GetLocalizedResource("There is no object with such ID: ({0}
    }
    if (item.StartDate <= DateTime.Now)
    {
        throw new ValidationAidaException(
            Localize.GetLocalizedResource(Localize.GetLocalizedResource("You cannot delete a pa
    }

    base.Delete(DB.TaxStatuses, itemID, true);
    //DB.TaxStatuses.Remove(item);
    DB.SaveChanges();
}
#endregion
```

Structure that Enforces Explicit Communication through Contracts

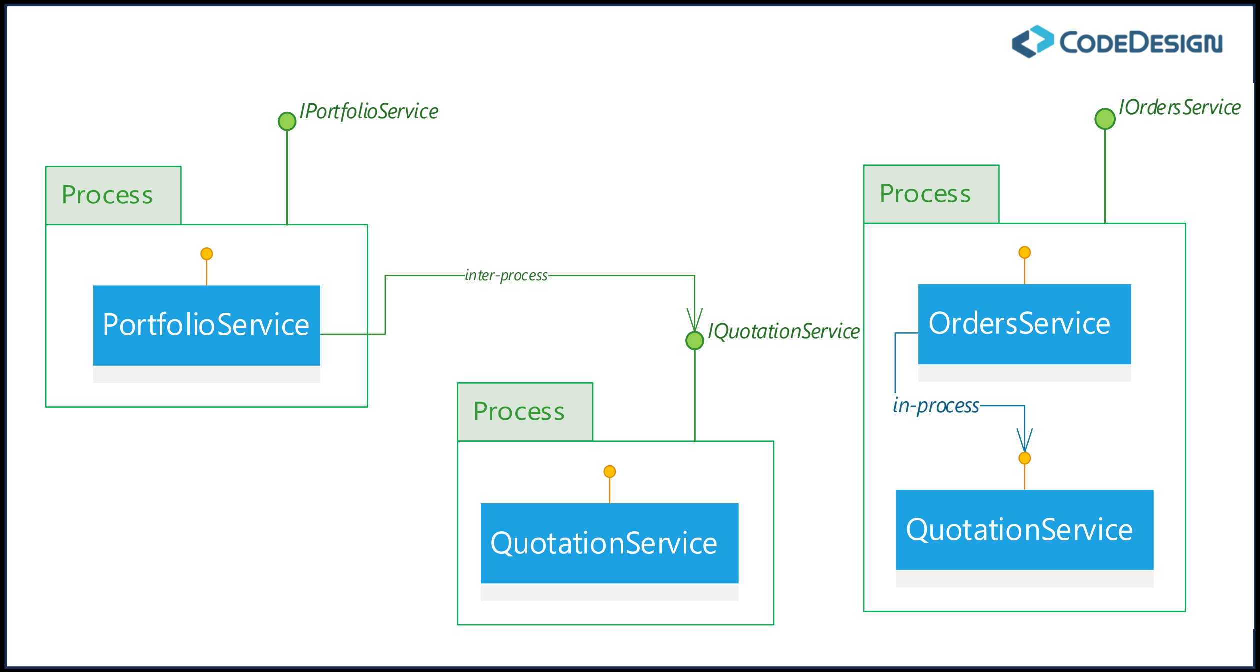


Does it have to be DISTRIBUTED (micro-services)?

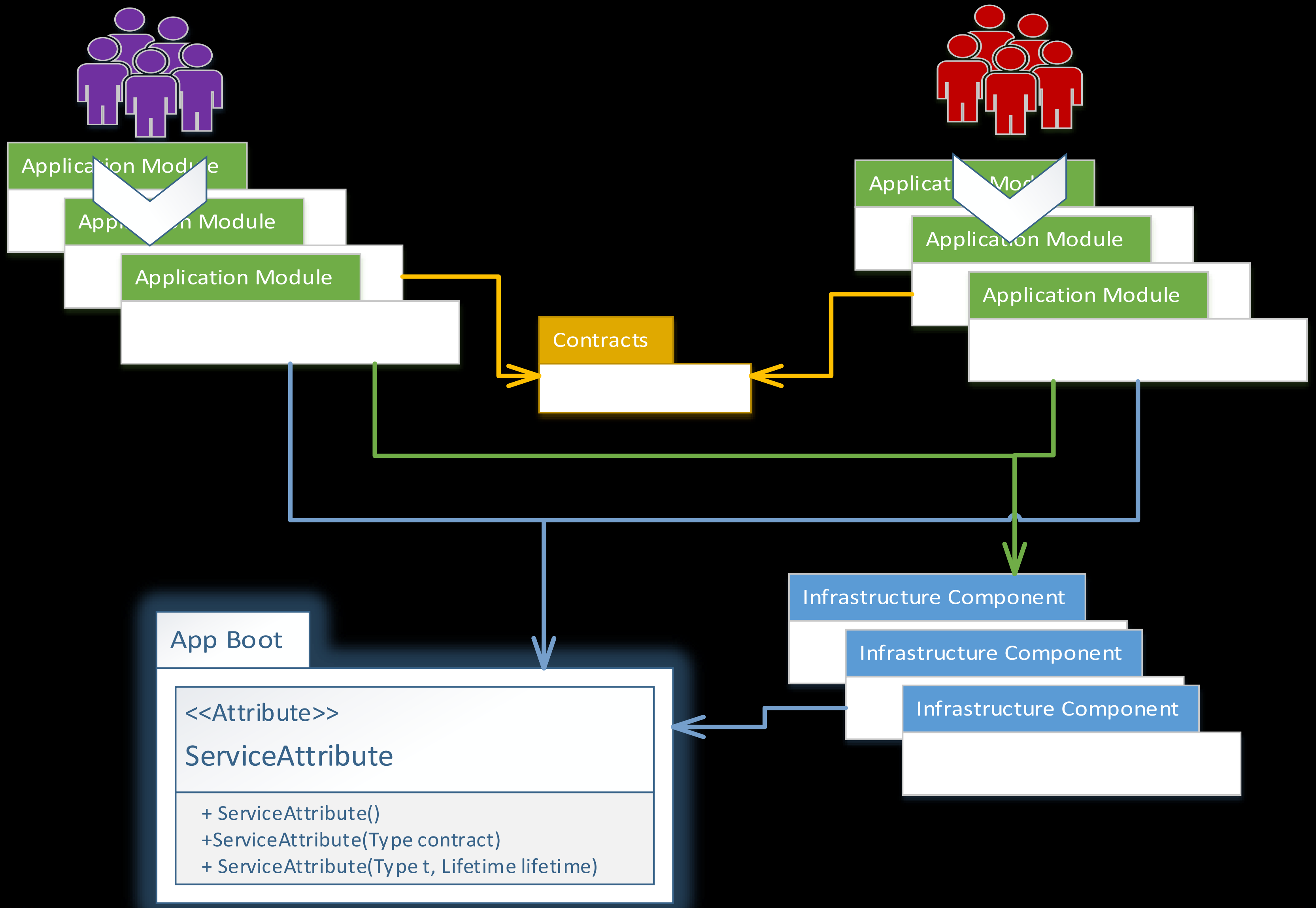
Monolith



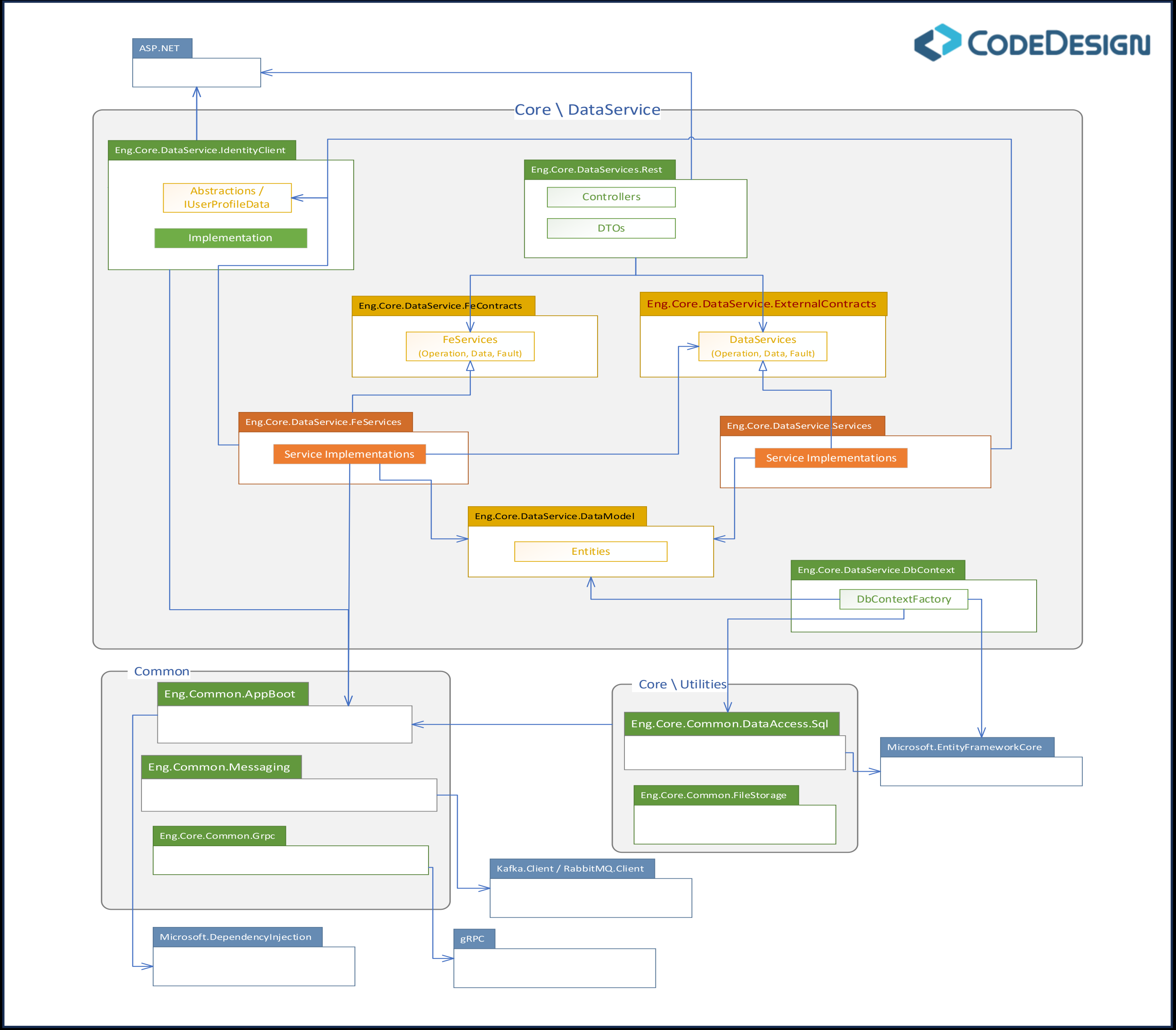
Micro-services



Team Scaleup – Code Ownership



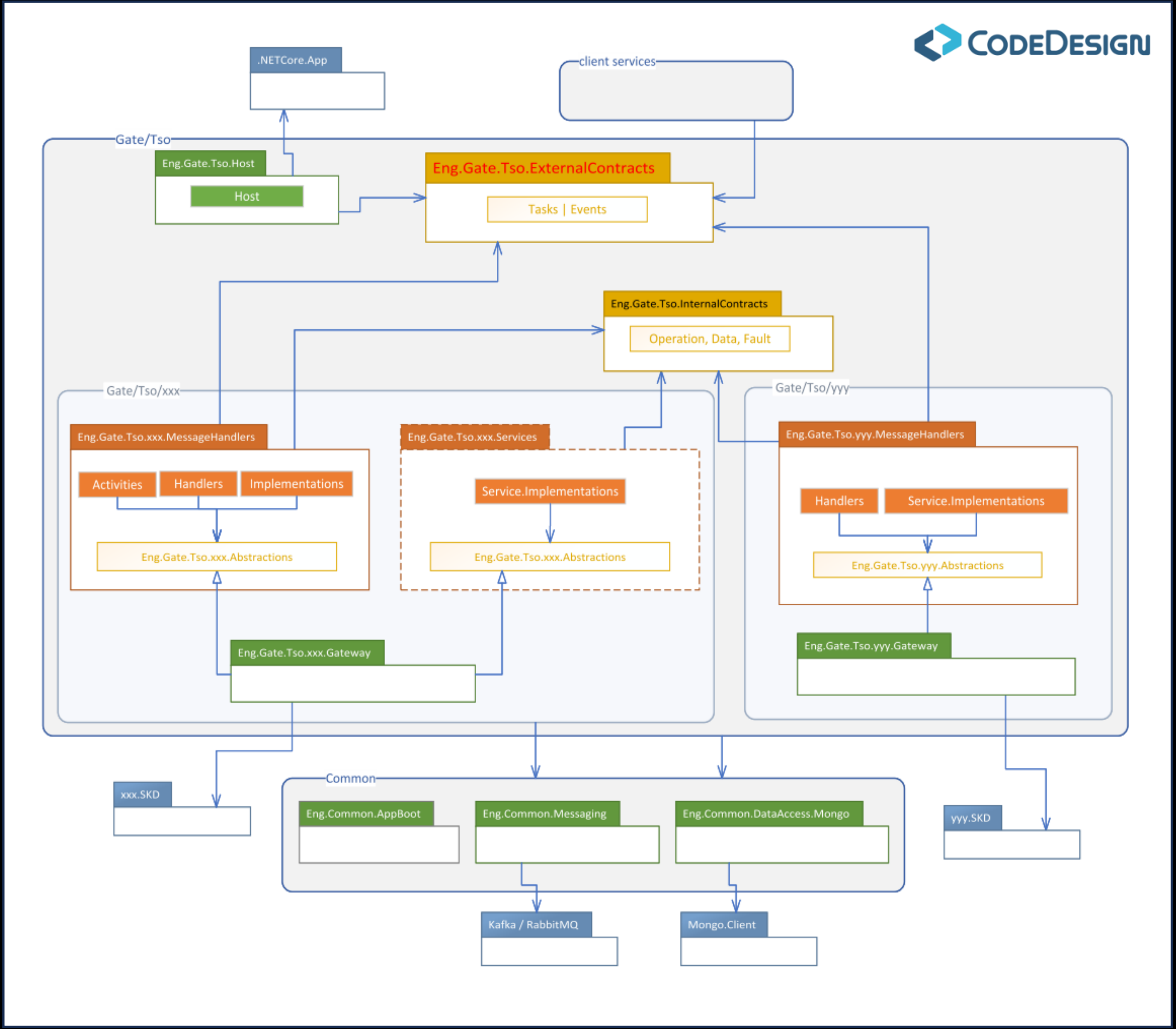
Common Structure and Conventions for ALL Services


CODEDESIGN
ASP.NET
Core \ DataService
Eng.Core.DataService.IdentityClient
Abstractions / IUserProfileData
Implementation
Eng.Core.DataServices.Rest
Controllers
DTOs
Eng.Core.DataService.FeContracts
FeServices (Operation, Data, Fault)
Eng.Core.DataService.ExternalContracts
DataServices (Operation, Data, Fault)
Eng.Core.DataService.FeServices
Service Implementations
Eng.Core.DataService.Services
Service Implementations
Eng.Core.DataService.DataModel
Entities
Eng.Core.DataService.DbContext
DbContextFactory
Common
Eng.Common.AppBoot
Eng.Common.Messaging
Eng.Core.Common.Grpc
Core \ Utilities
Eng.Core.Common.DataAccess.Sql
Eng.Core.Common.FileStorage
Microsoft.EntityFrameworkCore
Kafka.Client / RabbitMQ.Client
Microsoft.DependencyInjection
gRPC

- Separation of CONTRACTS from IMPLEMENTATION
- ExternalContracts by convention
- Clean Architecture principles – colour codes
- Conventions and mappings with folder structure
- Conventions for Build and Deploy
- Infrastructure categories
- Services categories

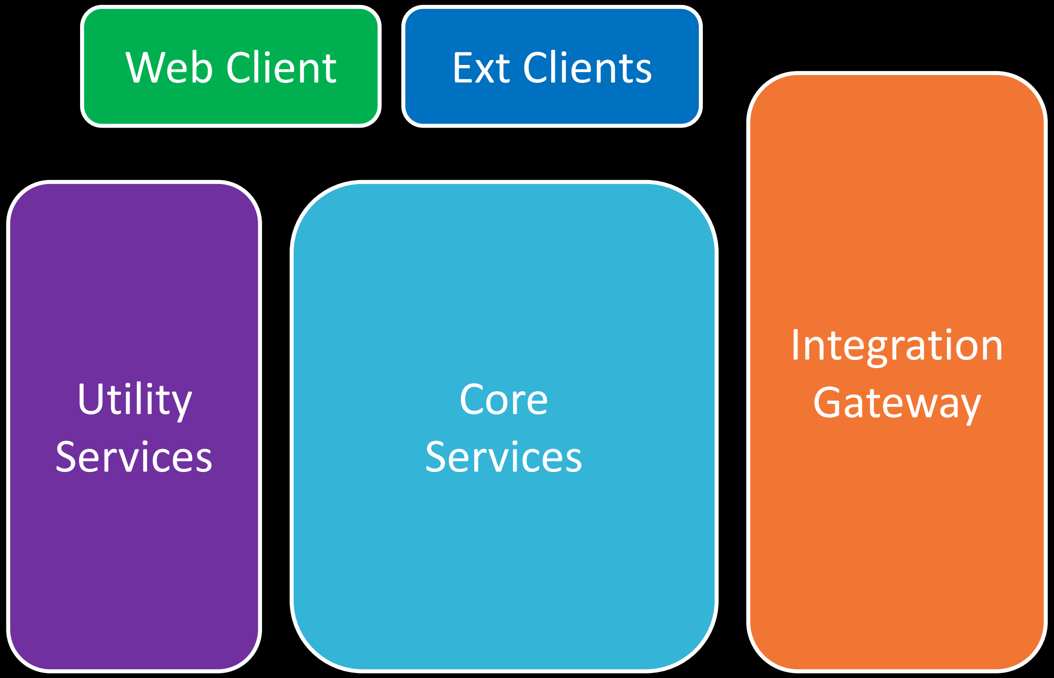
developerdays.eu

Common Structure and Conventions for ALL Services



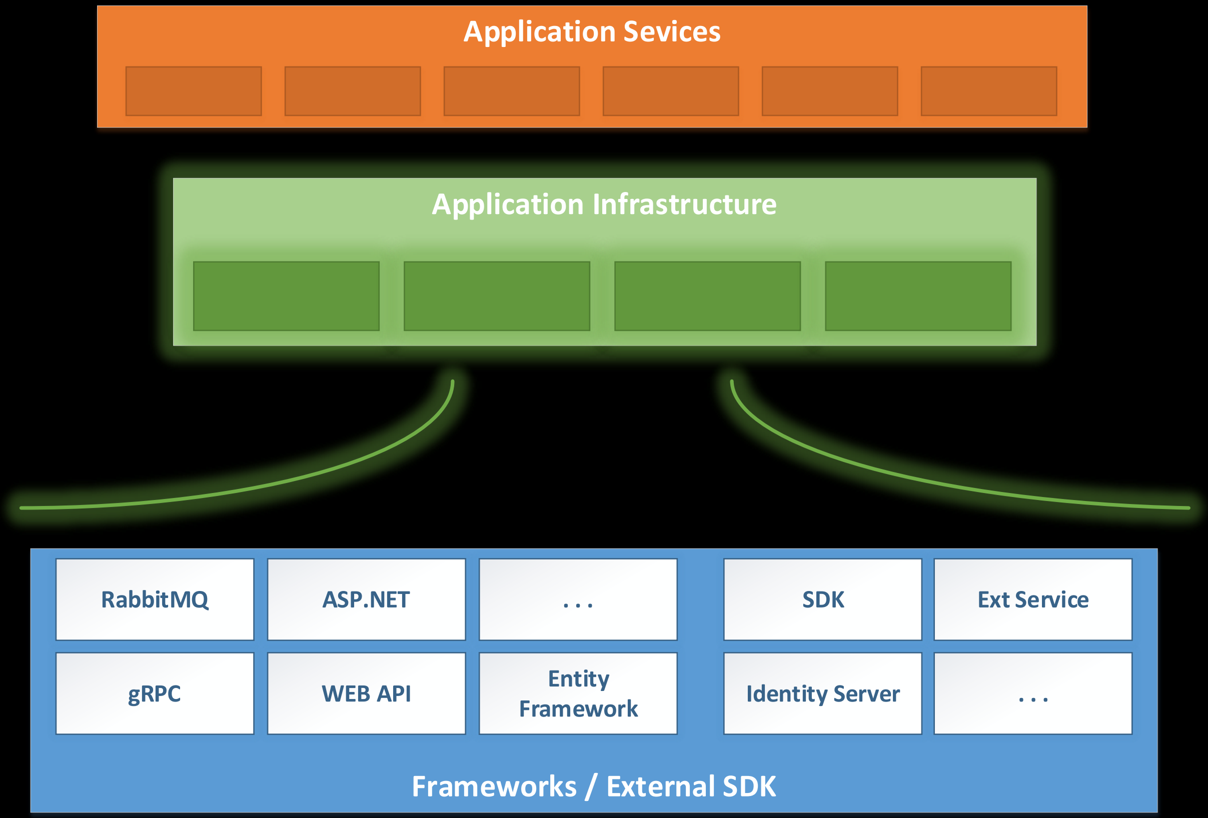
- Separation of CONTRACTS from IMPLEMENTATION
- ExternalContracts by convention
- Clean Architecture principles – colour codes
- Conventions and mappings with folder structure
- Conventions for Build and Deploy
- Infrastructure categories
- Services categories

Categories of Services



- Core Services implement the core behaviour
- Integration Gateway Services communication with External Systems
- Ext Clients provide REST API to customer apps
- Utility Services just utilities that have nothing specific to the business domain

App Infrastructure (Framework) of Tech Components



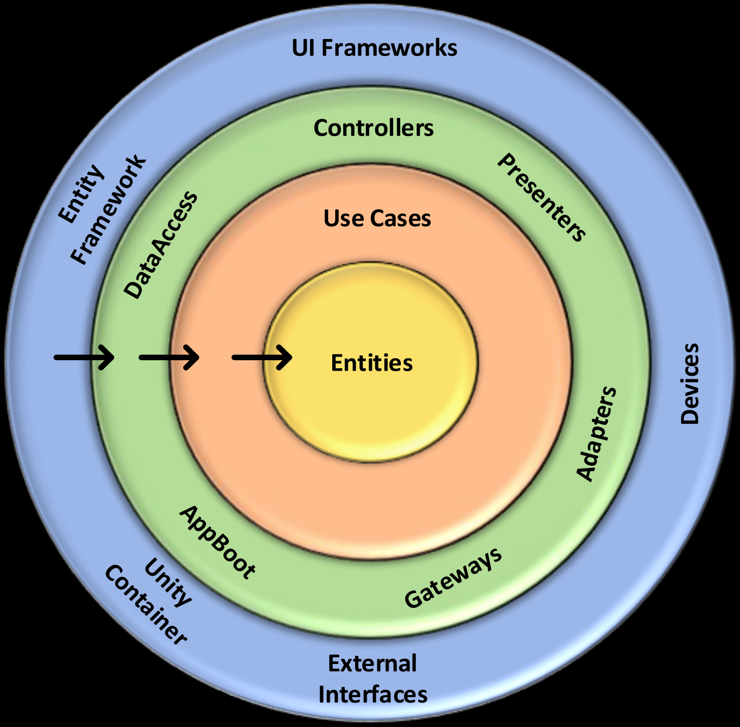
- *do not depend on Frameworks*

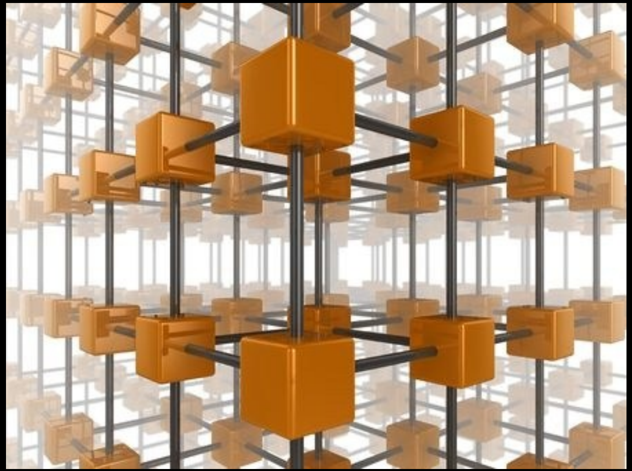
$$\underbrace{\qquad\qquad\qquad\qquad}$$

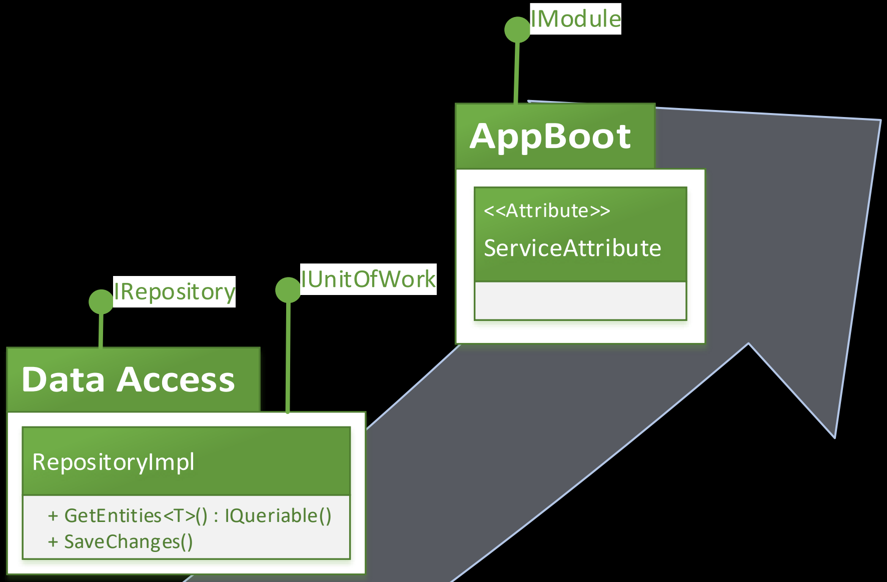
CONSISTENCY + STRUCTURE

HIDE COMPLEXITY

Implementing Clean Architecture through Structure





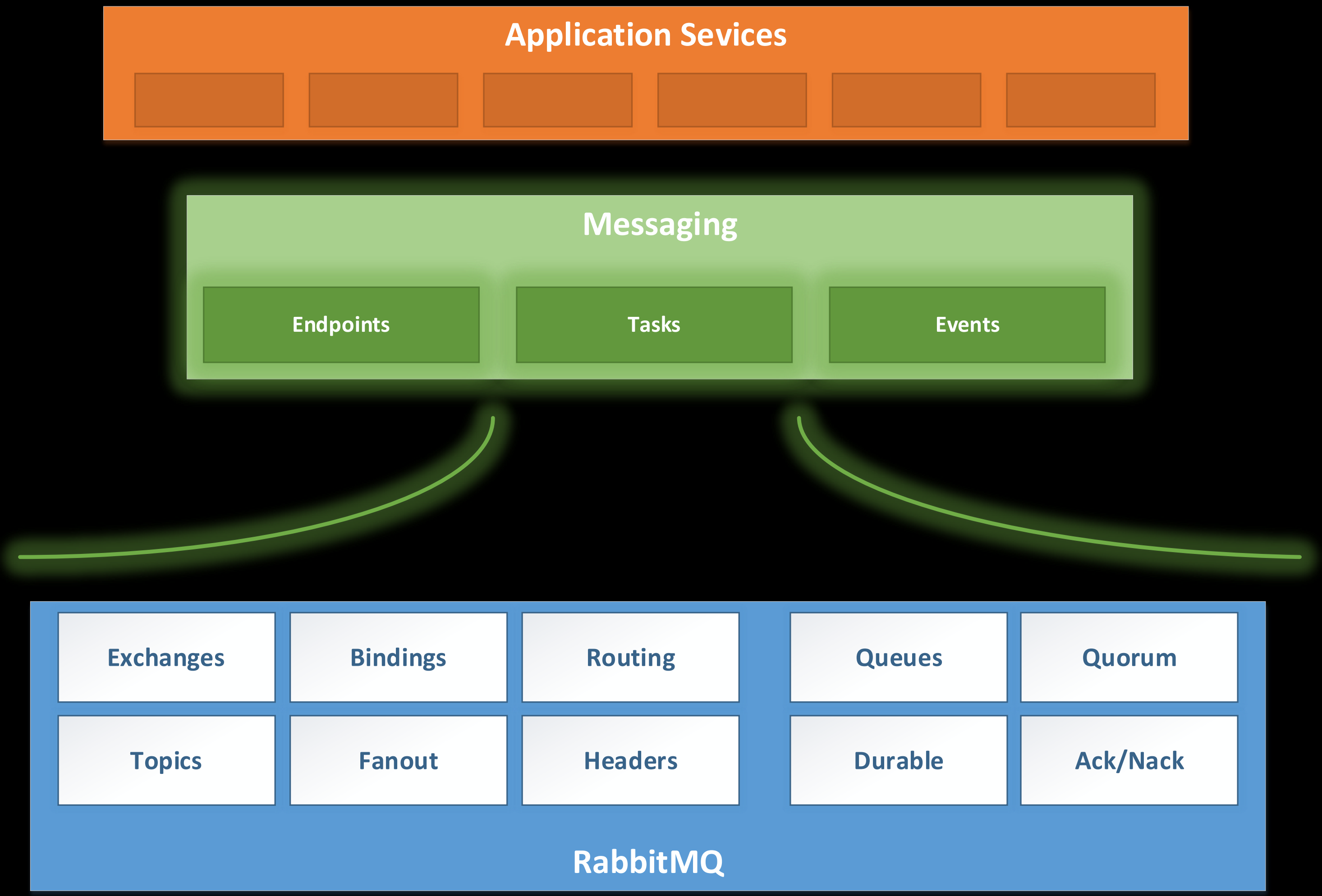


Hide external frameworks to enforce the way they are used

Use assemblies and references among them to **enforce rules**

Enforce Constructor Dependency Injection that encourages *Programming Against Interfaces*

Messaging over RabbitMQ



- Reliable Messaging across the system
- The developers that work on application services do NOT need to know the details and complexity of RabbitMQ
  - all types of Exchanges
  - all types of Queues
  - how construct the Routing Keys
  - how to build the Headers
  - Ack/Nack
  - Transactions, Durability

Long Term Development

feature complete
live!
CODEDESIGN

Challenges:

- ❖ invest in foundation vs deliver features
- ❖ team volatility & team scale-up
- ❖ adapt to changes in external systems APIs

CodeDesign

Please rate this session using

.NET DeveloperDays Mobile App
(available in AppStore & Google Play)

my LinkedIn

florin@onCodeDesign.com
linkedin.com/in/florincoros
oncodedesing.com/training
calendly.com/code-design/florin-short-call

onCodeDesign.com/DevDaysPL