



Application Infrastructure for Clean Architecture

Florin Coroş
florin@onCodeDesign.com
[linkedin.com/in/florincoros](https://www.linkedin.com/in/florincoros)



Florin Coroș

Solution Architect Consultant

Technical Trainer

Founder of Code Design

enjoing playing GO

enjoing traveling



Application Infrastructure for Clean Architecture

Florin Coroş
florin@onCodeDesign.com
[linkedin.com/in/florincoros](https://www.linkedin.com/in/florincoros)

Why is Architecture Important?



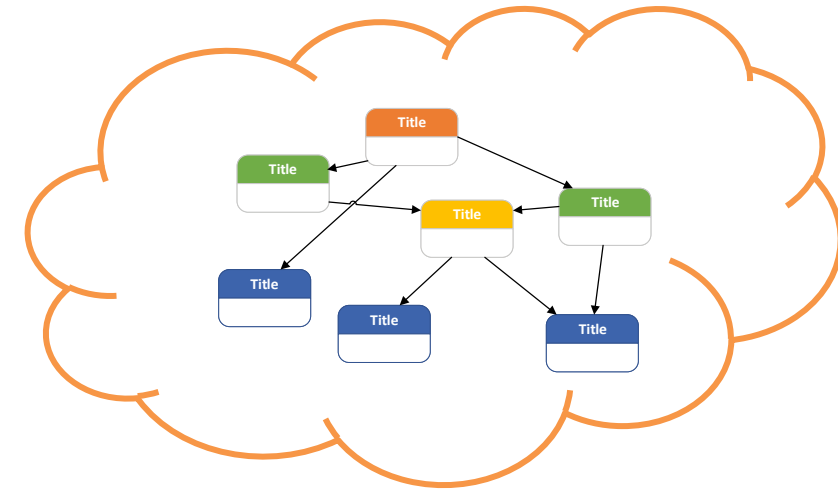
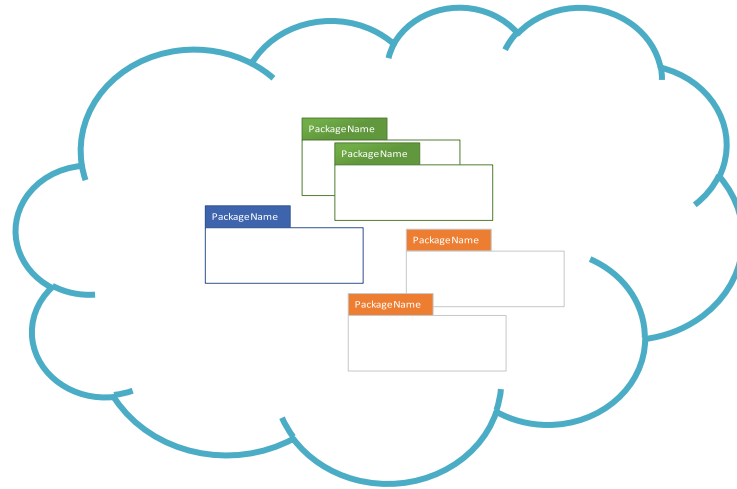
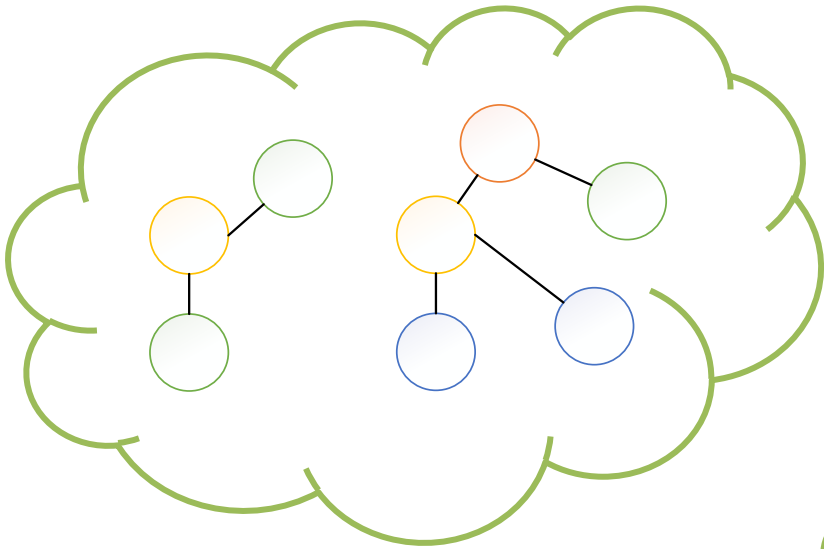
The goal of software architecture is to minimize the human resources required to build and maintain the required system

”

-- Robert C. Martin, Clean Architecture

4

Decompose - Separation of Concerns



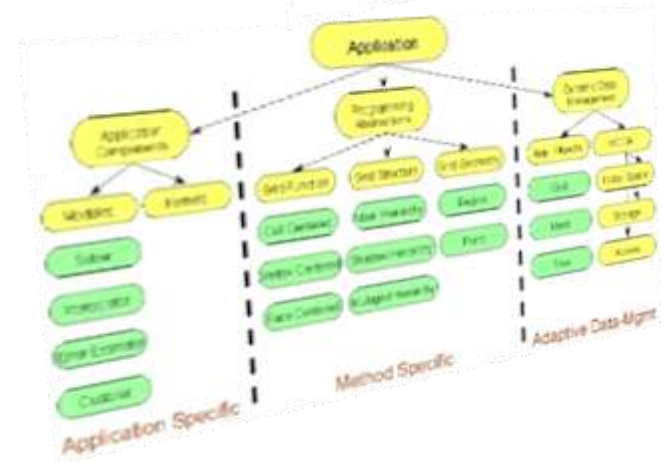
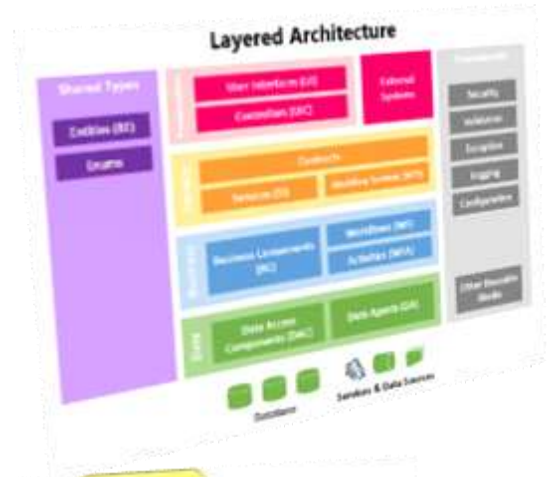
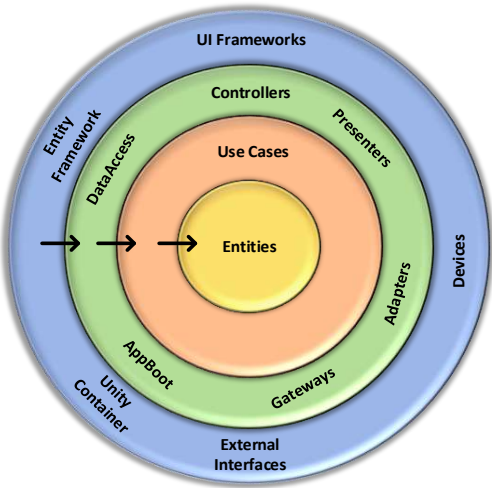
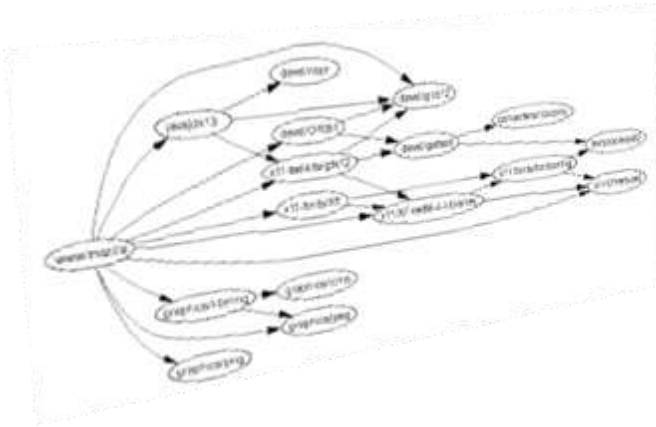
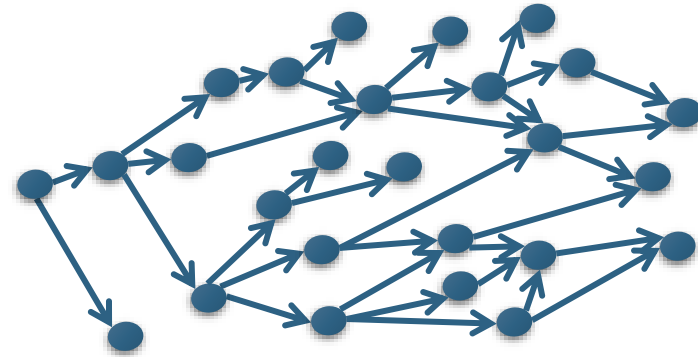
Manage the Complexity and Size



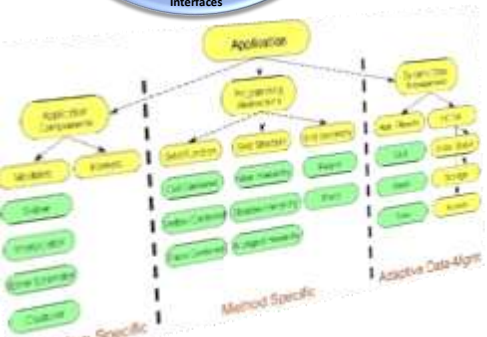
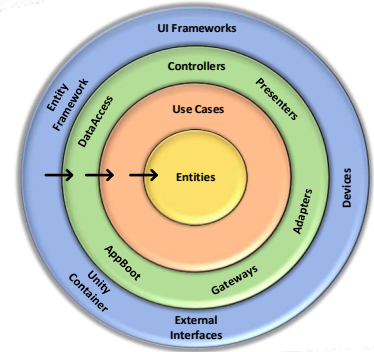
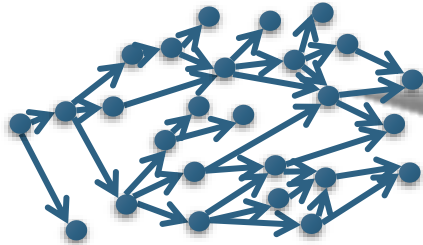
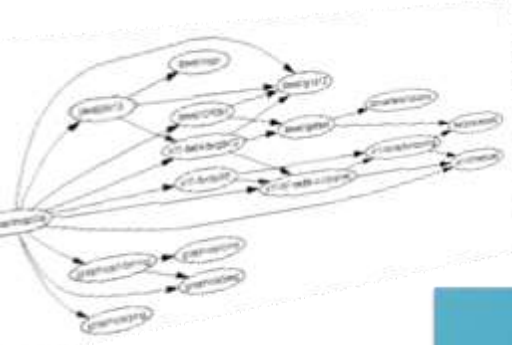
when projects do fail for reasons that are primarily technical, the reason is often

uncontrolled complexity

The Architecture is Separation of Concerns and a Set of Rules



Implementing the Architecture



```
private EfContext db = ...
public void UpdateTaxStatus(TaxStatus item)
{
    // validations
    if (DateTimeUtils.IsStartDateBeforeEndDate(item.StartDate, item.EndDate) == false)
    {
        throw new ValidationAidsException(Localize.GetLocalizedResource("Start date cannot ..."));
    }
    // check if some hotel period will overlap with another period
    list<Hotel> hotels = DB.TaxStatusesHotels.Where(c => c.TaxStatusID == item.ID).Select(
        () => new Hotel { ID = item.ID, StartDate = item.StartDate, EndDate = item.EndDate });
    foreach (Hotel hotel in hotels)
    {
        TaxStatusesHotel taxStatusesHotel;
        bool isOverlapping = IsHotelHavingOverlappingTaxStatus(hotel.ID, item.ID, item.St...);
        if (isOverlapping)
        {
            string errorMessage = string.Format(Localize.GetLocalizedResource("The Hotel ..."));
            taxStatusesHotel.Hotel.Name,
            taxStatusesHotel.TaxStatus.Name,
        }
    }
}

foreach (Hotel hotel in hotels)
{
    TaxStatusesHotel taxStatusesHotel;
    bool isOverlapping = IsHotelHavingOverlappingTaxStatus(hotel.ID, item.ID, item.StartDate, item.EndDate);
    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 ValidationAidsException(string.Format(errorMessage, item.ID), ScreenName);
    }
}

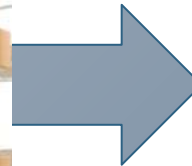
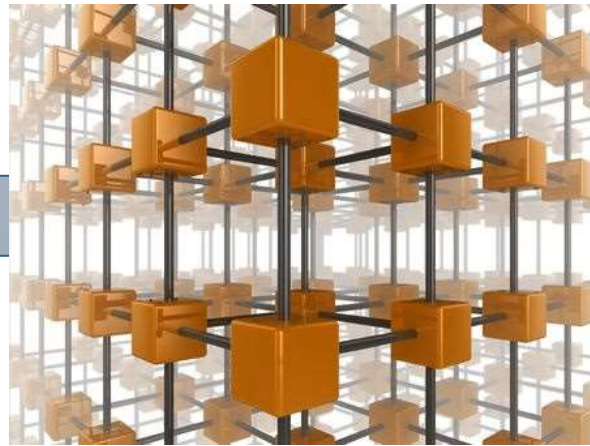
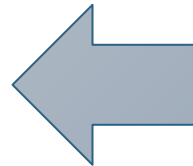
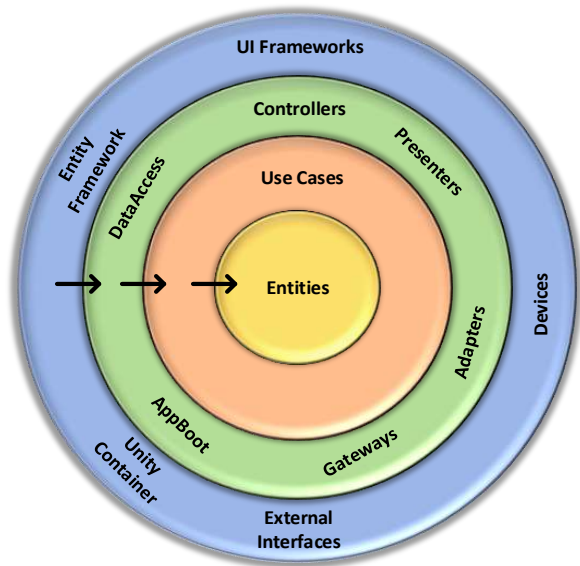
base.Update(item, true);

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

IDbCommand cmd.Parameters.Add(new SqlParameter("ID", itemID));
using (IDbCommand cmd = _db.CreateCommand())
{
    cmd.CommandText = "DELETE FROM TaxStatuses WHERE ID = @ID";
    cmd.ExecuteNonQuery();
}

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


Application Infrastructure Enforces the Architecture Implementation



```
public void UpdateTaxStatus(TaxStatus item)
{
    // validations
    if (DateHelper.IsStartDateBeforeEndDate(item.StartDate, item.EndDate) == false)
    {
        throw new ValidationException(Localize.GetLocalizedResource("Start date cannot be
    )
    // check if some hotel period will overlap with another period
    IList<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.StartDate
        if (isOverlapping)
            throw new ValidationException(string.Format(Localize.GetLocalizedResource("The Hotel '{0}'
    }
    foreach (Hotel hotel in hotels)
    {
        TaxStatusesHotel taxStatusesHotel;
        bool isOverlapping = IsHotelHavingOverlappingTaxStatus(hotel.ID, item.ID, item.StartDate
        if (isOverlapping)
        {
            string errorMessage = string.Format(Localize.GetLocalizedResource("The Hotel '{0}'
            taxStatusesHotel.HotelName,
            taxStatusesHotel.TaxStatus.StartDate.ToShortDateString(),
            taxStatusesHotel.TaxStatus.EndDate.ToShortDateString());
            taxStatusesHotel.TaxStatus.EndDate.ToShortDateString());
            throw new ValidationException(string.Format(errorMessage, item.ID), ScreenName
        }
    }
    base.Update(item, true);
}

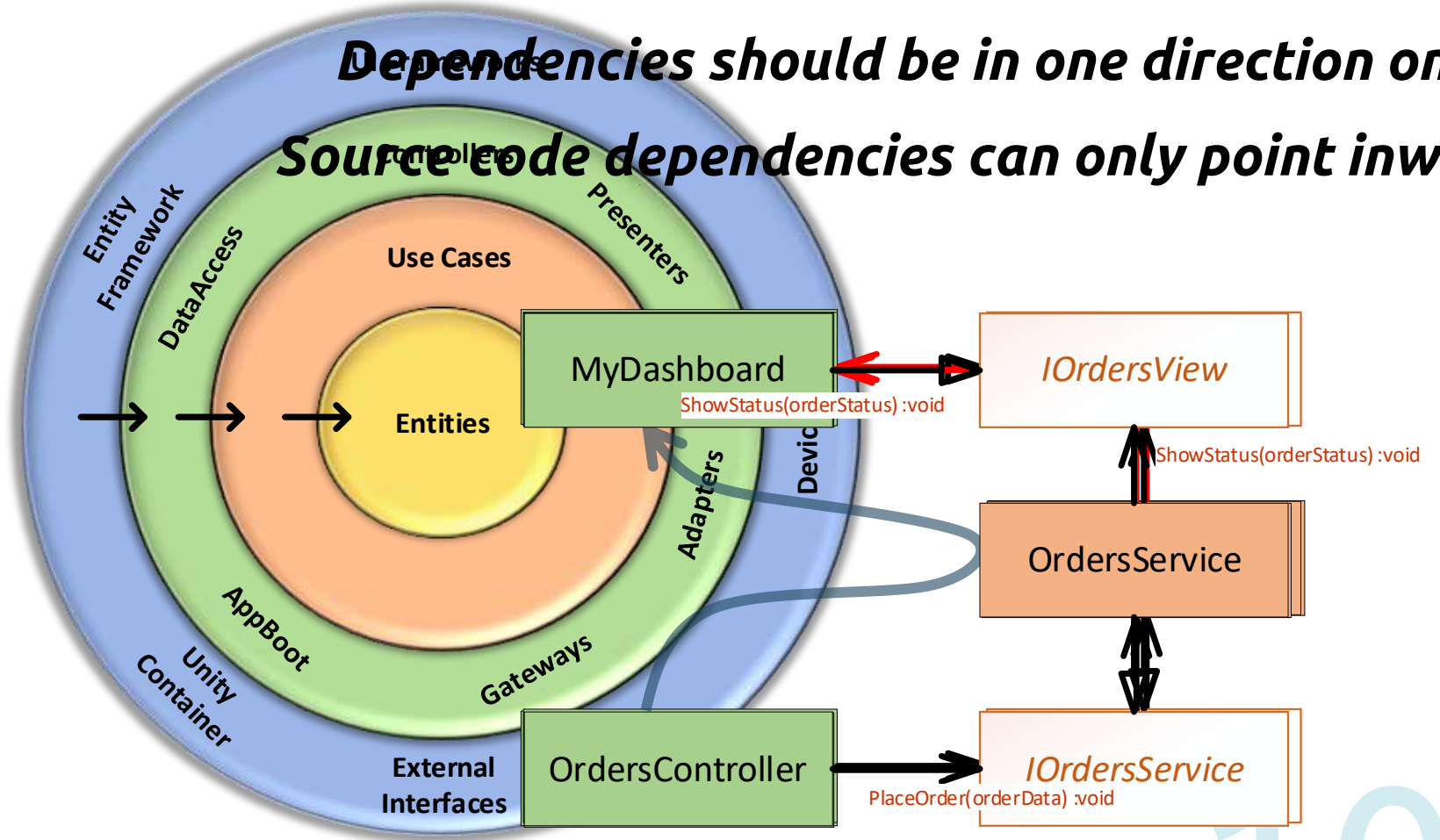
public void DeleteTaxStatus(long itemID)
{
    TaxStatus item = DB.TaxStatuses.FirstOrDefault(c => c.ID == itemID);
    if (item == null)
    {
        throw new ValidationException(
            string.Format(Localize.GetLocalizedResource("There is no object with such ID: {0}
    )
    if (item.StartDate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

#endregion
```

Clean Architecture



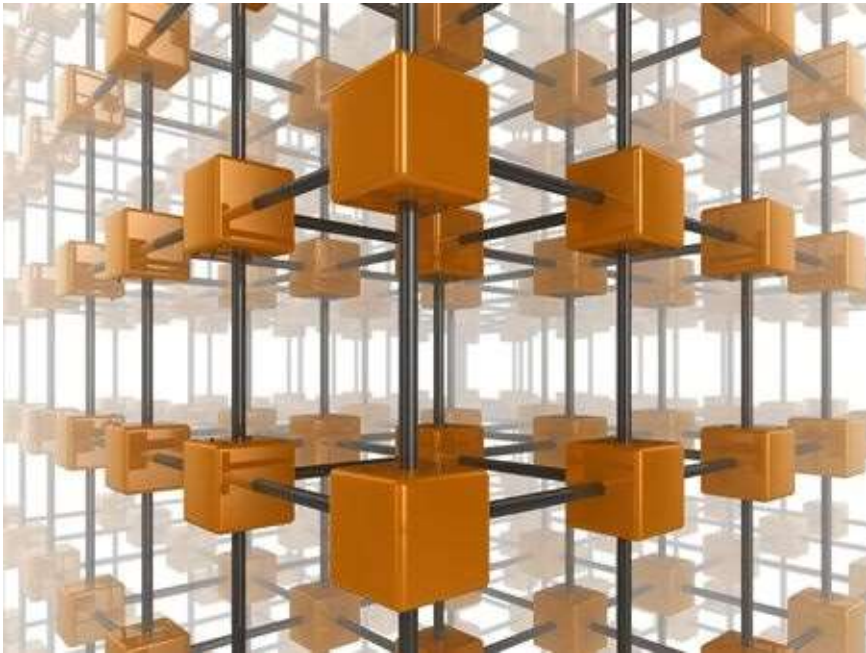
Dependencies should be in one direction only
Source code dependencies can only point inwards



Building an Application Infrastructure

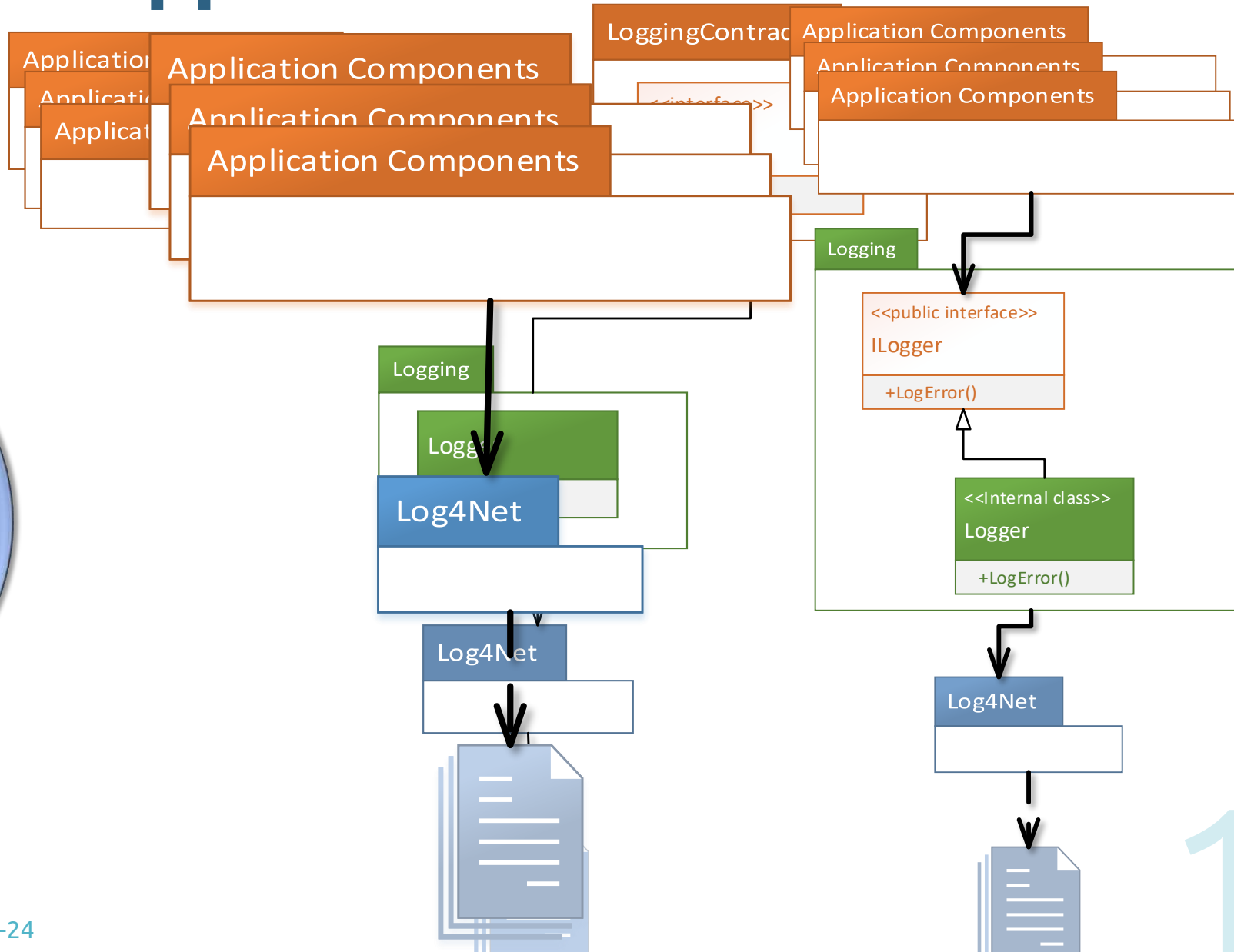
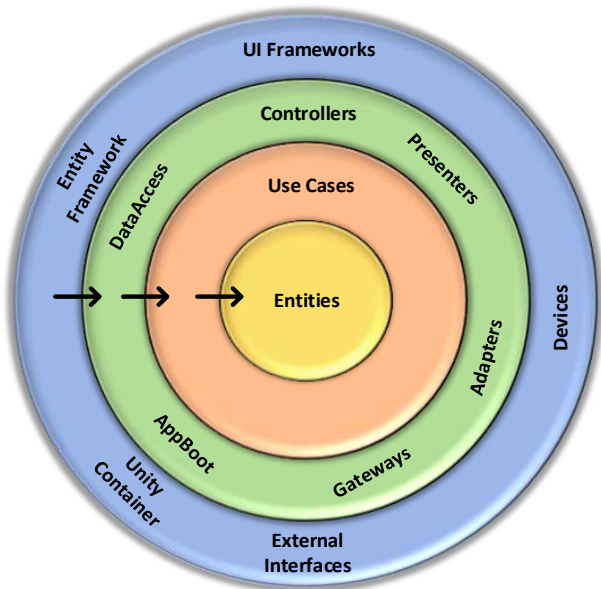


Create a structure that makes it difficult to write bad code and it makes it easy to write good code, code that follows the architecture and the design



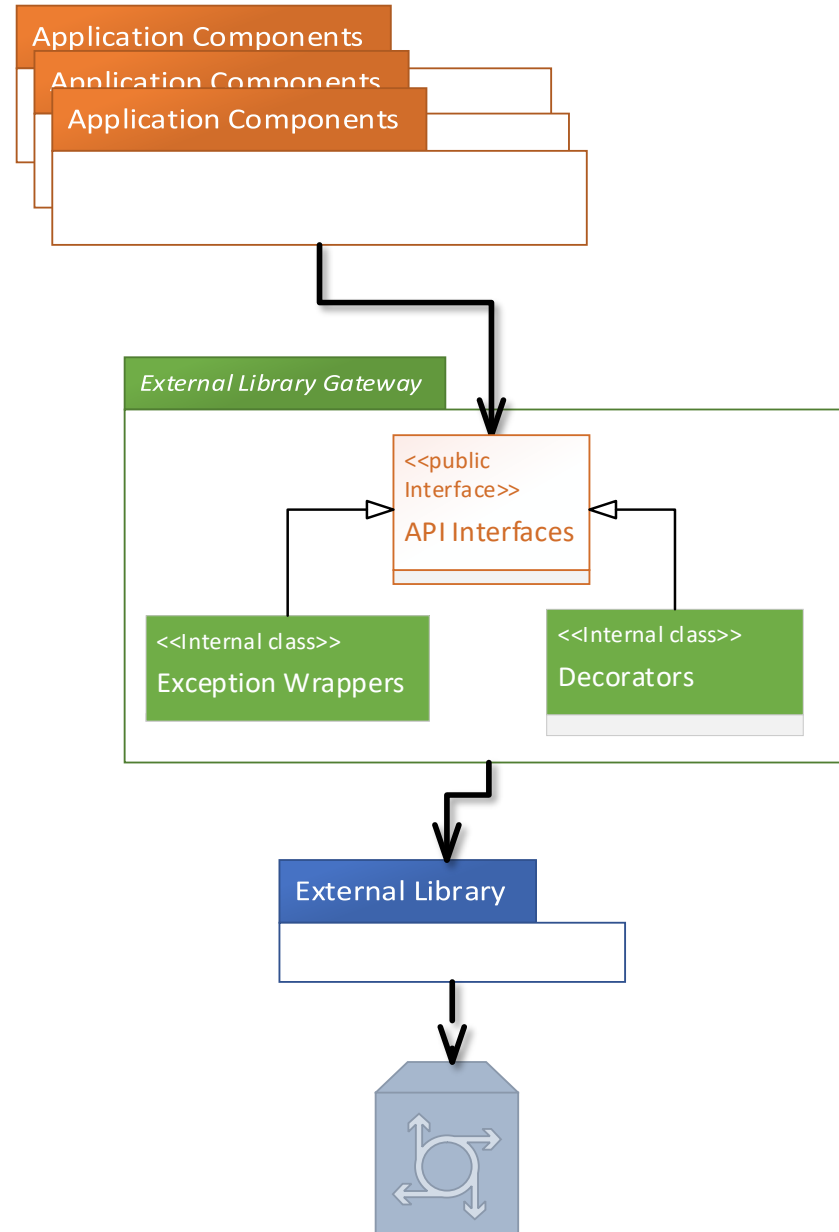
- **Hide external frameworks** to enforce the way they are used
- Use assemblies and references among them to **enforce rules**
- **Enforce *Constructor Dependency Injection*** that encourages *Programming Against Interfaces*

Creating the App Infra for Clean Architecture

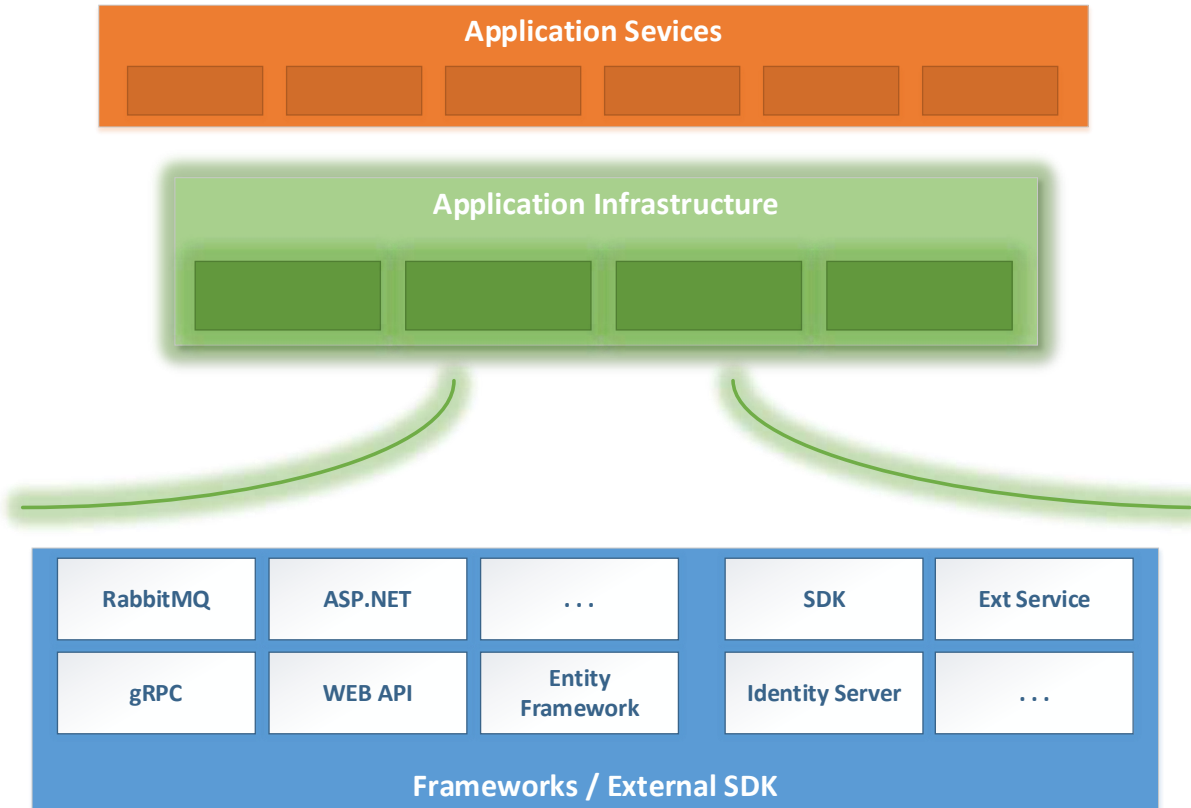


12

Hide External Libraries from the App Code



App Infrastructure is a Set of Tech Components



do not depend on Frameworks

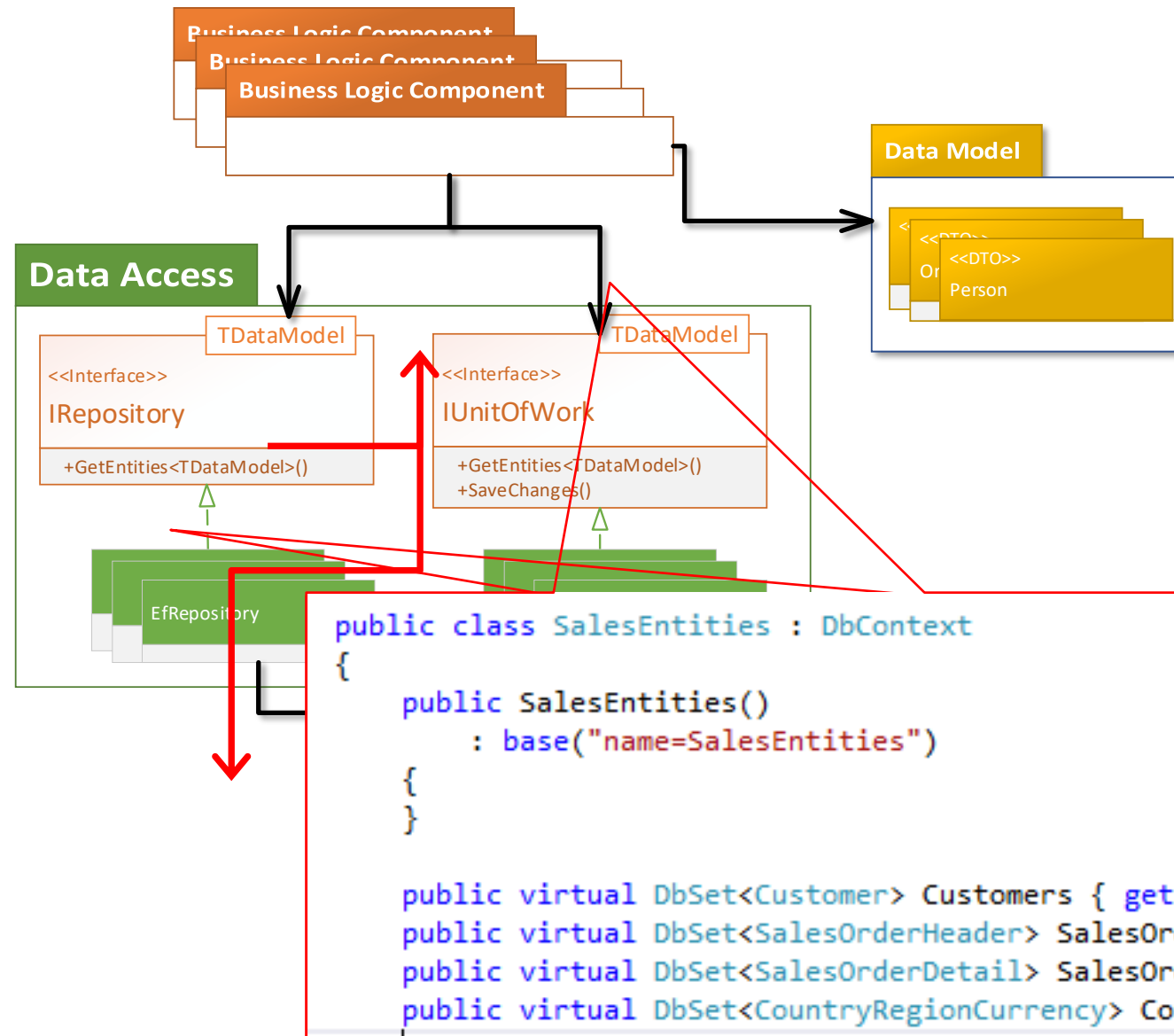


CONSISTENCY

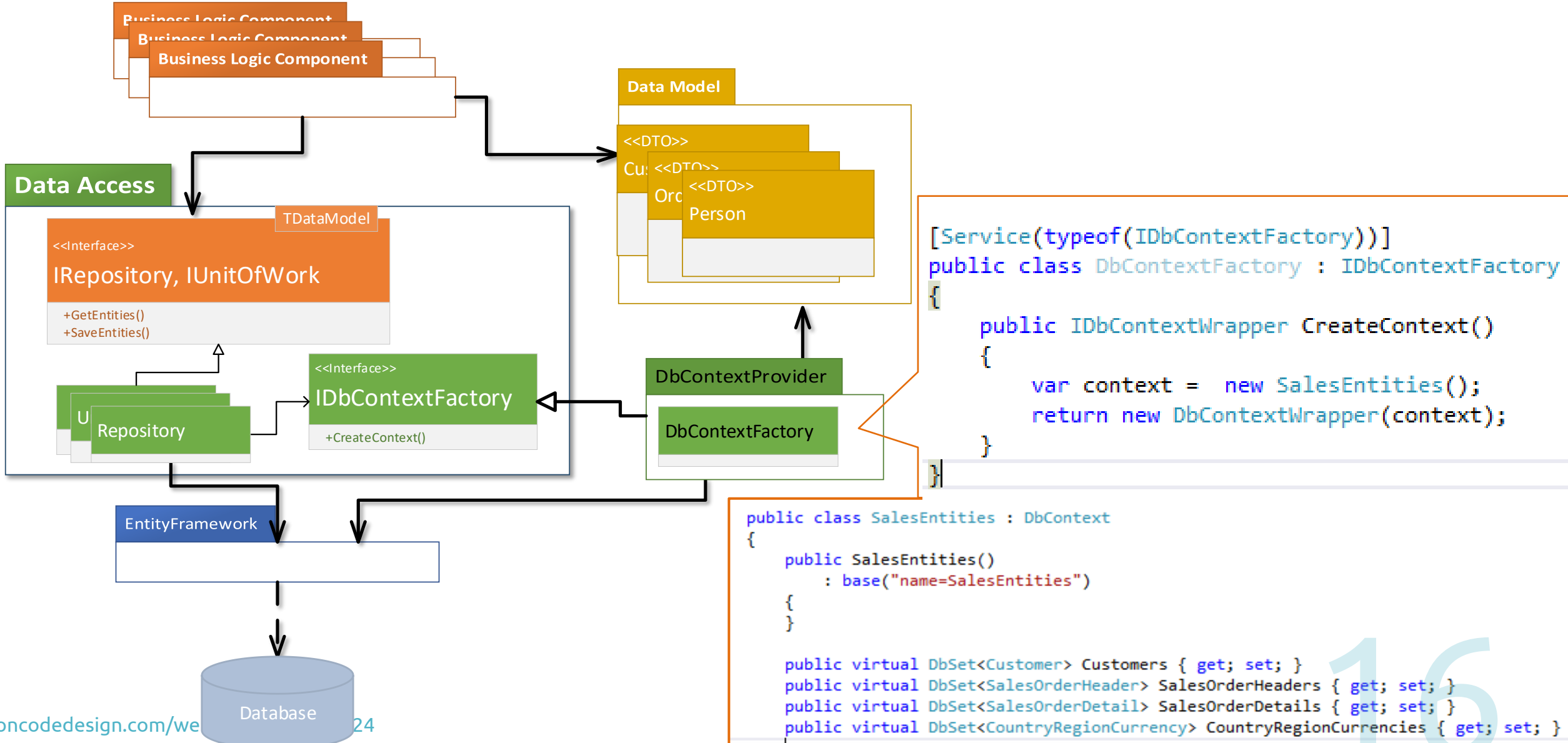
STRUCTURE

HIDE COMPLEXITY

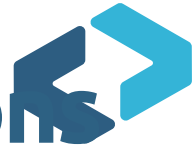
Enforce Separation of Data Access Concerns



DIP to Enforce Separation of Data Access Concern



AppBoot Hides the DI Framework under Abstractions

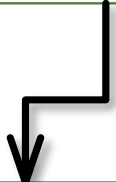


```
public interface IPriceCalculator
{
    int CalculateTaxes(Order o, Customer c);
    int CalculateDiscount(Order o, Customer c);
}

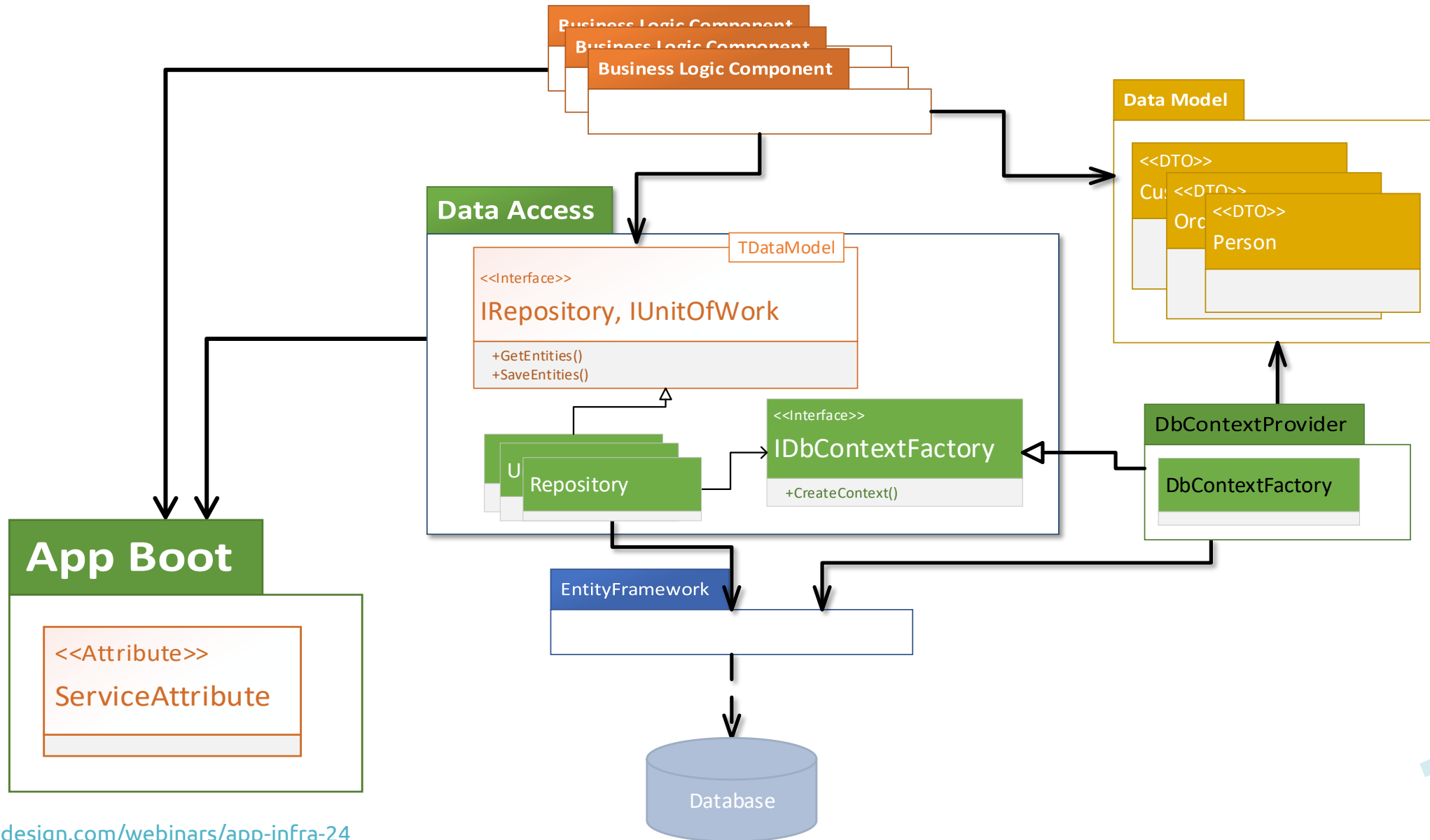
[Service(typeof(IPriceCalculator), Lifetime.Instance)]
internal class PriceCalculator : IPriceCalculator
{
    public int CalculateTaxes(Order o, Customer c)
    {
        return 10; // do actual calculation
    }

    public int CalculateDiscount(Order o, Customer c)
    {
        return 20; // do actual calculation
    }
}
```

Unity Container



AppBoot: DI Abstractions & Type Discovery



Patterns on how most of the code is written



```
[Service(typeof (IOrderingService))]
private class OrderingService : IOrderingService
{
    private readonly IRepository repository;
    private readonly IPriceCalculator calculator;
    private readonly IApprovalService orderApproval;

    public OrderingService(IRepository repository, IPriceCalculator calculator, IApprovalService orderApproval)
    {
        this.repository = repository;
        this.calculator = calculator;
        this.orderApproval = orderApproval;
    }

    public SalesOrderInfo[] GetOrdersInfo(string customerName)
    {
        var orders = repository.GetEntities<SalesOrderHeader>()
        ...
        return orders.ToArray();
    }

    public SalesOrderResult PlaceOrder(string customerName, OrderRequest request)
    {
        ...
    }
}
```

Patterns for Read-Only data



```
public class OrdersController : Controller
{
    private readonly IRepository repository;
    public OrdersController(IRepository repository)
    {
        this.repository = repository;
    }

    public IActionResult Index(string customer)
    {
        var orders = repository.GetEntities<SalesOrderHeader>()
            .Where(soh => soh.Customer.Person.LastName == customer)
            .Select(soh => new OrdersListViewModel
            {
                CustomerName = customer,
                Number = soh.SalesOrderNumber,
                SalesPersonName = soh.SalesPerson,
                DueDate = soh.DueDate,
            });

        return View(orders);
    }
}

```

Patterns for Read-Write data



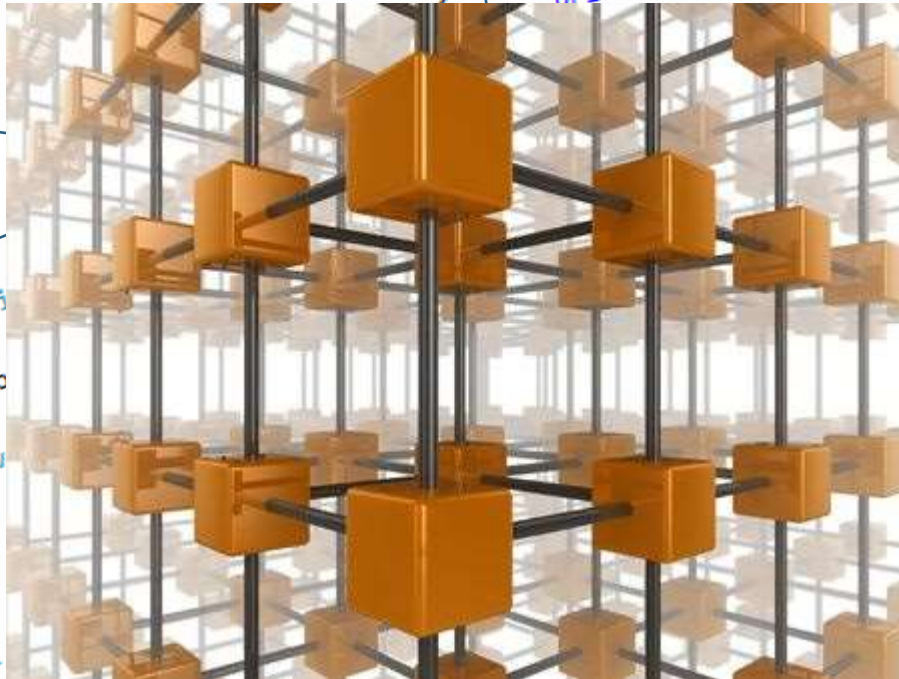
```
public class OrdersController : Controller
{
    ...
    [HttpPost]
    [ValidateAntiForgeryToken]
    public IActionResult PlaceOrder(OrderRequestViewModel model)
    {
        ...
        using (IUnitOfWork uof = repository.CreateUnitOfWork())
        {
            SalesOrderHeader order = uof.GetEntities<SalesOrderHeader>()
                .FirstOrDefault(o => o.CustomerID == c.ID && o.OrderDate.Month == DateTime.Now.Month);
            if (order == null)
            {
                order = new SalesOrderHeader {Customer = c};
                uof.Add(order);
            }
            AddRequestToOrder(model, order);

            uof.SaveChanges();
        }
        ...
    }
}
```

Create Development Patterns in Code

```
[Service(typeof (IPriceCalculator), Lifetime.Instance)]  
public class PriceCalculator : IPriceCalculator  
{  
    public decimal CalculateTaxes(OrderRequest o, Customer c)  
    {  
        // do actual calculation  
        return 10;  
    }  
}
```

```
[Service(typeof (IOrderingService))]  
class OrderingService : IOrderingService  
{  
    private readonly IRepository repository;  
    private readonly IPriceCalculator calculator;  
    private readonly IApprovalService orderApproval;  
    OrderingService(IRepository repository, IPriceCalculator calculator, IApprovalService orderApproval)  
    {  
        repository = repository;  
        calculator = calculator;  
        orderApproval = orderApproval;  
    }  
}
```

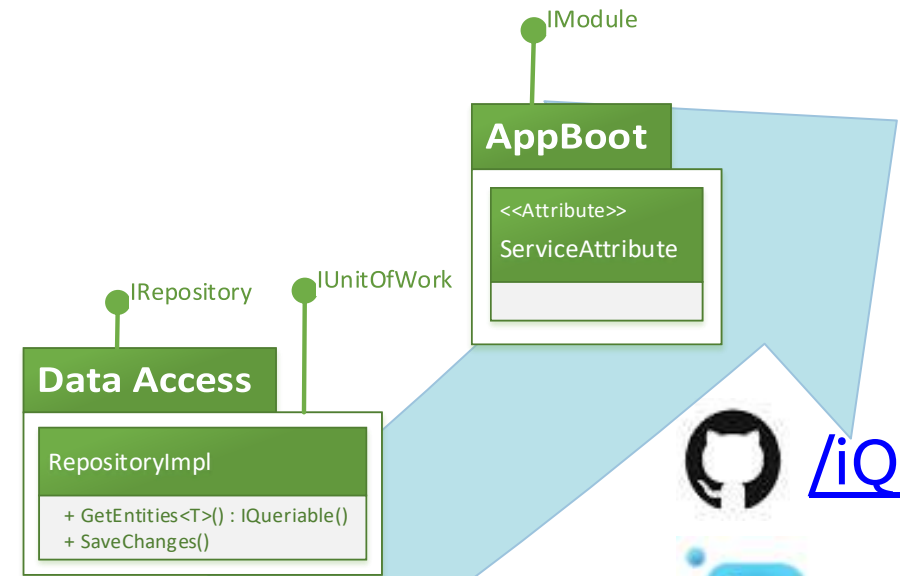
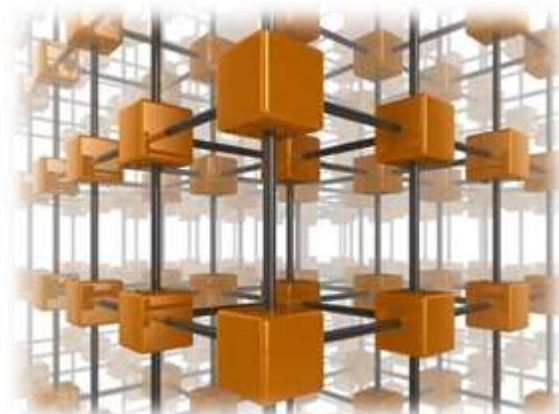
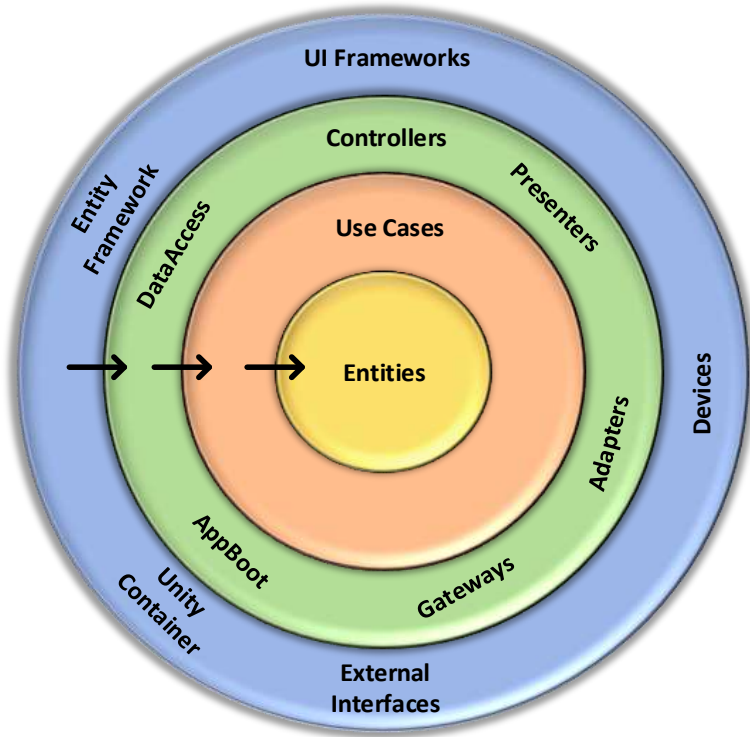


```
[Service(typeof (IApprovalService))]  
class CompositeApprovalService : IApprovalService  
{  
    private readonly IApprovalService[] approvals;  
    CompositeApprovalService(IApprovalService[] approvals)  
    {  
        this.approvals = approvals;  
    }  
    public bool Approve(ApproveRequest request)  
    {  
        foreach (var approval in approvals)  
        {  
            if (!approval.Approve(request))  
                return false;  
        }  
        return true;  
    }  
}
```

```
public void GetOrdersInfo(string customerName)  
{  
    var orders = repository.GetEntities<SalesOrderHeader>().  
        ToArray();  
}
```

```
public SalesOrderResult PlaceOrder(string customerName, OrderRequest request)  
{  
    var uow = repository.CreateUnitOfWork();  
    uow.Transaction.Begin();  
    try  
    {  
        ..  
    }  
    catch { }  
    uow.Transaction.Commit();  
    var taxes = calculator.CalculateTaxes(order, customer);  
    var discount = calculator.CalculateDiscount(order, customer);  
    ..  
    return new SalesOrderResult {State = OrderResultState.Invalid};  
}
```

Implementing Clean Architecture with Application Infrastructure



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

27



florin@onCodeDesign.com

[linkedin.com/in/florincoros](https://www.linkedin.com/in/florincoros)

oncodedesing.com/training

calendly.com/code-design/florin-short-call



Application Infrastructure for Clean Architecture

Florin Coroș

Solution Architect Consultant
Technical Trainer