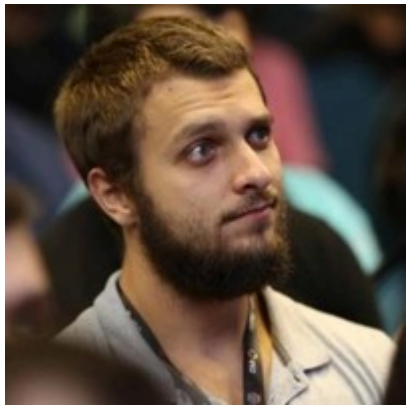
THE DEVELOPER'S CONFERENCE

Entregando testes de valor

Leonardo Prange

Marco Nicolodi

Sobre nós



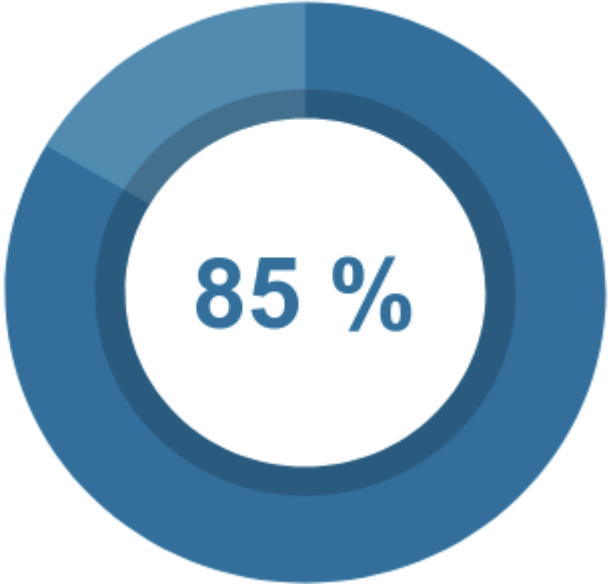
Desenvolvedores

THE DEVELOPER'S CONFERENCE

Line of Business ASP.NET Core API

THE
DEVELOPER'S
CONFERENCE

Coverage



THE DEVELOPER'S CONFERENCE

Parece bom

O que acontecia

- Mecheu == quebrou
- Teste não passava confiança
- Dificil de entender/ler/refatorar/olhar/cheirar/TUDO
- Degradavel
- Diminui velocidade da entrega
- “//” era a salvação

THE DEVELOPER'S CONFERENCE

```
[Fact]
public async void Update_ShouldUpdateDocument()
{
    //Arrange
    var context = InMemoryDocContextFactory.Create();
    var unitOfWork = new DocUnitOfWork(context, _logger);

    var elaboration = new Stage() { Type = StageType.Elaboration, Name = "Elaboration" };
    var approval = new Stage() { Type = StageType.Approval, Name = "Approval" };
    var consensus = new Stage() { Type = StageType.Approval, Name = "Consensus" };

    context.Stages.Add(elaboration);
    context.Stages.Add(approval);
    context.Stages.Add(consensus);

    var oldCategory = new Category()
    {
        Initials = "CAT",
        Name = "Category",
        Stages = new List<CategoryStage>() {
            new CategoryStage() { Order = 1, Stage = elaboration },
            new CategoryStage() { Order = 2, Stage = approval },
        },
    };
    var newCategory = new Category()
    {
        Initials = "DOG",
        Name = "Dogcary",
        Stages = new List<CategoryStage>() {
            new CategoryStage() { Order = 1, Stage = elaboration },
            new CategoryStage() { Order = 2, Stage = consensus },
        }
    };

    context.Categories.Add(oldCategory);
    context.Categories.Add(newCategory);
```

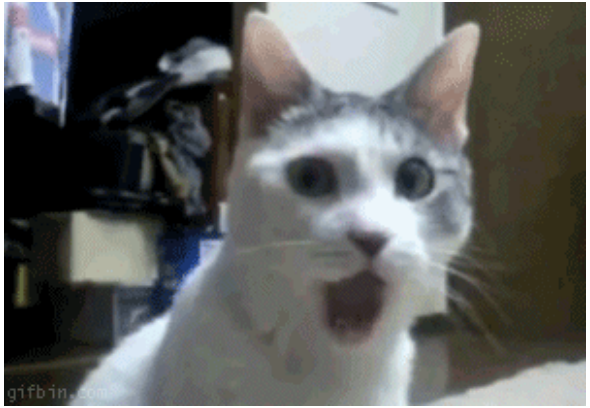
THE DEVELOPER'S CONFERENCE

```
var document = new Document()
{
    CategoryId = oldCategory.Id,
    SuggestedCode = new SuggestedCode() { Code = 5, DocumentId = 1 },
    ProcessGuid = ProcessMother.AutomatedTest().Id,
    Title = "Arroz",
    Code = "Codecept",
    Stages = new List<DocumentStage>() {
        new DocumentStage() {
            StageId = elaboration.Id,
            Responsibles = new List<DocumentStageResponsible>()
            {
                new DocumentStageResponsible() { ResponsibleGuid = UserMother.MrCatraId },
            },
        },
        new DocumentStage()
        {
            StageId = approval.Id,
            Responsibles = new List<DocumentStageResponsible>()
            {
                new DocumentStageResponsible() { ResponsibleGuid = UserMother.MrCatraId },
                new DocumentStageResponsible() { ResponsibleGuid = UserMother.PabloEscobarId },
            },
        },
    },
    Revisions = new List<Revision>()
    {
        new Revision() {
            Stages = new List<RevisionStage>(),
            PublishDate = DateTime.Now,
            RevisionRequestHistory = new List<RevisionRequest>()
        },
    }
};

context.Documents.Add(document);
```

THE DEVELOPER'S CONFERENCE

```
var mutation = new UpdateDocumentMutation()
{
    Category = newCategory.Id,
    Code = "CAT0006",
    Title = "Title",
    Process = ProcessMother.AutomatedTest().Id,
    Stages = new List<UpdateDocumentMutationStage>() {
        new UpdateDocumentMutationStage()
        {
            Id = elaboration.Id,
            Responsibles = new List<Guid>() { UserMother.PabloEscobarId },
            DaysToDueDate = 27
        },
        new UpdateDocumentMutationStage()
        {
            Id = consensus.Id,
            Responsibles = new List<Guid>() { UserMother.MrCatraId },
            DaysToDueDate = 7,
        },
    },
};

var business = new Mock<IDocumentBusiness>().Object;

context.SaveChanges();
var documentService = new DocumentWriteService(unitOfWork, null, business, null, null, context, null, null, null, null)

//Act & Assert
var updatedDocument = await documentService.Update(document.Id, mutation);
Assert.Equal(updatedDocument.Code, mutation.Code);
Assert.Equal(updatedDocument.CategoryId, mutation.Category);
Assert.Equal(updatedDocument.Title, mutation.Title);
Assert.Equal(updatedDocument.ProcessGuid, mutation.Process);
Assert.Equal(6, updatedDocument.SuggestedCode.Code);
updatedDocument.Stages.First().StageId.Should().Equals(elaboration.Id);
Assert.Equal(27, updatedDocument.Stages.First().DaysToDueDate);
updatedDocument.Stages.Last().StageId.Should().Equals(consensus.Id);
Assert.Equal(7, updatedDocument.Stages.Last().DaysToDueDate);
updatedDocument.Stages.First().Responsibles.Count().Equals(1);
updatedDocument.Stages.First().Responsibles.First().ResponsibleGuid.Equals(UserMother.PabloEscobarId);
updatedDocument.Stages.Last().Responsibles.Count().Equals(1);
updatedDocument.Stages.Last().Responsibles.First().ResponsibleGuid.Equals(UserMother.MrCatraId);
```

THE DEVELOPER'S CONFERENCE



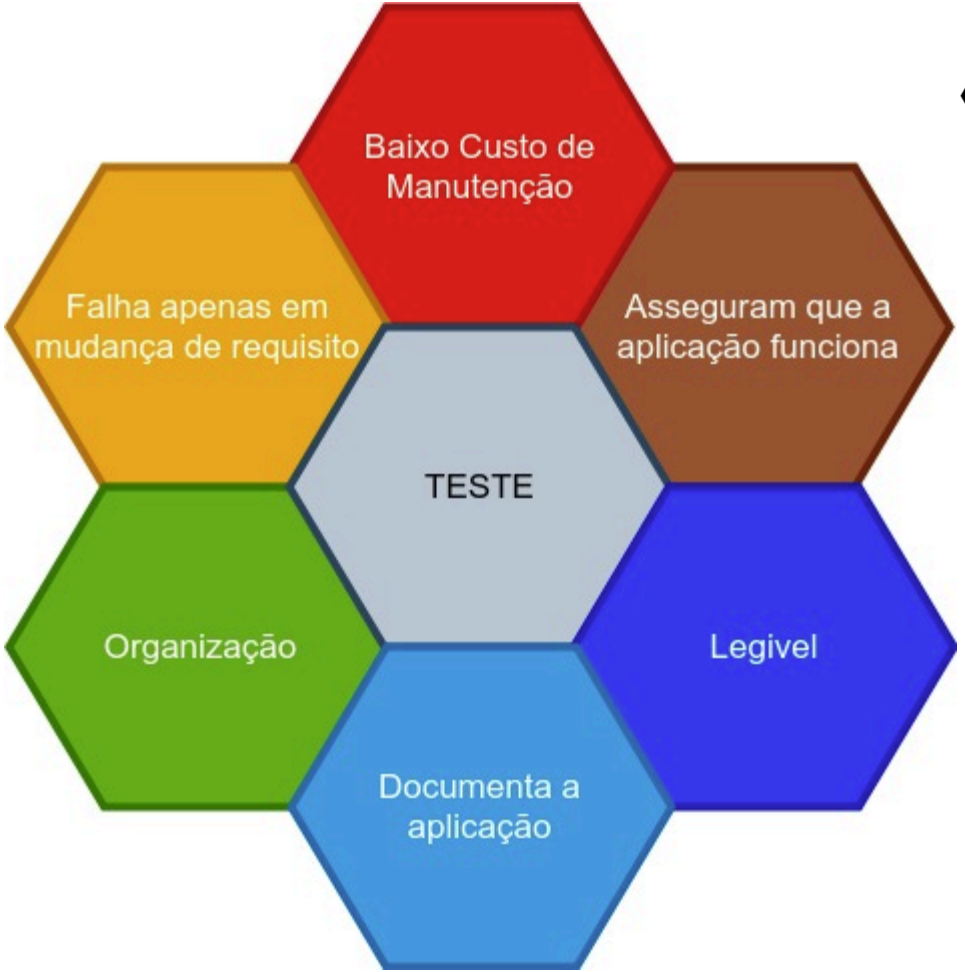
119 linhas

THE DEVELOPER'S CONFERENCE

Como saímos deste cenário

THE DEVELOPER'S CONFERENCE

Quais as características de um bom teste?



THE DEVELOPER'S CONFERENCE

THE DEVELOPER'S CONFERENCE

Entender

THE DEVELOPER'S CONFERENCE

```
[Fact]
public async void RejectRevisionRequestTransfer_ShouldThrowNotFoundException_WhenDocumentDoesNotExist()

[Fact]
public async void RejectRevisionRequestTransfer_ShouldThrowNotFoundException_WhenDocumentDoesNotHaveAElaborationStage()

[Fact]
public async void RejectRevisionRequestTransfer_ShouldThrowNotFoundException_WhenThereIsNoOpenRevisionRequest()

[Fact]
public async void RejectRevisionRequestTransfer_ShouldCallBusinessAndUpdateRequestAndPublishRejectedRevisionRequestTransferEvent()
```

Use a propriedade **DisplayName** para cenários de casos de uso

Documentação

THE DEVELOPER'S CONFERENCE

```
[Fact (DisplayName=@"
    GIVEN I have a document
    WHEN I update information
    THEN the document must have the new information
")]
0 references | Run Test | Debug Test
public async void Update_ShouldUpdateDocument()
{
```

THE DEVELOPER'S CONFERENCE

```
[Fact(DisplayName = @"
    GIVEN
        that SteveJobs is responsible for the elaboration stage of the document
        AND SteveJobs and GeorgeBush are responsibles for the approval stage of the document
    WHEN
        I replace SteveJobs with MrCatra
    THEN
        MrCatra should be the new responsible for the elaboration stage
        AND GeorgeBush and MrCatra should be the new responsibles for the approval stage
")]
public async void Replace_ShouldReplaceResponsibles()
{
```

THE DEVELOPER'S CONFERENCE

Fixtures

THE DEVELOPER'S CONFERENCE

```
[Fact]
public async void Update_ShouldUpdateDocument()
{
    //Arrange
    var context = InMemoryDocContextFactory.Create();
    var unitOfWork = new DocUnitOfWork(context, _logger);

    var elaboration = new Stage() { Type = StageType.Elaboration, Name = "Elaboration" };
    var approval = new Stage() { Type = StageType.Approval, Name = "Approval" };
    var consensus = new Stage() { Type = StageType.Approval, Name = "Consensus" };

    context.Stages.Add(elaboration);
    context.Stages.Add(approval);
    context.Stages.Add(consensus);

    var oldCategory = new Category()
    {
        Initials = "CAT",
        Name = "Category",
        Stages = new List<CategoryStage>() {
            new CategoryStage() { Order = 1, Stage = elaboration },
            new CategoryStage() { Order = 2, Stage = approval },
        },
    };
    var newCategory = new Category()
    {
        Initials = "DOG",
        Name = "Dogcary",
        Stages = new List<CategoryStage>() {
            new CategoryStage() { Order = 1, Stage = elaboration },
            new CategoryStage() { Order = 2, Stage = consensus },
        }
    };

    context.Categories.Add(oldCategory);
    context.Categories.Add(newCategory);
```

THE DEVELOPER'S CONFERENCE

```
var document = new Document()
{
    CategoryId = oldCategory.Id,
    SuggestedCode = new SuggestedCode() { Code = 5, DocumentId = 1 },
    ProcessGuid = ProcessMother.AutomatedTest().Id,
    Title = "Arroz",
    Code = "Codecept",
    Stages = new List<DocumentStage>() {
        new DocumentStage() {
            StageId = elaboration.Id,
            Responsibles = new List<DocumentStageResponsible>()
            {
                new DocumentStageResponsible() { ResponsibleGuid = UserMother.MrCatraId },
            },
        },
        new DocumentStage()
        {
            StageId = approval.Id,
            Responsibles = new List<DocumentStageResponsible>()
            {
                new DocumentStageResponsible() { ResponsibleGuid = UserMother.MrCatraId },
                new DocumentStageResponsible() { ResponsibleGuid = UserMother.PabloEscobarId },
            },
        },
    },
    Revisions = new List<Revision>()
    {
        new Revision() {
            Stages = new List<RevisionStage>(),
            PublishDate = DateTime.Now,
            RevisionRequestHistory = new List<RevisionRequest>()
        },
    }
};

context.Documents.Add(document);
```

THE DEVELOPER'S CONFERENCE

Modelo único levam a fixtures maiores

THE DEVELOPER'S CONFERENCE

Bounded contexts simplificam as entidades e os testes

THE DEVELOPER'S CONFERENCE

```
var document = new Document()
{
    CategoryId = oldCategory.Id,
    SuggestedCode = new SuggestedCode() { Code = 5, DocumentId = 1 },
    ProcessGuid = ProcessMother.AutomatedTest().Id,
    Title = "Arroz",
    Code = "Codecept",
    Stages = new List<DocumentStage>() {
        new DocumentStage() {
            StageId = elaboration.Id,
            Responsibles = new List<DocumentStageResponsible>()
            {
                new DocumentStageResponsible() { ResponsibleGuid = UserMother.MrCatraId },
            },
        },
        new DocumentStage()
        {
            StageId = approval.Id,
            Responsibles = new List<DocumentStageResponsible>()
            {
                new DocumentStageResponsible() { ResponsibleGuid = UserMother.MrCatraId },
                new DocumentStageResponsible() { ResponsibleGuid = UserMother.PabloEscobarId },
            },
        },
    },
    Revisions = new List<Revision>()
    {
        new Revision() {
            Stages = new List<RevisionStage>(),
            PublishDate = DateTime.Now,
            RevisionRequestHistory = new List<RevisionRequest>()
        },
    }
};

context.Documents.Add(document);
```

THE DEVELOPER'S CONFERENCE

Object Mothers have essentially one job: give birth to good defaults and test-ready entities.

```
var document = DocumentMother.DocumentWithOnePendingRevisionInElaboration();
```

THE DEVELOPER'S CONFERENCE

12 references

```
public static Document DocumentWithOnePendingRevisionInElaboration(
    int id = 1,
    string code = "CEA001",
    string title = "Document Title",
    User elaborator = null,
    List<User> approvers = null,
    List<File> files = null,
    Process process = null
)
{
    var categoryElaborationApproval = CategoryMother.ElaborationApproval();
    process = process ?? ProcessMother.AutomatedTest();

    var document = Document(
        id: id,
        code: code,
        title: title,
        process: process,
        category: categoryElaborationApproval
    );

    elaborator = elaborator ?? UserMother.MrCatra();
    approvers = approvers ?? new List<User>() { UserMother.PabloEscobar() };
    files = files ?? new List<File>();

    AddDocumentStages(document, elaborator, approvers);

    document.Revisions.Add(RevisionMother.OneCyclePendingElaboration(document, files));

    return document;
}
```

THE DEVELOPER'S CONFERENCE

Coesão

THE DEVELOPER'S CONFERENCE

O quanto os membros de uma classe estão relacionadas com ela

```
public interface IDocumentService
{
    Task<Page<DocumentListViewModel>> List(PaginationOptions paginationOptions, SortOptions sortOptions,
    Task<DocumentDetailsViewModel> FindDetails(int id);
    Task<DocumentFormViewModel> FindForm(int id);
    Task<List<TaskViewModel>> GetLoggedUserTasks();
    Task<CurrentStageViewModel> CurrentStage(int documentId);
    List<StageViewModel> GetCurrentStageStages(int documentId);
    int Count(DocumentFilterOptions filterOptions);
    string GetSuggestedCode(int categoryId);
    Task<Document> Insert(InsertDocumentRequest request);
    Task<Document> Update(int id, UpdateDocumentRequest request);
    Task CompleteElaboration(int documentId, CompleteElaborationRequest request);
    Task MarkFileAsViewed(int documentId);
    Task UpdateElaborationFile(int documentId, File file);
    Task ApproveOrReproveRevision(int documentId, bool approve, RevisionApprovalRequest request);
    Task RequestRevision(int documentId, RequestRevisionRequest request);
    Task RejectRevisionRequestTransfer(int documentId, RejectRevisionRequestTransferRequest request);
    Task RejectRevisionRequest(int documentId, RejectRevisionRequestRequest request);
    Task StartRevision(int documentId, RevisionObservationMutation observation);
    Task TransferRevisionRequest(int documentId, TransferRevisionRequest request);
    Task CancelRevision(int documentId);
}
```

```
public DocumentWriteService(
    IDocumentRepository repository,
    IDocumentSpecification specification,
    RequestContext requestContext,
    IStorageClient storageClient,
    DocContext context,
    IEventBus eventBus,
    IQualyteamLogger logger,
    IFileService fileService,
    IMediator mediator,
    IPaymentGateway paymentGateway,
    IPdfConverter pdfConverter
)
{
    _repository = repository;
    _specification = specification;
    _requestContext = requestContext;
    _storageClient = storageClient;
    _context = context;
    _eventBus = eventBus;
    _logger = logger;
    _fileService = fileService;
    _mediator = mediator;
    _paymentGateway = paymentGateway;
    _pdfConverter = pdfConverter;
}
```

THE DEVELOPER'S CONFERENCE

Builders

THE DEVELOPER'S CONFERENCE

```
var documentService = new DocumentWriteService(unitOfWork, null, business, null, null, context, null, null, null, null);
```

```
var documentService = new DocumentWriteServiceBuilder()
    .WithUnitOfWork(unitOfWork)
    .WithBusiness(business)
    .WithContext(context)
    .Build();
```

THE DEVELOPER'S CONFERENCE

Asserções & Use Fluent Assertions

THE DEVELOPER'S CONFERENCE

Asserções únicas ou múltiplas?


É TRETA, É TRETA,
É TRETAAAAA

Asserções

Únicas

- Muitos testes para o mesmo cenário
- Análise mais rápida do real problema

Múltiplas

- Ajuda em testes mais complexos
- Mais difícil saber o que quebrou

THE DEVELOPER'S CONFERENCE

```
revision.Changes.Should().Be(requestPayload.Changes);
elaborationResult.Approved.Should().BeTrue("because we just completed the elaboration stage");
revision.Stages.Should()HaveCount(2, "because the next stage has been created");
createdStage.Order.Should().Be(2, "because it  happens after the elaboration");
createdStage.StageId.Should().Be(StageMother.ApprovalStageid, "because the next stage is the approval stage");
```

THE DEVELOPER'S CONFERENCE

```
documentFromDatabase
    .GetLastPublishedRevision()
    .GetOpenRevisionRequest()
    .ResponsibleGuid
    .Should()
    .Be(UserMother.MrCatraId, "because we replaced SteveJobs with MrCatra and SteveJobs was the requester of this revision");

documentFromDatabase
    .GetLastPublishedRevision()
    .GetClosedRevisionRequest()
    .ResponsibleGuid
    .Should()
    .Be(UserMother.SteveJobsId, "because we shouldnt change history");
```

THE DEVELOPER'S CONFERENCE

Como resultado

THE DEVELOPER'S CONFERENCE

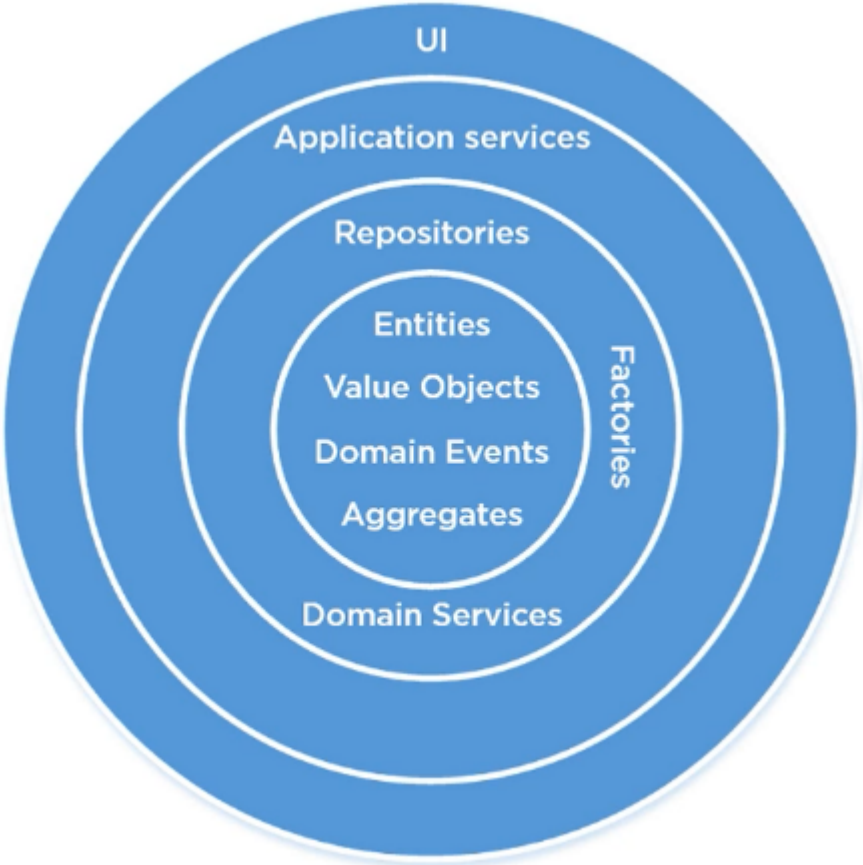
```
[Fact (DisplayName=@"
    GIVEN I have a document
    WHEN I update your information
    THEN the document must have the new information
")]
0 references | Run Test | Debug Test
public async void Update_ShouldUpdateDocument()
{
    var document = DocumentMother.DocumentWithOnePublishedAndOnePendingRevisionInApproval();

    context.Documents.Add(document);

    var mutation = DocumentMother.DocumentWithNoStages();

    context.SaveChanges();
    var documentService = new DocumentWriteServiceBuilder()
        .WithUnitOfWork(unitOfWork)
        .WithBusiness(business)
        .WithContext(context)
        .Build();

    var updatedDocument = await documentService.Update(document.Id, mutation);

    updatedDocument.Code.Should().Be(mutation.Code);
    updatedDocument.Title.Should().Be(mutation.Title);
    updatedDocument.SuggestedCode.Code.Should().Be(6);

    var firstStage = updatedDocument.Stages.First();
    firstStage.DaysToDueDate.Should().Be(27);
    firstStage.Responsibles.Should().HaveCount(1);
    firstStage.Responsibles.First().ResponsibleGuid.Should().Be(UserMother.PabloEscobarId);

    var lastStage = updatedDocument.Stages.Last();
    lastStage.DaysToDueDate.Should().Be(7);
    lastStage.Responsibles.Should().HaveCount(1);
    lastStage.Responsibles.First().ResponsibleGuid.Should().Be(UserMother.MrCatraId);
}
```

THE
DEVELOPER'S
CONFERENCE

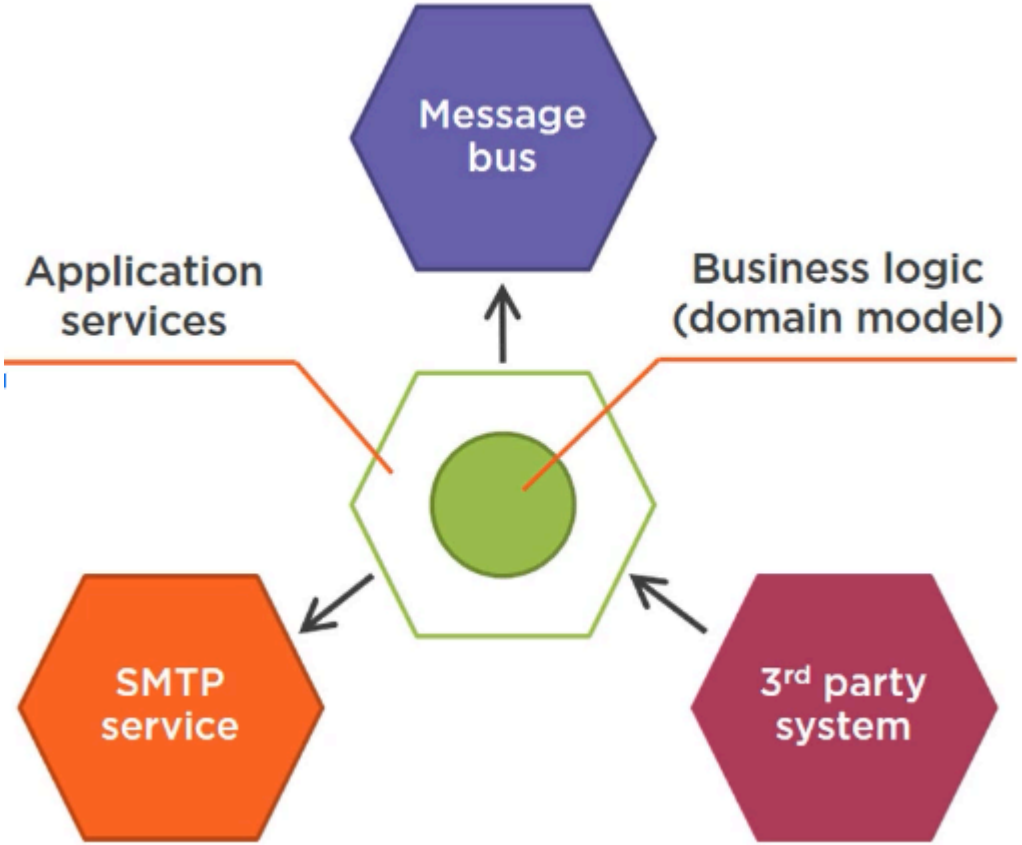
Melhoramos... mas queríamos ir além

THE DEVELOPER'S CONFERENCE

Arquiteturas

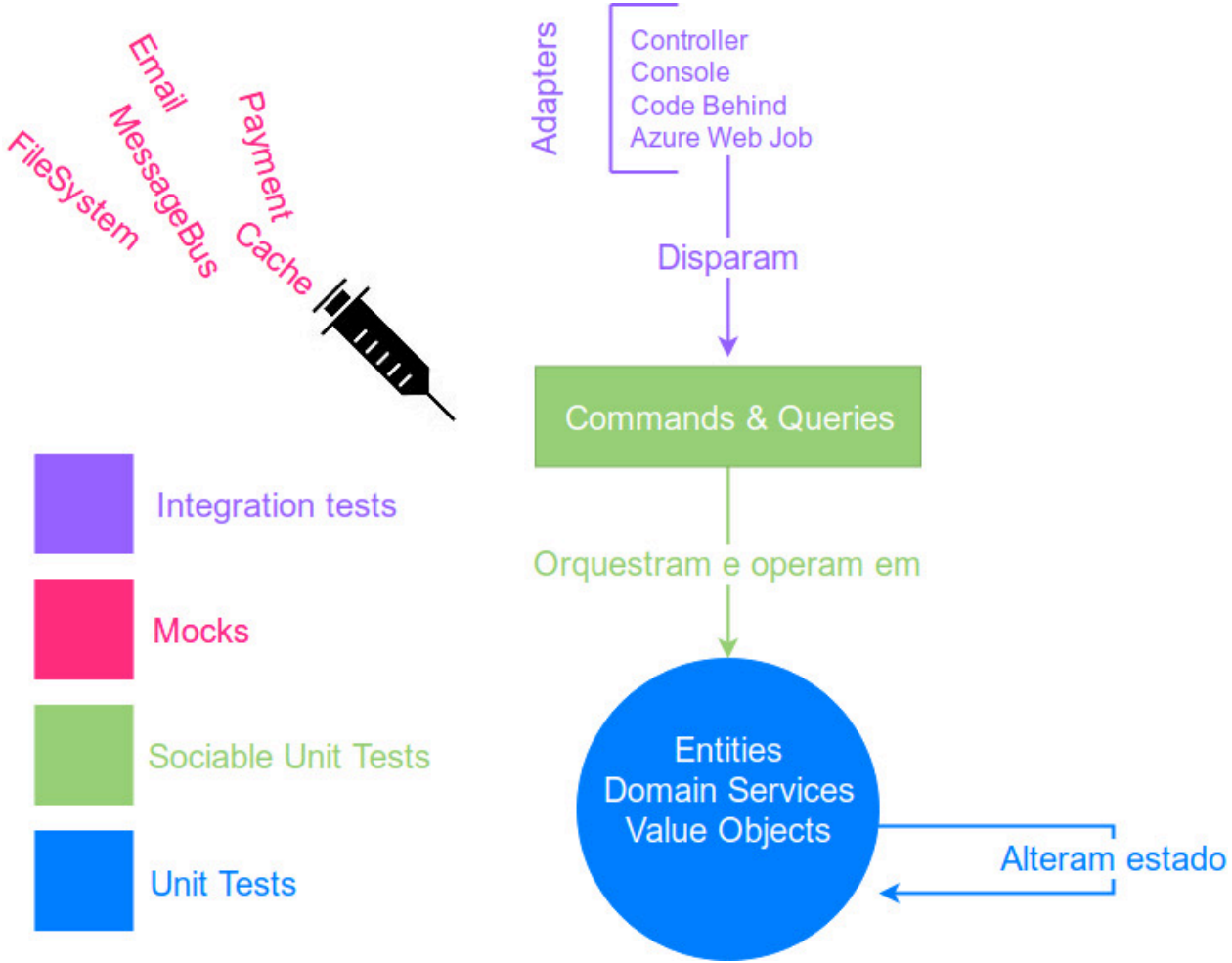


THE DEVELOPER'S CONFERENCE


Message bus
Application services
Business logic (domain model)
SMTP service
3rd party system

THE DEVELOPER'S CONFERENCE

Proteja seu dominio das **extremidades**



THE DEVELOPER'S CONFERENCE

THE DEVELOPER'S CONFERENCE

Classes de serviço de aplicação não são coesas

THE DEVELOPER'S CONFERENCE

Use MediatR para orquestração de casos de uso

```
public class AplicarAVagaCommandHandler : IRequestHandler<AplicarAVagaCommand>
{
    public AplicarAVagaCommandHandler(
        IRepositorioDeCandidatos _repositorioDeCandidatos,
        IRepositorioDeVagas _repositorioDeVagas,
        IEventBus _eventBus,
        ILogger _logger
    ){
        //Atribui dependencias
    }

    public async Task Handle(AplicarAVagaCommand comando)
    {
        var candidato = await _repositorioDeCandidatos.Encontrar(comando.IdCandidato);
        var vaga = await _repositorioDeVagas.Encontrar(comando.IdVaga);

        candidato.Aplicar(vaga);

        _eventBus.Publicar(candidato.EventosLevantados);
        _logger.Info($"Candidato {candidato.Nome} aplicou a vaga {vaga.Identificacao}");
    }
}
```

THE DEVELOPER'S CONFERENCE

Encapsulamento

THE DEVELOPER'S CONFERENCE

```
[Fact]
public async void Update_ShouldUpdateDocument()
{
    //Arrange
    var context = InMemoryDocContextFactory.Create();
    var unitOfWork = new DocUnitOfWork(context, _logger);

    var elaboration = new Stage() { Type = StageType.Elaboration, Name = "Elaboration" };
    var approval = new Stage() { Type = StageType.Approval, Name = "Approval" };
    var consensus = new Stage() { Type = StageType.Approval, Name = "Consensus" };

    context.Stages.Add(elaboration);
    context.Stages.Add(approval);
    context.Stages.Add(consensus);

    var oldCategory = new Category()
    {
        Initials = "CAT",
        Name = "Category",
        Stages = new List<CategoryStage>() {
            new CategoryStage() { Order = 1, Stage = elaboration },
            new CategoryStage() { Order = 2, Stage = approval },
        },
    };
    var newCategory = new Category()
    {
        Initials = "DOG",
        Name = "Dogcary",
        Stages = new List<CategoryStage>() {
            new CategoryStage() { Order = 1, Stage = elaboration },
            new CategoryStage() { Order = 2, Stage = consensus },
        }
    };

    context.Categories.Add(oldCategory);
    context.Categories.Add(newCategory);
```

THE DEVELOPER'S CONFERENCE

```
public class Category
{
    public int Id { get; set; }
    public string Initials { get; set; }
    public string Name { get; set; }
    public virtual ICollection<CategoryStage> Stages { get; set; }
}
```

```
public class Category
{
    public string Initials => Name.Substring(0, 2).ToUpper();
    public string Name { get; protected set; }
    public virtual ICollection<CategoryStage> Stages { get; }

    private Category(string name, ICollection<CategoryStage> stages)
    {
        Name = name;
        Stages = stages;
    }
    protected Category() { }

    public void ChangeName(string name) => Name = name;

    public static Category Basic(string name, string initials)
    => new Category(name, initials, Stages.Stages());

    public static Category Enterprise(string name, string initials)
    => new Category(name, initials, Stages.Enterprise());
}
```

THE DEVELOPER'S CONFERENCE

THE
DEVELOPER'S
CONFERENCE

```
var context = InMemoryDocContextFactory.Create();

context.Categories.Add(Category.Basic("Dogcary"));
context.Categories.Add(Category.Enterprise("Category"));
```

THE
DEVELOPER'S
CONFERENCE

The **new** keyword considered harmful

THE DEVELOPER'S CONFERENCE

Mocks

THE DEVELOPER'S CONFERENCE

Setups diminuem a velocidade de escrita

THE DEVELOPER'S CONFERENCE

```
var specificationMock = new Mock<ISpecification>();
specificationMock.Setup(s => s.IsSatisfiedBy(It.IsAny<Document>())).Returns(true);

var waterMarkStrategyMock = new Mock<IWaterMarkStrategy>();
waterMarkStrategy.Setup(w => w.Generate()).Returns(Watermark.None);

var mapperMock = new Mock<IMapper>();
mapperMock.Setup(s => s.Map(It.IsAny<Document>())).Returns(new ControlledCopyDTO(sut.Id, sut.Name))

var handler = new PrintControlledCopyCommandHandler(
    specificationMock.Object,
    waterMarkStrategyMock.Object,
    mapperMock.Object
);
```

THE
DEVELOPER'S
CONFERENCE

```
var handler = new PrintControlledCopyCommandHandler(
    new DocumentIsPrintable(),
    new NullWaterMarkStrategy(),
    AutoMapperFactory.Create(),
);
```

THE DEVELOPER'S CONFERENCE

Te prendem a um detalhe de implementação

THE DEVELOPER'S CONFERENCE

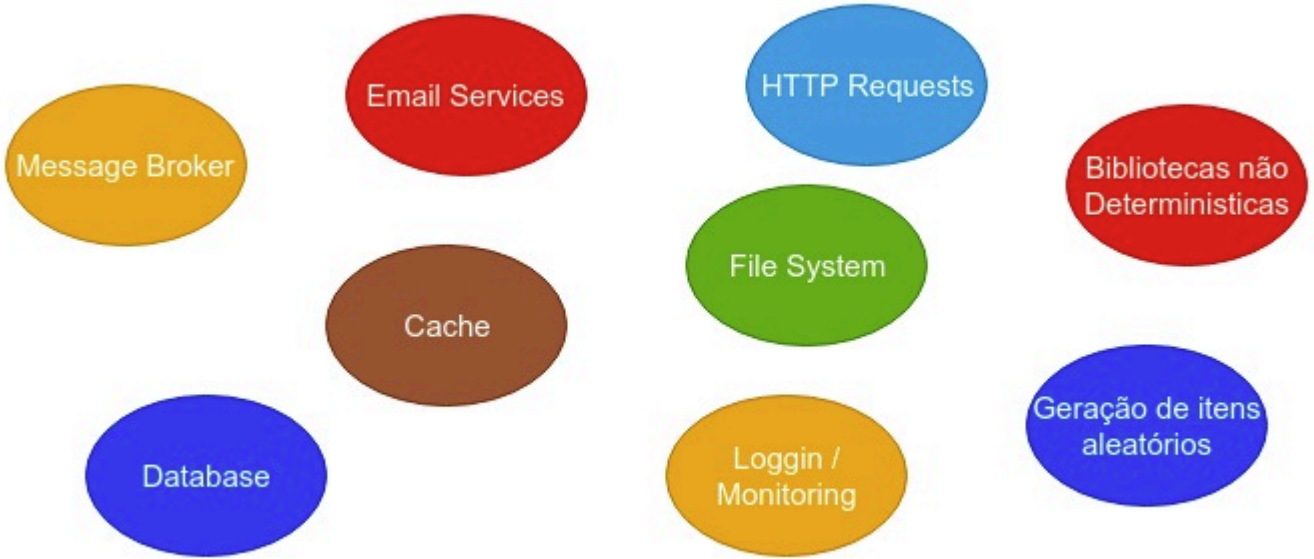
```
var specificationMock = new Mock<ISpecification>();
specificationMock.Setup(s => s.IsSatisfiedBy(It.IsAny<Document>)).Returns(true);

var handler = new PrintControlledCopyCommandHandler(specificationMock.Object);

var result = await handler.Handle(new PrintControlledCopyCommand(revisionId), CancellationToken.None);

result.Status.Should().Be(Result.Success)
```

THE DEVELOPER'S CONFERENCE

Diminuem a confiança de que o código funciona

THE DEVELOPER'S CONFERENCE

Faça mocks de **dependências instáveis**


Email Services
HTTP Requests
Message Broker
Bibliotecas não Deterministicas
File System
Cache
Geração de itens aleatórios
Database
Loggin / Monitoring

THE DEVELOPER'S CONFERENCE

Nossa concepção de cobertura também foi alterada

100% Coverage =

THE DEVELOPER'S CONFERENCE

É o que tem pra hoje.

THE DEVELOPER'S CONFERENCE

Testes automatizados fazem parte da cultura **ágil**

THE DEVELOPER'S CONFERENCE

Seu time deve **amar** a suite de testes

THE DEVELOPER'S CONFERENCE

Devem **auxiliar** a refatoração

THE DEVELOPER'S CONFERENCE

Refatorar é lindo.
Manter funcionando é arte

Obrigado

THE DEVELOPER'S CONFERENCE

Contatos

/in/leonardo-prange/

/in/marconicolodi/

medium.com/qualyteam-engineering

THE DEVELOPER'S CONFERENCE