Organizing Persistence Logic to Support DDD Design

Julie Lerman

EF Core Expert and Serial DDD Advocate

@julielerman | thedatafarm.com

Drawing from experience across the DDD community

Additional Guidance in this Path

EF Core 6 Encapsulation

Vladimir Khorikov

Overview

Repositories in DDD and in EF Core

The repository designed for our solution

What about searches?

Communicating between bounded contexts

Mapping guard keys in EF Core 6 & 7

Repositories in DDD and EF Core

Repository

**Representation of all objects of a certain type as a conceptual set …
like a collection with more elaborate querying capability.**

Eric Evans: Domain-Driven Design, 2003 Addison-Wesley

Repositories are a Key Asset of Tactical Design

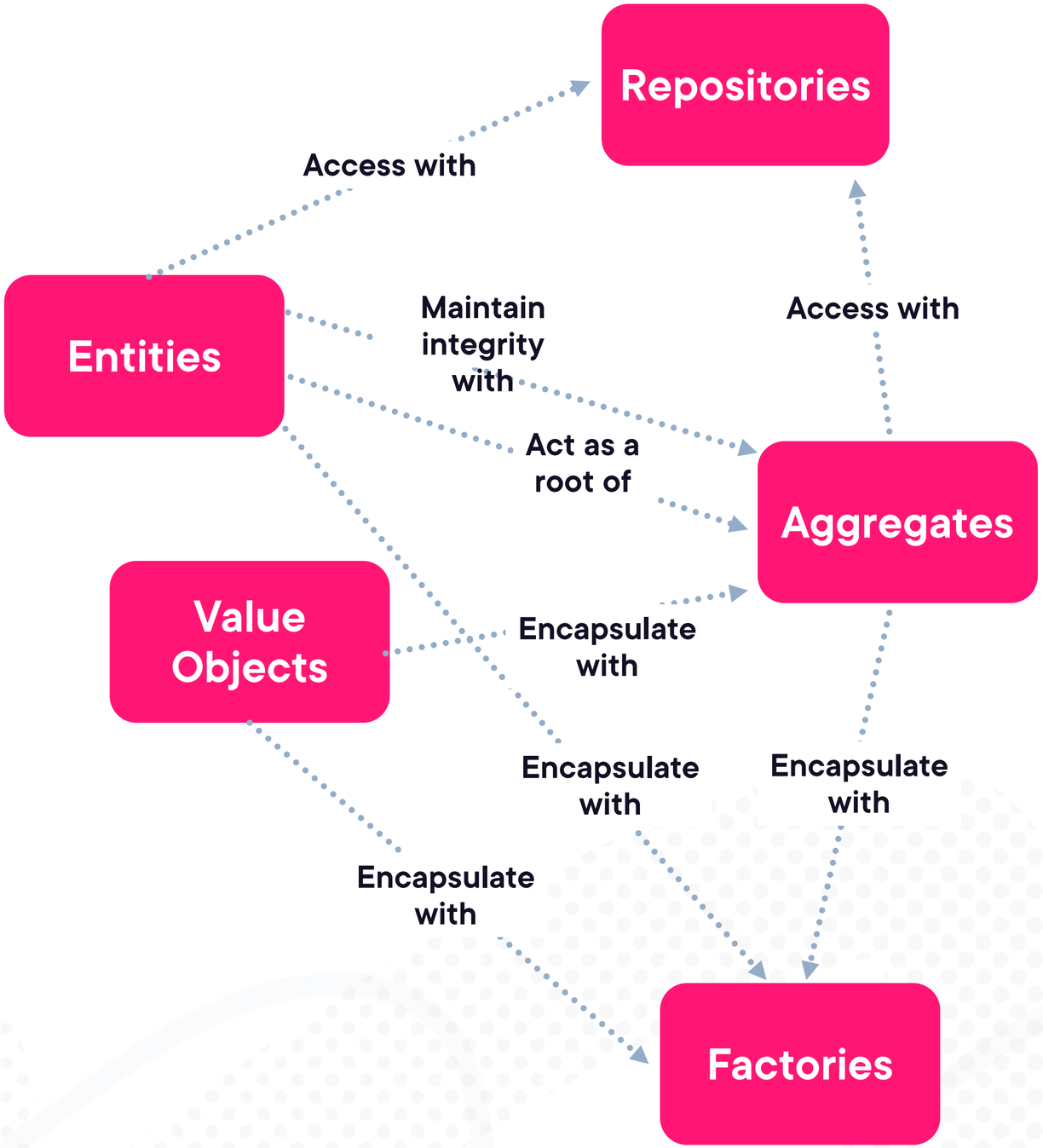
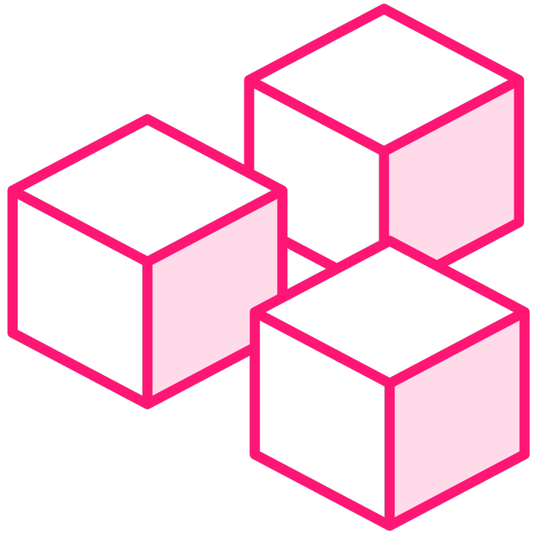
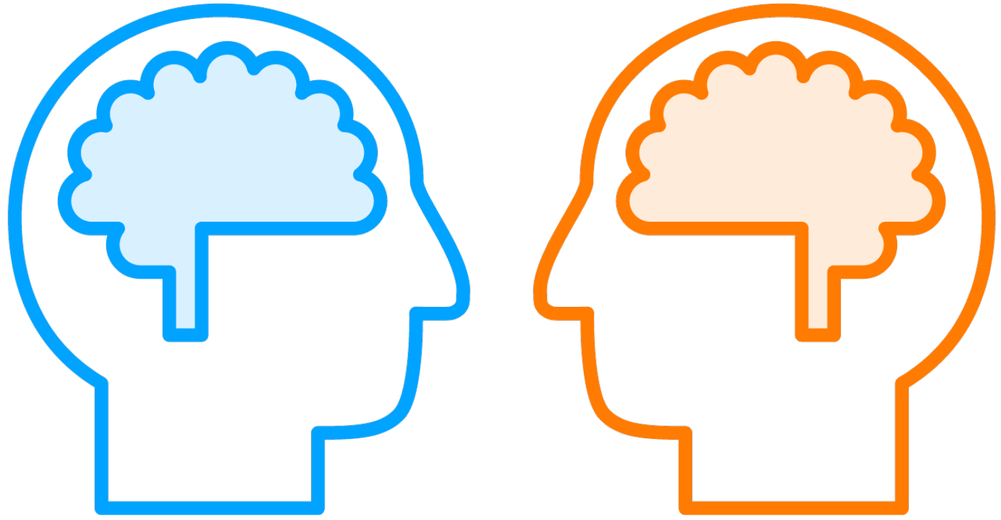


Image based on DDD Mind Map in
Domain-Driven Design, Eric Evans, Addison-Wesley, 2003

Choosing Your Repository Pattern



Many Options



Many Worthy Opinions

Repository with EF Core Considerations

DbSet is a repository.
No need to encapsulate it.

Generic repository

Intelligent repository
(Steve Smith's Specification repo*)

Customized repository-like service

*An EF Core version within https://github.com/ardalis/Specification

Considerations for Keeping the Aggregate Valid

Should you ever retrieve or update a version on its own?

What if it’s read-only?

..or a contract with only the current version?

What are the needs described by the domain experts?

Additional Guidance in this Path

EF Core 6 Encapsulation

Vladimir Khorikov

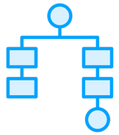
My Original Deep Dive into The Great Repository Debate

EF6 in the Enterprise (2016)

Julie Lerman

Understanding Needs & Considerations for Our Contract Service

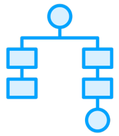
Service Needs for Our App



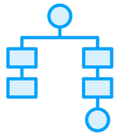
Add a new contract aggregate

Flexible search of contract

Find selected contract aggregate by key



Update a contract aggregate



Add contract versions*

*Cosmos provider does not currently support partial updates

“We will never delete a contract”

Some Considerations for Our Design

It makes sense to retrieve a contract with only its latest version and be able to add versions and keep the aggregate valid.

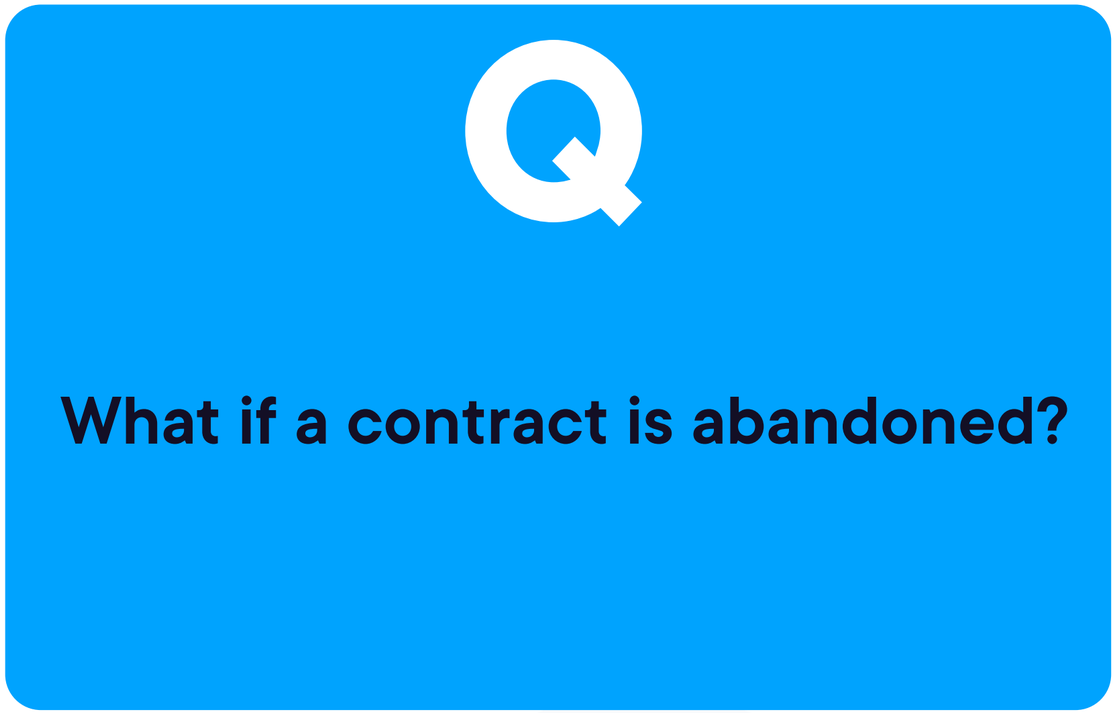
Will retrieving the full aggregate cause a performance issue?

What about looking at older versions...read only?

At what point are we asking things of the Contracting BC that are really for reporting...

... or require a focused models and DbContext within the same bounded context?

Because We Don’t Delete Contracts …

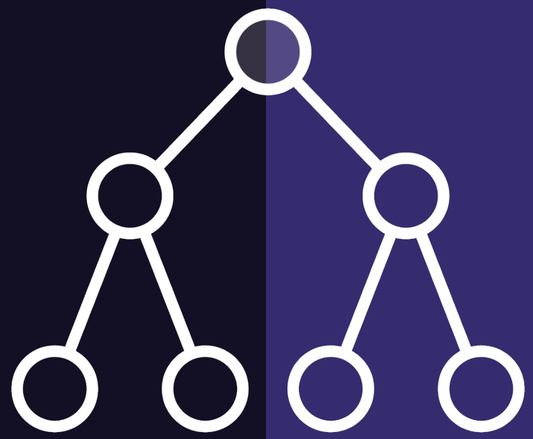


YAGNI

You Ain’t Gonna Need It

Stay focused on what you learned from the domain experts, rather than planning for every possible use case. Evolve later as needed.

Exploring the Contract Service



```
_context.Entry(contract).State
      = EntityState.Modified
```

◀ Only Contract gets marked Modified

```
_context.Entry(version).State
      = EntityState.Added
```

◀ Only the ContractVersion is Added

```
_context.Add(version)
```

◀ ContractVersion, Authors & Specs are Added

You May Be Wondering ....

Why are older versions ignored in the update?

Aggregate enforces the rule that old versions can never be edited.

In DDD, clarify and simplify domain problems

Possible extra effort in infrastructure

Less side effects via the domain models

Smoother collaboration with domain experts

```
public void
 FinalVersionSignedByAllParties()
{
   Completed = true;
   CompletedDate= DateTime.Now;
   FinalVersionId = CurrentVersionId;

   var contractSignedEvent
    = new ContractSignedEvent(
    CurrentVersion(), CompletedDate);
   Events.Add(contractSignedEvent);
}

 private void
   ContractContext_SavedChanges(
    object? sender,
    SavedChangesEventArgs e)
{  //handle any events held by entities
}
```

- **Finalizing contract will trigger an event!**

- **We'll be publishing events after changes are successfully saved to the database**

```
modelBuilder.Entity<Contract>().Navigation(c => c.Versions).AutoInclude();
```

AutoInclude Mapping Introduced in EF Core 6

Will always eager load the specified navigations
Does not align with the needs of the Contracting bounded context.

Testing the Contract Service

Implementing Contract Search

Do Searches Belong in the Aggregate-Based Repo?

What are results?

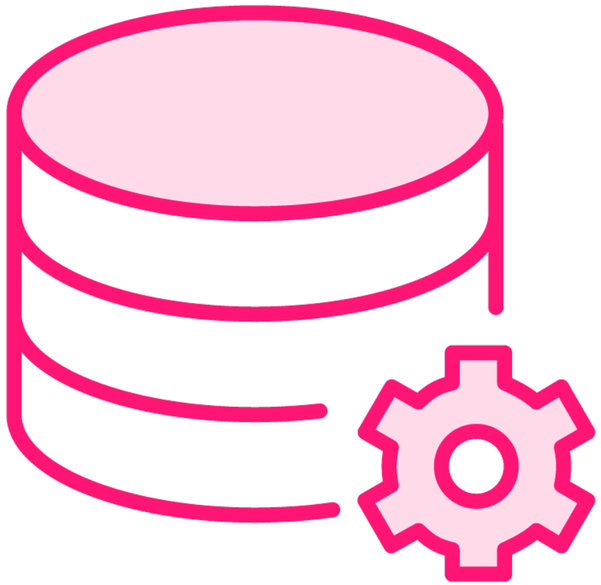
List [of Id & highlights] from which to select one contract and retrieve it

Can be difficult to express

Traversing into dependents and value objects

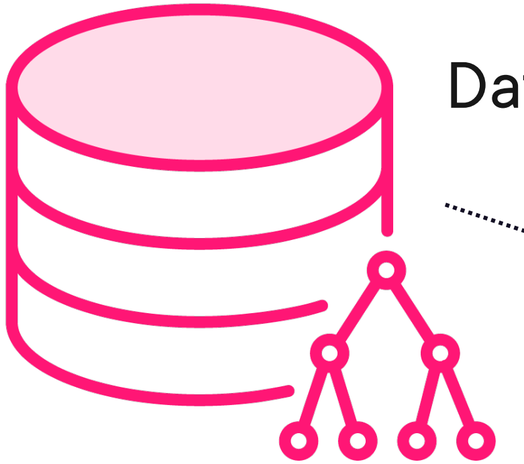
Recommendation

Create a separate read-only DbContext & search class designed for Contract filtering.



Leverage the database (and DB smarties) to implement the complexity of these searches

Constructing Search for Contract Pick Lists

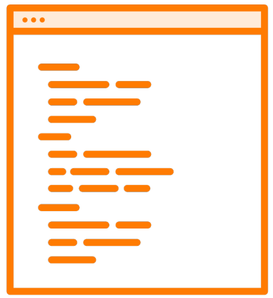


Database View: Exposes search columns & pick list data

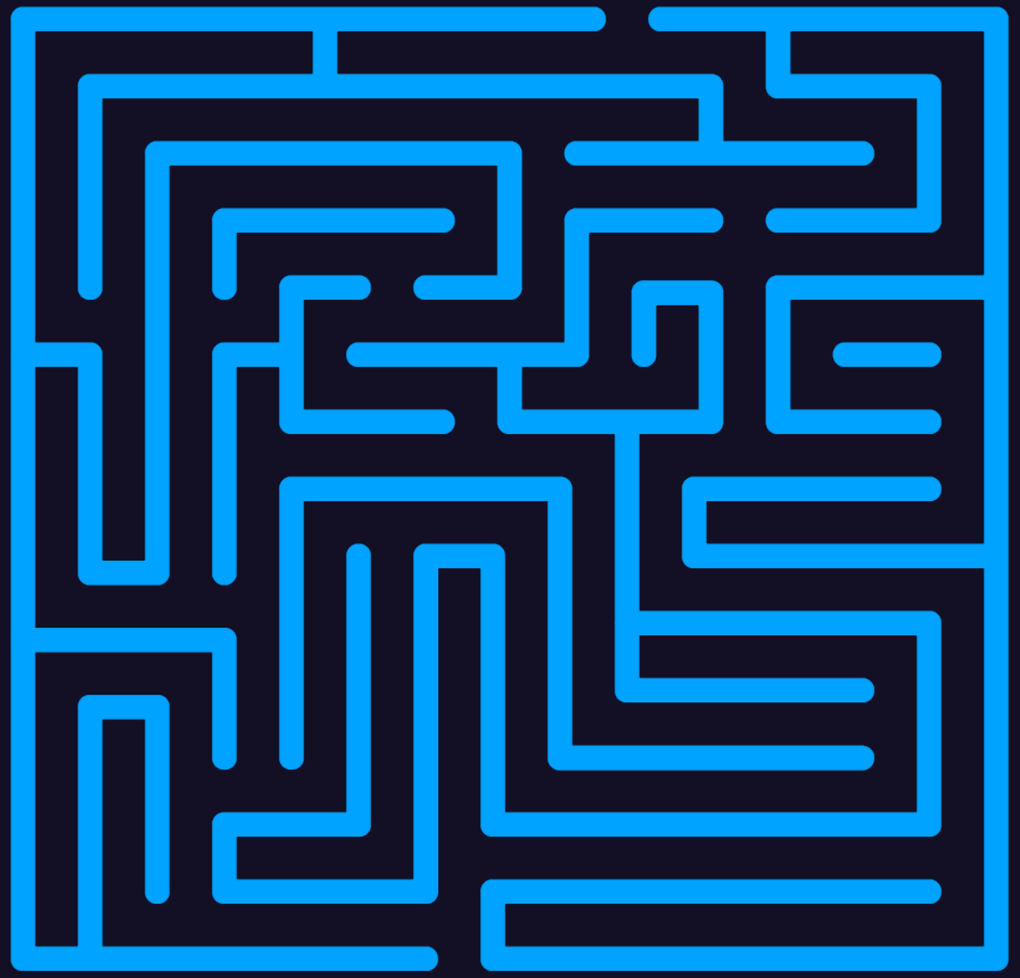
Stored Procedures to query from the view, filtering on LastName, Contract date, etc.

Return ContractId + string of contract highlights

SearchResults model & DbContext

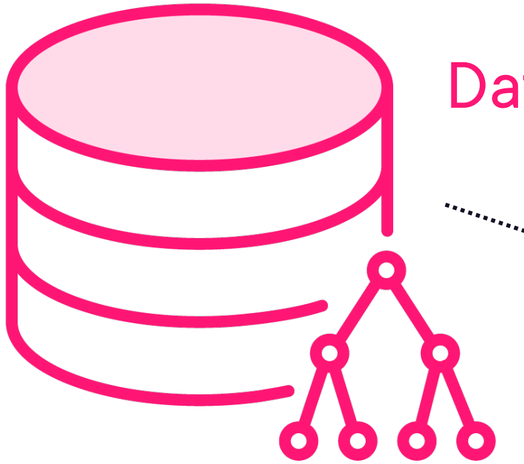


Search logic executes the stored procs & return lists of SearchResults



It was soooooo hard!

Constructing Search for Contract Pick Lists

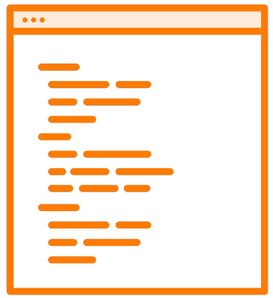


Database View: Exposes search columns & pick list data

Stored Procedures to query from the view, filtering on LastName, Contract date, etc.

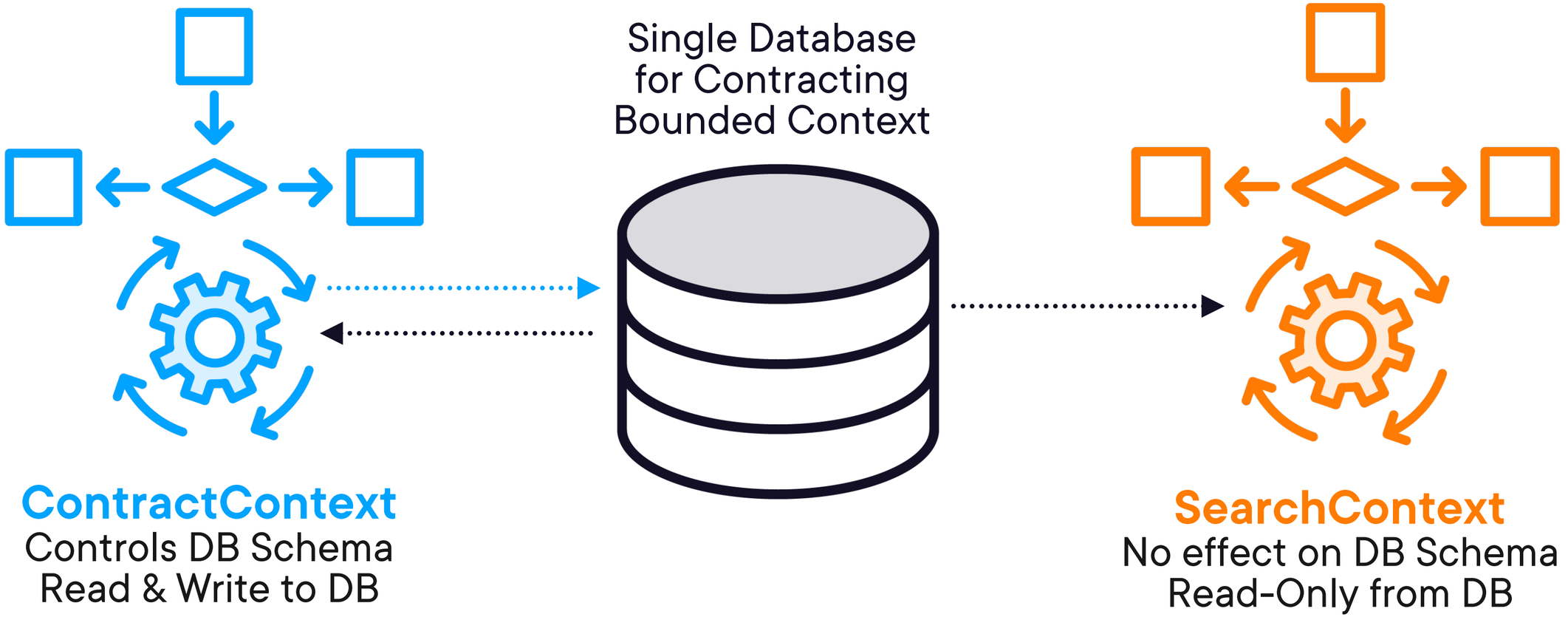
Return ContractId + string of contract highlights

SearchResults model & DbContext



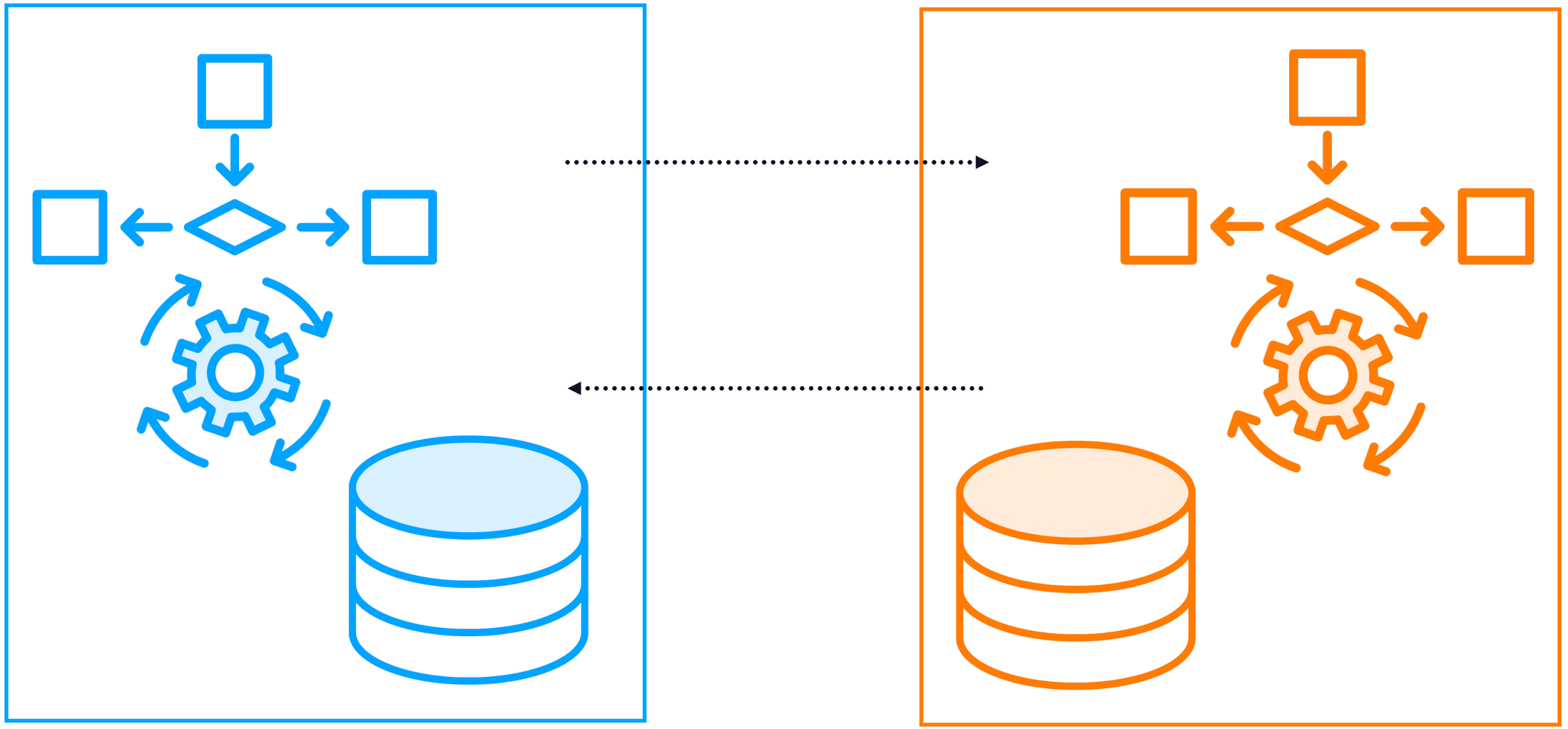
Search logic executes the stored procs & return lists of SearchResults

Search & Contract Contexts Use the Same Database

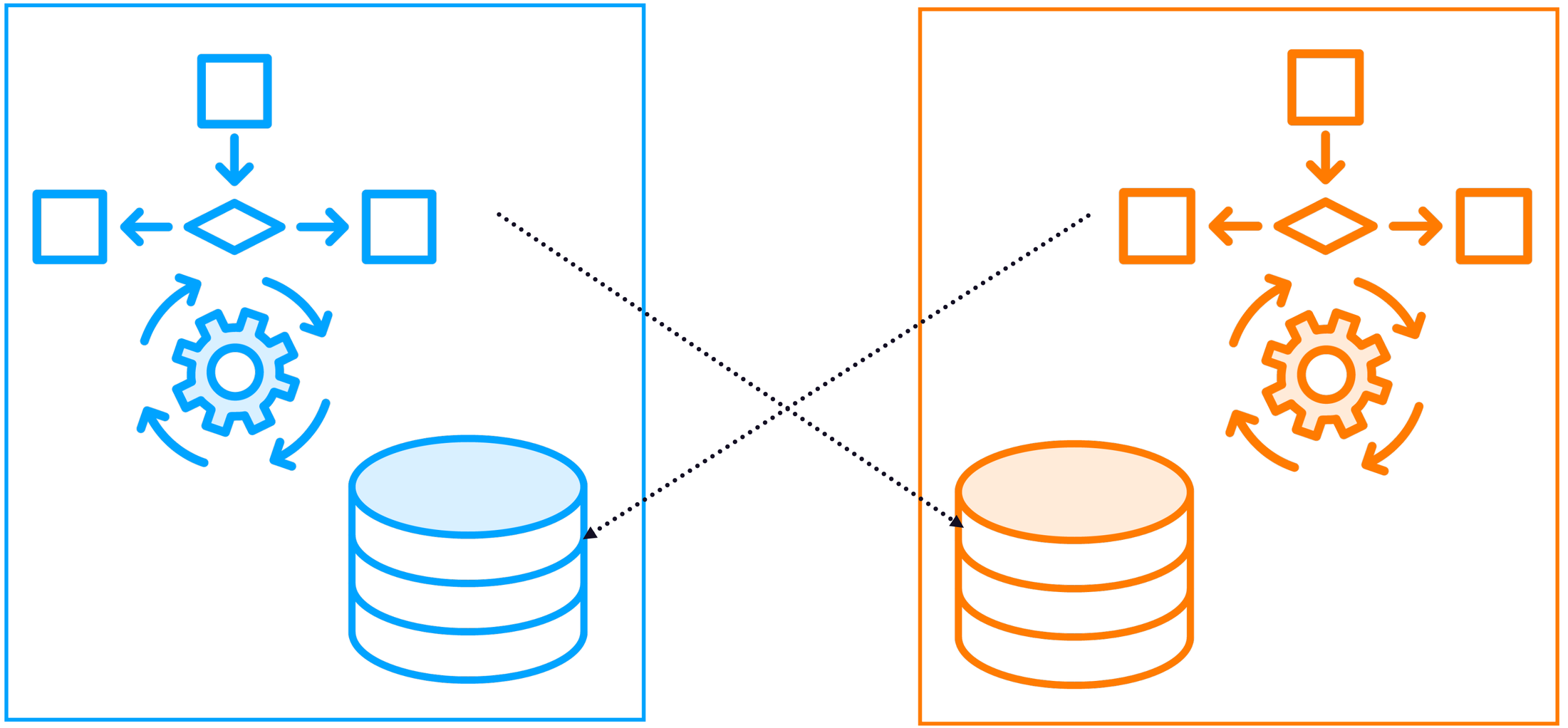


Communicating Between Bounded Contexts

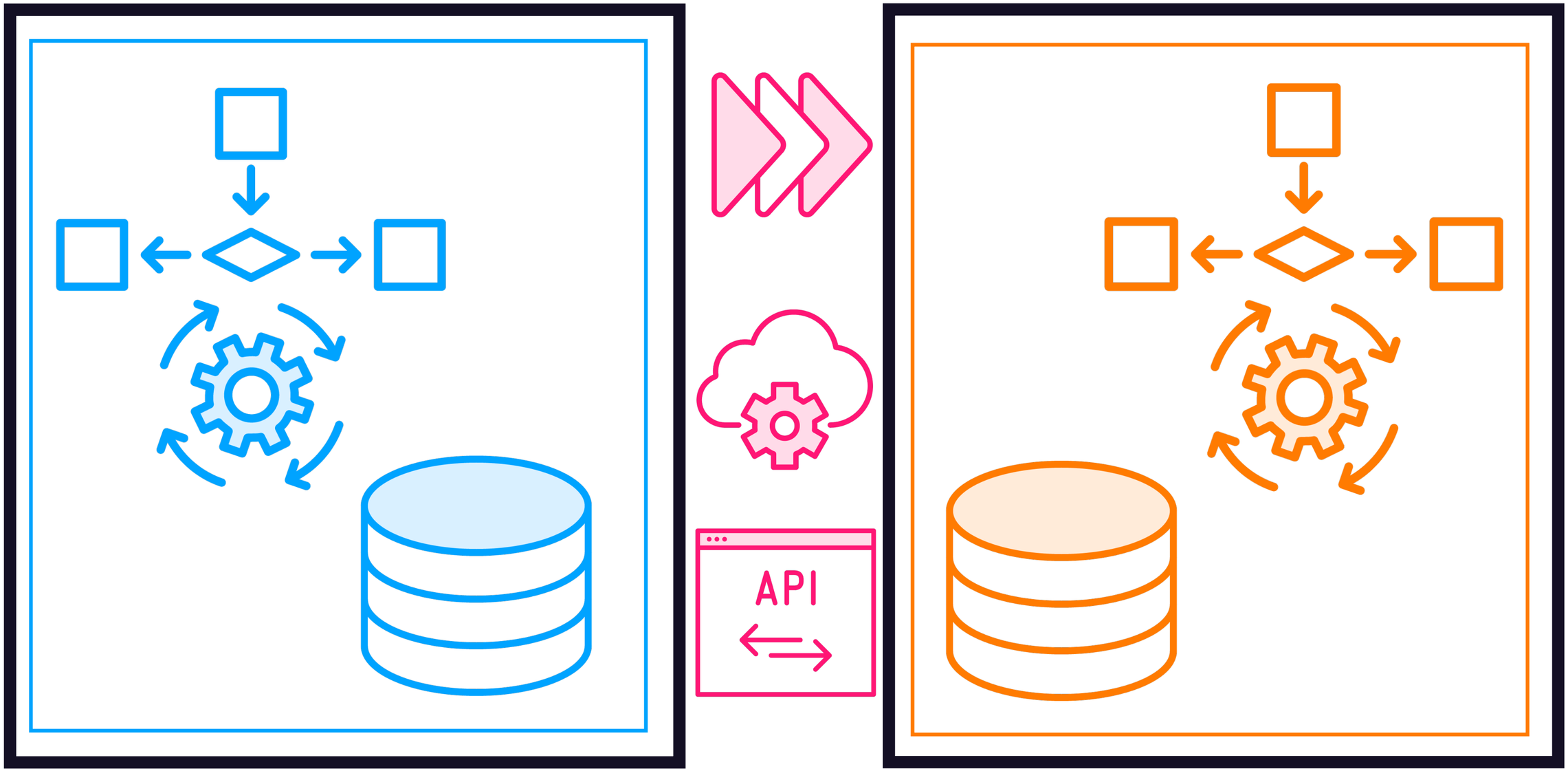
The Apps for Our Bounded Contexts



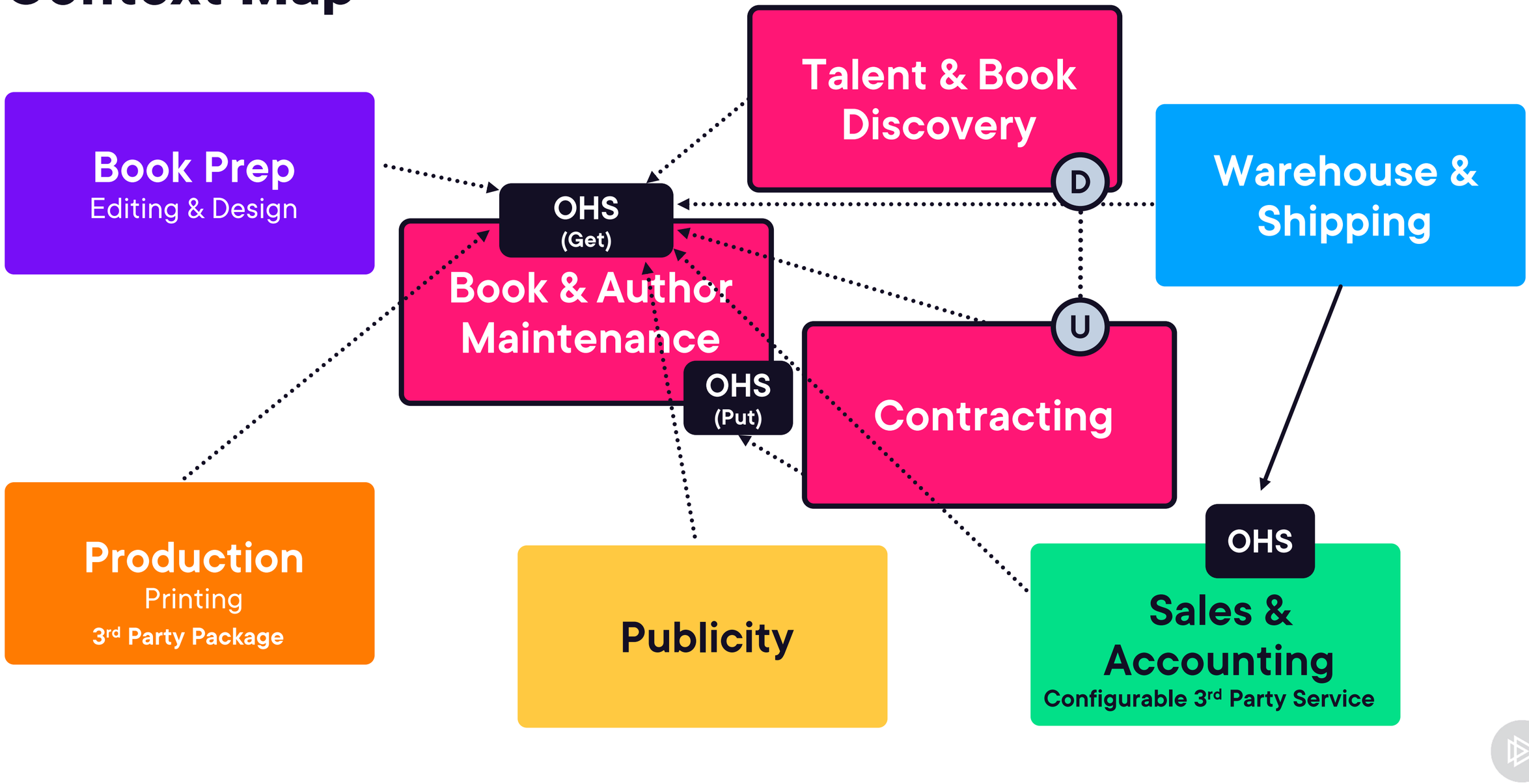
The Apps for Our Bounded Contexts



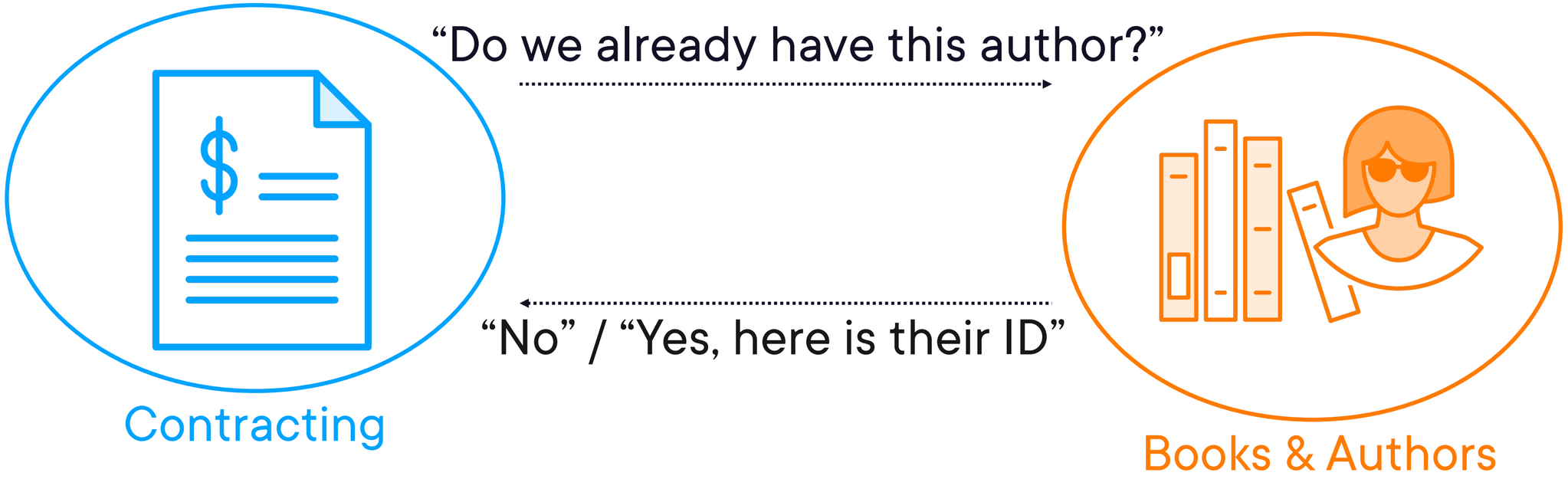
Various Tried and True Ways to Communicate



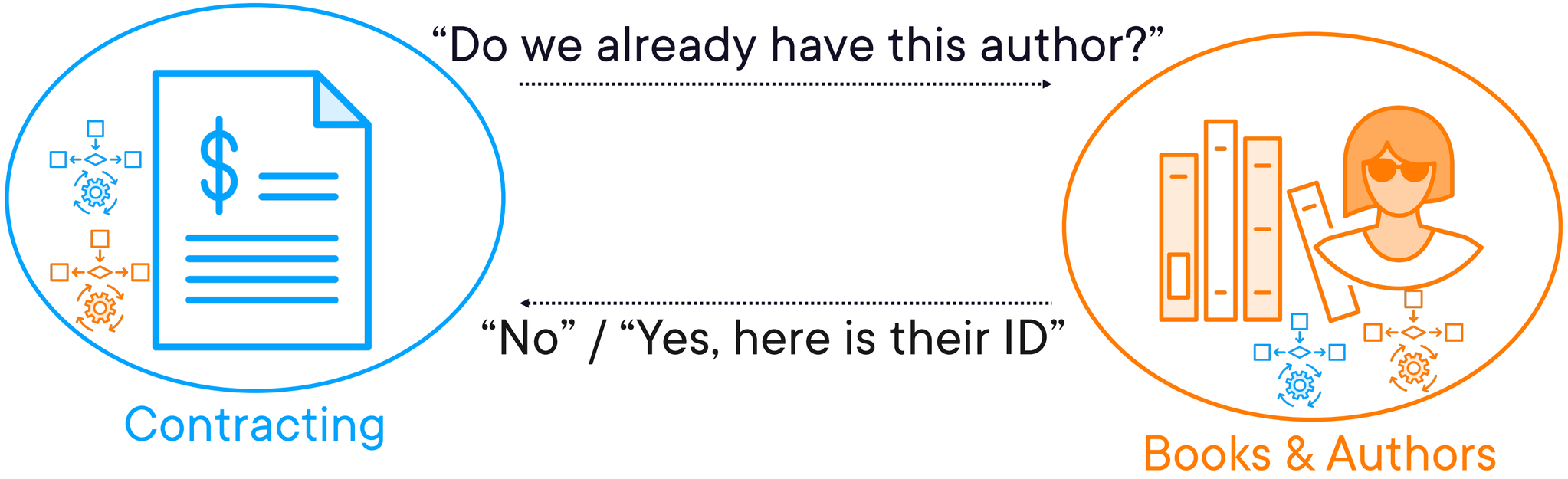
Context Map


Book Prep
Editing & Design
Talent & Book Discovery
D
Warehouse & Shipping
OHS
(Get)
Book & Author Maintenance
OHS
(Put)
U
Contracting
Production
Printing
3rd Party Package
Publicity
OHS
Sales & Accounting
Configurable 3rd Party Service

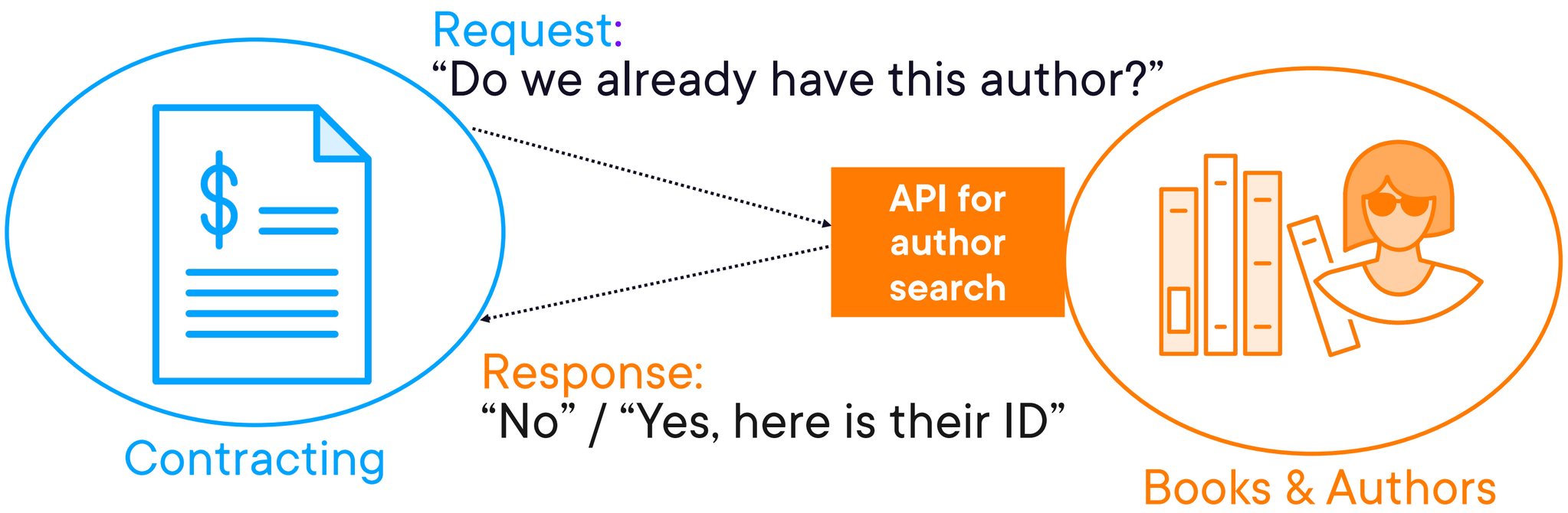
Checking for Existing Authors



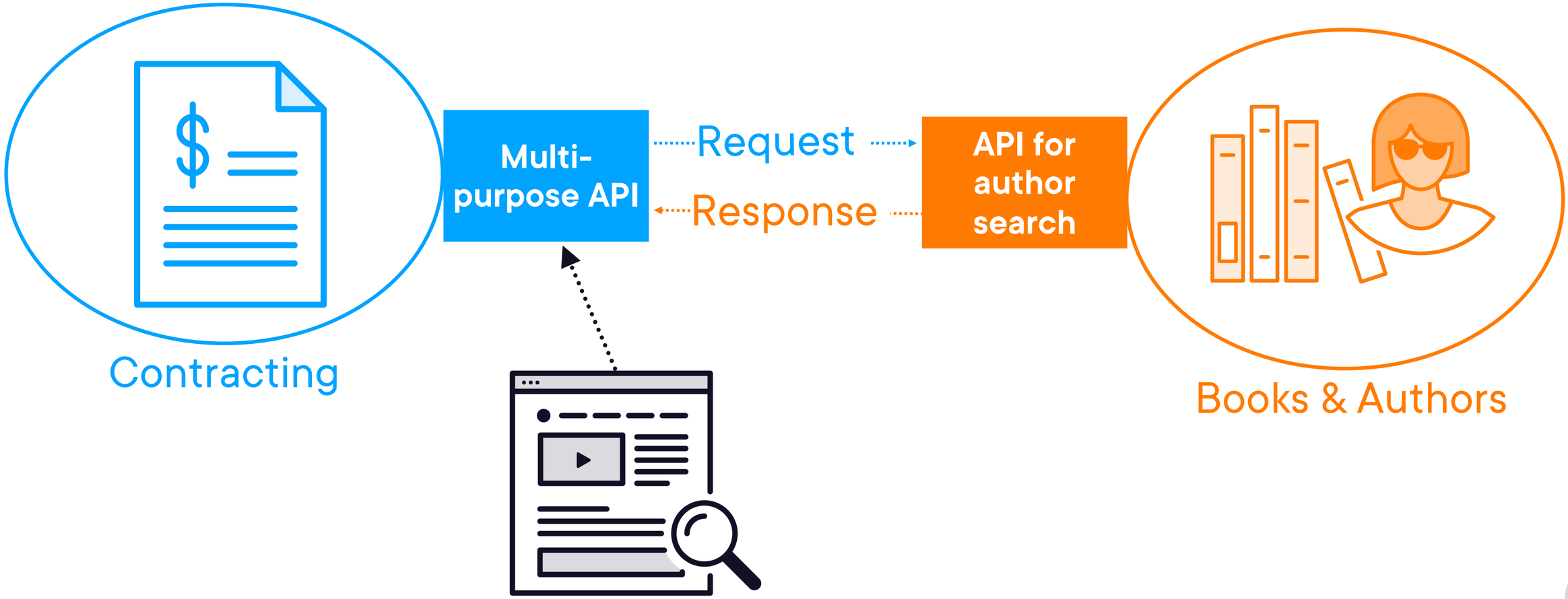
Checking for Existing Authors



Checking for Existing Authors

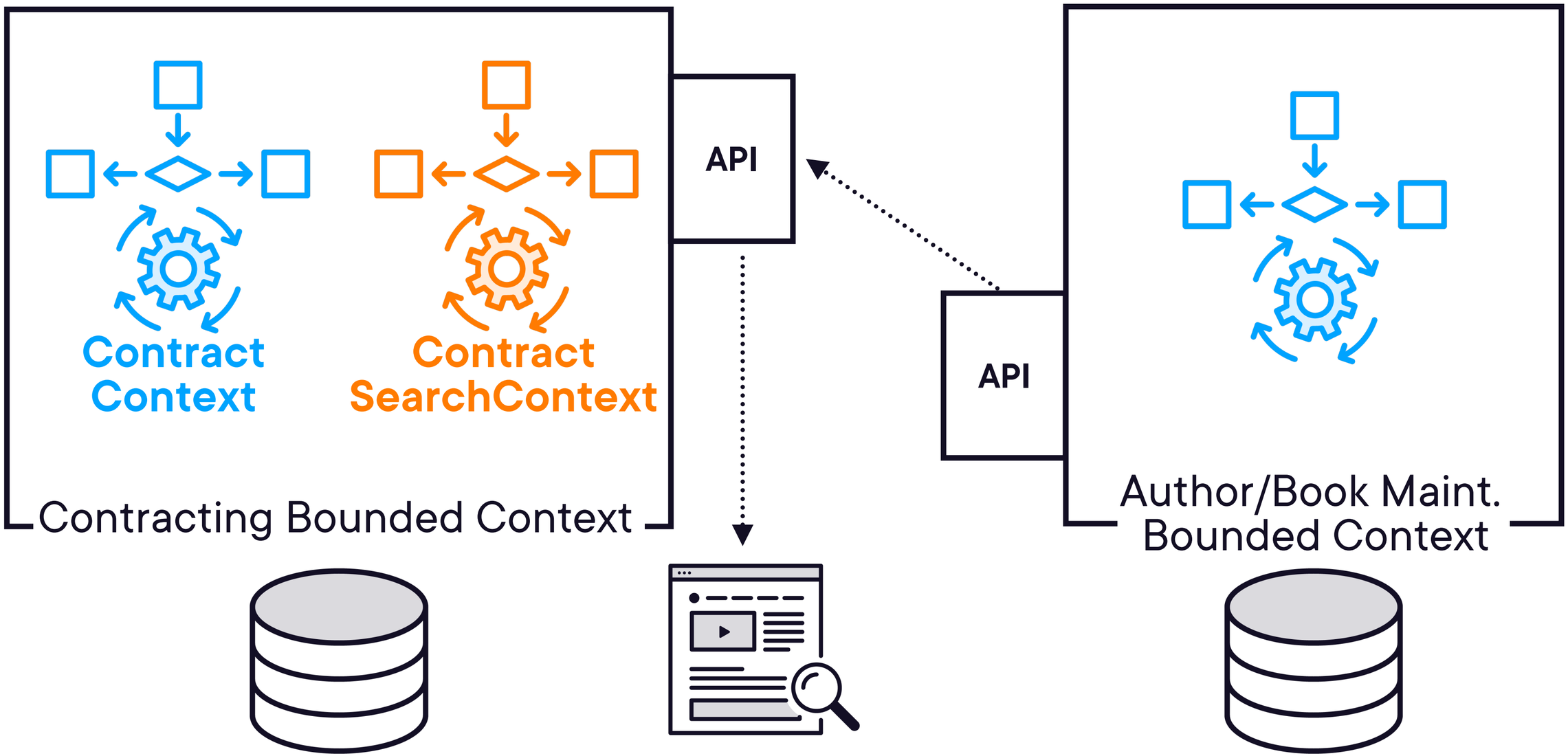


Checking for Existing Authors When Building the UI

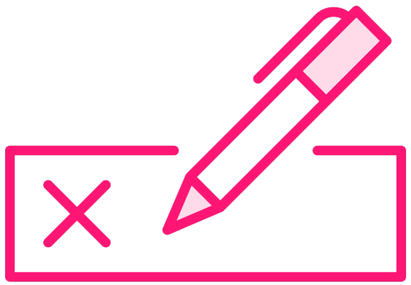


Leveraging SaveChanges in an Events & Messaging Workflow

High-Level View of Existing Architecture



Contract Signing Triggers an Event [High Level Workflow]

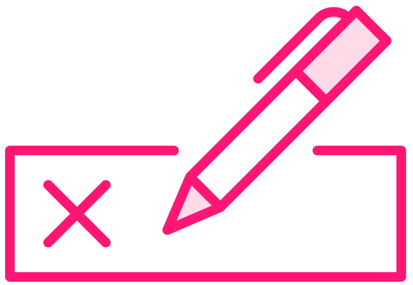


Contract signed

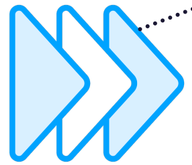
Book & [new] authors added to Maintenance BC

Send email to author(s)

Contract Signing Triggers an Event [High Level Workflow]



Contract signed



Book & [new] authors added to Maintenance BC via message queue

Send email to author(s) via local service

No Message Queue in This Demo

- Contracting project publishes an event.
- AuthorBook project has logic to consume message
- There is no plumbing for the message queue

Domain-Driven Design Fundamentals

by Steve Smith and Julie Lerman

Course Overview		1m 51s
Introducing Domain-Driven Design		24m 37s
Modeling Problems in Software		45m 12s
Elements of a Domain Model		32m 21s
Understanding Value Objects & Services in the Model		22m 55s
Tackling Complexity with Aggregates		34m 40s
Working with Repositories		49m 56s
Adding in Domain Events and Anti-corruption Layers		29m 41s
Evolving the Application Easily Thanks to DDD		46m 44s
Introduction and Overview		54s
Reviewing Our Current System Design		1m 47s
Addressing a New Feature with the Domain Expert		2m 0s
Planning Our Implementation Steps		1m 17s
Introducing Message Queues	✓	3m 18s
Sending a Message to the Queue	✓	2m 34s
Reading From the Message Queue and Acting on the Message	✓	3m 46s

Course authors

Steve Smith

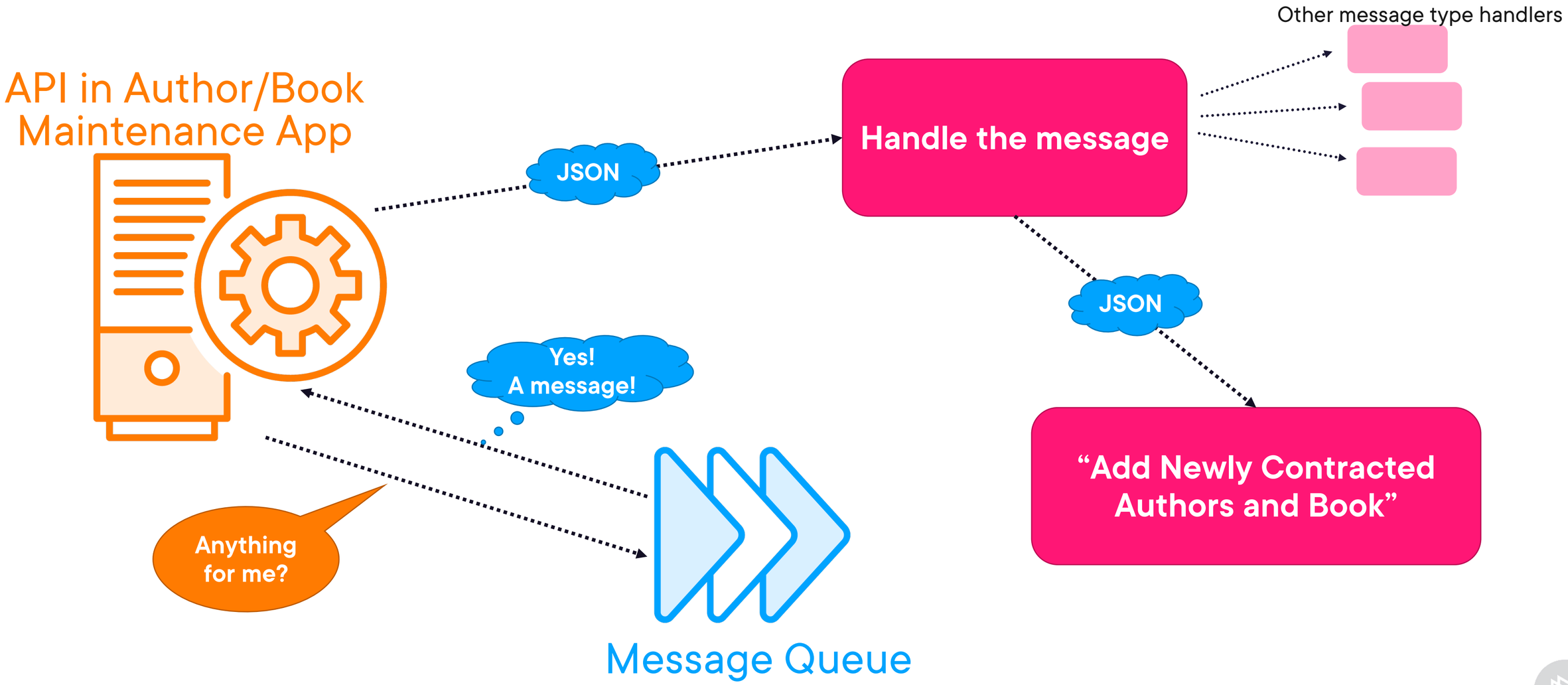
Julie Lerman

Course info

Level	Beginner
Rating	★★★★½ (347)
Duration	4h 48m
Released	13 May 2021

Share course

Responding to a Message from Another Bounded Context



Mapping Guarded Keys

Why I Didn’t Use This Pattern in the Pub Demo

Doesn’t apply to every solution

Limited support in EF Core 6

Expanded support in EF Core 7

Guarding Identities

Encapsulating identities into their own classes helps avoid mixing up their values with any other random scalar value

More about Value Conversion

EF Core 6 Fundamentals

Julie Lerman

Mapping Guarded Keys...

...when values are assigned in code

EF Core 6

...when values are database generated, too!

EF Core 7

Mapping Guarded Keys...

EF Core 6	EF Core 7
...when values are assigned in code	...when values are database generated, too!
Need ValueConverters	Need ValueConverters & ValueGeneratedOnAdd

Review

Generic repos are okay for CRUD, but a focused service is a better option for complex needs.

Search is best handled separately from your aggregates.

Expose & consume APIs to share data across boundaries.

Leverage event handlers and message queues to share events across boundaries.

EF Core 6 & 7 provide support for mapping guard keys.

Resources from This Module

EF Core Specification Repository by Steve Smith
github.com/ardalis/Specification

Effective Aggregate Design article series by Vaughn Vernon
dddcommunity.org/library/vernon_2011/

Domain-Driven Design Fundamentals (on Pluralsight)

EF Core 6 and Domain-Driven Design

Julie Lerman

EF Core Expert and Serial DDD Advocate

@julielerman | thedatafarm.com