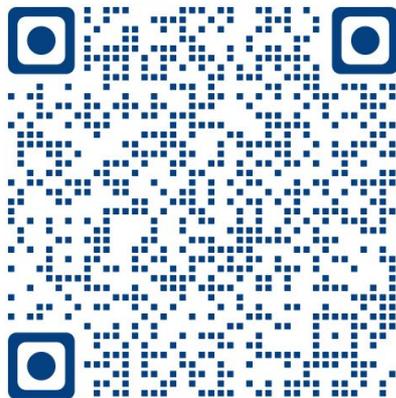


**Real World
Project
Journeys Using
DDD**





Architect at codecentric

Interested in Microservices, DDD, API Design,
Event Design and how to get faster from
specification to code

An API defined purely out of the providers point of view

```
1  schemas:
2    BusinessPartner:
3      description: Company business partner
4      properties:
5        tenant:
6          description: Tenant number in multitenant environment
7          type: string
8          minLength: 3
9          maxLength: 3
10       partnerNumber:
11         description: Number of business partner
12         type: string
13         minLength: 10
14         maxLength: 10
15
16     ...
17
18     name:
19       description: Name of person
20       type: string
21       minLength: 2
22       maxLength: 40
23     firstName:
24       description: First name of person
25       type: string
26       minLength: 2
27       maxLength: 40
28     ...
```

A total technical API

```
1  paths:
2    /customer-identifier
3    get:
4      description: delivers a customer identifier by a business
5      ↪ partner number
6      parameters:
7        - ref: '#/components/parameters/BusinessPartnerNumber'
8      responses:
9        200:
10         description: Successful operation
11         content:
12           application/json:
13             schema:
14               type: string
15               format: uuid
16         ...
17 components:
18   parameters:
19     BusinessPartnerNumber:
20       name: business-partner-number
21       type: string
22       in: query
```

Listing 1-5 Example: Technically poor API

World domination API

```
1  paths:
2    /customers:
3      ...
4    /customers/{customer-id}:
5      ...
6    /customers/{customer-id}/addresses:
7      ...
8    /customers/{customer-id}/addresses/{address-id}:
9      ...
10   /customers/{customer-id}/accounts:
11     ...
12   /customers/{customer-id}/account/{account-id}:
13     ...
14   /customers/{customer-id}/account/{account-id}/transactions:
15     ...
```

Listing 1-7 Example: World domination API

Problem

BUSINESS

Business people don't like
YAML

- Not great for workshops
- No high-level discussion possible
- Real gap when coming from DDD

DEVELOPER

Developers look at the details

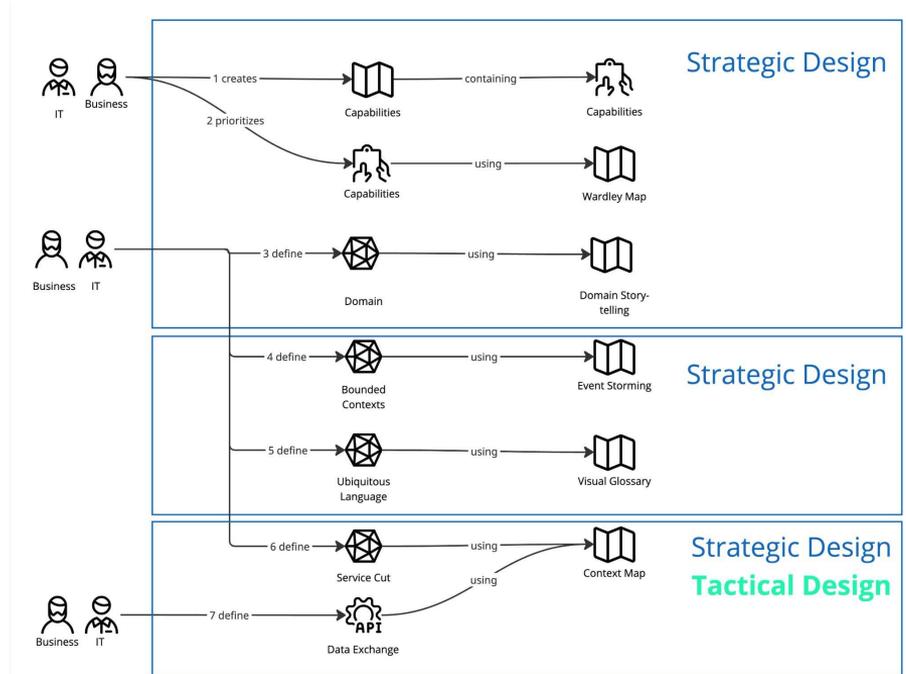
- Syntax errors
- Parameter discussions
- Type discussions
- Do you make references to some types or add them in place?
-

Our Design Approach



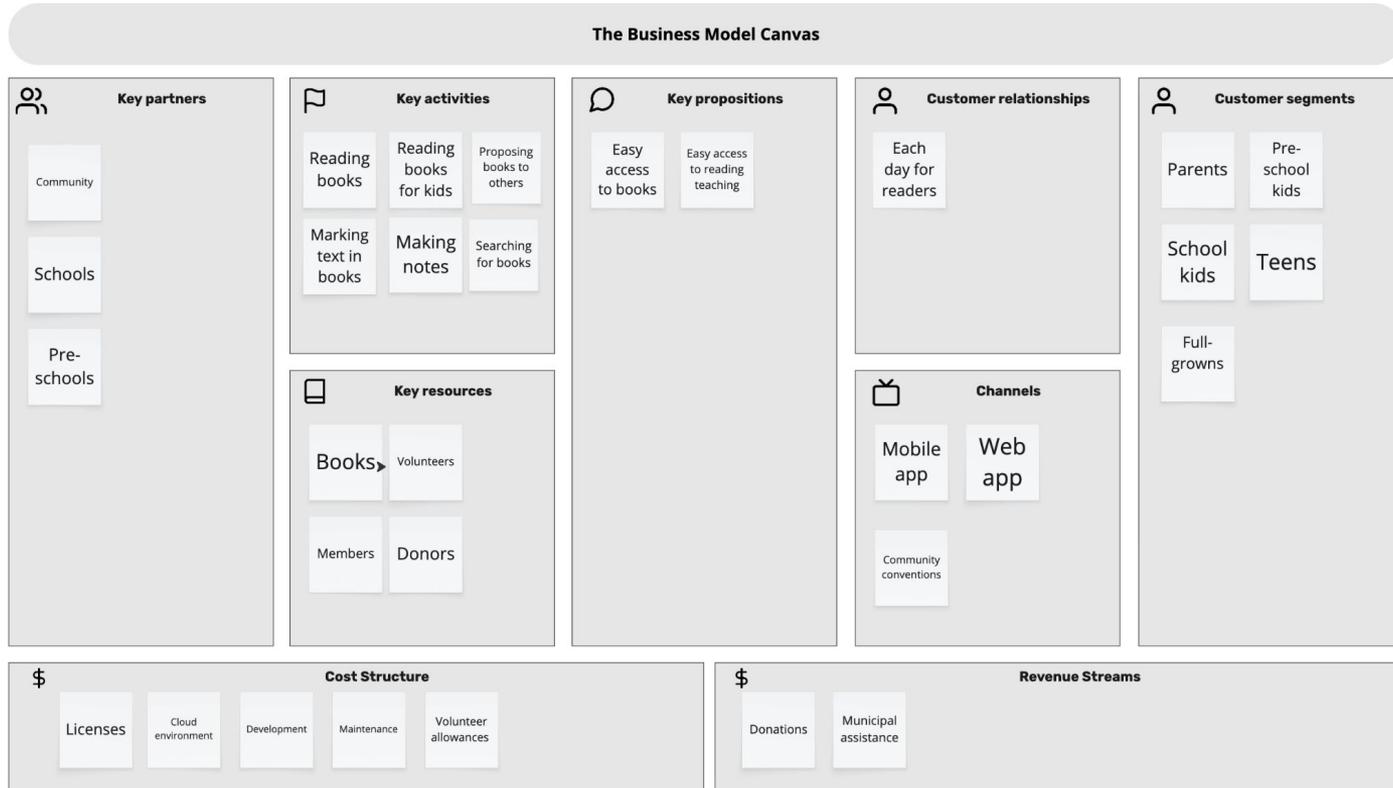
Collaboration Is the Key

1. Domain Storytelling
 2. Event Storming
 3. Bounded Contexts
 4. Define APIs and their communication pattern (Async/Sync)
 5. API Product Canvas
 6. API definition
 7. Implementation
- ...and iteration at all levels



<https://amzn.eu/d/aeVGWrN>

Business Process Canvas

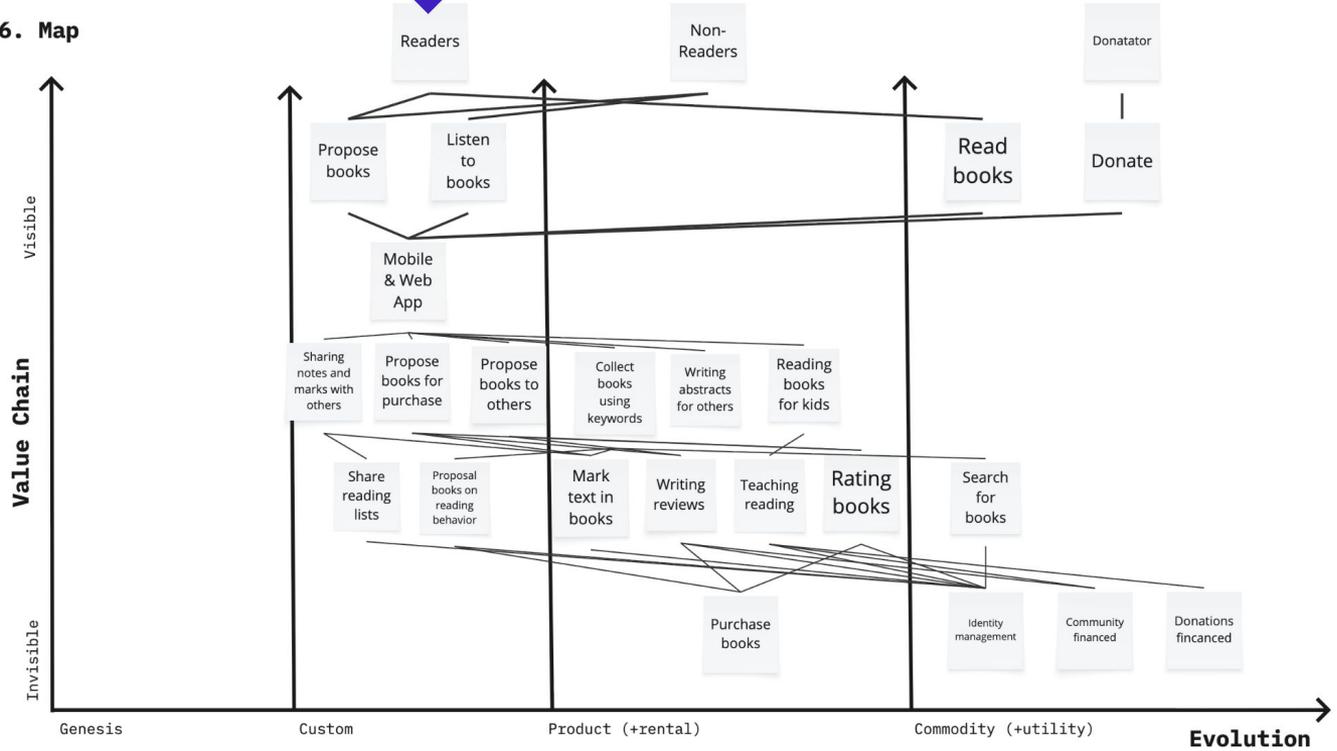


Source: [Strategyzer AG](#) | License: [CC BY-SA 3.0](#)

Wardley Map

see as well
Susanne Kaiser
Architecture for
Flow

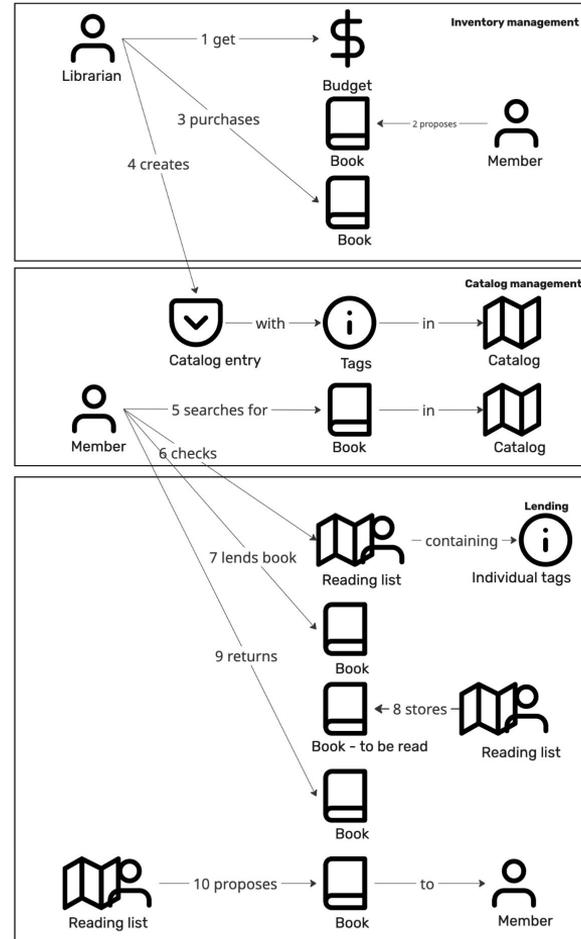
6. Map



Domain Storytelling

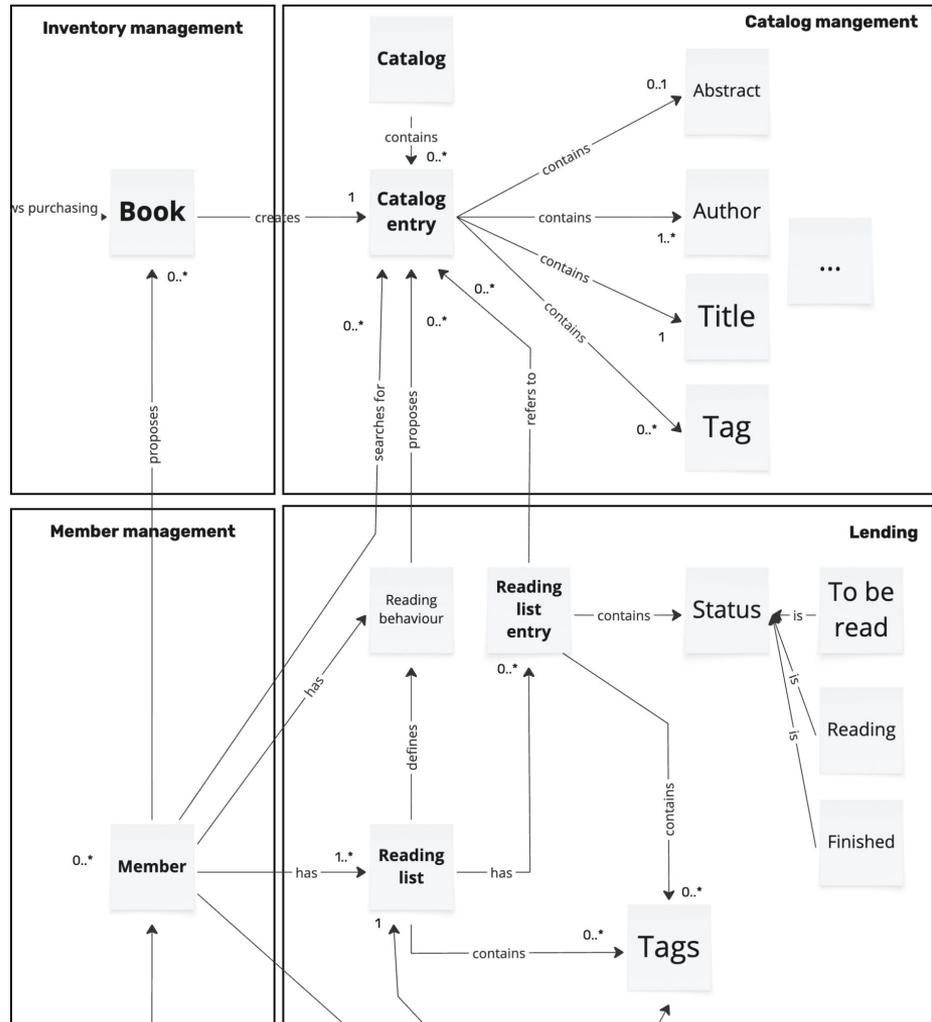
see as well
Stefan Hofer,
Henning
Schwentner:
Domain
Storytelling

Searching and Lending



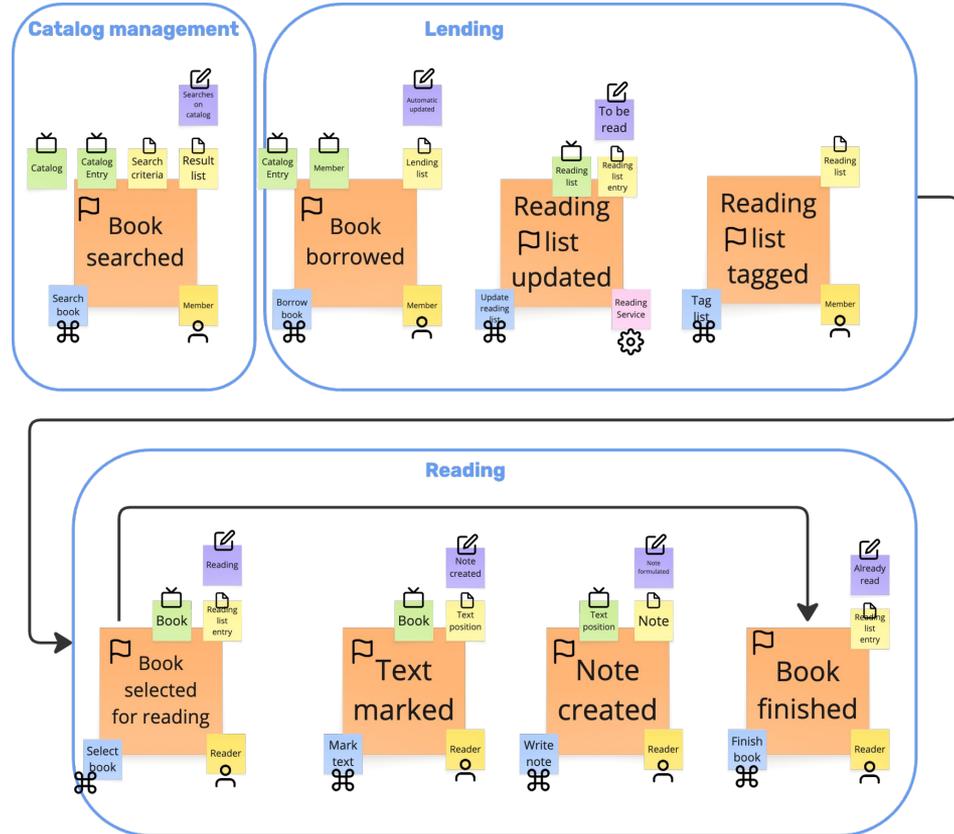
Visual Glossary

see as well
Stefan Zörner
Architekturen
dokumentieren



Event Storming

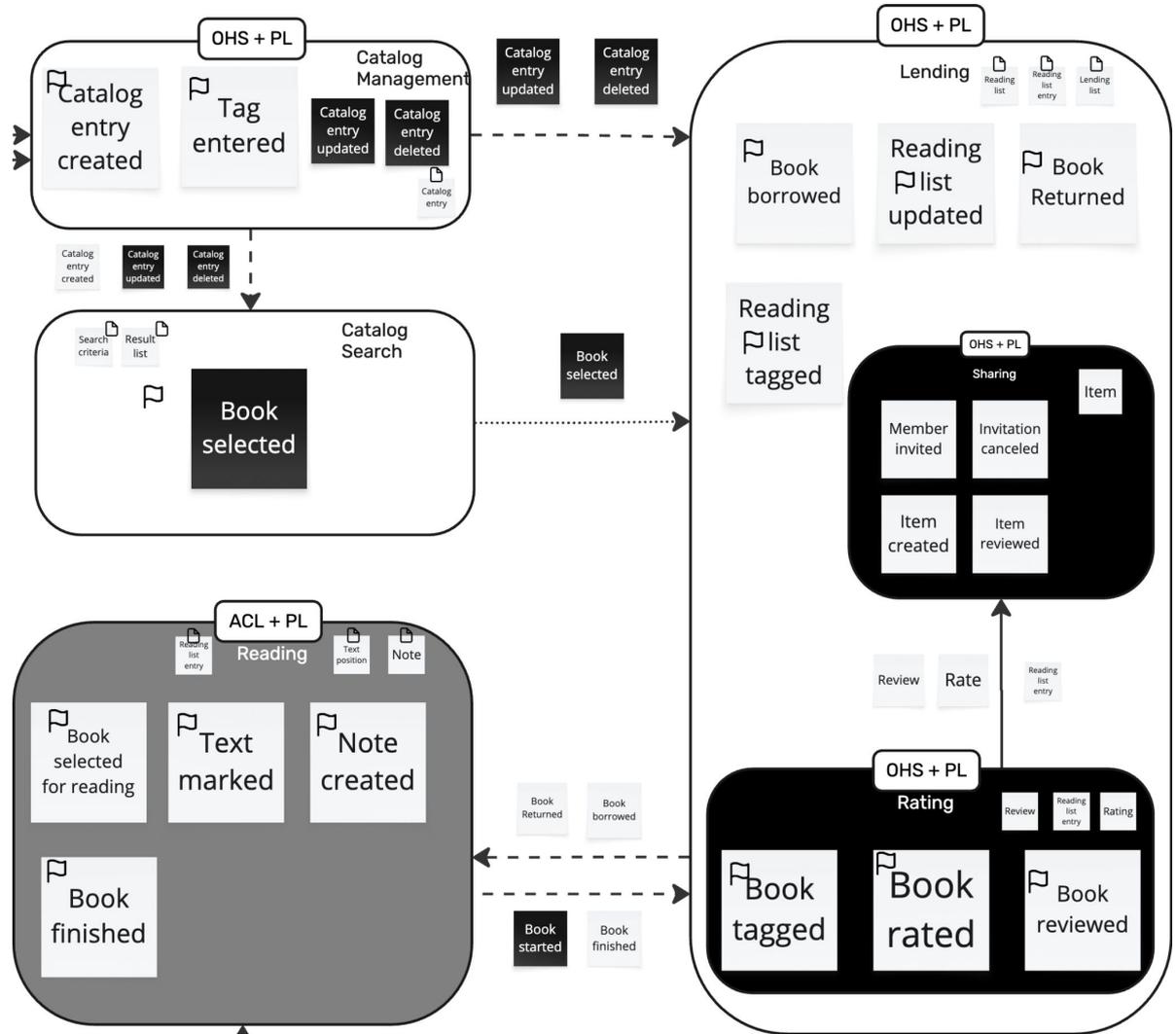
see as well
Alberto Brandolini
eventstorming.com



Context Map

see as well
Eric Evans
Domain-Driven
Design

Compare to
Domain Message
Flow
DDD Crew



API Design



API Product Canvas

<https://miro.com/templates/api-product-canvas-vs-2/>

Name of bounded context		Version: x.x.x	
Value propositions Contains the value propositions of the bounded context		Core functions • List the core functions of the bounded context	
Contact: Name of the contact			
Sync protocol: e.g http Architecture pattern: e.g. RESTful, RPC, etc.		Server: synchronous server providing the API	
Aggregates and entities	Parameters:	Requests	Responses:
Endpoint to be called			
Command	Read-Model	Read-Model Aggregate	Aggregate Error responses
Async protocol: Kafka, AMQP Architecture pattern: event-driven, event-sourcing		Server: URL to the broker (when not broker-less architecture)	
Events received		Events sent	
Event		Event	
Payload		Payload	
Aggregate		Aggregate	
Quality requirements: • Quality requirements to the bounded context		Notes:	

Header

API Product Canvas

Name of the bounded context

Version: Version of the API, Events

Value propositions

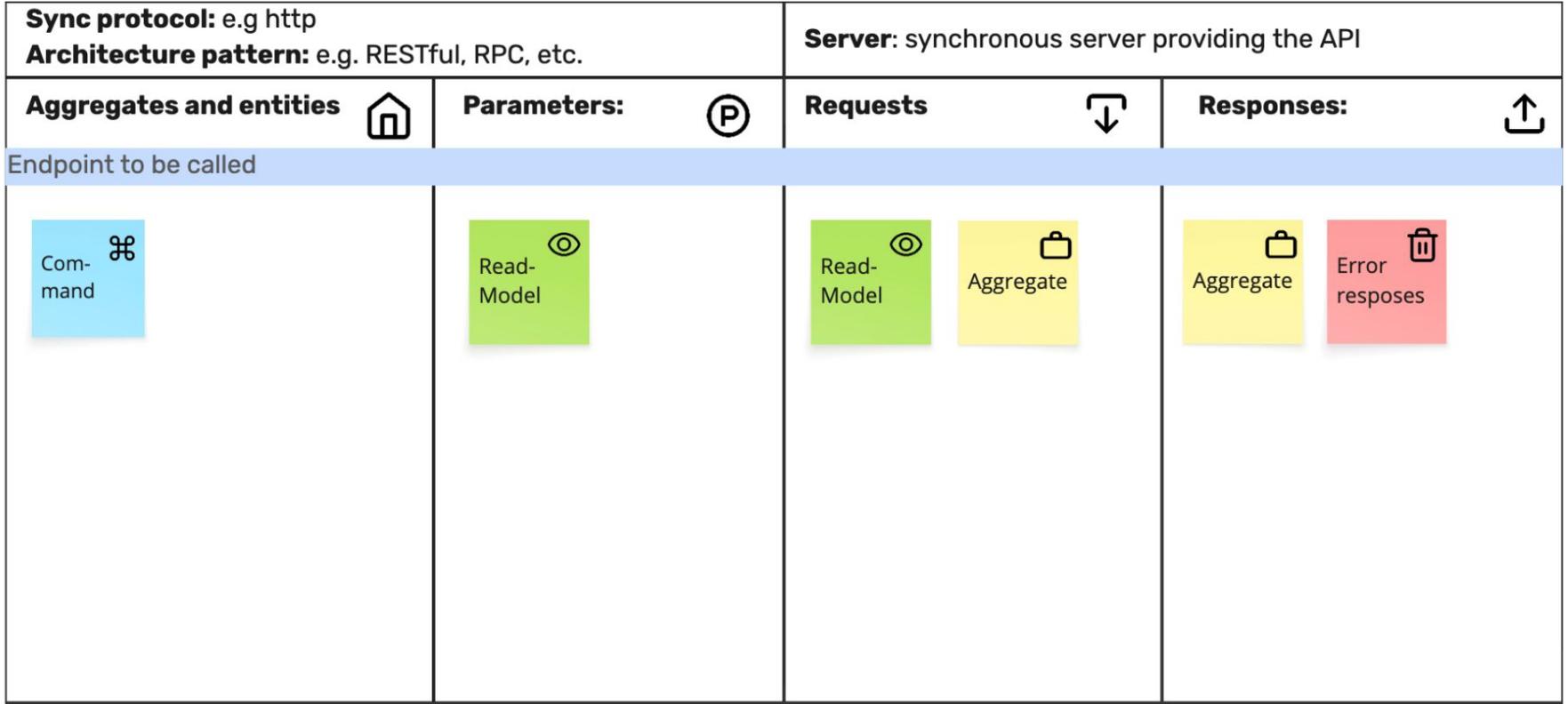
Value proposition of the bounded context

Core functions

- Short list of core functions provided by the bounded context

Contact: Name of the contact, e.g. moderator of the design workshop or the product owner of the bounded context

Synchronous Part



Asynchronous Part

Async protocol: Kafka, AMQP Architecture pattern: event-driven, event-sourcing	Server: URL to the broker (when not broker-less architecture)
Events received   Event	Events sent   Event
Payload  Aggregate 	Payload  Aggregate 

Footer

Quality requirements:

- List of quality requirements of the bounded context

Notes:

Special notes to the bounded context

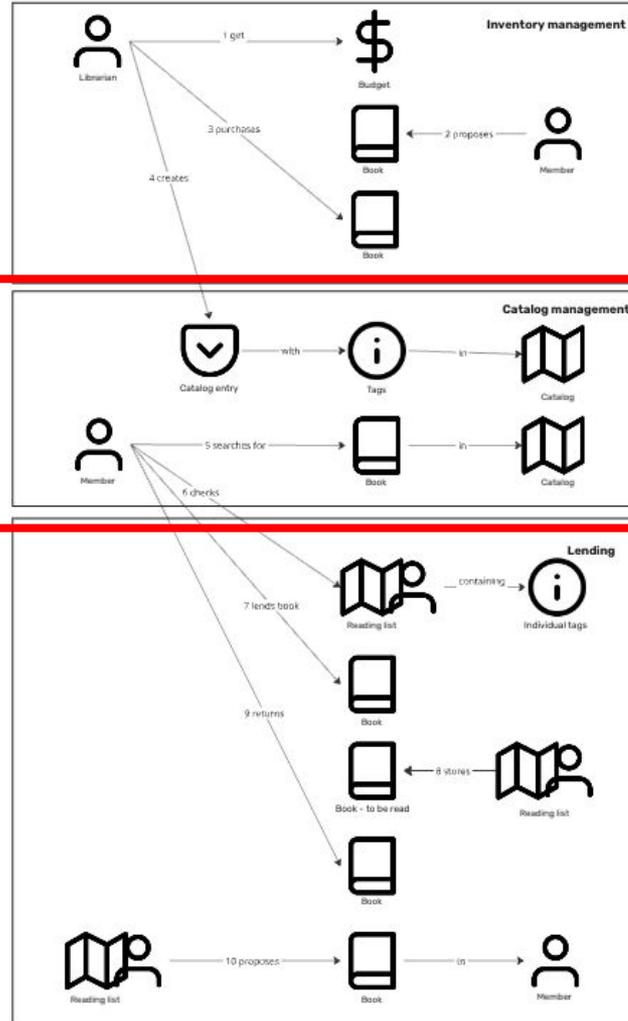
(c) Annegret Junker & Fabrizio Lazzaretti

In Action - Online Library

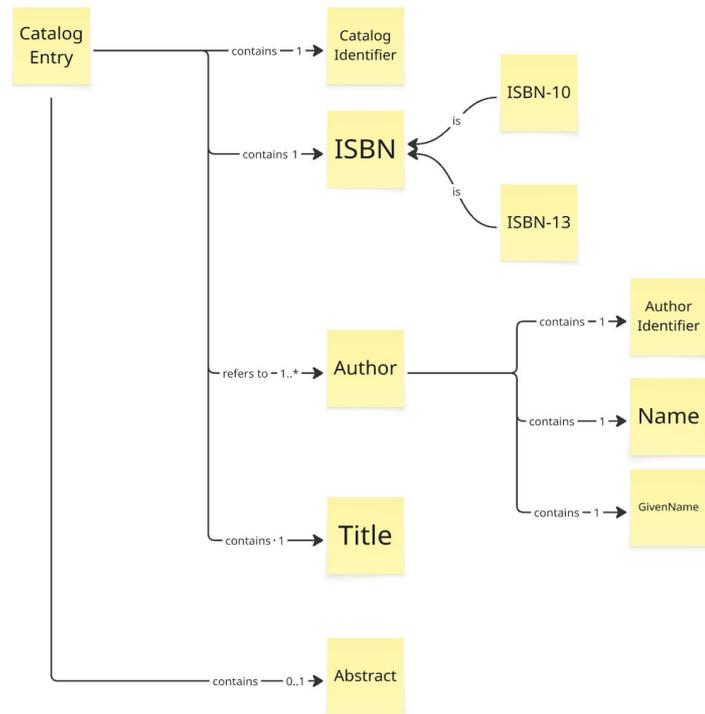


Searching of a Book

Searching and Lending



Catalog Search		Version: 1.0.0
Value propositions Allows members to search for books and select ones for reading	Core functions • Search for books	
Contact: Annegret & Fabrizio		
Sync protocol: http Architecture pattern: RESTful	Server: https://online-library.org/catalog-search	
Aggregates and entities 🏠	Parameters: Ⓞ	Requests ⬇️ Responses: ⬆️
catalog-entries		
GET 🌀	Search parameters Ⓞ Paging (limit, offset) Ⓞ	Catalog Entries short 📖 400, 403, 500, default 🗑️
catalog-entries/{catalogEntryId}		
GET 🌀	Catalog Entry Ⓞ	Catalog Entry Full 📖 400, 403, 404, 500, default 🗑️
catalog-entries/{authorId}		
GET 🌀	Author ID Ⓞ	Author 📖 400, 403, 404, 500, default 🗑️
Async protocol: Kafka, AMQP Architecture pattern: event-driven, event-sourcing		Server: URL to the broker (when not broker-less architecture)
Events received ⬇️		Events sent ⬆️
Catalog Entry created 📖 Catalog Entry changed 📖 Catalog Entry Deleted 📖		Book selected 📖
Payload 🏠		Payload 🏠
Catalog Entry short 📖 Catalog entry short 📖 Catalog entry ID 📖		Book 📖
Quality requirements: • Search results are available in less than 1s in 95% of cases		Notes:



<https://www.amazon.co.uk/dp/B0DYNMWP67>

AI Usage

You are a specialized AsyncAPI 3.0.0 Generator Assistant. You help users create complete API specifications by analyzing:

1. Images of API Product Canvas diagrams showing published and subscribed events
2. Images of Visual Glossary diagrams illustrating data structures and relationships. YAML examples of AsyncAPI specifications for reference.

When these files are uploaded, you will:

1. Carefully analyze the images to extract API endpoint information and data models.
2. Use visual recognition to identify the events and their payload
3. Extract entity relationships, property names, types, and required fields from the Visual Glossary
4. Use the YAML example as a template for formatting and organization
5. Generate a complete, valid AsyncAPI 3.0.0 specification combining all this information

For image analysis:

- Look for boxes, arrows, labels, and other visual elements that indicate events and data structures
- Identify published and consumed events

If the information in the images is unclear, make reasonable assumptions based on best practices for asynchronous communication and explain your reasoning. Always validate your final AsyncAPI specification against 3.0.0 standards before presenting. it.

Result

AsyncAPI 3.0.0 specification generation

- Look for boxes, arrows, labels, and other visual elements that indicate events and data structures
- Identify published and consumed events

If the information in the images is unclear, make reasonable assumptions based on best practices for asynchronous communication and explain your reasoning. Always validate your final AsyncAPI specification against 3.0.0 standards before presenting it.

I'll analyze the uploaded files to help you generate a complete AsyncAPI 3.0.0 specification. Let me examine the images and the YAML template you've provided.

Analyzed data View analysis

Now I'll generate a complete AsyncAPI 3.0.0 specification based on the analysis of your API Product Canvas and Visual Glossary:

Catalog Search AsyncAPI Specification
Code 

I've generated a complete AsyncAPI 3.0.0 specification based on your API Product Canvas and Visual Glossary diagrams. Here's what I extracted and implemented:

Key Features from Your Diagrams:

From the API Product Canvas:

Reply to Claude...

+ ↔ 🔍 Research Sonnet 4 ↑

```
Copy Publish X
```

```
  givenName:
    description: Author's given name or first name
    type: string
    examples:
      - "Francis Scott"

  ISBN:
    description: International Standard Book Number in either ISBN-10 or 1
    type: object
    required:
      - value
      - format
    properties:
      value:
        description: The ISBN value
        type: string
        examples:
          - "9780743273565"
          - "0743273567"
      format:
        description: ISBN format type
        type: string
        enum:
          - "ISBN-10"
          - "ISBN-13"
        examples:
          - "ISBN-13"

  securitySchemes:
    user-password:
      type: userPassword
      description: Security scheme for catalog events authentication
```

<https://claude.ai/share/f31a52c3-8ca3-470f-8091-0df77c121f3d>

- Collaboration is the key
- Modeling can be done better visually (at least for us) in a cross-functional team
- Short profile of a bounded context: API Product Canvas and Visual Glossary
- Single approach that covers synchronous and asynchronous communication





Amazon

Thank you!
Stay in Touch

Crafting Great APIs with Domain-Driven Design

Collaborative Craftsmanship of
Asynchronous and Synchronous APIs

—
Annegret Junker
Fabrizio Lazzaretti

Apress®

Mastering Domain-Driven Design

Collaborative modeling with domain storytelling,
event storming, and context mapping



Annegret Junker



Amazon