SINTEF

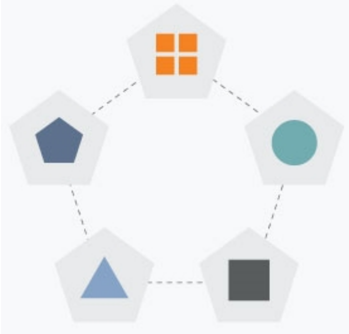
MIGRATING MONOLITHIC APPLICATIONS TO MICROSERVICES-BASED CUSTOMIZABLE MULTI-TENANT APPLICATIONS

Sindre Grønstøl Haugeland[1], Phu H. Nguyen[2*], Franck Chauvel[2] , Hui Song[2]

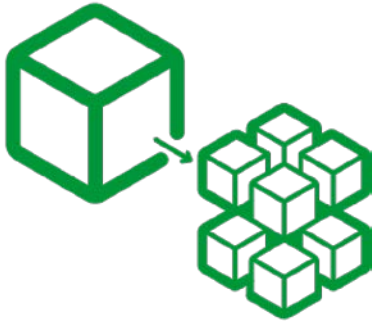
[1] University of Oslo, Oslo, Norway, sindrgro@ifi.uio.no
[2] SINTEF, Oslo, Norway, firstname.lastname@sintef.no , [*] Presenter

Online @ the International Conference on Microservices,
September 8-10, 2020

This talk presents 1) Why migrating to Microservices; 2) Our Migration Approach ; and 3) Lessons learnt and Ideas for Future Work

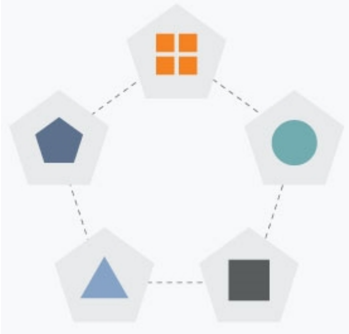


Why migrating to Microservices?



Our Migration Approach

Conclusion & Future Work



Why migrating to Microservices?

TWO main reasons for adopting our migration approach here

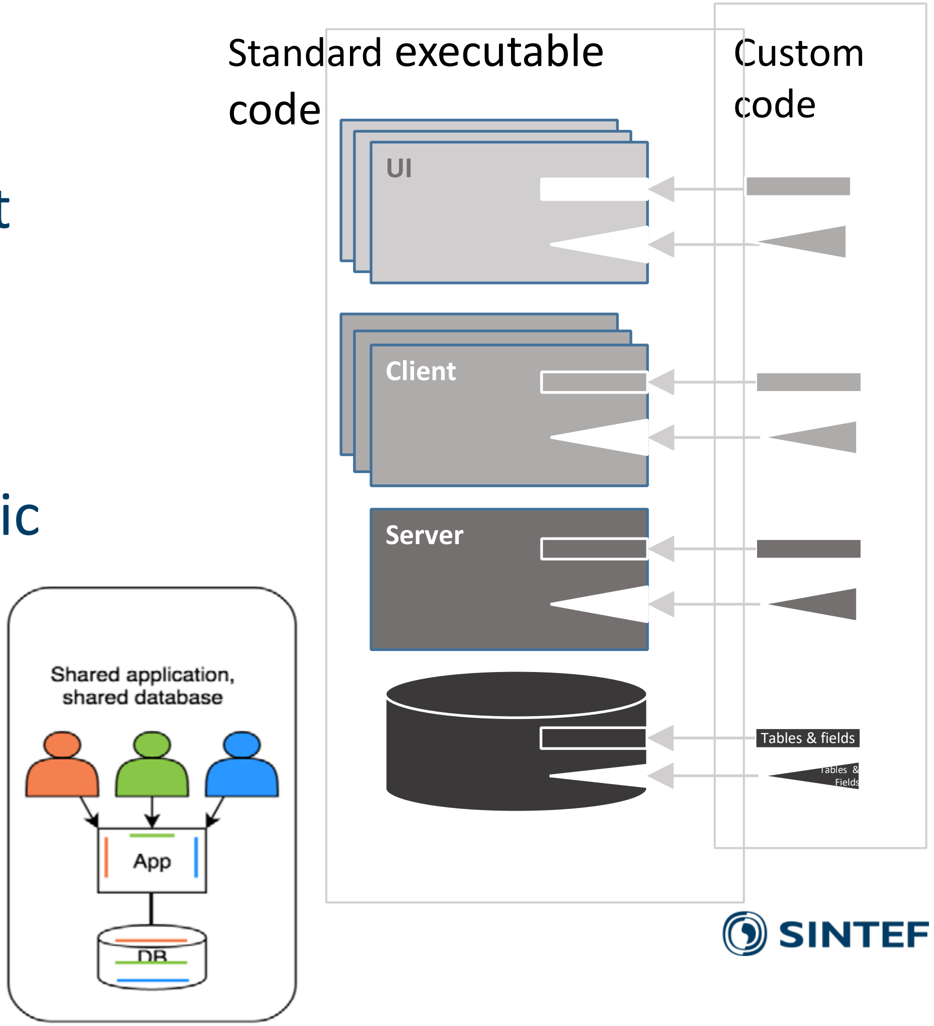
The FIRST reason is a common one: All the benefits of the Microservices architecture, e.g.,

- Loosely coupled, highly cohesive services,
  that can be deployed and developed independently
- Solves some of the issues with monolithic applications with regards to speed of delivery and maintaniability
- You name it…

The SECOND reason is a very specific one

To enable (deep) customization in the multi-tenant Software-as-a-Service (SaaS) context!

- Inspired from our R&D project with industry.
- How can microservices be used for tenant-specific customization?
  - Separation of application layers
  - Separation of functional logic
  - Independent context



More background about enabling deep customization for multi-tenant SaaS

- P. H. Nguyen, H. Song, F. Chauvel, R. Muller, S. Boyar, and E. Levin, “Using microservices for non-intrusive customization of multi-tenant SaaS,” in Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Tallinn, Estonia, Aug. 2019, pp. 905–915, doi: 10.1145/3338906.3340452.
- H. Song, P. H. Nguyen, and F. Chauvel, “Using Microservices to Customize Multi-Tenant SaaS: From Intrusive to Non-Intrusive,” p. 18 pages, 2020, doi: 10.4230/OASICS.MICROSERVICES.2017-2019.1.
- H. Song, P. H. Nguyen, F. Chauvel, J. Glattetre, and T. Schjerpen, “Customizing Multi-Tenant SaaS by Microservices: A Reference Architecture,” in 2019 IEEE International Conference on Web Services (ICWS), Jul. 2019, pp. 446–448, doi: 10.1109/ICWS.2019.00081.
- H. Song, F. Chauvel, and P. H. Nguyen, “Using Microservices to Customize Multi-tenant Software-as-a-Service,” in Microservices: Science and Engineering, A. Bucchiarone, N. Dragoni, S. Dustdar, P. Lago, M. Mazzara, V. Rivera, and A. Sadovykh, Eds. Cham: Springer International Publishing, 2020, pp. 299–331.

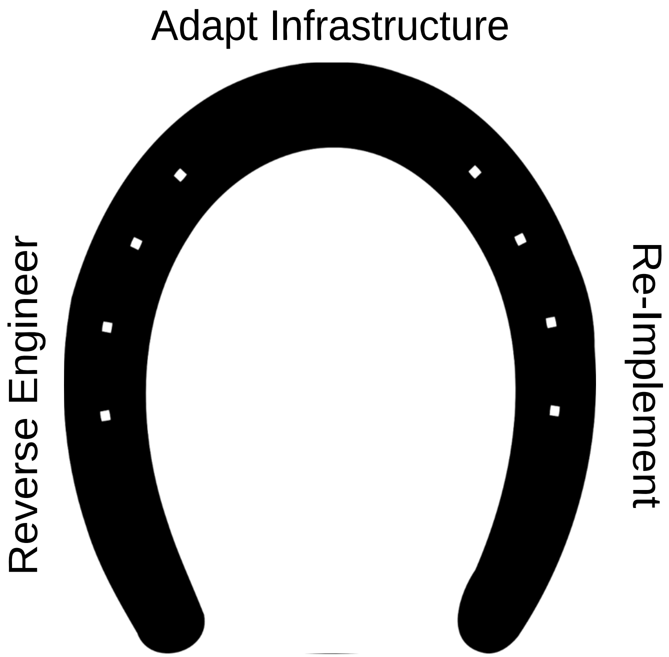
Existing Migration Approaches

Strangler Approach

- Wraps application in intercepting layer
- Redirect incoming requests to extracted services

Blueprint Approach

- Three phases
  - Analysis
  - Adaption of infrastructure
  - Re-implementation
- Template for further modification

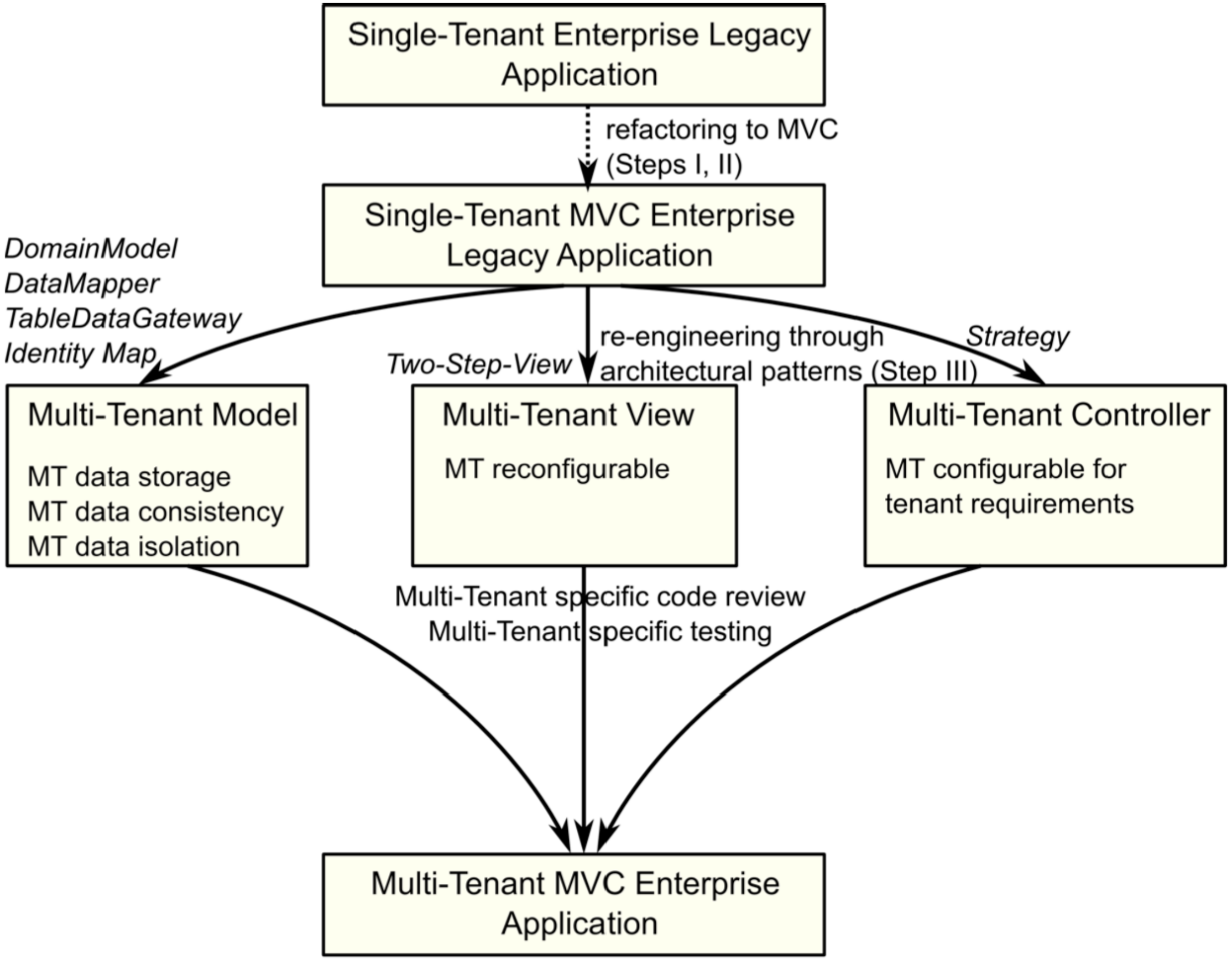


“StranglerFigApplication” martinfowler.com. https://martinfowler.com/bliki/StranglerFigApplication.html (accessed Jun. 13, 2020).

Eberhard Wolff. Migrating monoliths to microservices: A survey of approaches. 2019.

Existing Approaches

- Single to Multi-tenant approach
- Split into three steps
  - Convert to MVC
  - Introduce tenant-specific Model, View and Controller
  - Tenant-specific QA

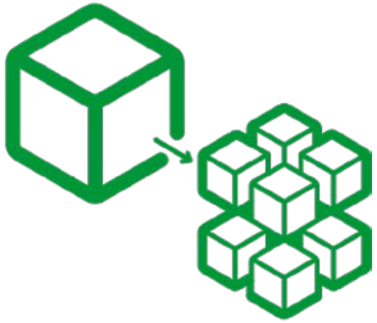


Single-Tenant Enterprise Legacy Application
refactoring to MVC (Steps I, II)
Single-Tenant MVC Enterprise Legacy Application
DomainModel
DataMapper
TableDataGateway
Identity Map
Two-Step-View
re-engineering through architectural patterns (Step III)
Strategy
Multi-Tenant Model
MT data storage
MT data consistency
MT data isolation
Multi-Tenant View
MT reconfigurable
Multi-Tenant Controller
MT configurable for tenant requirements
Multi-Tenant specific code review
Multi-Tenant specific testing
Multi-Tenant MVC Enterprise Application

The re-engineering process for enabling multi-tenancy in a single tenant legacy application by applying a combination of enterprise application architecture patterns (Fowler 2003).

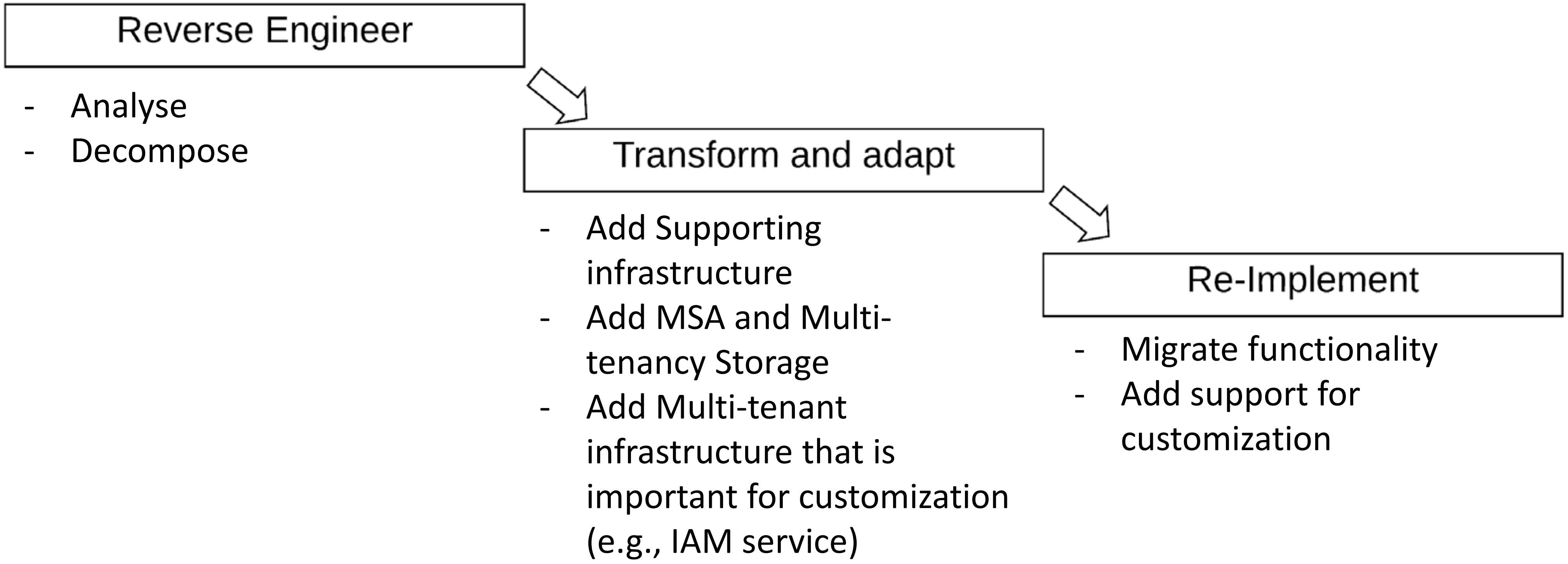
Furda, Andrei, et al. "Reengineering data-centric information systems for the cloud–a method and architectural patterns promoting multitenancy." Software Architecture for Big Data and the Cloud. Morgan Kaufmann, 2017. 227-251.

Fowler, M. (2003). Patterns of enterprise application architecture. Boston, Addison-Wesley.



Our Migration Approach

Suggested migration approach to MSA-based customizable multi-tenant SaaS



Reverse Engineer

- Analyse
- Decompose

Transform and adapt

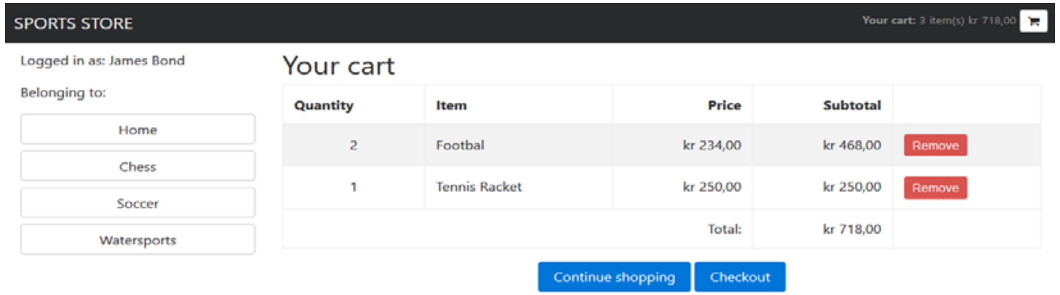
- Add Supporting infrastructure
- Add MSA and Multi-tenancy Storage
- Add Multi-tenant infrastructure that is important for customization (e.g., IAM service)

Re-Implement

- Migrate functionality
- Add support for customization

A Running Example: The SportStore Application

Shopping Cart View:



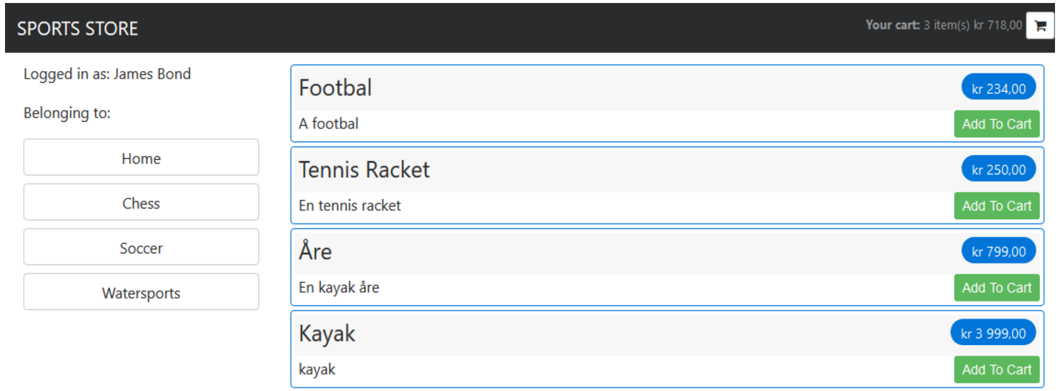
Logged in as: James Bond

Belonging to:

Your cart

Quantity	Item	Price	Subtotal
2	Footbal	kr 234,00	kr 468,00
1	Tennis Racket	kr 250,00	kr 250,00
		Total:	kr 718,00

Product Catalog View:



Logged in as: James Bond

Belonging to:

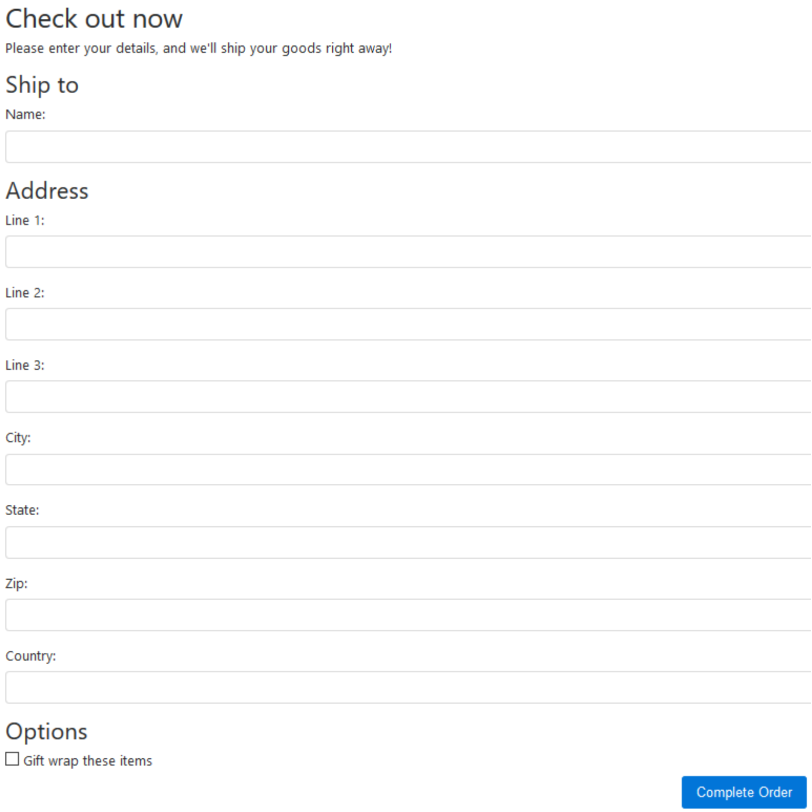
Footbal kr 234,00
A footbal

Tennis Racket kr 250,00
En tennis racket

Åre kr 799,00
En kayak åre

Kayak kr 3 999,00
kayak

Checkout View



Check out now

Please enter your details, and we'll ship your goods right away!

Ship to

Name:

Address

Line 1:

Line 2:

Line 3:

City:

State:

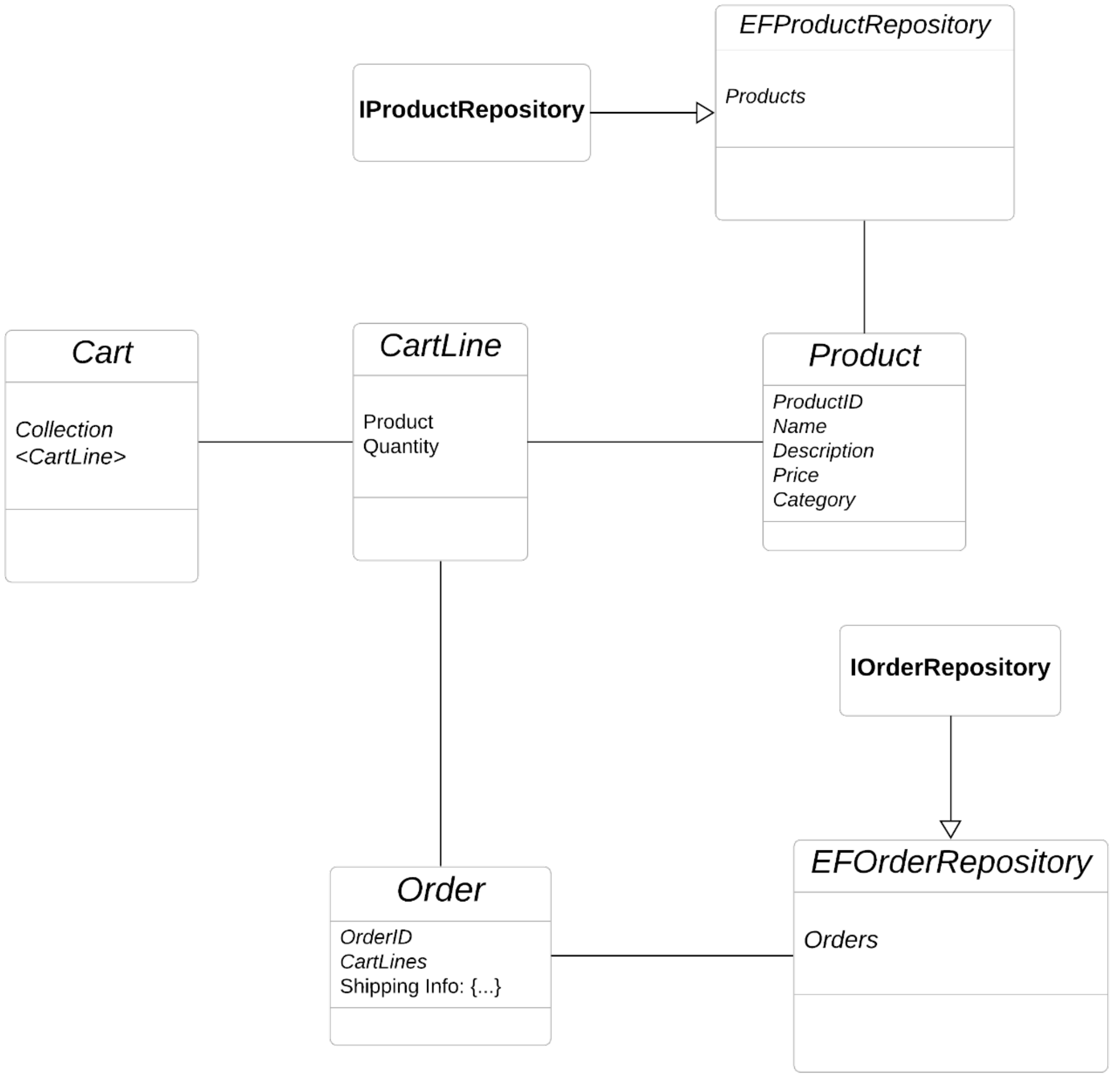
Zip:

Country:

Options

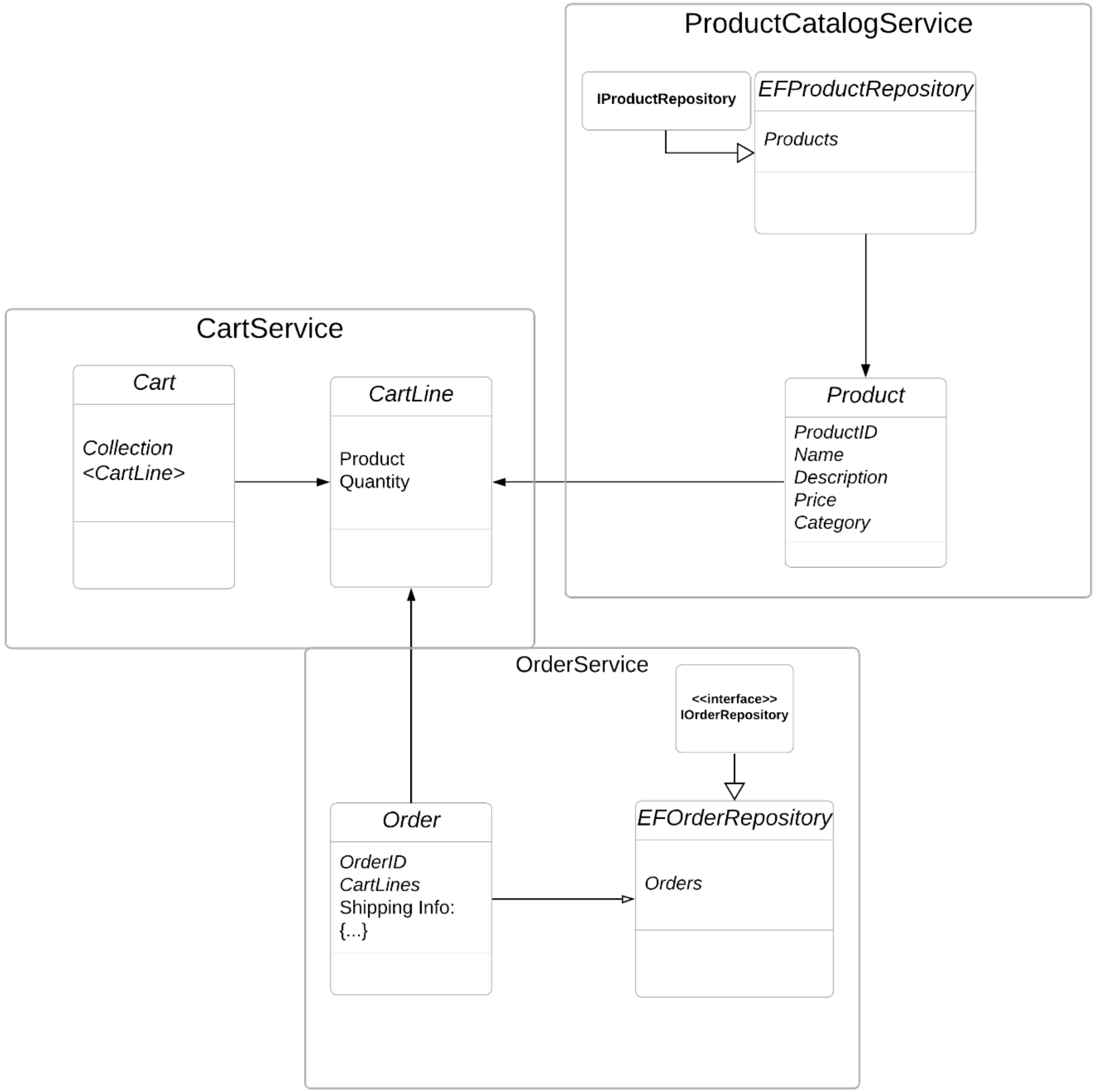
□ Gift wrap these items

Sanderson, S. (2011). *Pro Asp. net MVC 2 Framework*. Apress.



Phase 1: Reverse Engineering

- Analysis
- Looking at source code to determine the architecture
- Finding internal dependencies in the application



Phase 1: Reverse Engineering

Decompose into services:

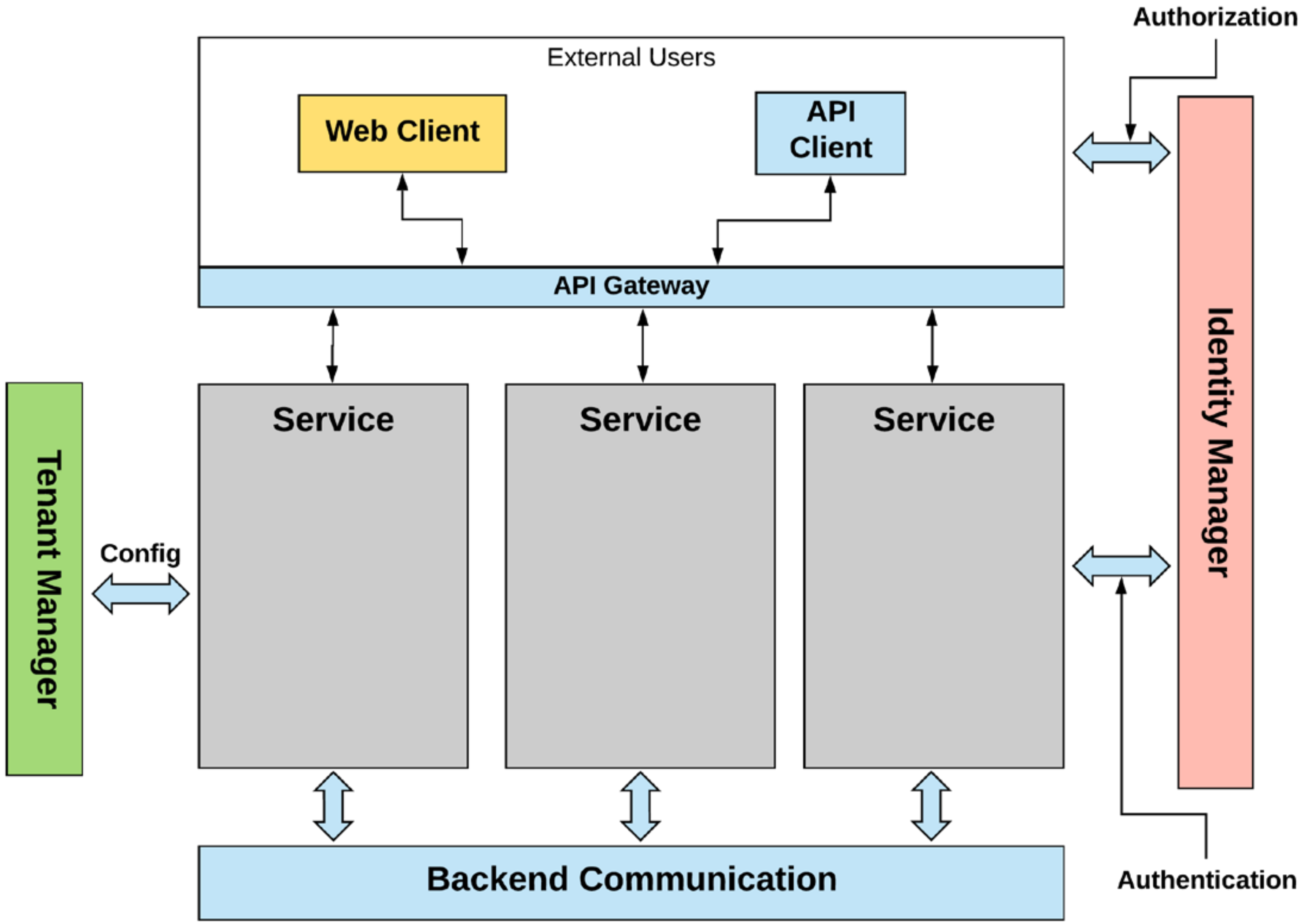
Identify potential services

- Cart
- ProductCatalog
- Ordering

Find boundary resources

- Product
- Cartline

Phase 2: Adapt Infrastructure



Additional Infrastructure

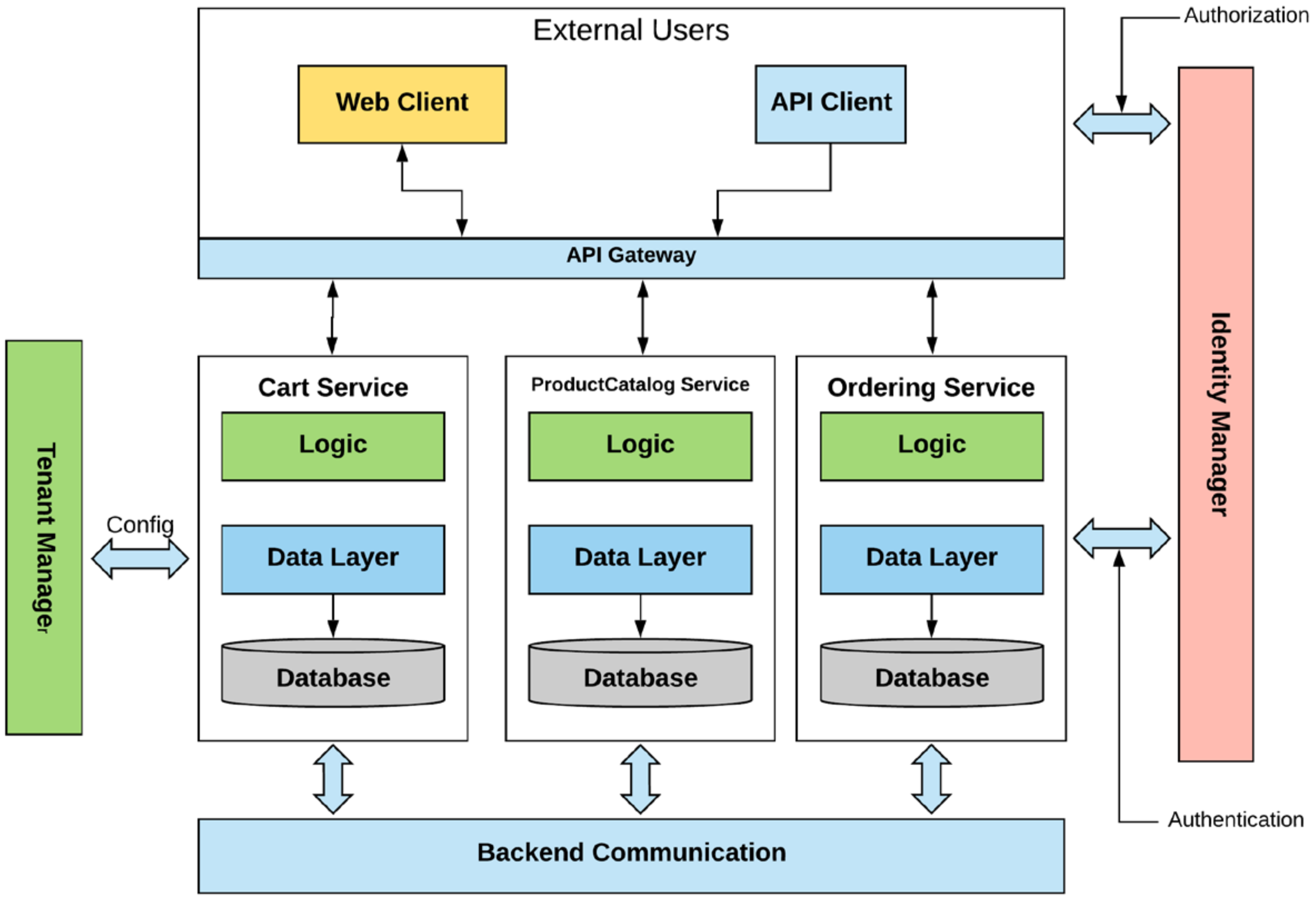
For MSA

- API-Gateway
- Message Broker/Exchange

For Multi-Tenancy

- Identity Manager
- Tenant Manager

Phase 3: Re-Implementation



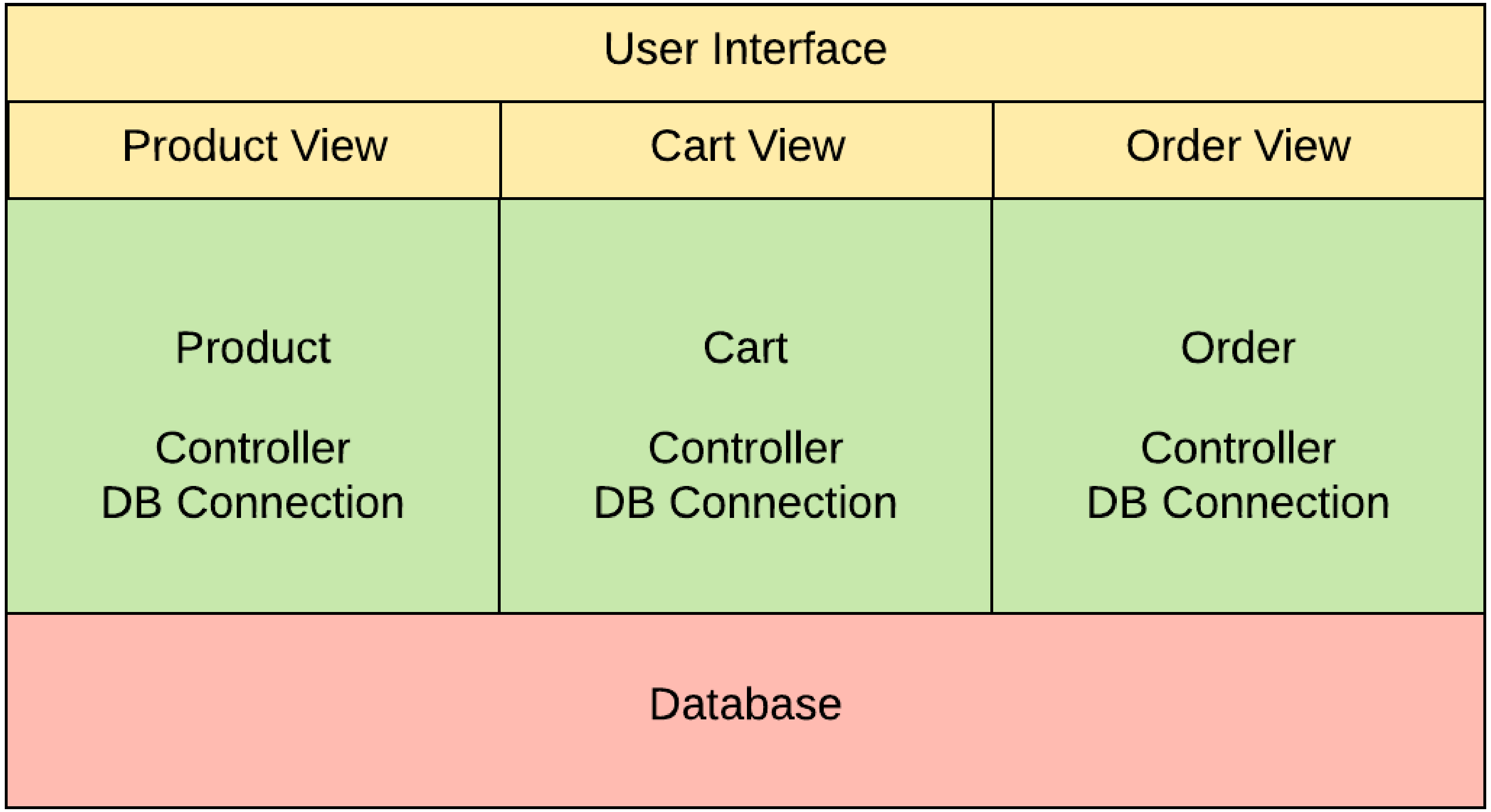
Target architecture

- Services behind API-Gateway
- Message broker for orchestration between services

Customization Ready

- Separation of application layers
- Use IAM and Tenant Manager to retrive tenant customization

The migration


User Interface

Product View

Cart View

Order View

Product

Controller
DB Connection

Cart

Controller
DB Connection

Order

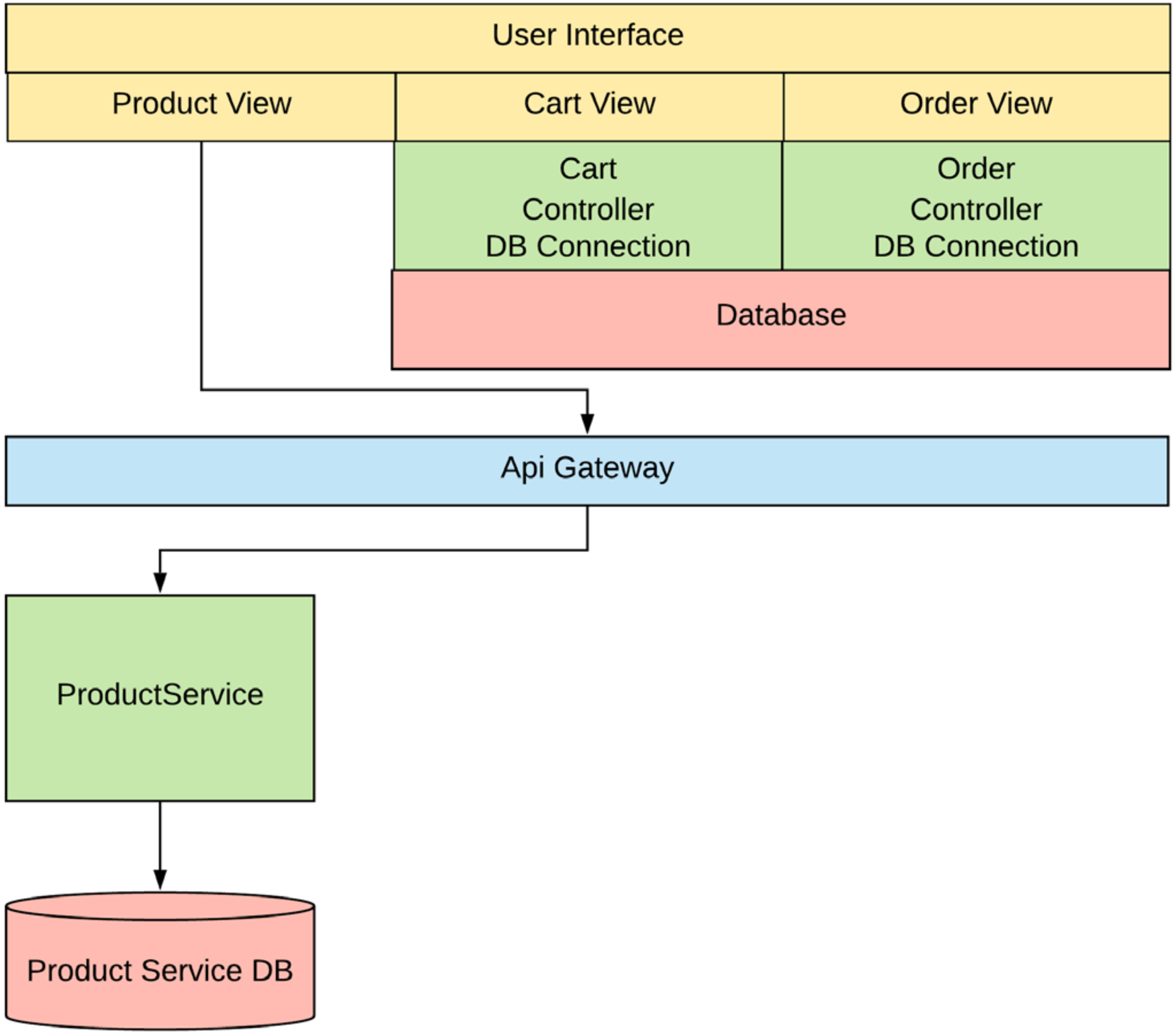
Controller
DB Connection

Database

Initial State:

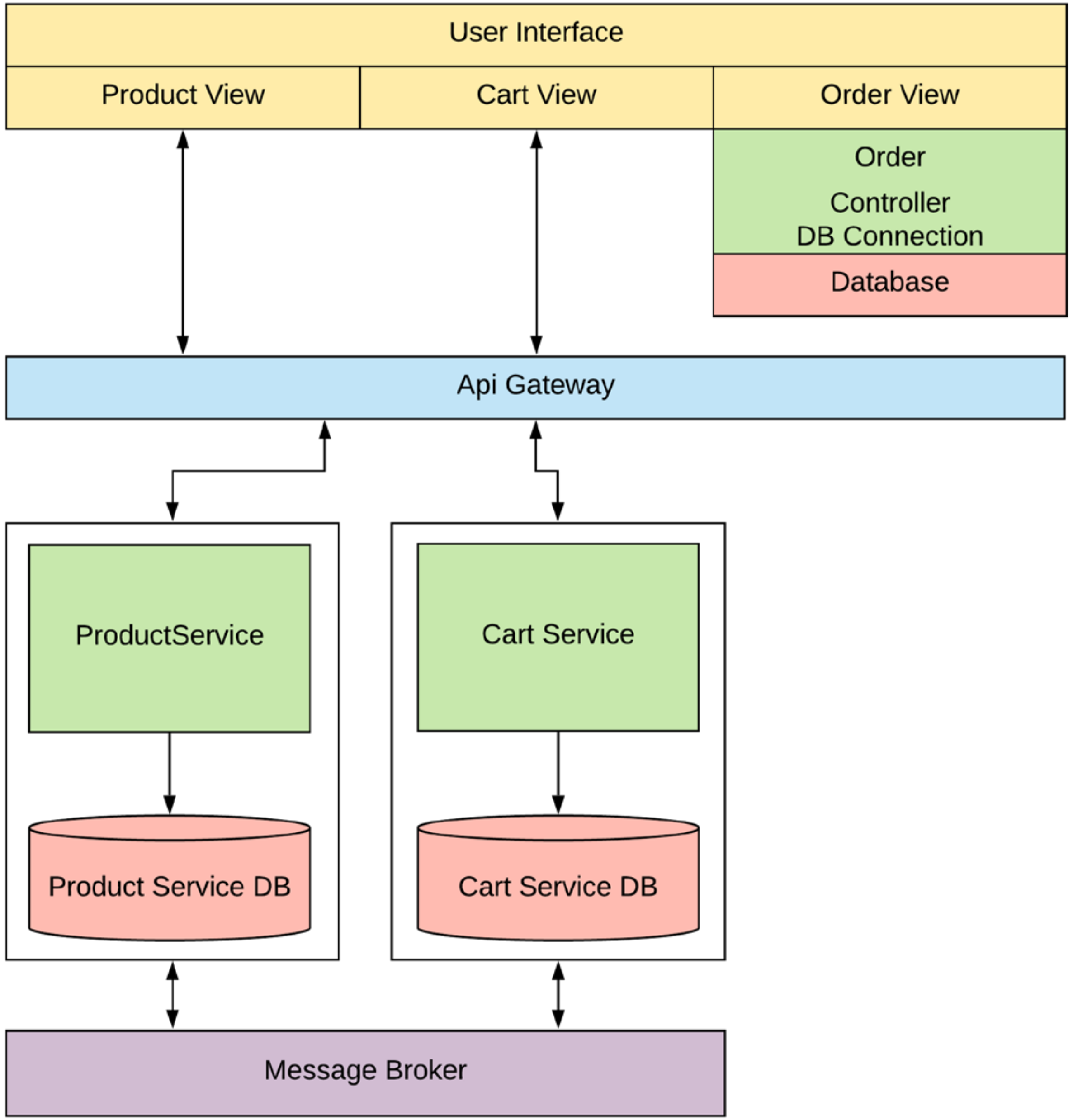
- Monolithic
- UI, Logic and Storage in one deployment

The migration



Second state:

- Start extracting services
- Autonomous deployment
- Reroute through Gateway



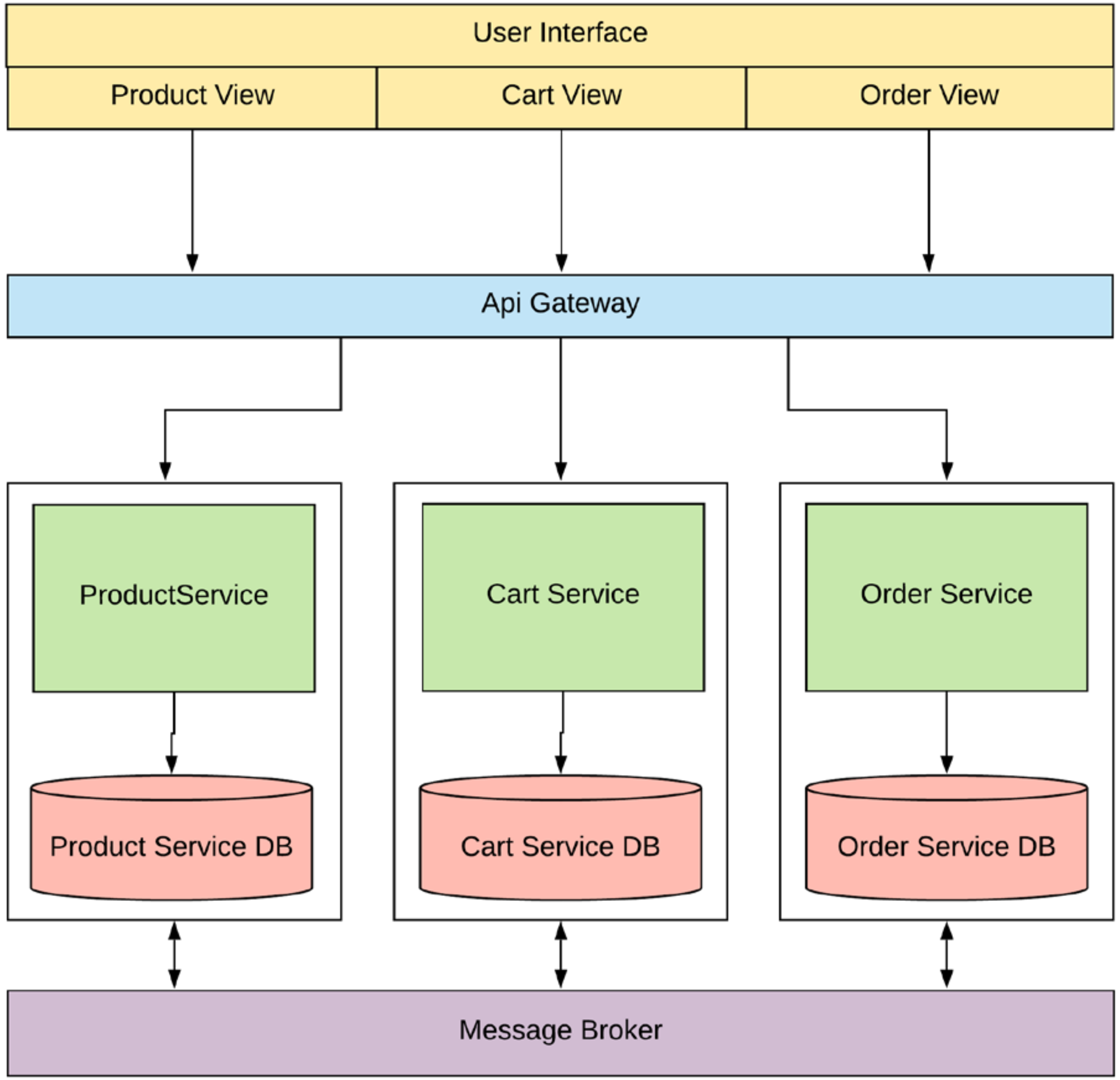
The migration

Third state:

- Second service extracted
- Introduce MB
- Orchestration between services

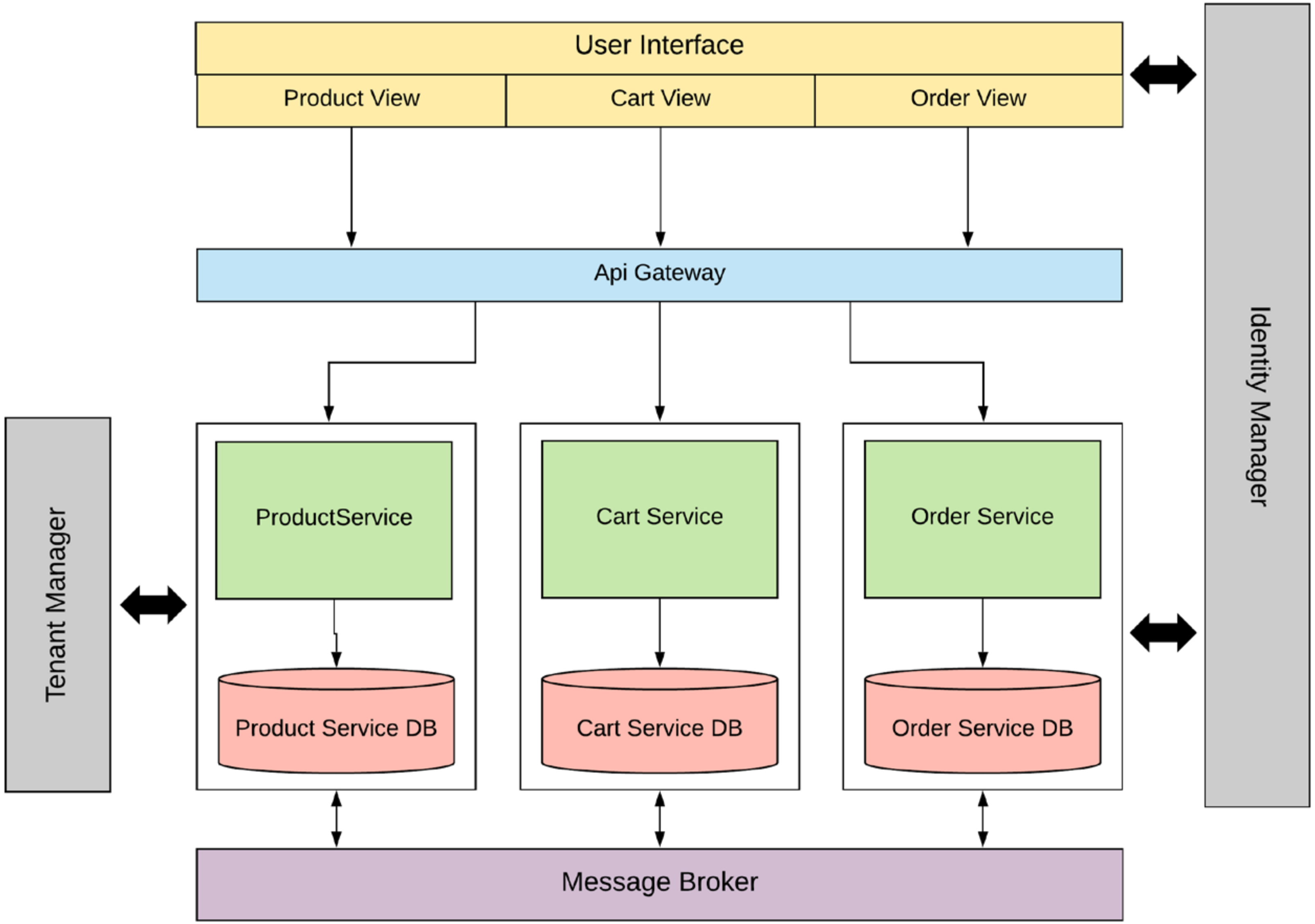
The migration

- Fourth state:
- Third service extracted
- Application has migrated to microservices



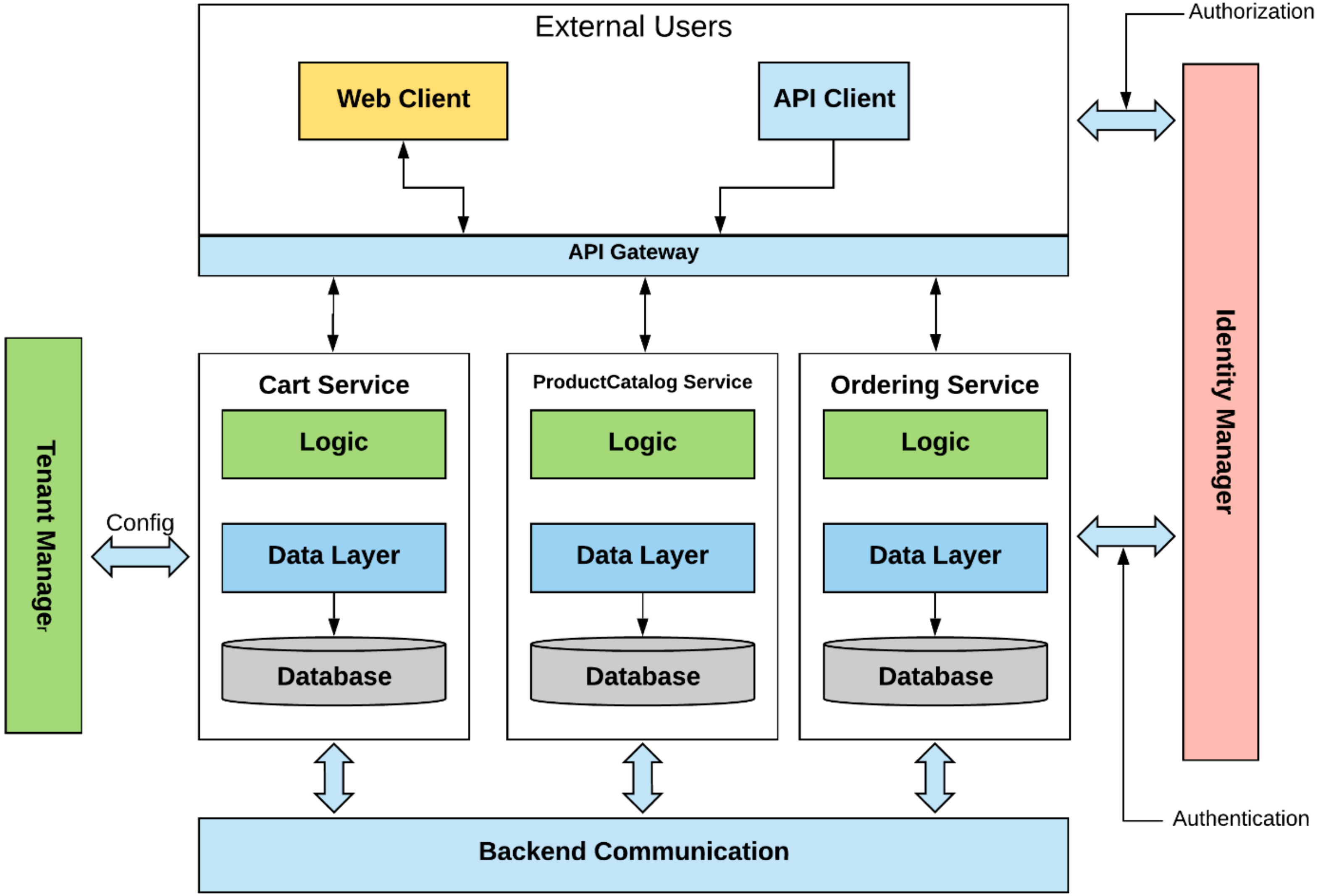
The migration

- Fifth state:
- All services extracted
- Additional infrastructure for multi-tenancy added



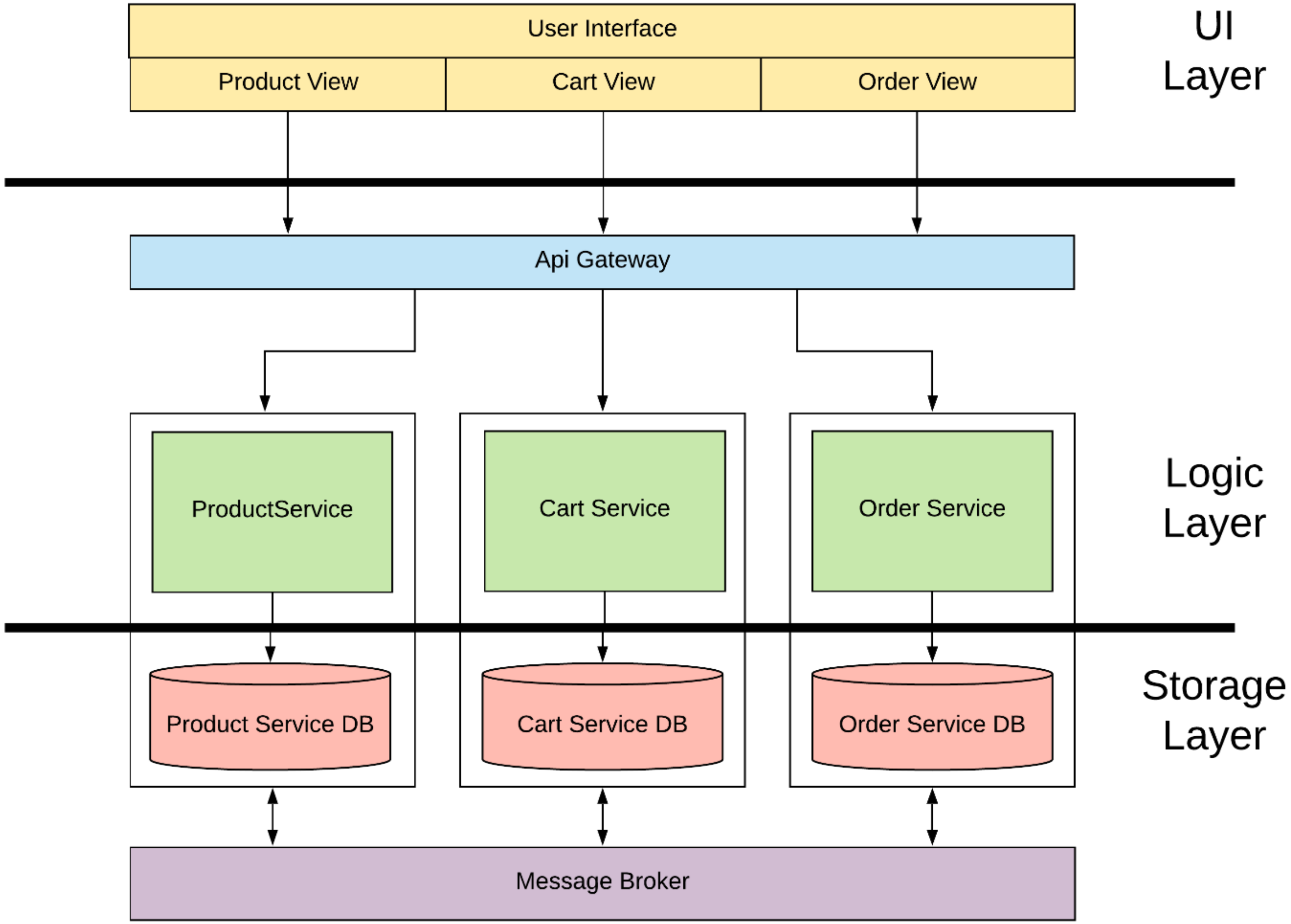
Results

Final Architecture



Results

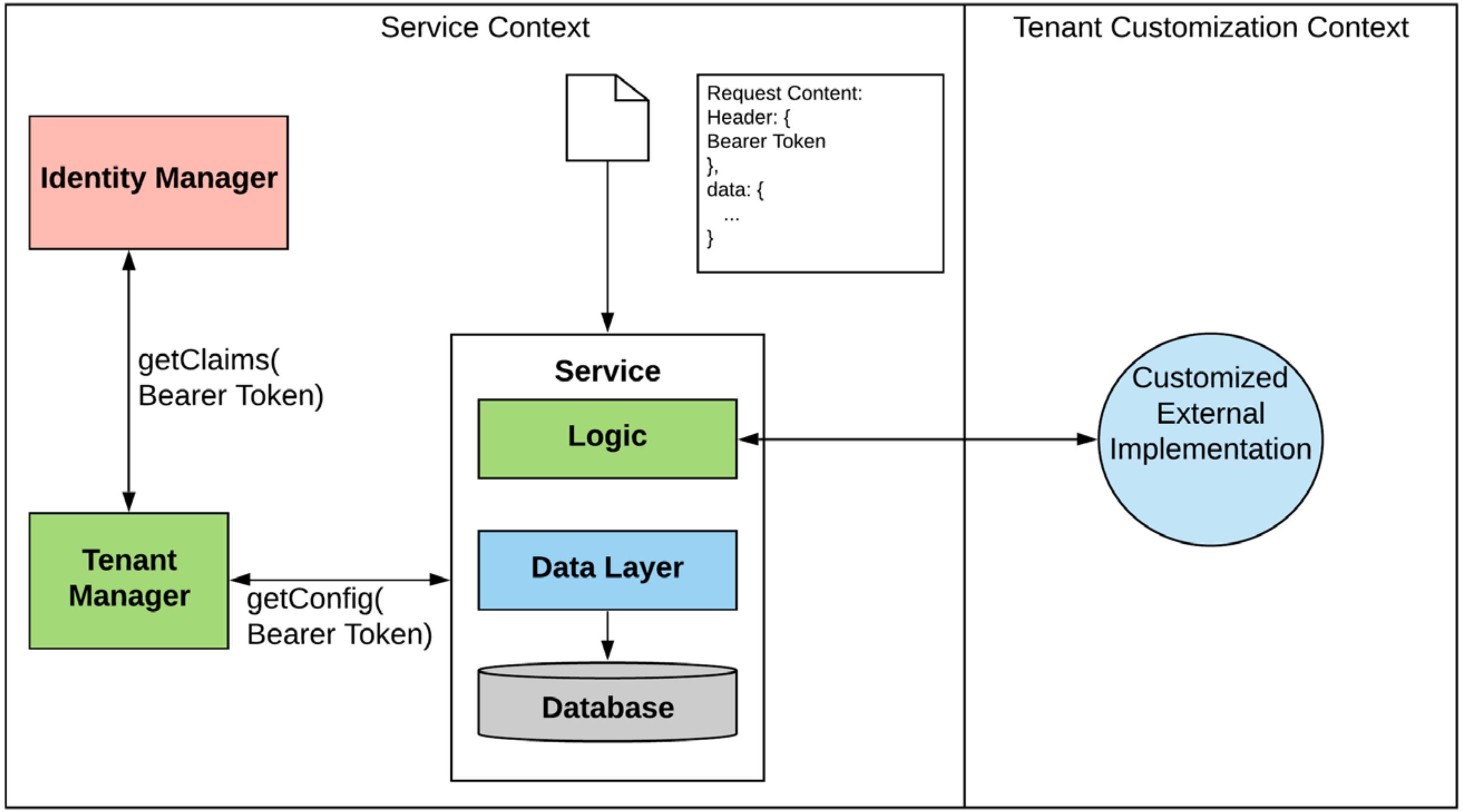
Separation of Layers



P. H. Nguyen, H. Song, F. Chauvel, R. Muller, S. Boyar, and E. Levin, “Using microservices for non-intrusive customization of multi-tenant SaaS,” in Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Tallinn, Estonia, Aug. 2019, pp. 905–915, doi: 10.1145/3338906.3340452.

Results

Customization & Tenant-Isolation



Request Content:
Header: {
Bearer Token
},
data: {
...
}

Nordli E.T., Nguyen P.H., Chauvel F., Song H. (2020) Event-Based Customization of Multi-tenant SaaS Using Microservices. In: Bliudze S., Bocchi L. (eds) Coordination Models and Languages. COORDINATION 2020. Lecture Notes in Computer Science, vol 12134. Springer, Cham. https://doi.org/10.1007/978-3-030-50029-0_11

Tenant isolation for security, especially together with deep customization enabled

- We can manage all the tenants’ customization microservices, in how they are authorized to customize the main product for a specific tenant,
- Need administrating and monitoring the customization microservices at runtime.
- Deploying customization microservices on separate containers/clusters/hosts for different tenants and the main product is also very important for tenant isolation.

Tenant Isolation at app level and cluster level

- Ralph Squillace had a talk on Leap 2019 in Redmond on “Slicing and Dicing: Tenants and Azure Kubernetes Service (AKS)”. He discussed tenant isolation with regard to Kubernetes and Docker isolation. He concluded that Tenant isolation must be handled at an application level.
- Furthermore, separate clusters/hosts should be used to achieve adequate isolation, as Docker shares cores without adequate virtualization isolation. (https://www.microsoft.com/nb-no/leap/default.aspx)
  https://docs.microsoft.com/en-us/azure/aks/best-practices

Conclusion & Future Work

Conclusion & Future Work

MIGRATING MONOLITHIC APPLICATIONS TO MICROSERVICES-BASED CUSTOMIZABLE MULTI-TENANT APPLICATIONS

Sindre Grønstøl Haugeland[1], Phu H. Nguyen[2*], Franck Chauvel[2] , Hui Song[2]

[1] University of Oslo, Oslo, Norway, sindrgro@ifi.uio.no
[2] SINTEF, Oslo, Norway, firstname.lastname@sintef.no , [*] Presenter

Online @ the International Conference on Microservices, September 8-10, 2020

1) Our Approach has combined migration to MSA and transition to Multi-tenancy with an aim for enabling customization;

2) The main goal is to add the necessary infrastructure to allow (deep) customization of the application behaviour in multi-tenant SaaS context;

3) The target MSA-based multi-tenant SaaS is suitable to enable tenant-specific deep customization using tenant-specific customization microservices.

Tenant-isolation must be at app level and cluster level. We are still working on a systematic approach for tenant isolation.

The work presented in this talk has received funding from the Research Council of Norway under the grant agreement numbers 256594 (Cirrus).

SINTEF

Technology for a better society