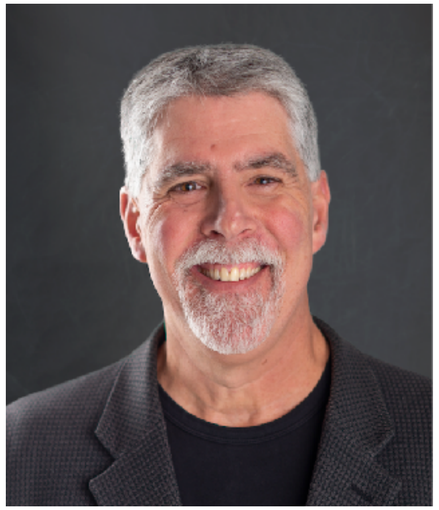
O'REILLY®

LIVE ONLINE TRAINING

Microservices Architecture and Design



Mark Richards

Independent Consultant

Hands-on Software Architect / Published Author

Founder, DeveloperToArchitect.com

https://www.linkedin.com/in/markrichards3

@markrichardssa

v8e

Apache ZooKeeper™

CONSUL

ANSIBLE

cloudstack
open source cloud computing

docker

CHEF™

kubernetes
by Google

Dropwizard

Prometheus

spring
boot

Goals
The desired results
plans and commits to achieve
the end toward which effort is
establishing specific, realistic
time-targeted objectives.

gain a full understanding of microservices from an architecture and design standpoint

gain a full understanding of microservices from an architecture and design standpoint

gain an understanding of the implications and challenges of the microservices architecture style

gain a full understanding of microservices from an architecture and design standpoint

gain an understanding of the implications and challenges of the microservices architecture style

validate whether microservices is the right choice for your company and understand the alternatives

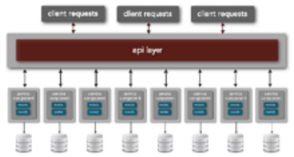
course agenda

course agenda

architectural
modularity

course agenda

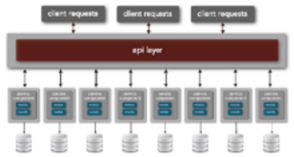
architectural
modularity



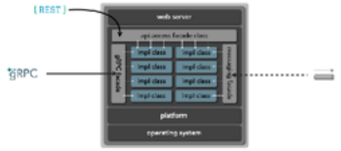
microservices
core concepts

course agenda

architectural
modularity



microservices
core concepts

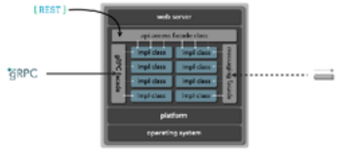


service
design

course agenda

architectural
modularity

microservices
core concepts

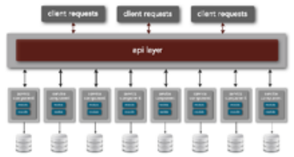


service
design

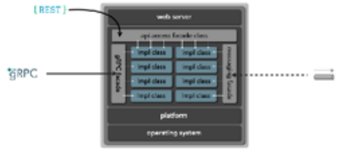
service
granularity

course agenda

architectural
modularity



microservices
core concepts



service
design

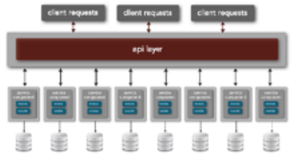
service
granularity

[FunctionalService]
[OrchestrationService]
[AggregationService]
[AdapterService]
[InfrastructureService]
[DataService]

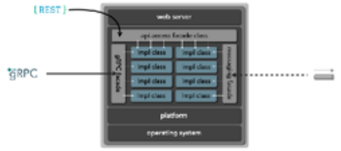
service
types

course agenda

architectural
modularity



microservices
core concepts



service
design

service
granularity

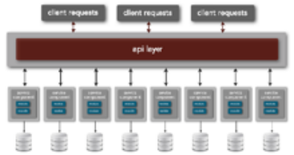
```
[FunctionalService]
[OrchestrationService]
[AggregationService]
[AdapterService]
[InfrastructureService]
[DataService]
```

service
types

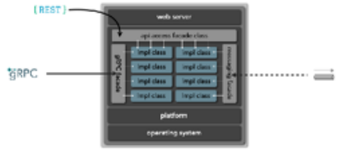
identifying
services

course agenda

architectural
modularity



microservices
core concepts



service
design

service
granularity

[FunctionalService]
[OrchestrationService]
[AggregationService]
[AdapterService]
[InfrastructureService]
[DataService]

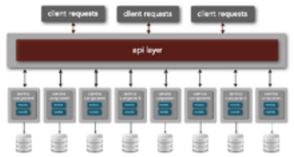
service
types

identifying
services

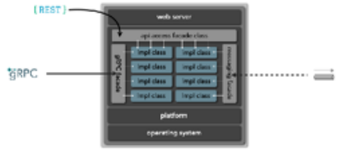
code sharing
techniques

course agenda

architectural
modularity



microservices
core concepts



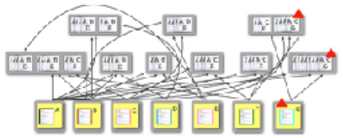
service
design

service
granularity

[FunctionalService]
[OrchestrationService]
[AggregationService]
[AdapterService]
[InfrastructureService]
[DataService]

service
types

identifying
services

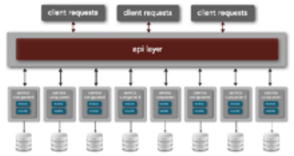


code sharing
techniques

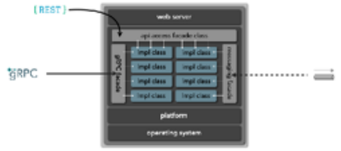
decomposing
data

course agenda

architectural modularity



microservices core concepts



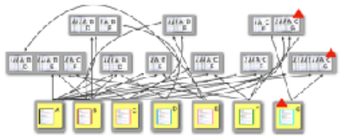
service design

service granularity

```
[FunctionalService]
[OrchestrationService]
[AggregationService]
[AdapterService]
[InfrastructureService]
[DataService]
```

service types

identifying services



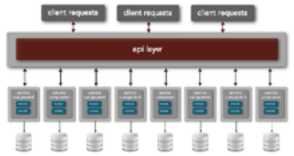
code sharing techniques

decomposing data

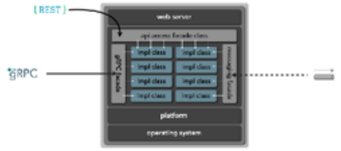
data ownership and access

course agenda

architectural modularity



microservices core concepts



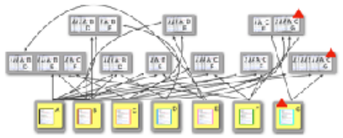
service design

service granularity

```
[FunctionalService]
[OrchestrationService]
[AggregationService]
[AdapterService]
[InfrastructureService]
[DataService]
```

service types

identifying services



code sharing techniques

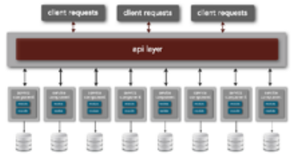
decomposing data

data ownership and access

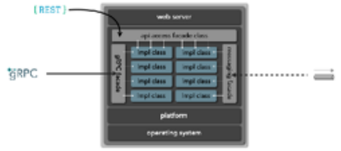
managing workflows

course agenda

architectural modularity



microservices core concepts



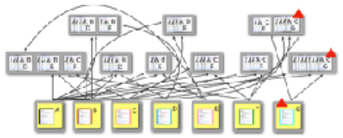
service design

service granularity

```
[FunctionalService]
[OrchestrationService]
[AggregationService]
[AdapterService]
[InfrastructureService]
[DataService]
```

service types

identifying services



code sharing techniques

decomposing data

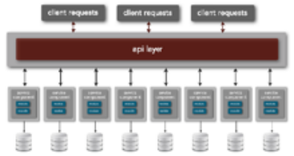
data ownership and access

managing workflows

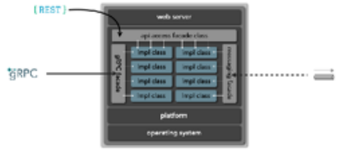
hybrid architectures

course agenda

architectural modularity



microservices core concepts



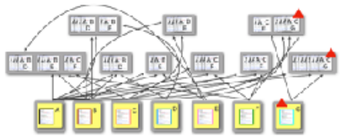
service design

service granularity

```
[FunctionalService]
[OrchestrationService]
[AggregationService]
[AdapterService]
[InfrastructureService]
[DataService]
```

service types

identifying services



code sharing techniques

decomposing data

data ownership and access

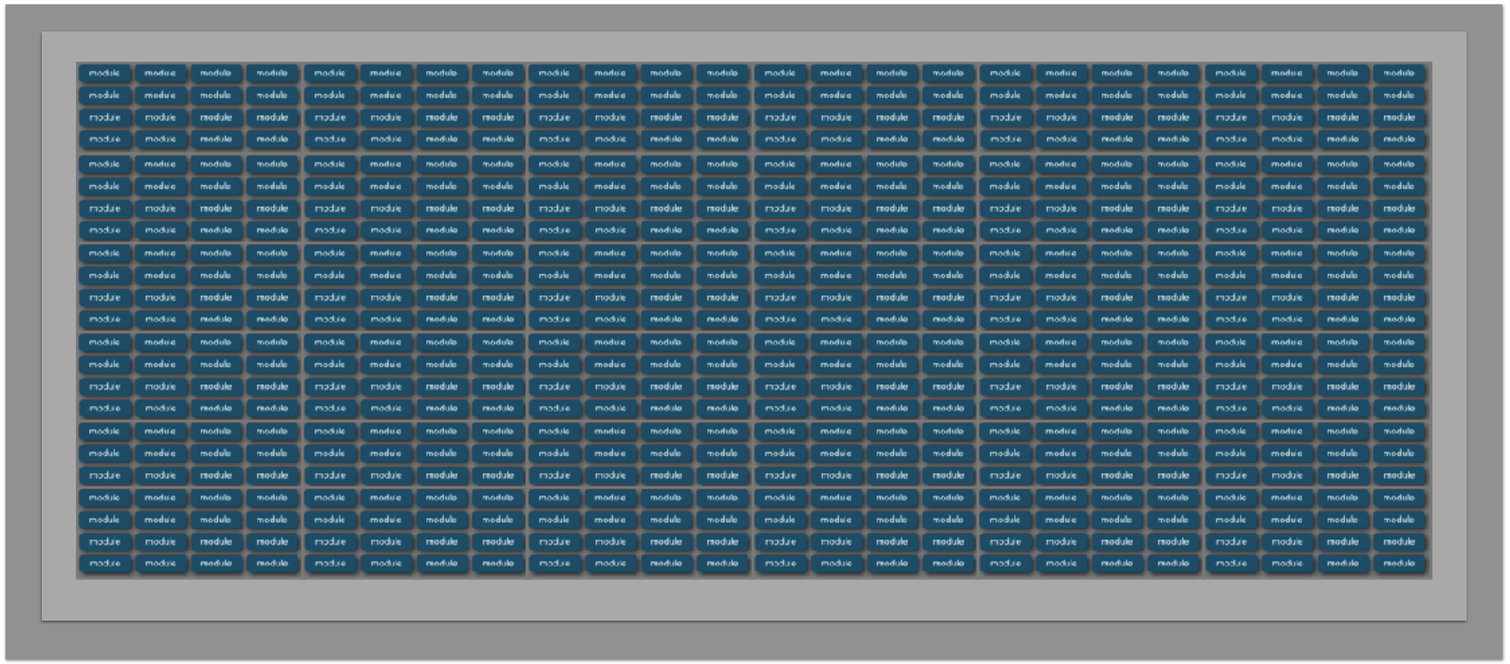
managing workflows

hybrid architectures

final words of advice

Architectural Modularity

architectural modularity



architectural modularity

INOX
STAINLESS
MADE IN

VICTORINOX
SWITZERL
STAIN
RO

WENGER

WENGER

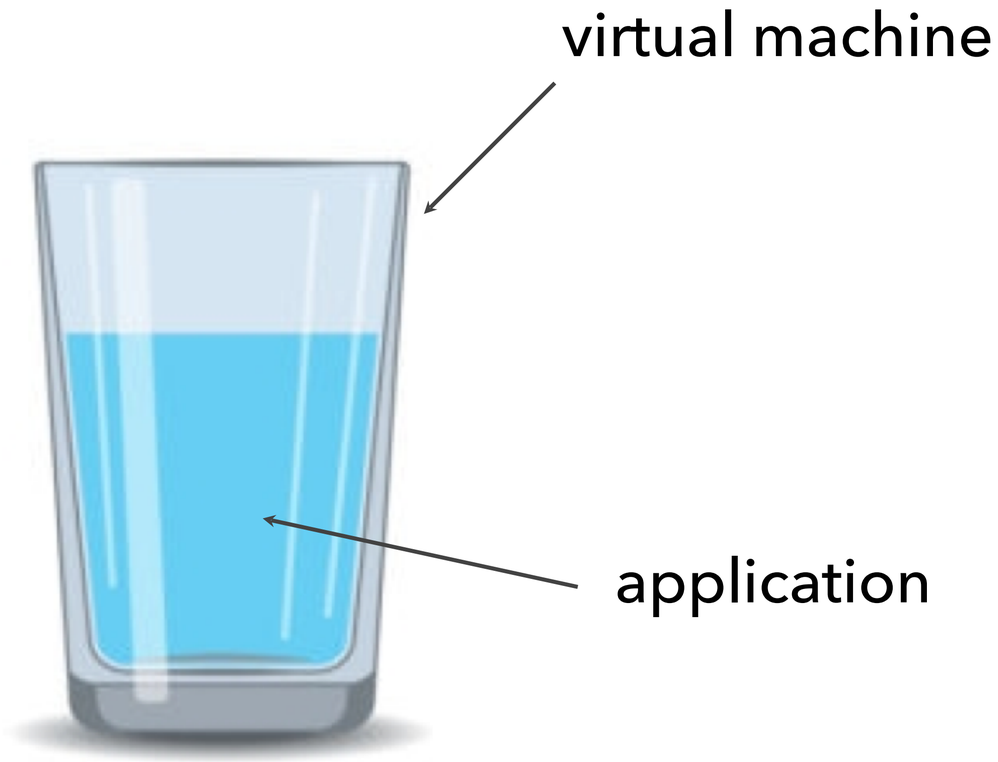
WENGER

architectural modularity

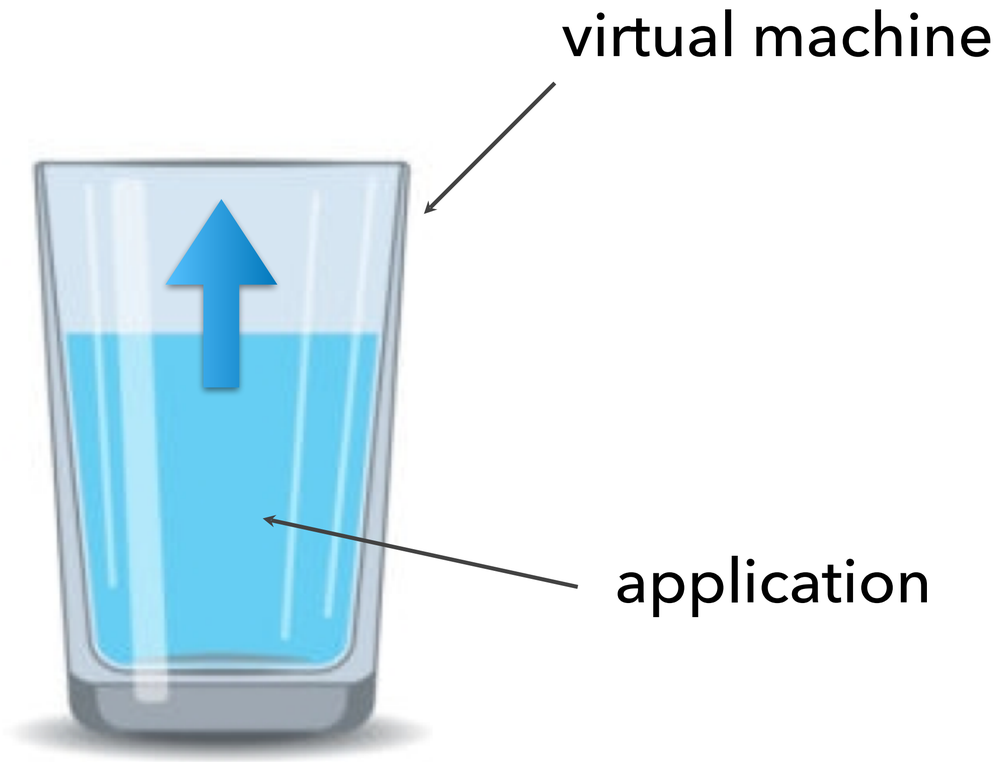
architectural modularity

virtual machine

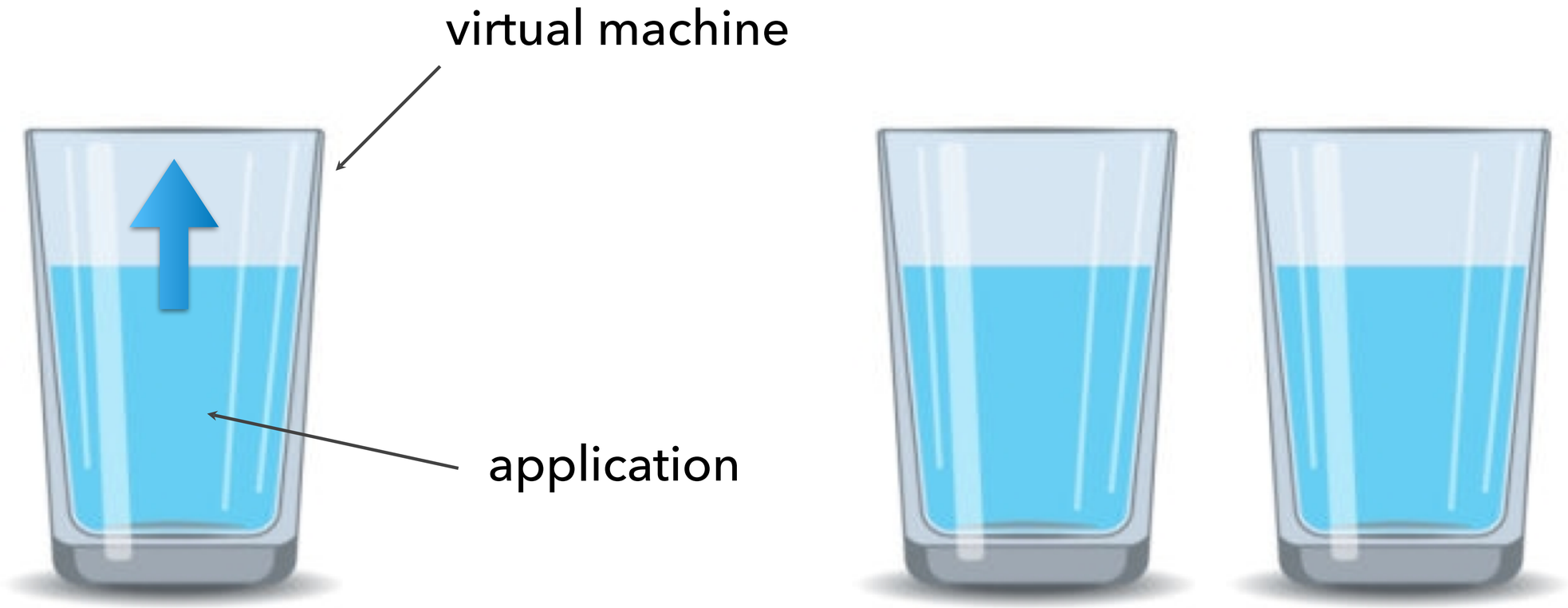
architectural modularity



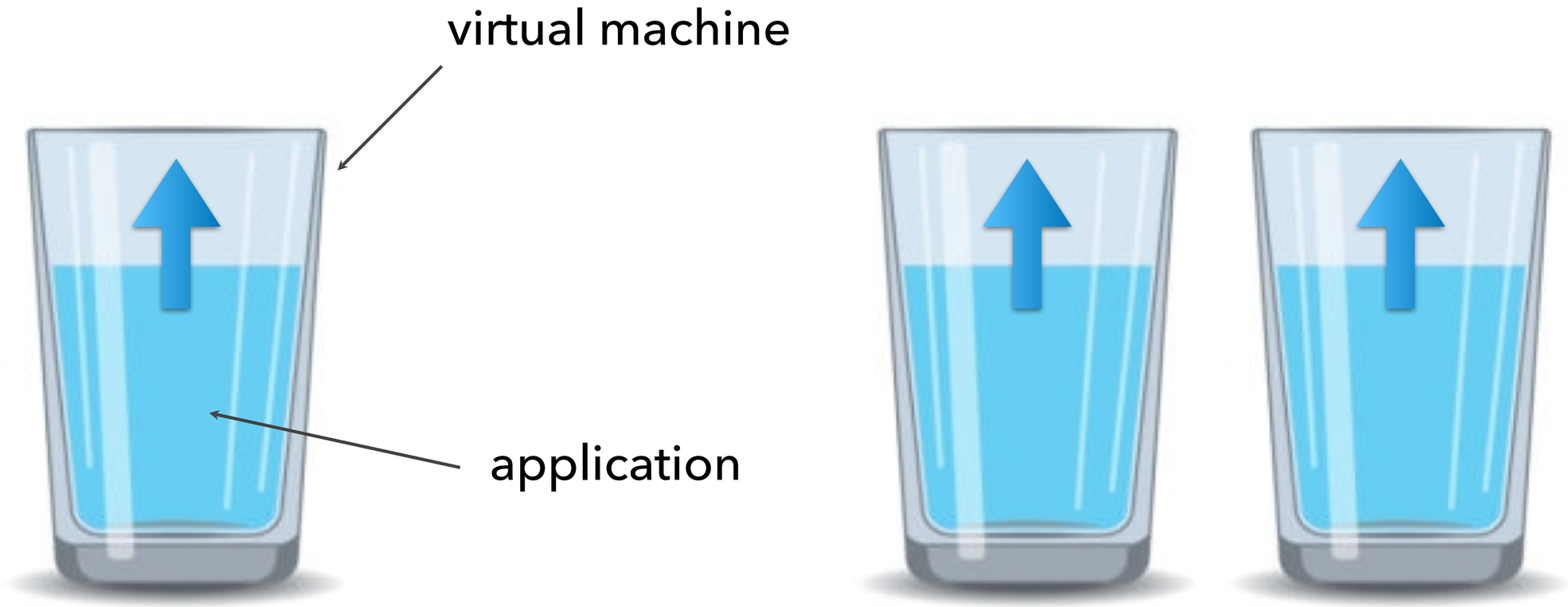
architectural modularity



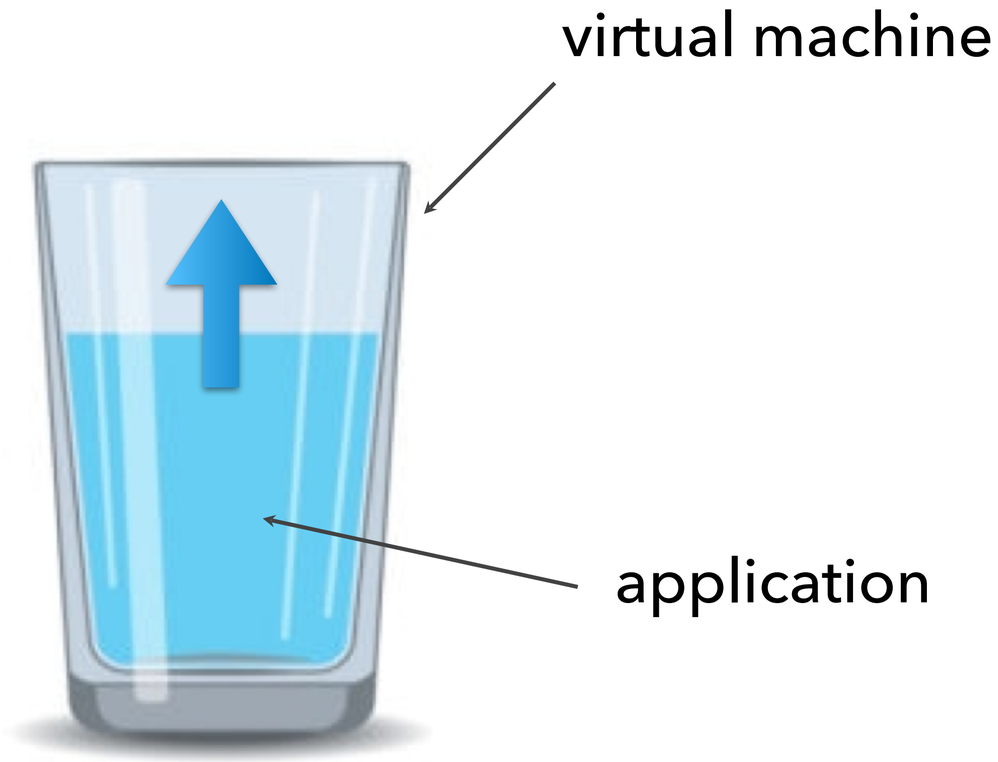
architectural modularity



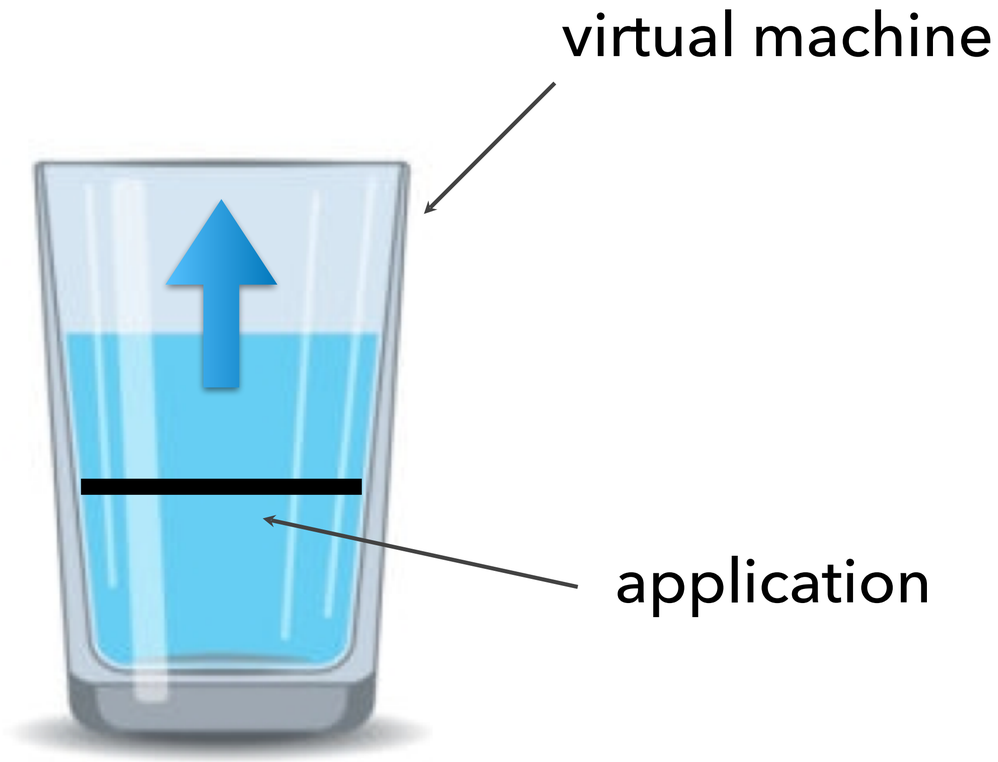
architectural modularity



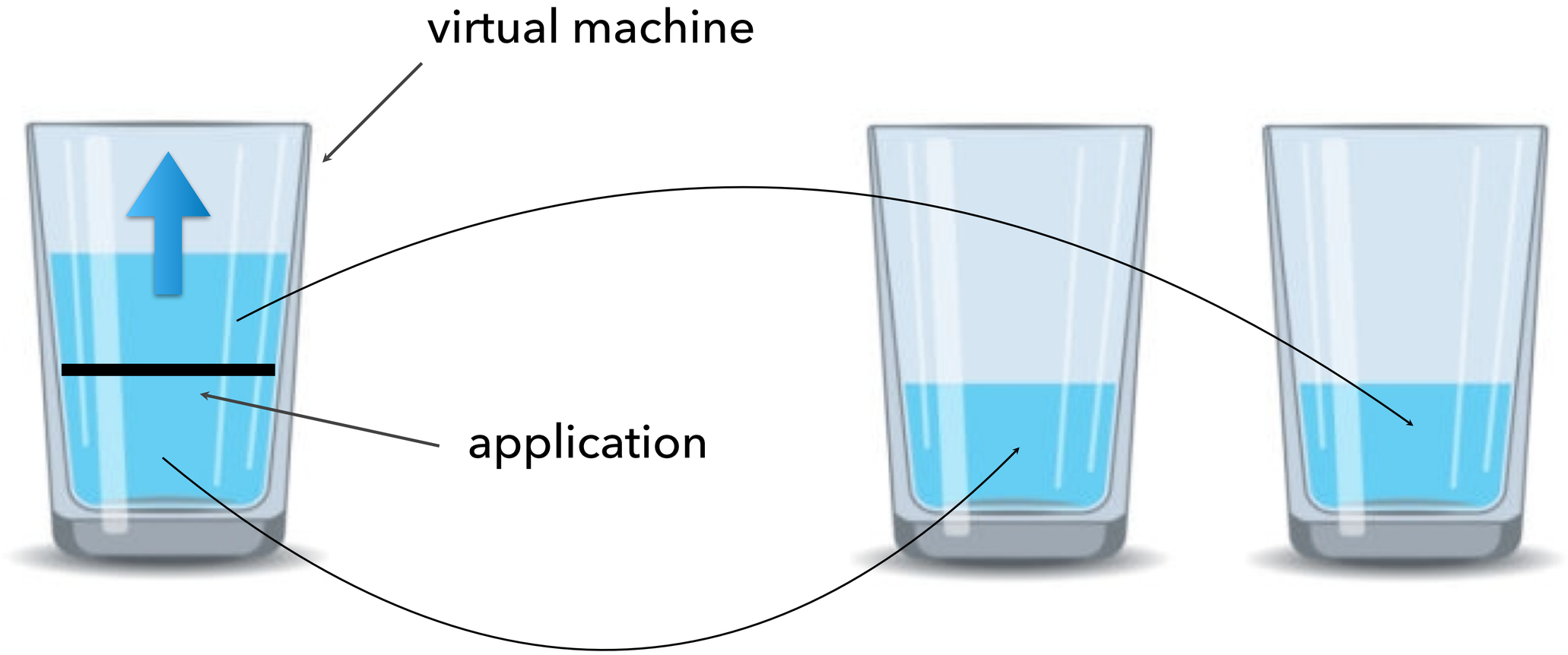
architectural modularity



architectural modularity

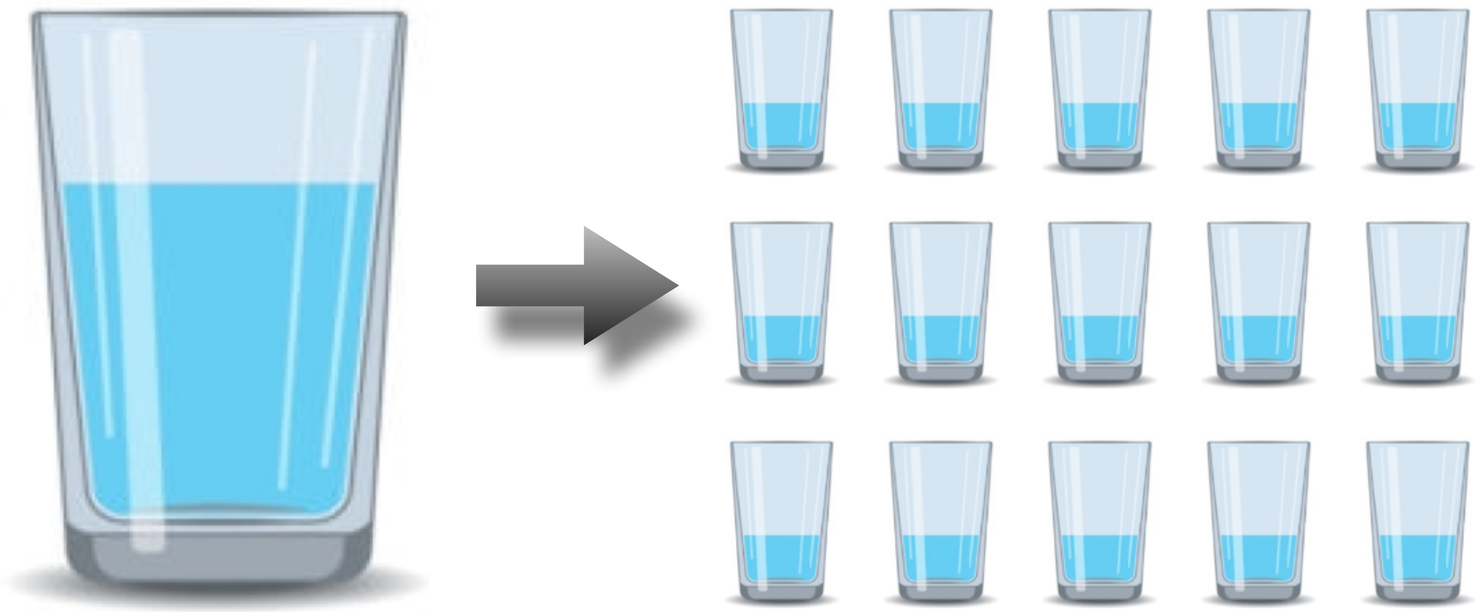


architectural modularity

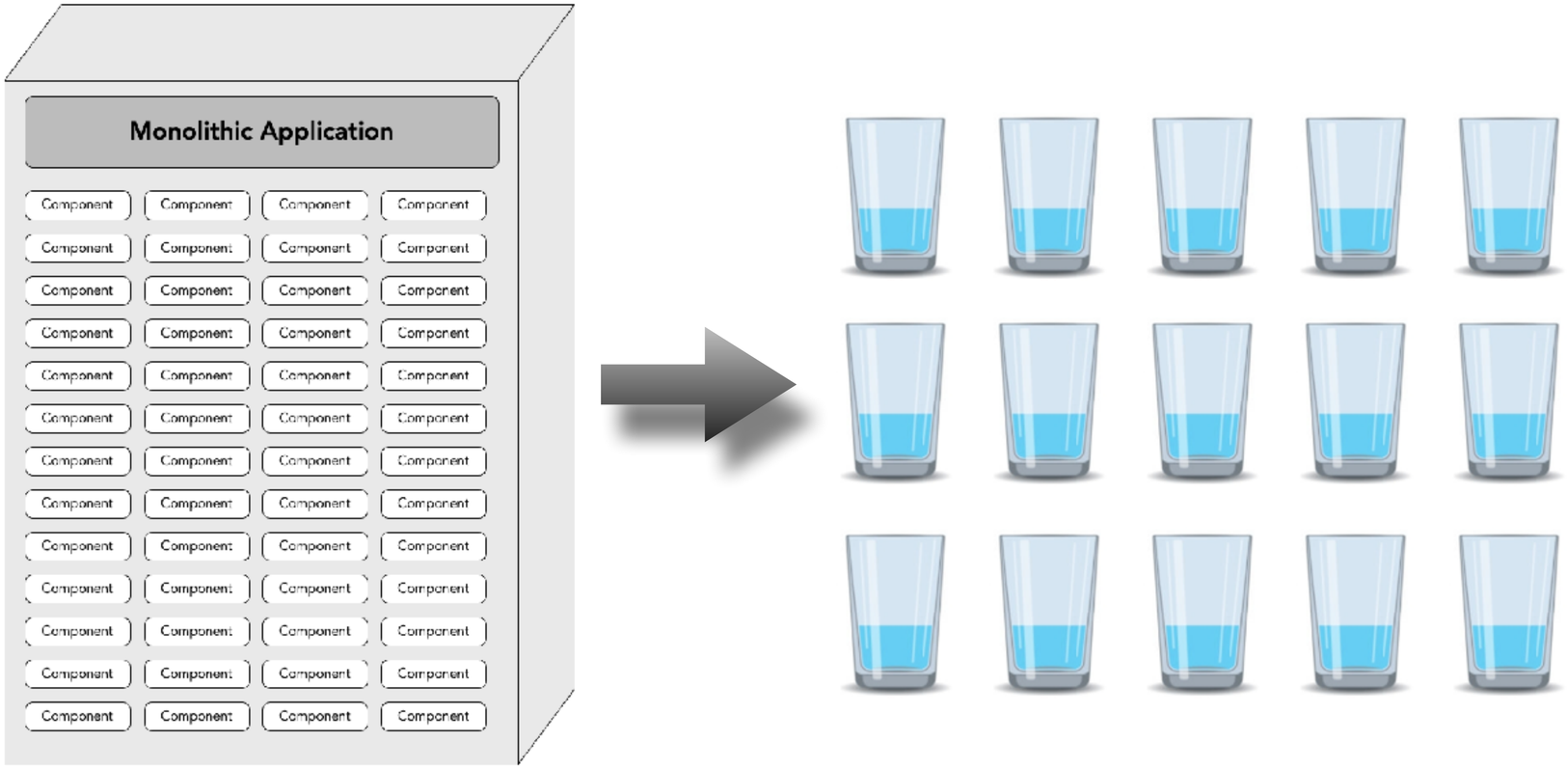


architectural modularity

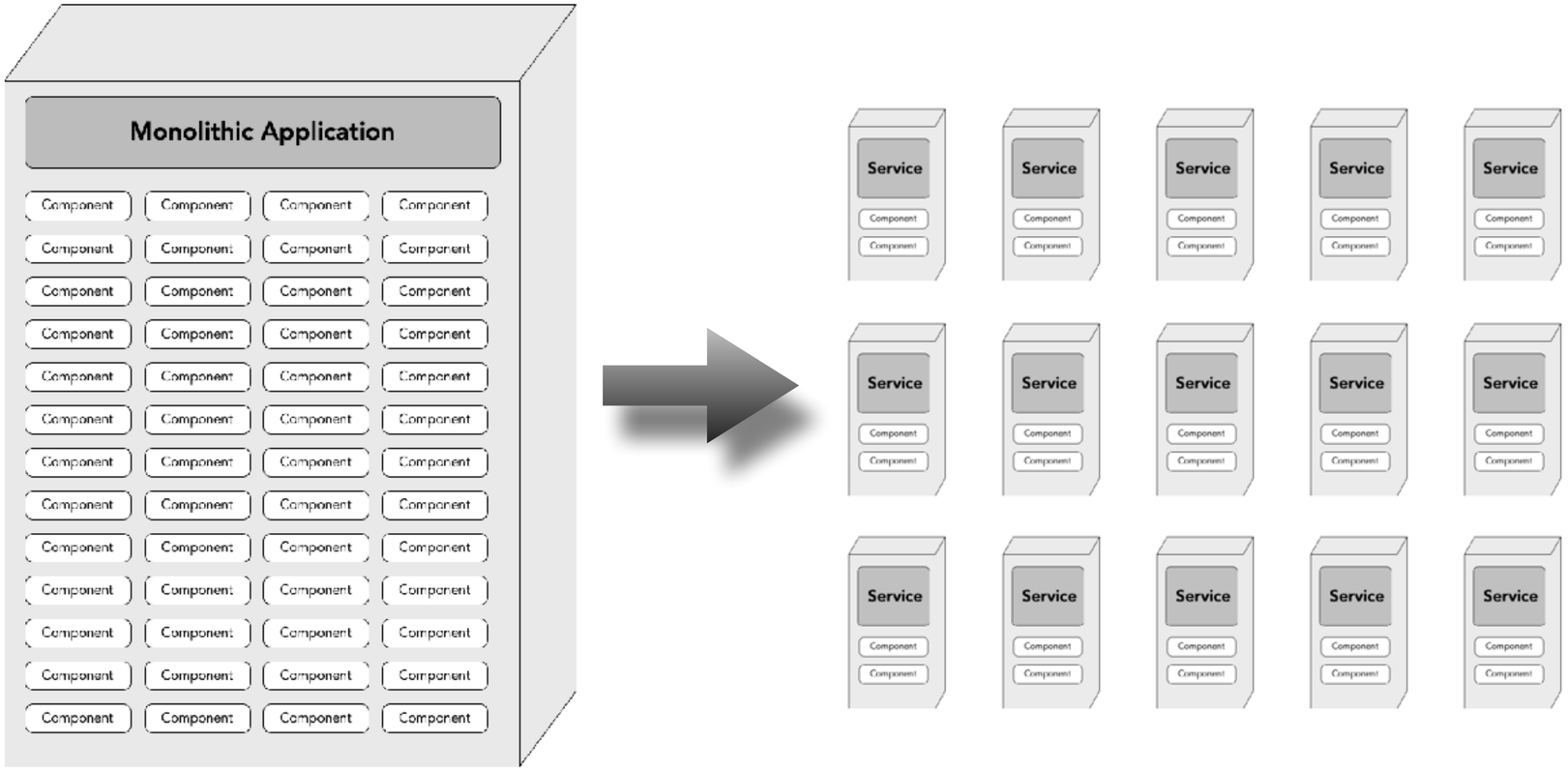
architectural modularity



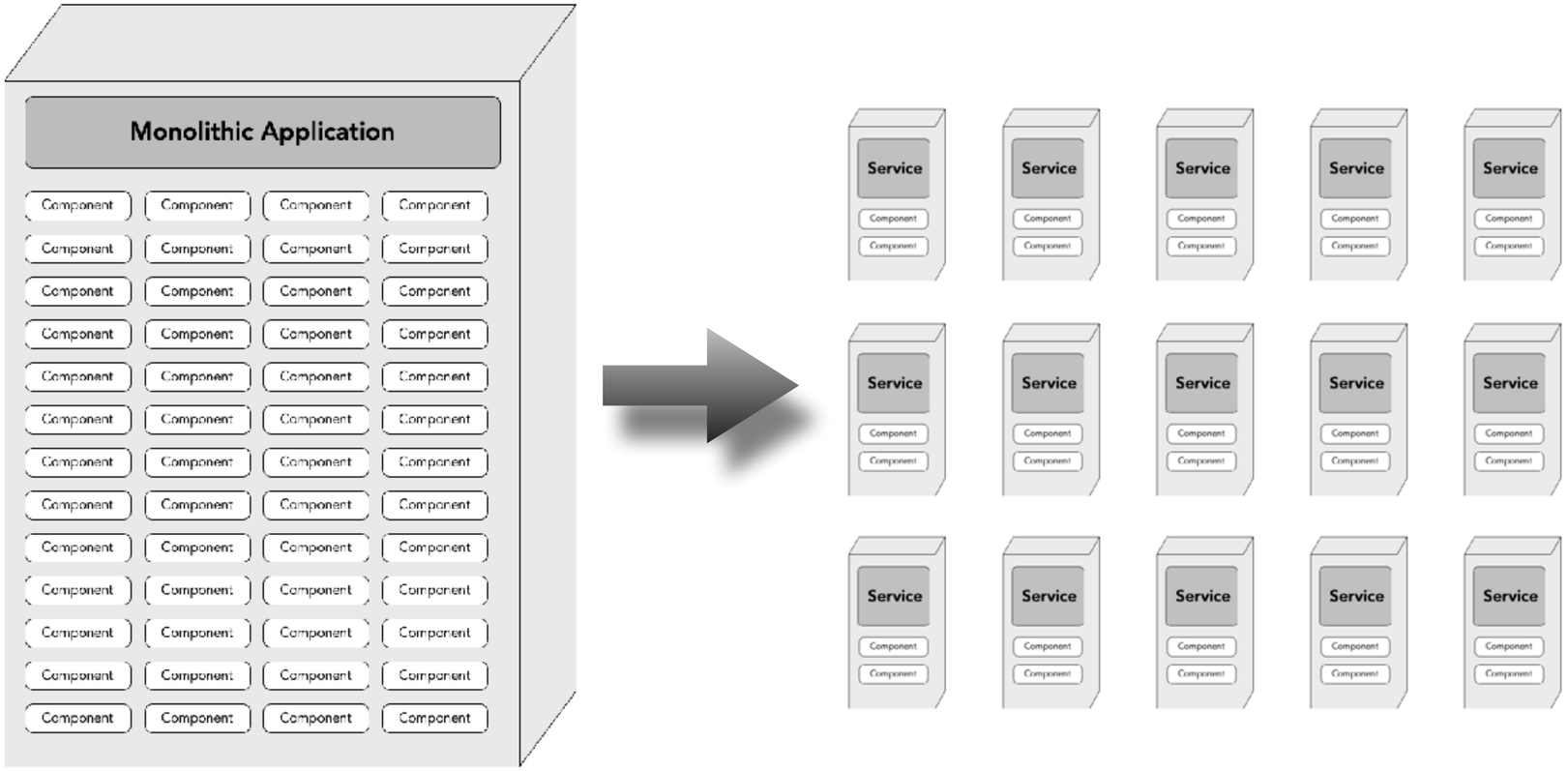
architectural modularity



architectural modularity

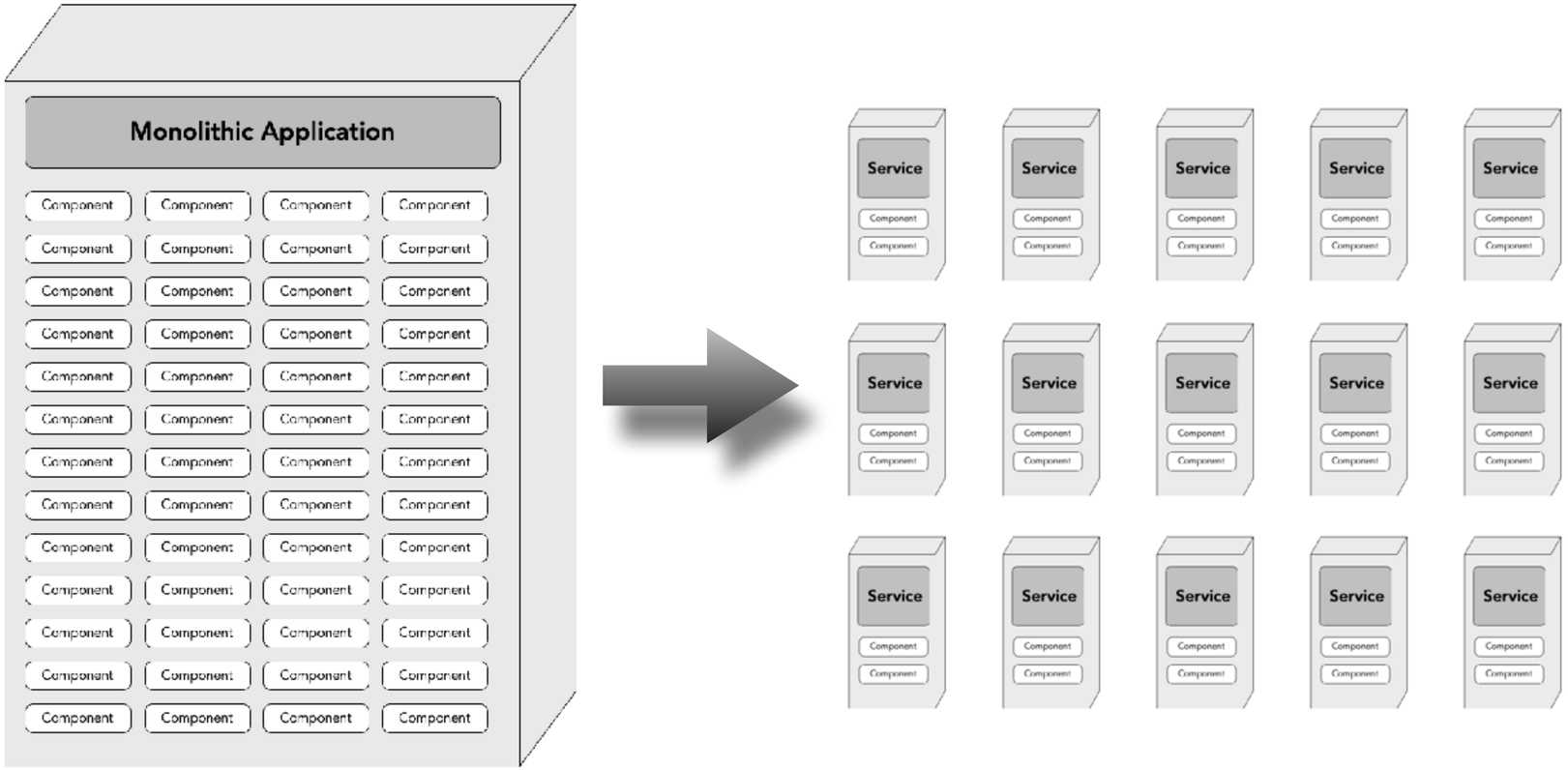


architectural modularity



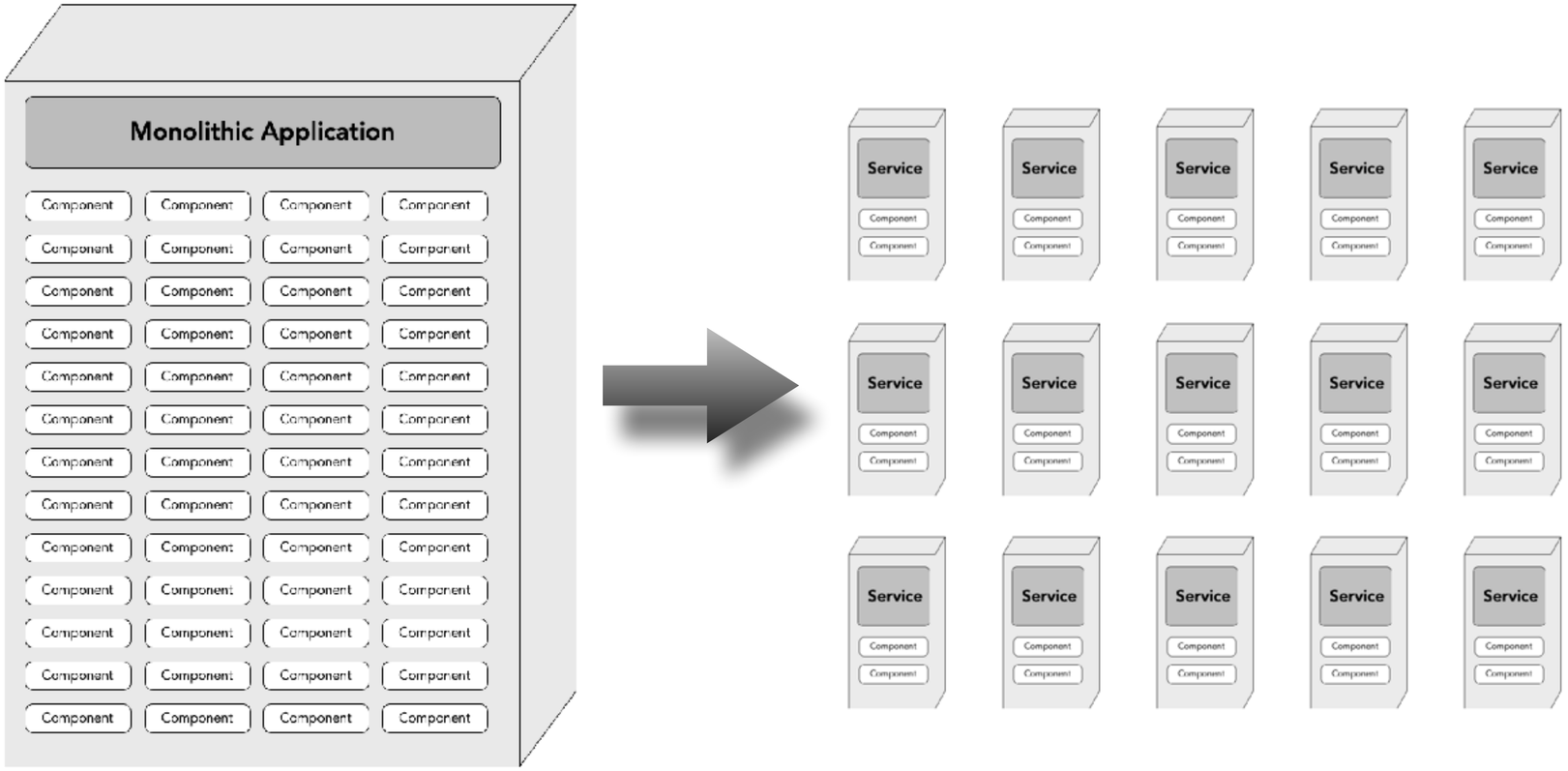
✓ maintainability

architectural modularity



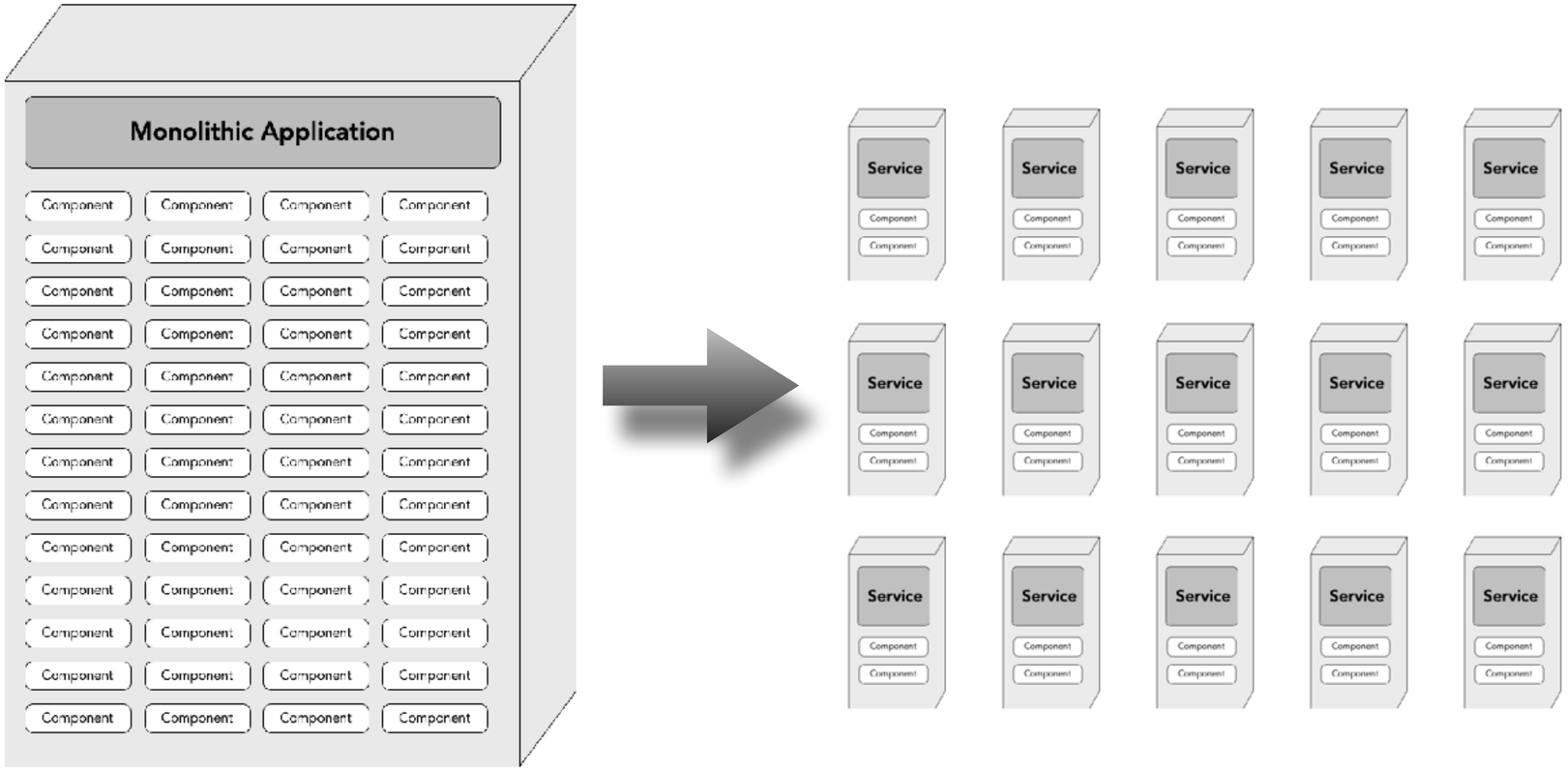
- ✓ maintainability
- ✓ testability

architectural modularity



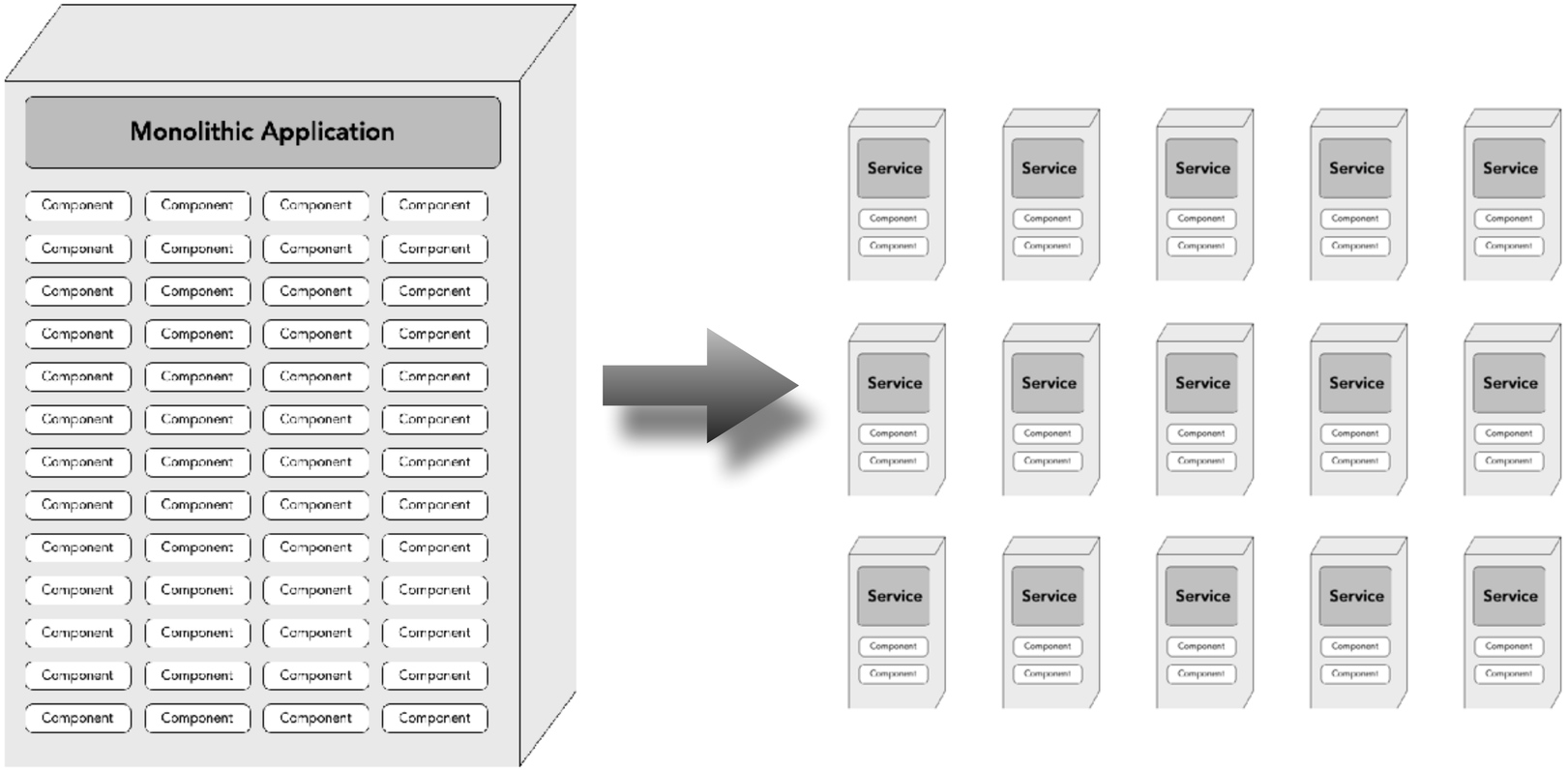
- ✓ maintainability
- ✓ testability
- ✓ deployability

architectural modularity



- ✓ maintainability
- ✓ testability
- ✓ deployability
- ✓ scalability

architectural modularity

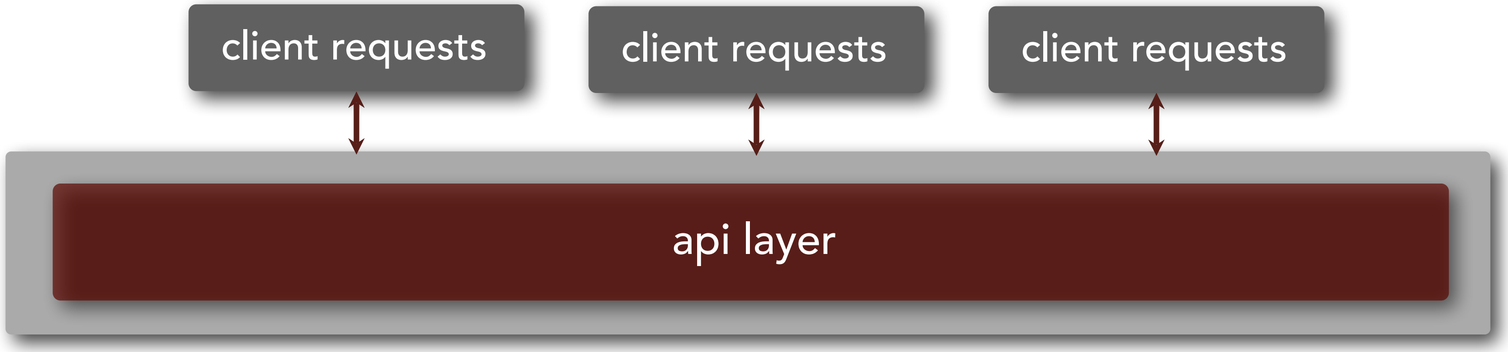


- ✓ maintainability
- ✓ testability
- ✓ deployability
- ✓ scalability
- ✓ availability

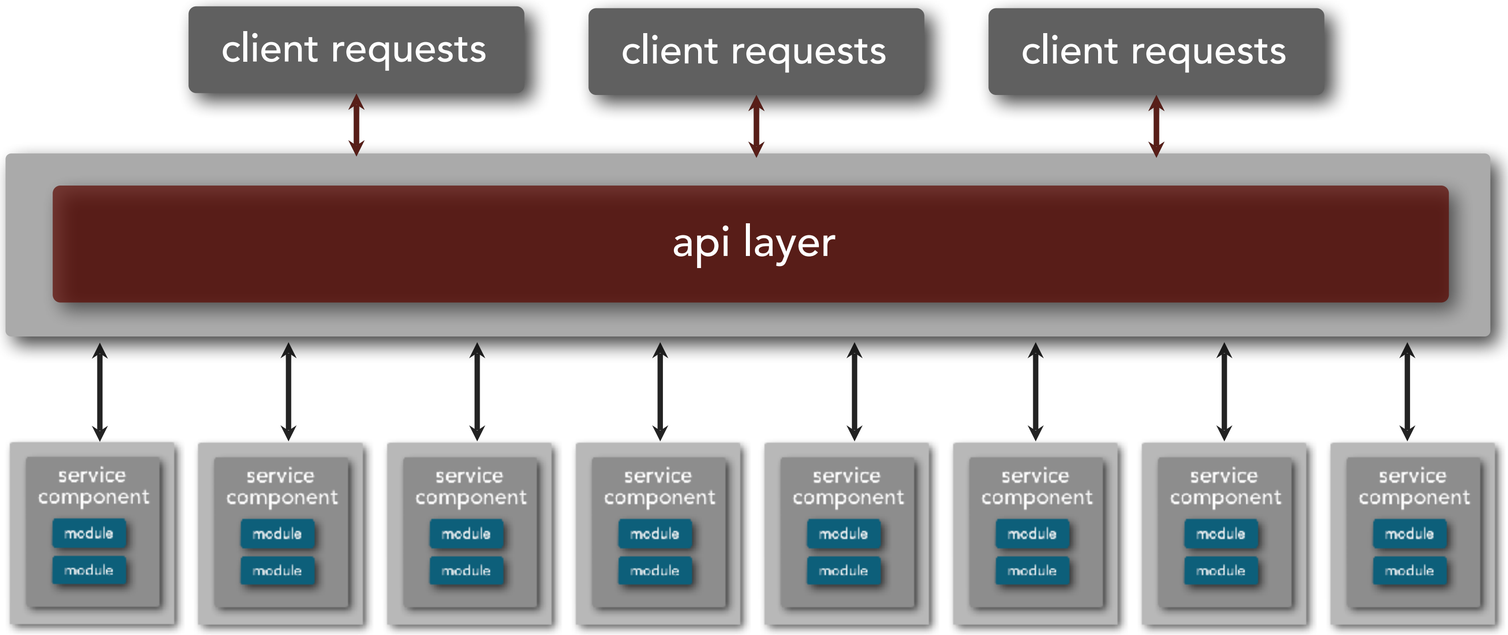
Microservices
Core Concepts

microservices core concepts

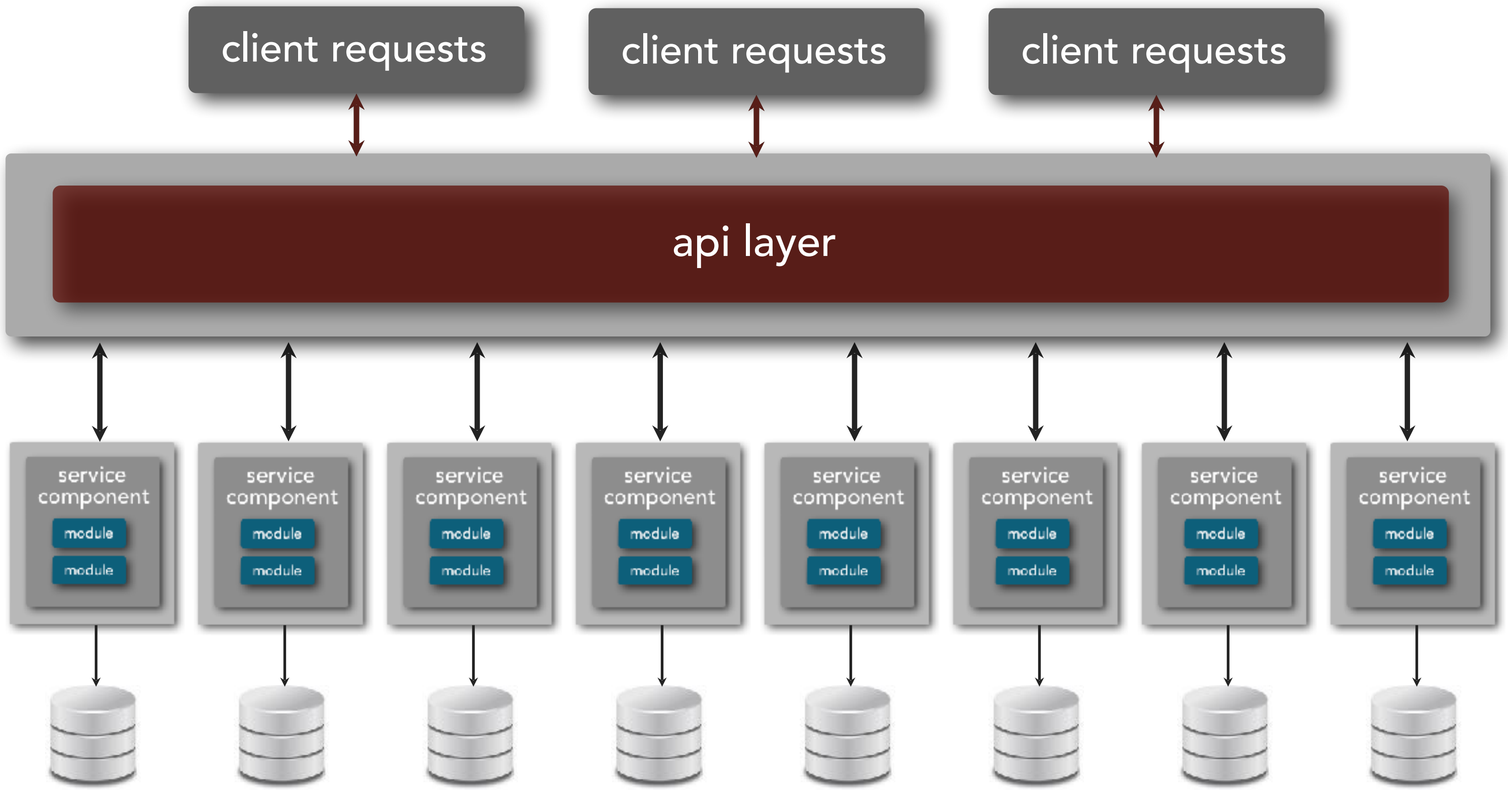
microservices core concepts



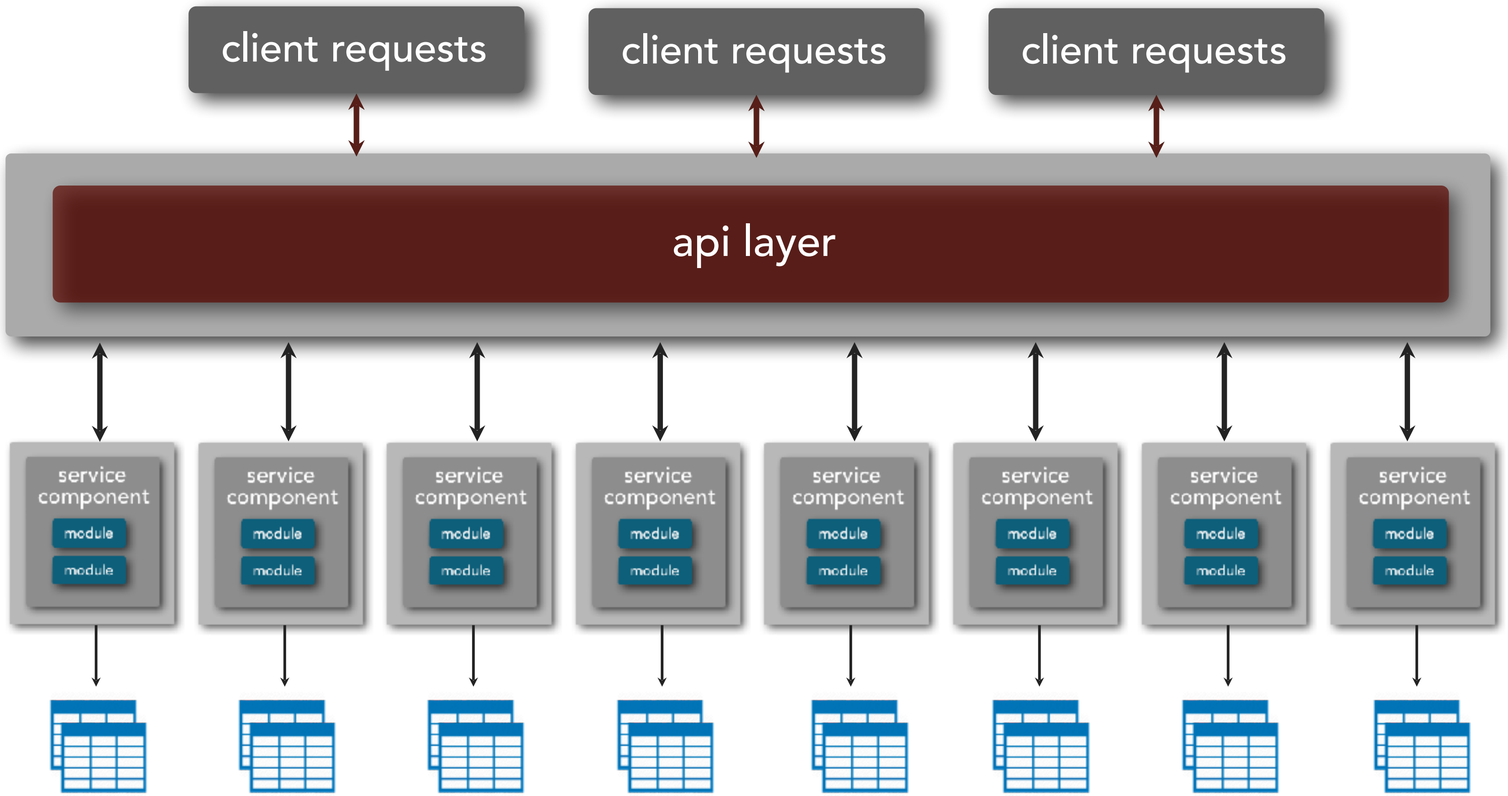
microservices core concepts



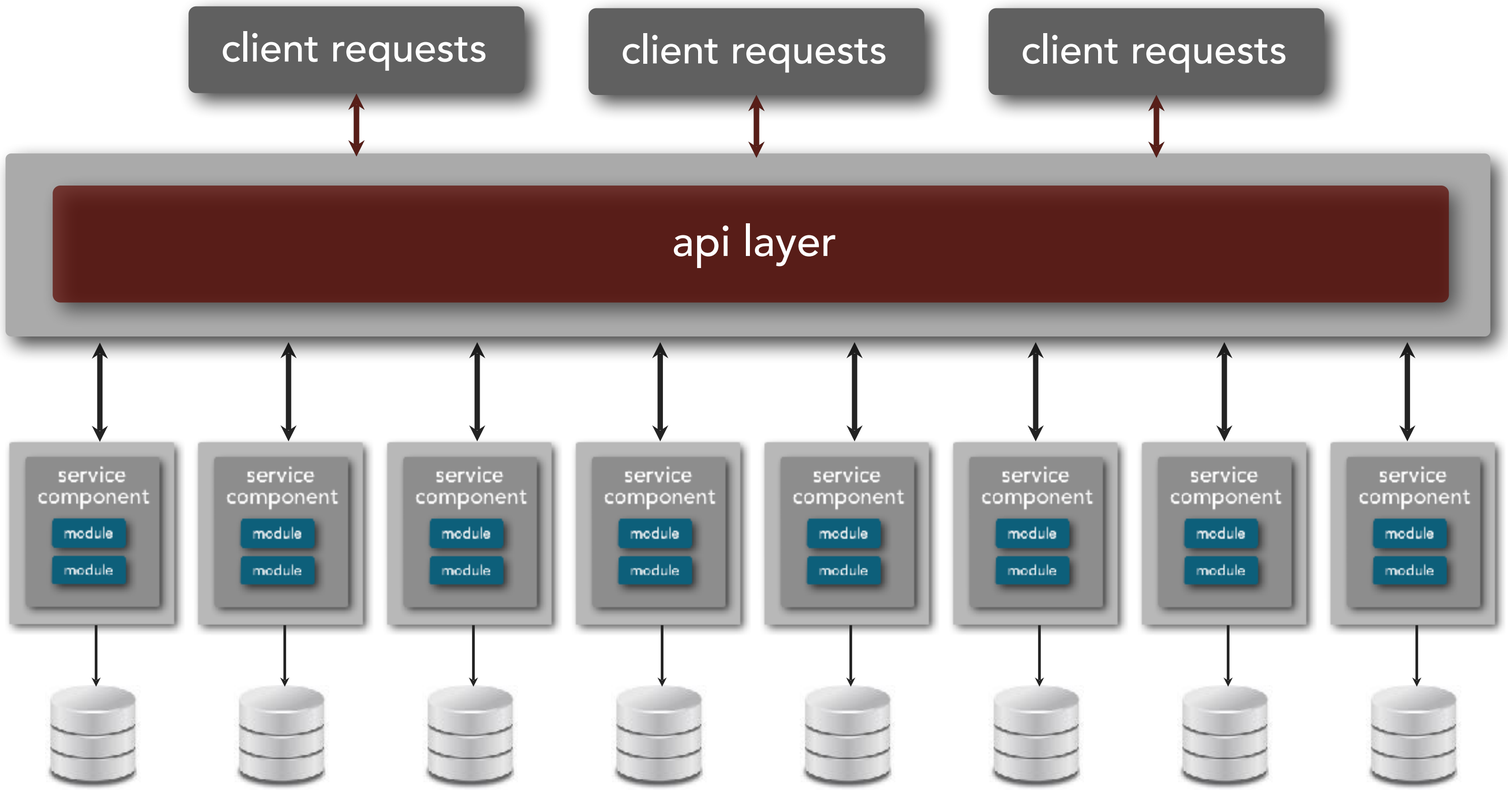
microservices core concepts



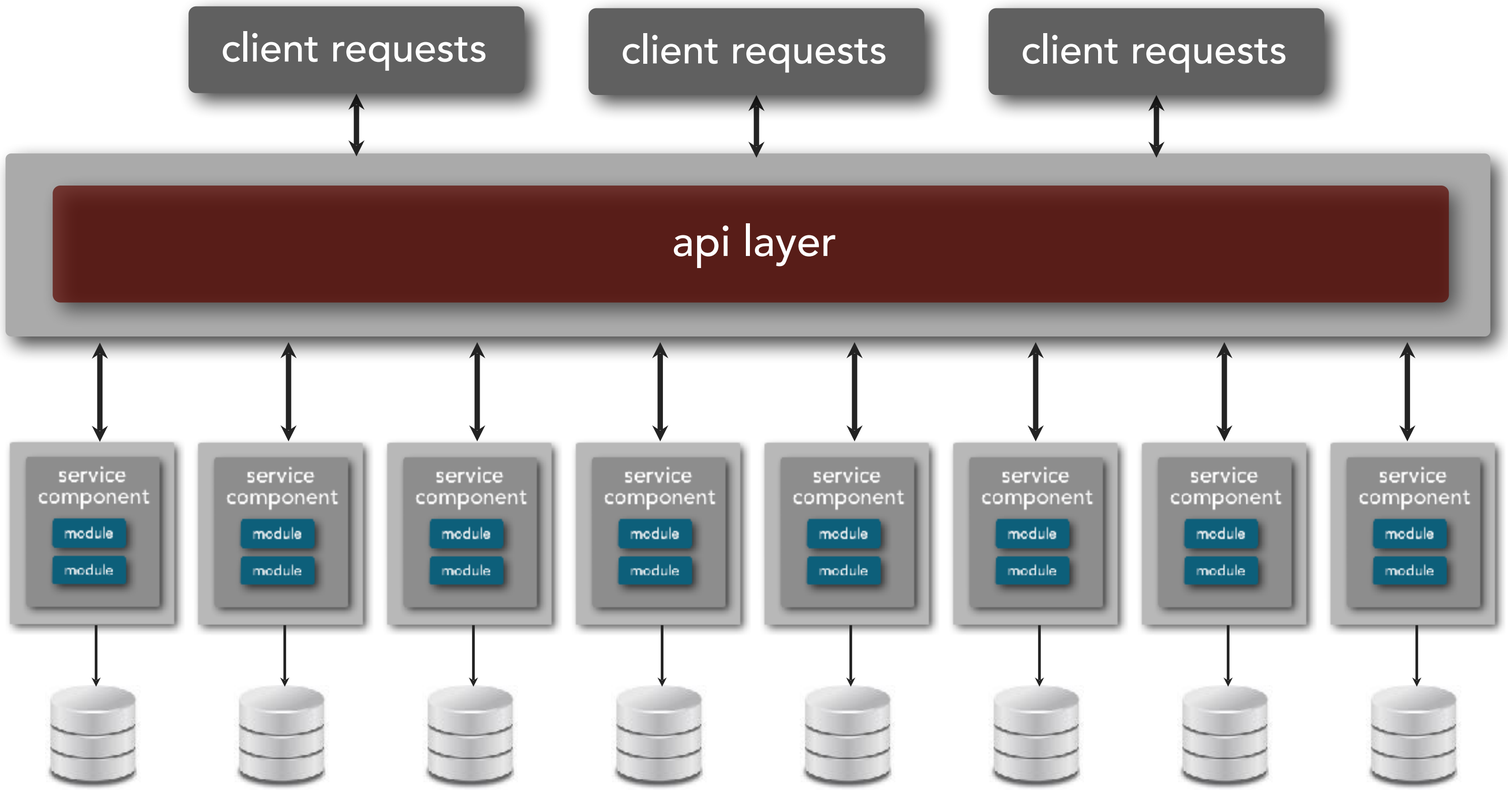
microservices core concepts



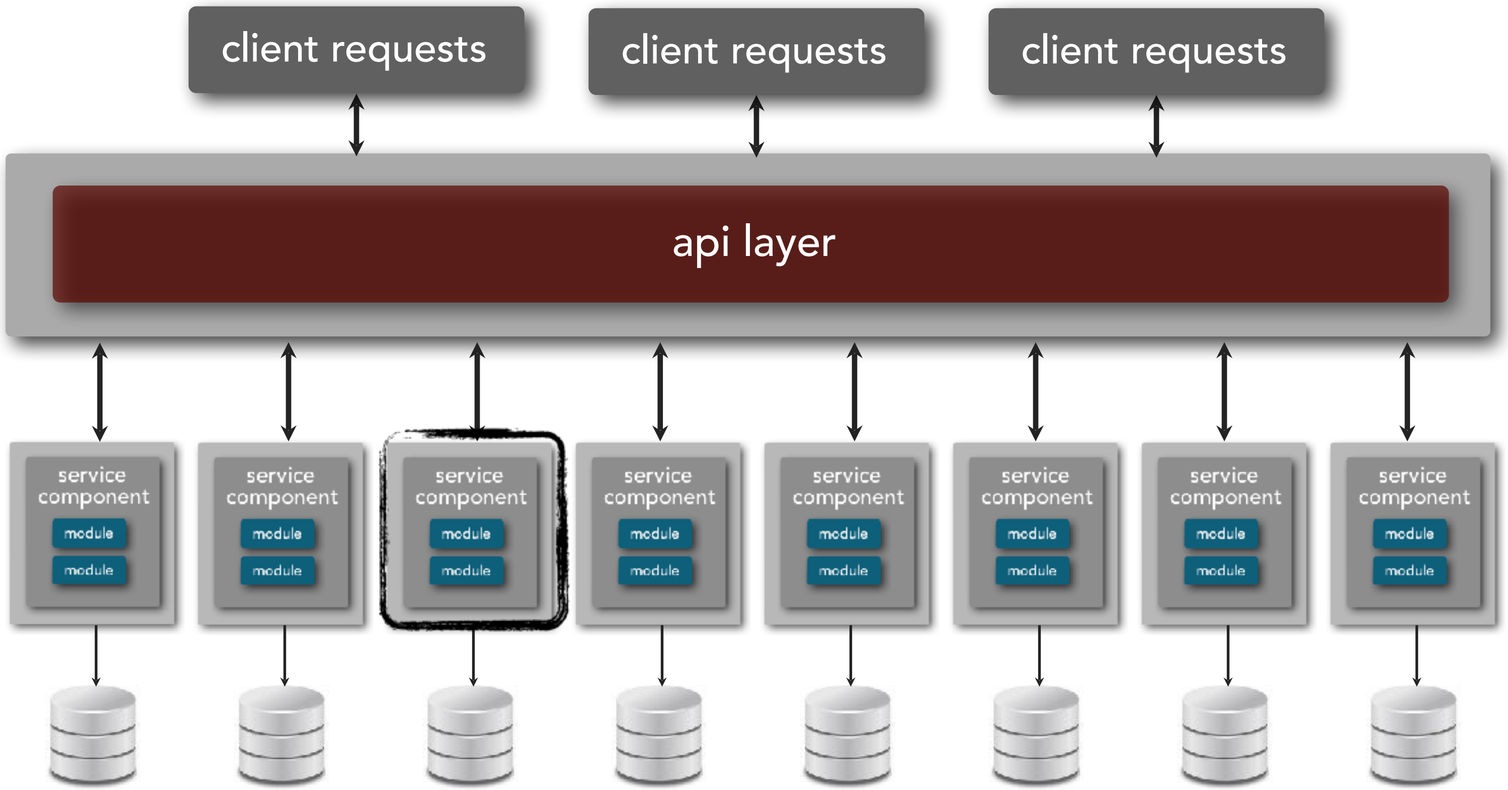
microservices core concepts



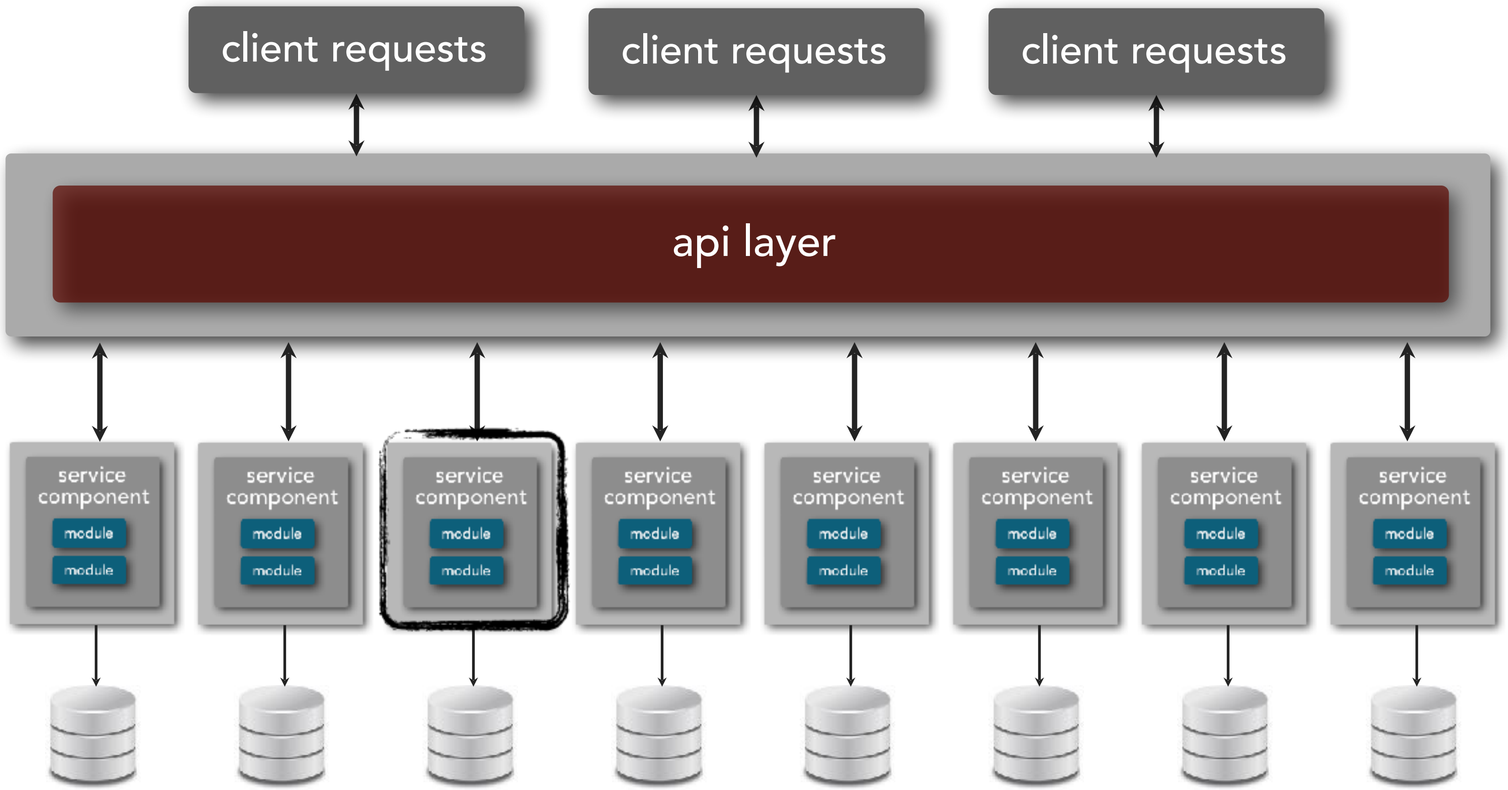
microservices core concepts



microservices core concepts

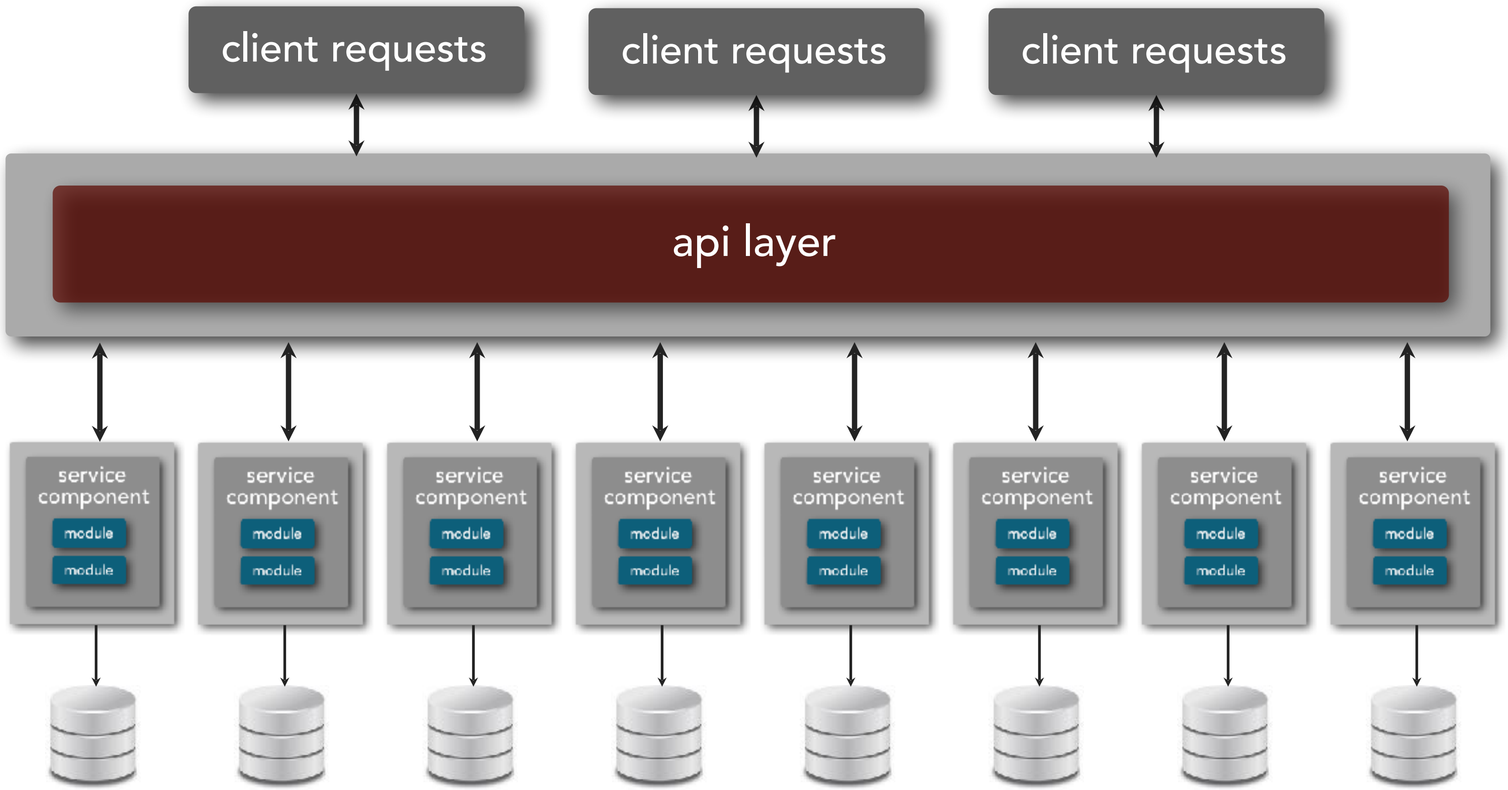


microservices core concepts



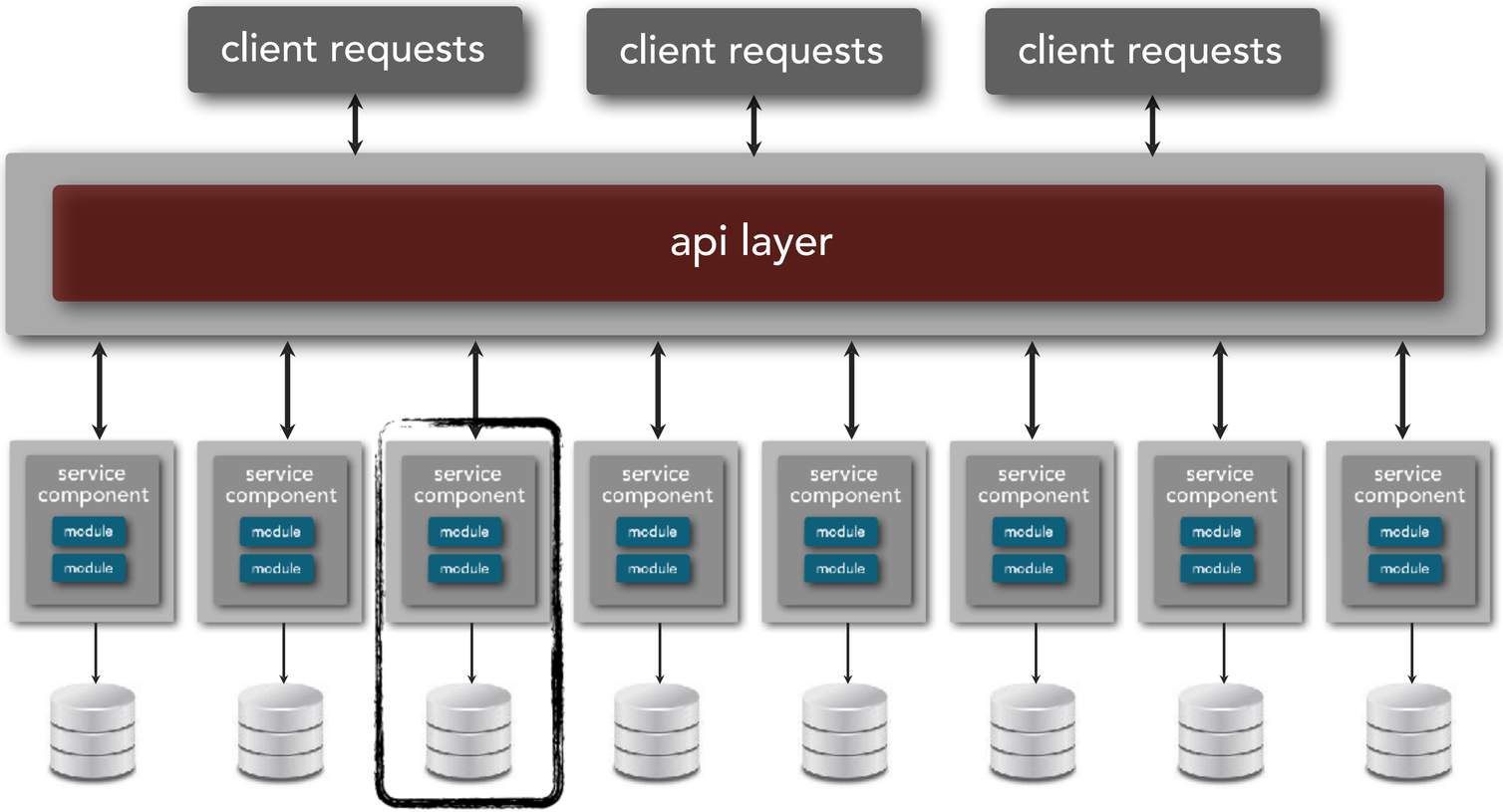
single-purpose, separately deployed unit of software that does one thing really well

microservices core concepts



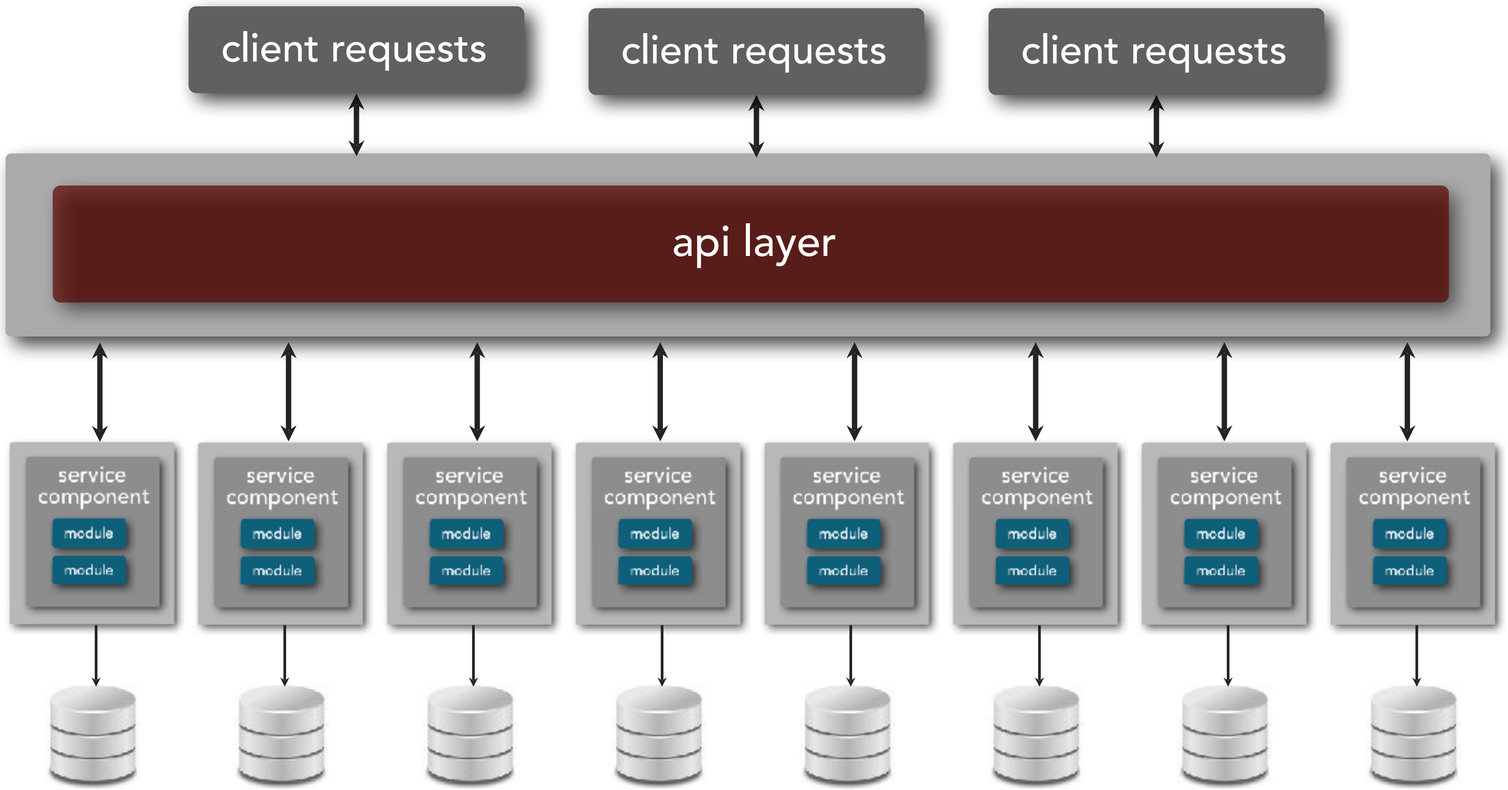
bounded context (share nothing architecture)

microservices core concepts



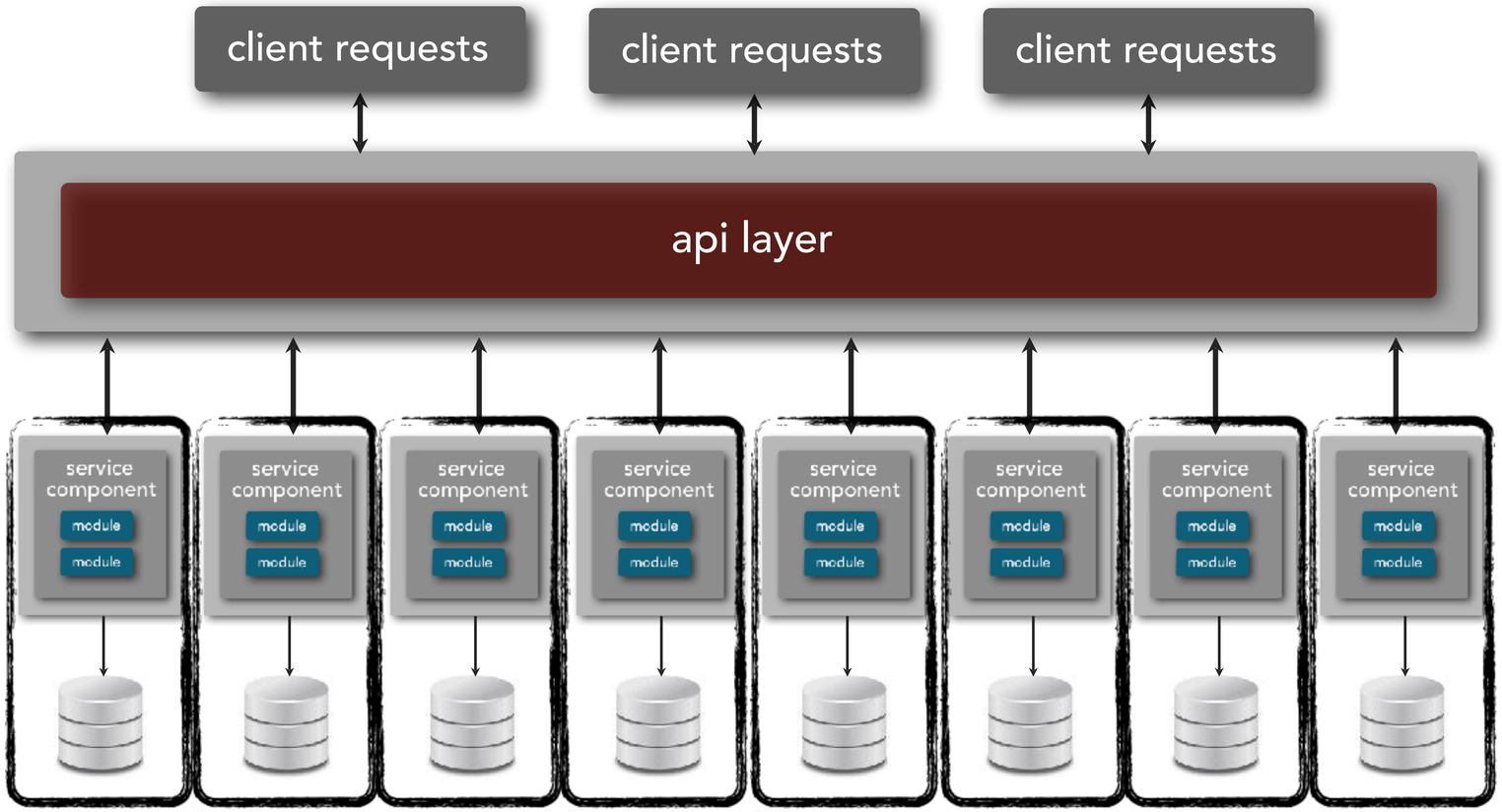
bounded context (share nothing architecture)

microservices core concepts



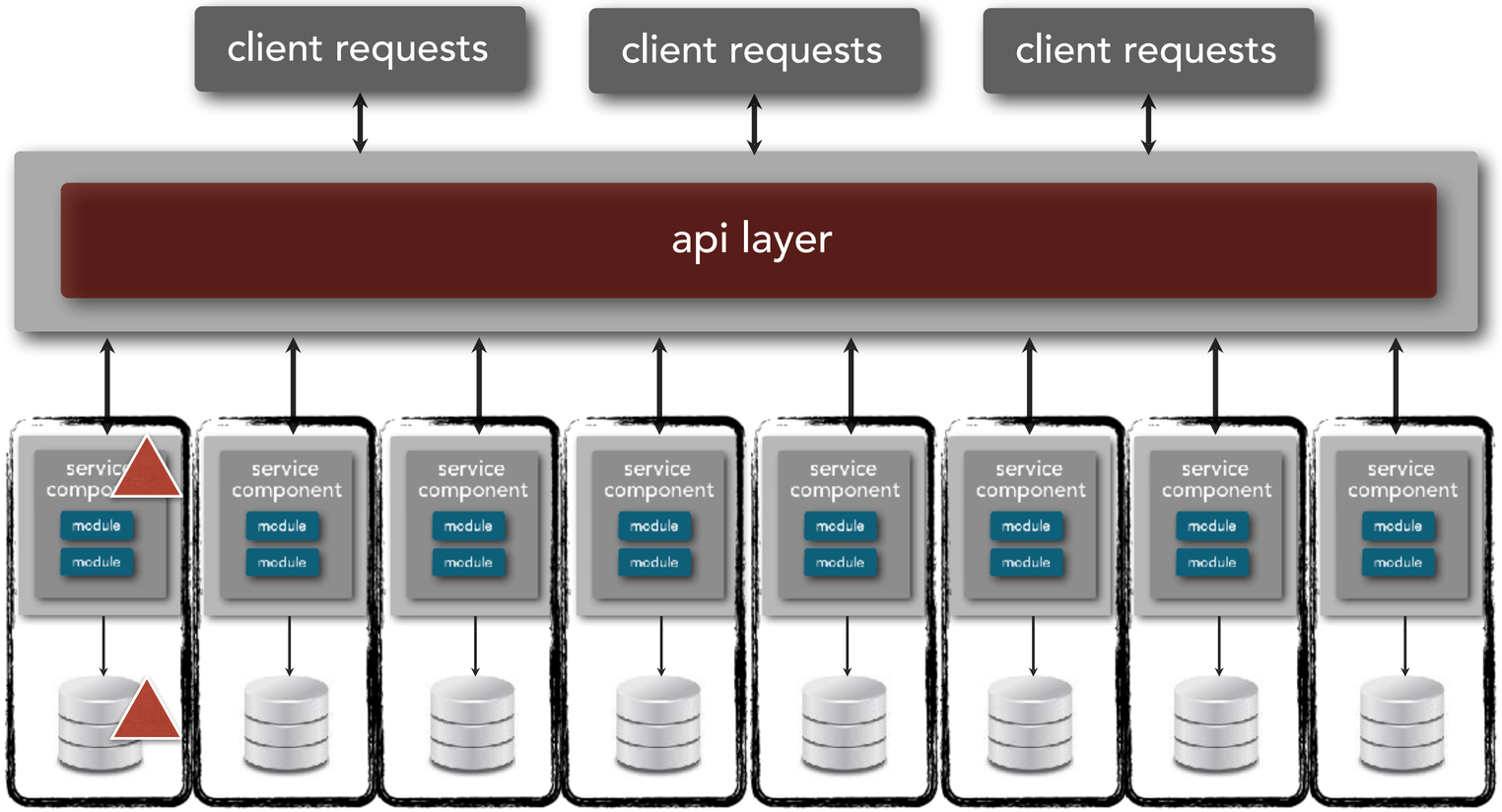
bounded context (share nothing architecture)

microservices core concepts



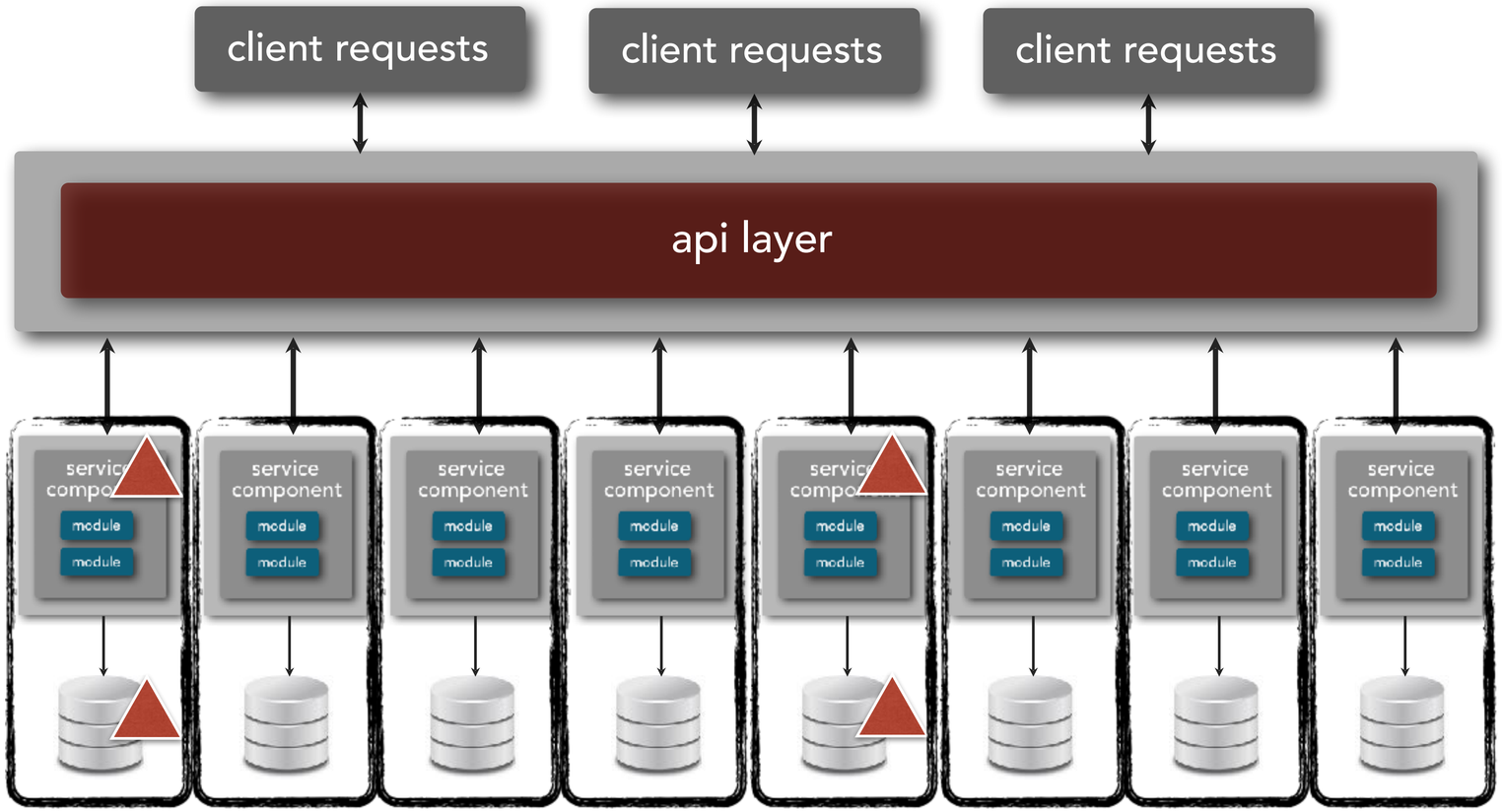
bounded context (share nothing architecture)

microservices core concepts



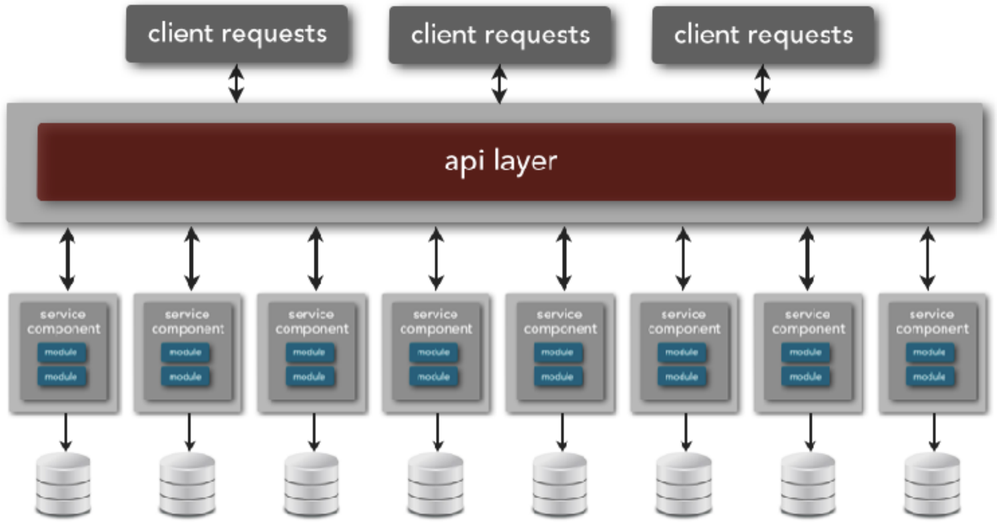
bounded context (share nothing architecture)

microservices core concepts

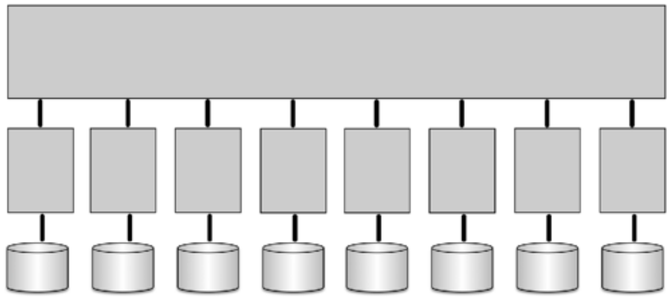


bounded context (share nothing architecture)

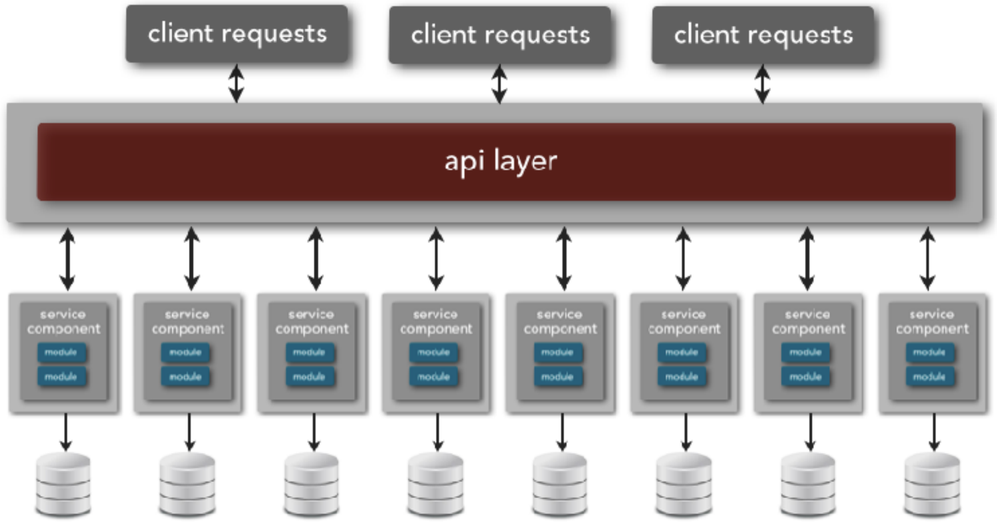
microservices core concepts



single purpose functions
deployed as separate units

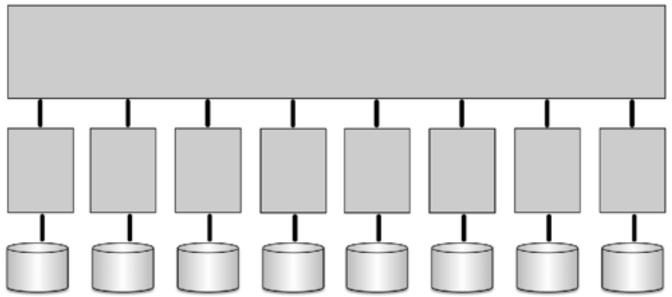


microservices core concepts

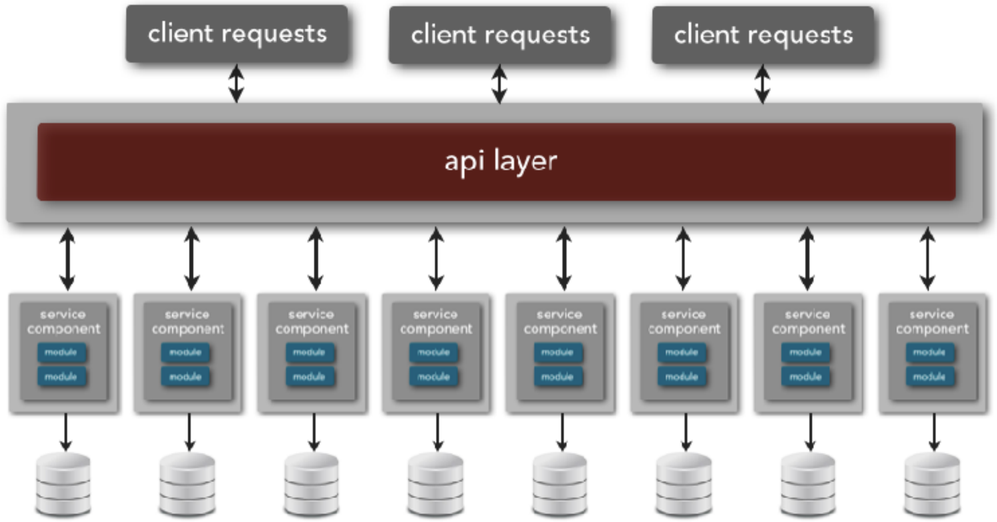


maintainability ★★★★★
deployability ★★★★★
testability ★★★★★

single purpose functions
deployed as separate units

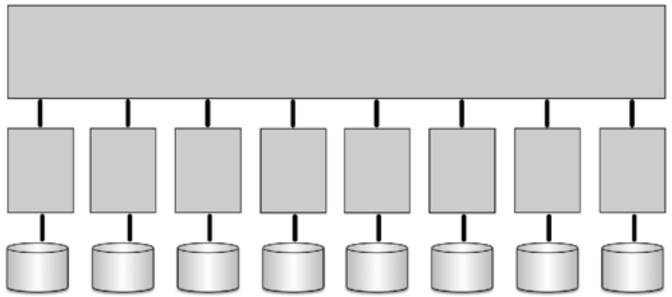


microservices core concepts

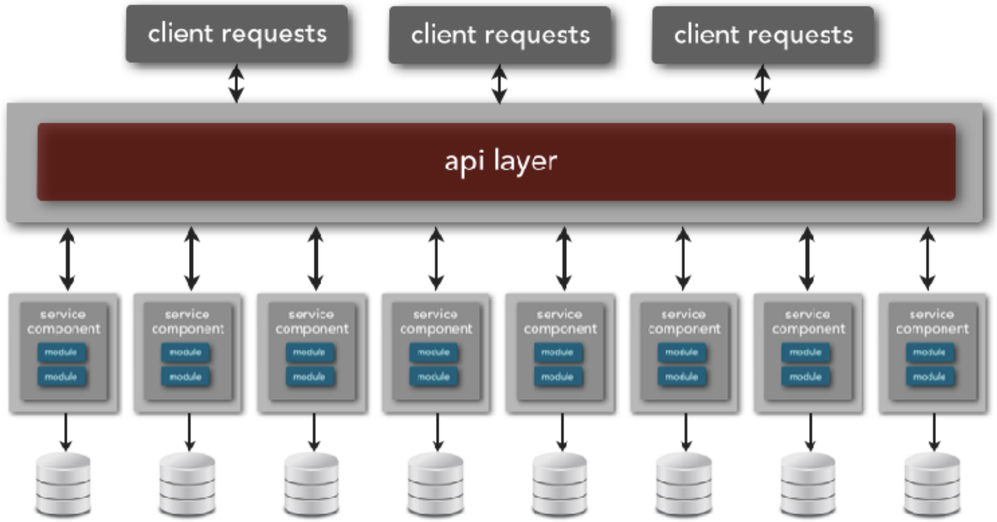


maintainability ★★★★★
deployability ★★★★★
testability ★★★★★
fault-tolerance ★★★★★

single purpose functions
deployed as separate units

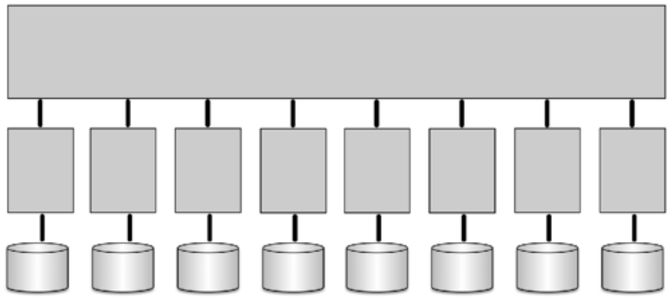


microservices core concepts

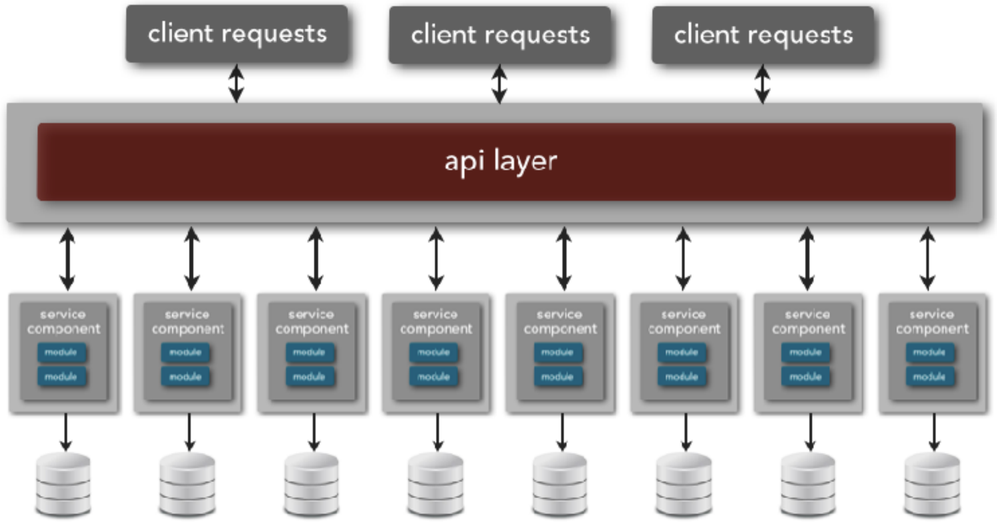


maintainability ★★★★★
deployability ★★★★★
testability ★★★★★
fault-tolerance ★★★★★
scalability ★★★★★

single purpose functions
deployed as separate units

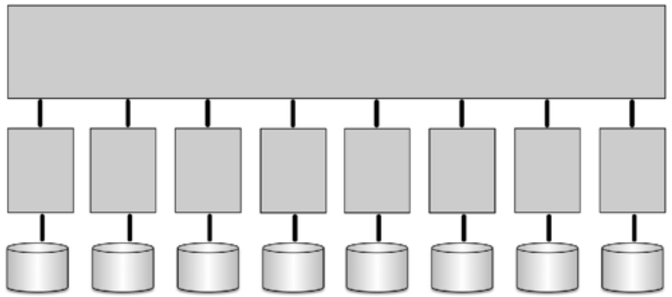


microservices core concepts

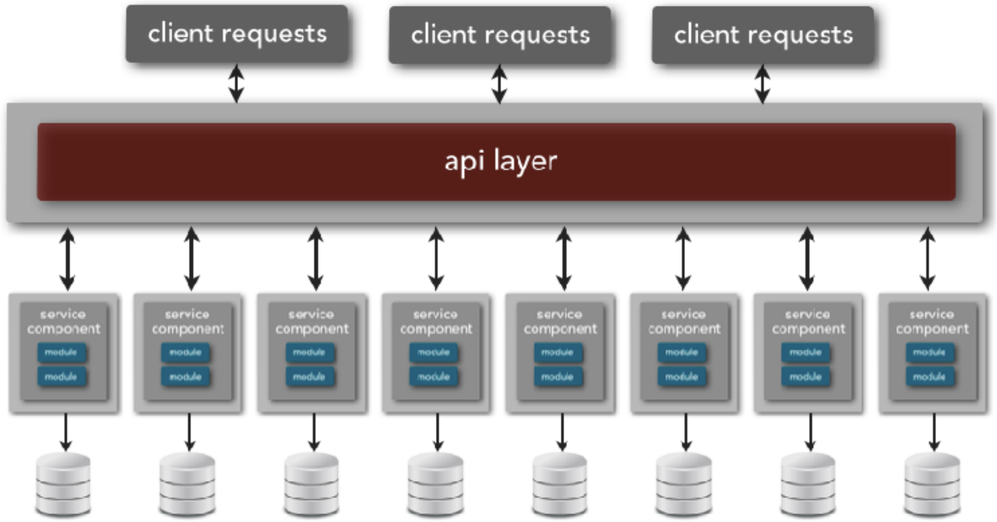


maintainability ★★★★★
deployability ★★★★★
testability ★★★★★
fault-tolerance ★★★★★
scalability ★★★★★
evolvability ★★★★★

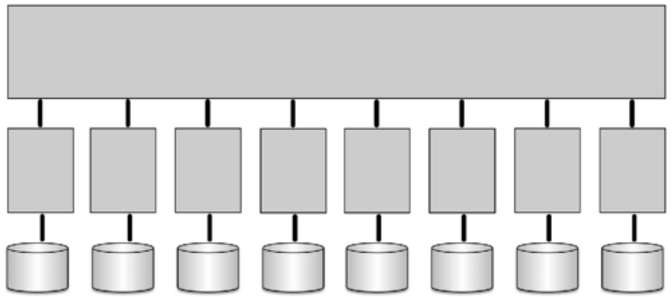
single purpose functions
deployed as separate units



microservices core concepts



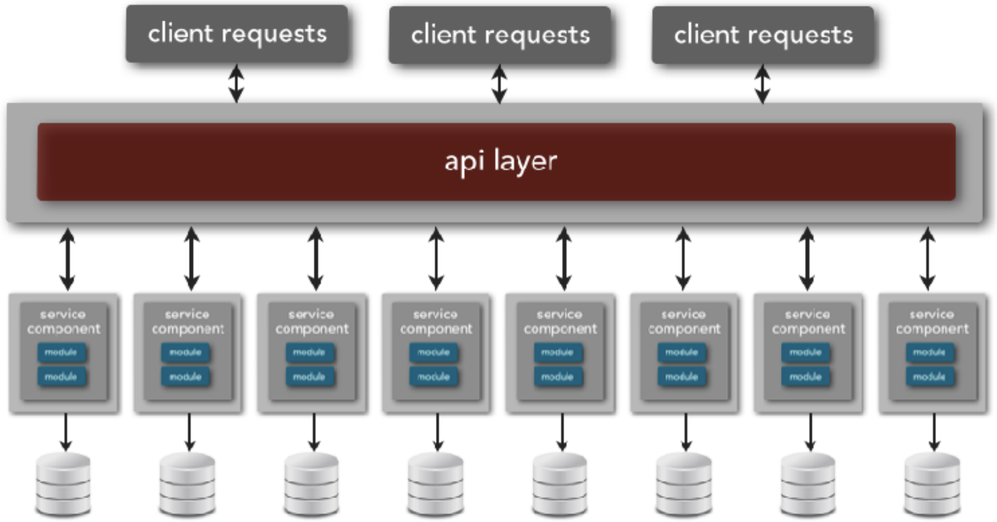
single purpose functions
deployed as separate units



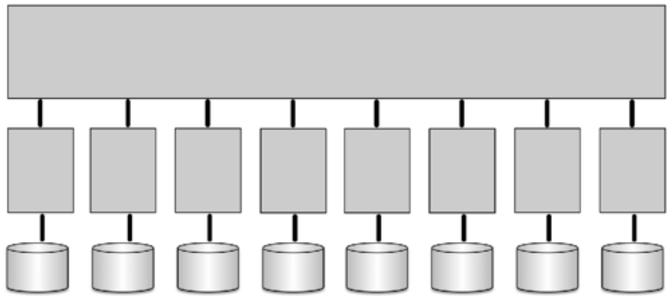
maintainability ★★★★★
deployability ★★★★★
testability ★★★★★
fault-tolerance ★★★★★
scalability ★★★★★
evolvability ★★★★★

performance ★★

microservices core concepts



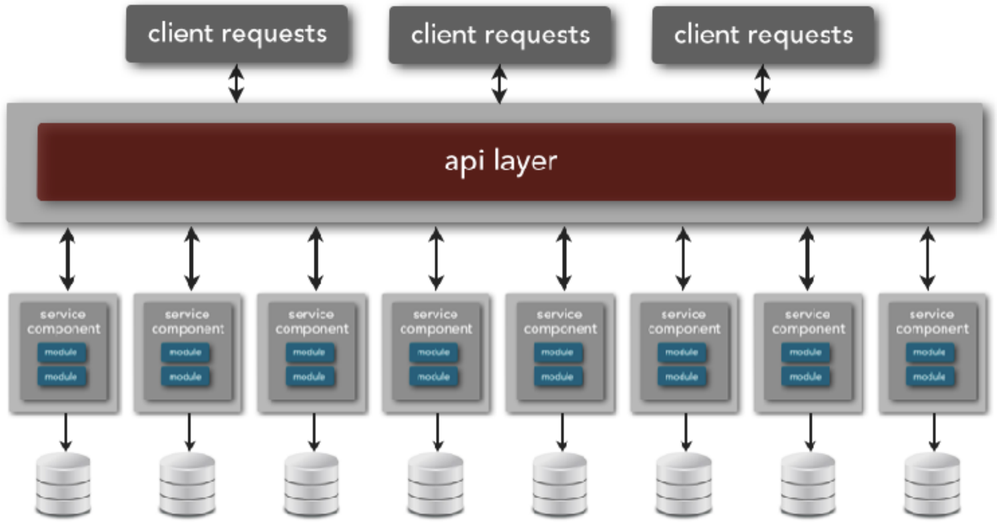
single purpose functions
deployed as separate units



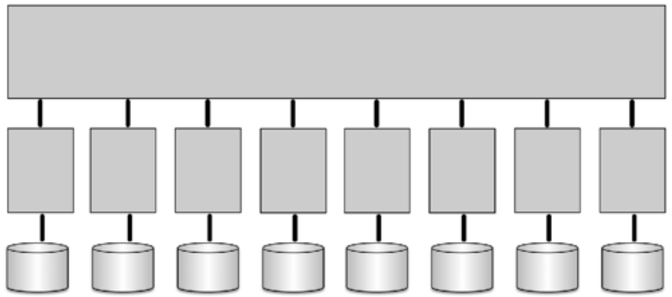
maintainability ★★★★★
deployability ★★★★★
testability ★★★★★
fault-tolerance ★★★★★
scalability ★★★★★
evolvability ★★★★★

performance ★★
simplicity ★

microservices core concepts

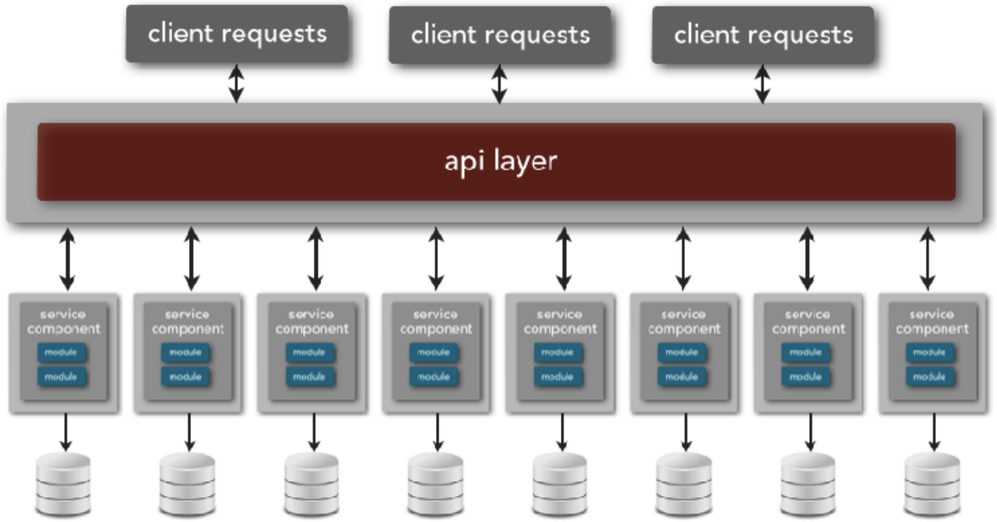


single purpose functions
deployed as separate units

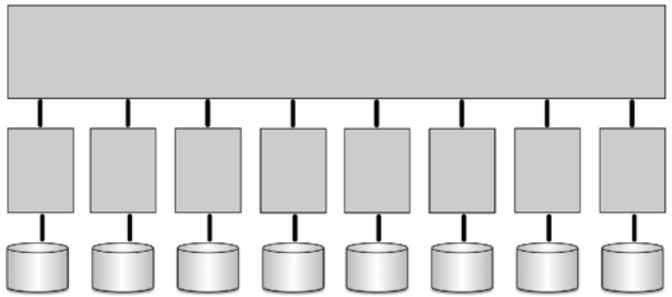


maintainability	★★★★★
deployability	★★★★★
testability	★★★★★
fault-tolerance	★★★★★
scalability	★★★★★
evolvability	★★★★★
performance	★★
simplicity	★
cost	★

microservices core concepts



single purpose functions
deployed as separate units

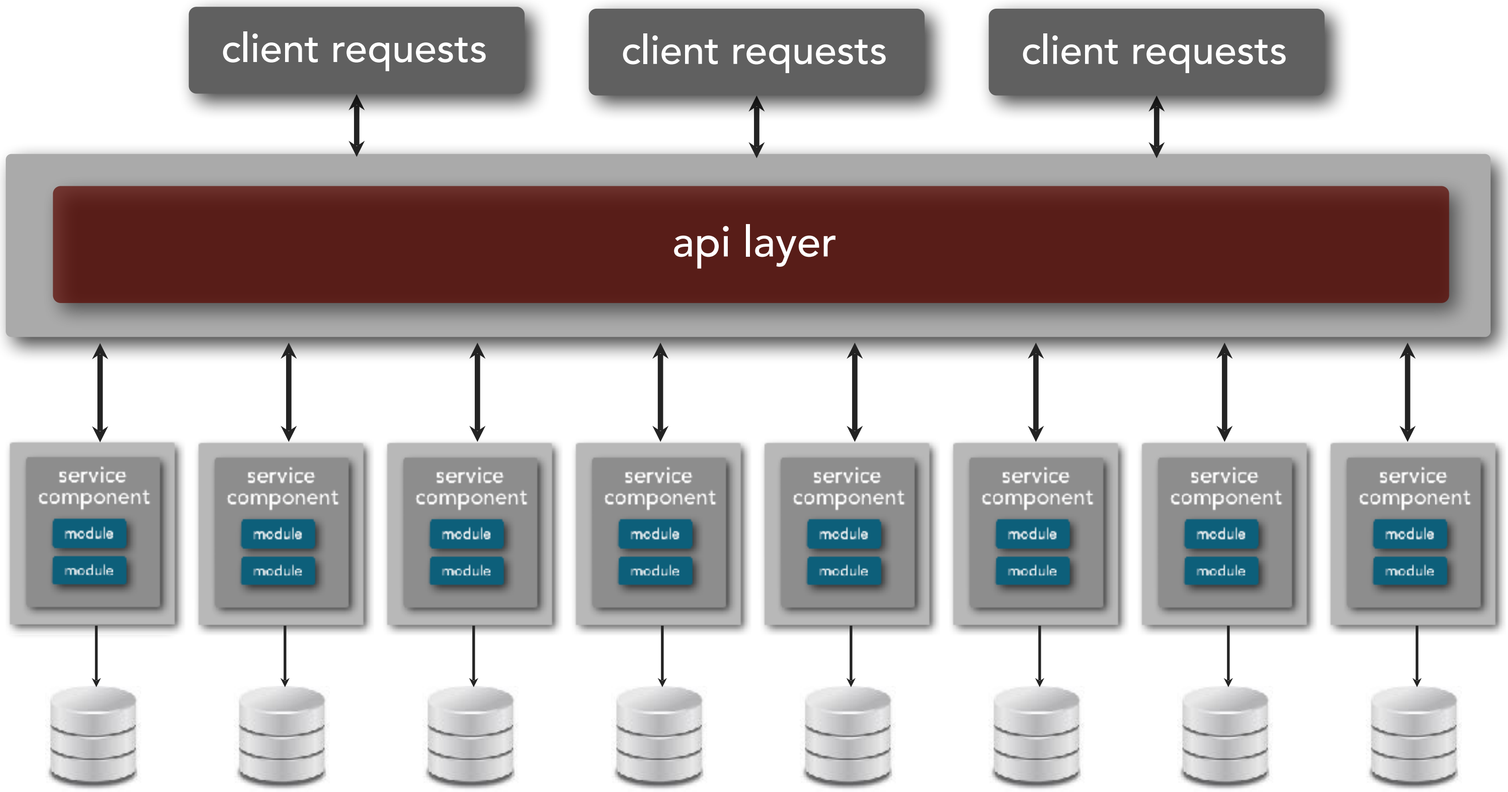


maintainability ★★★★★
deployability ★★★★★
testability ★★★★★
fault-tolerance ★★★★★
scalability ★★★★★
evolvability ★★★★★

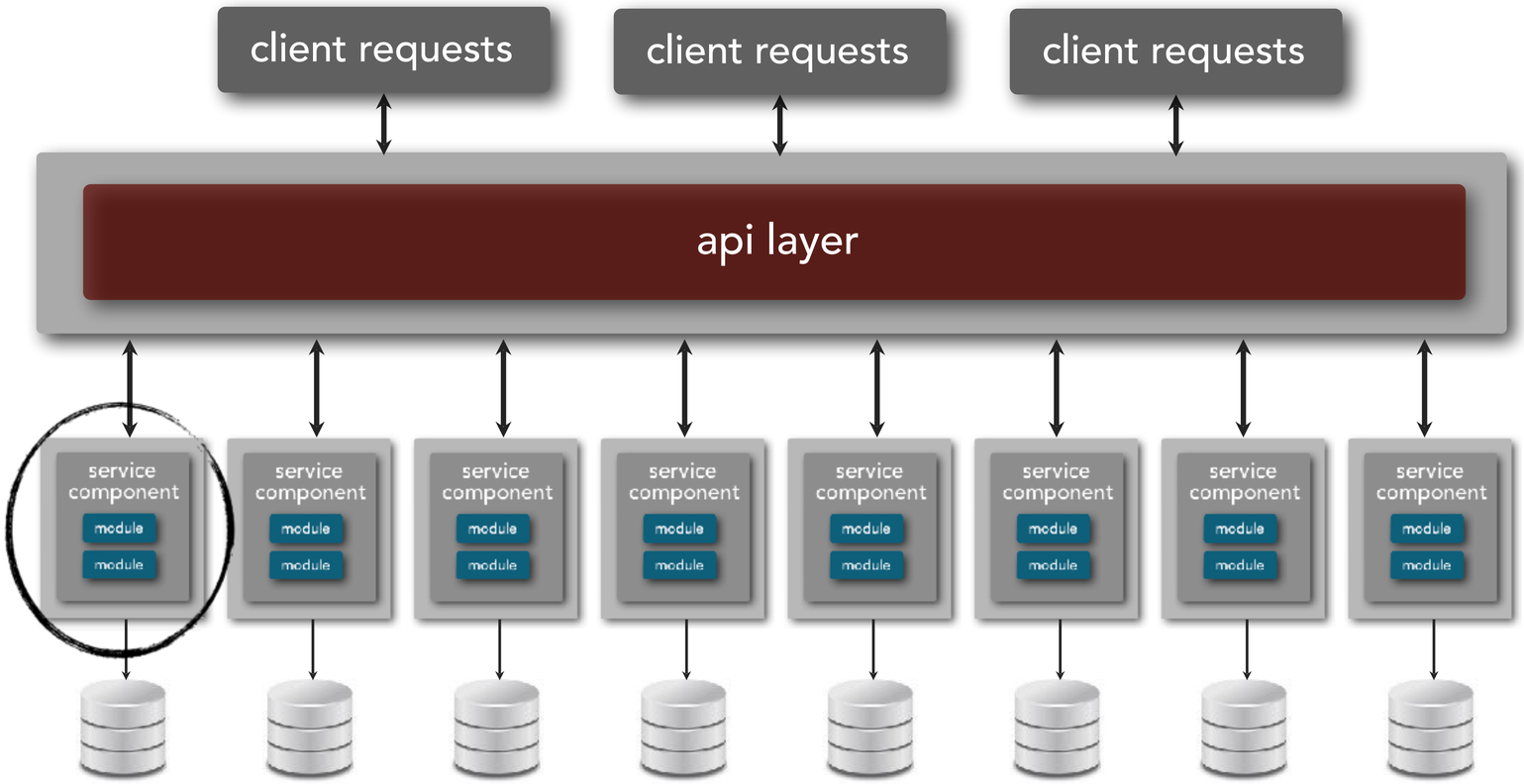
performance ★★
simplicity ★
cost ★
workflow ★

Service Design

service design



service design

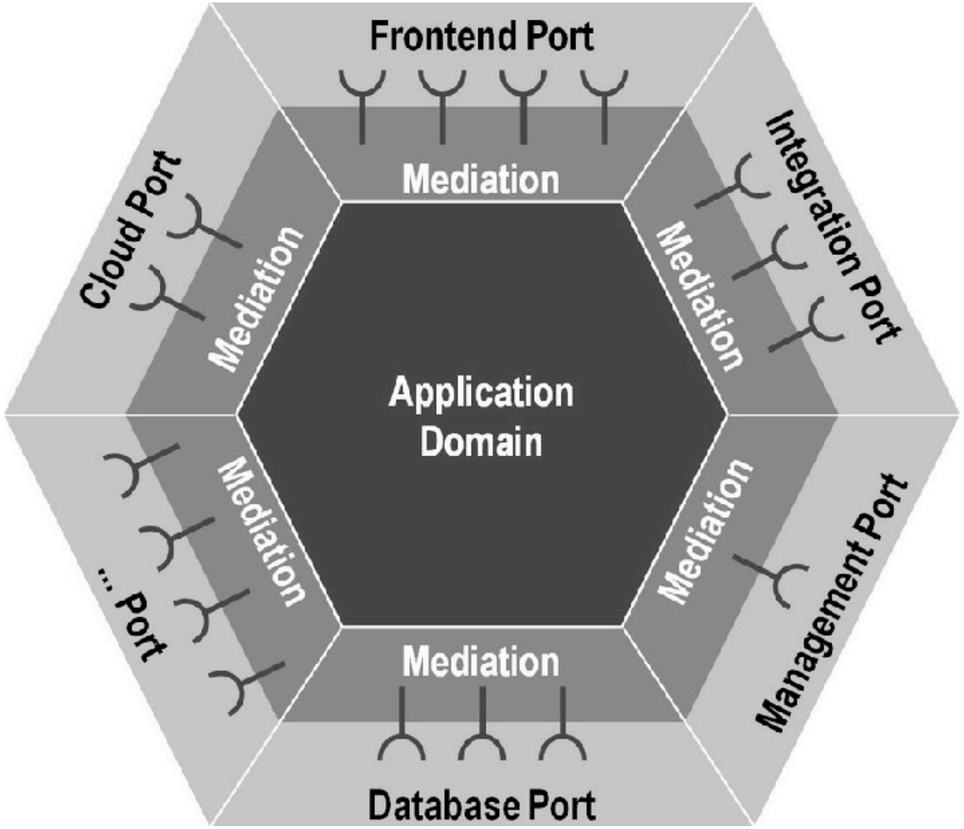


service design

micro-hexagonal design **to support** protocol agnostic processing

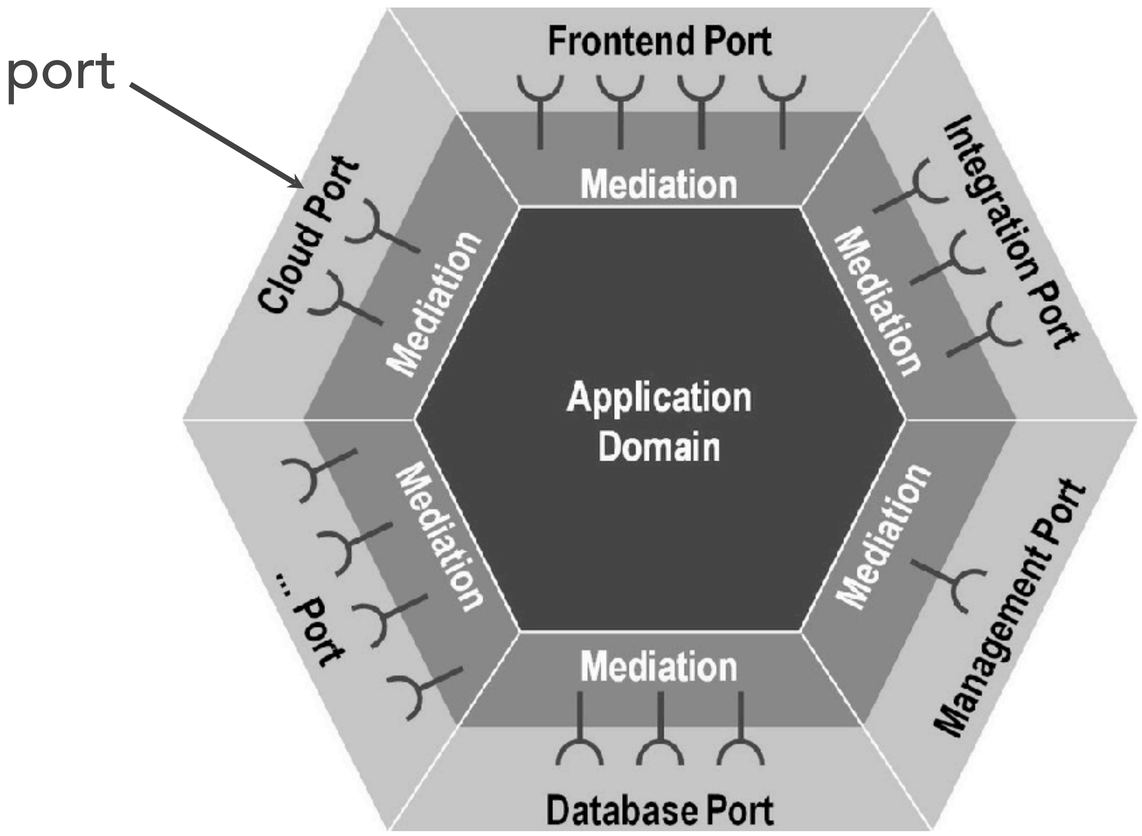
service design

micro-hexagonal design **to support** protocol agnostic processing



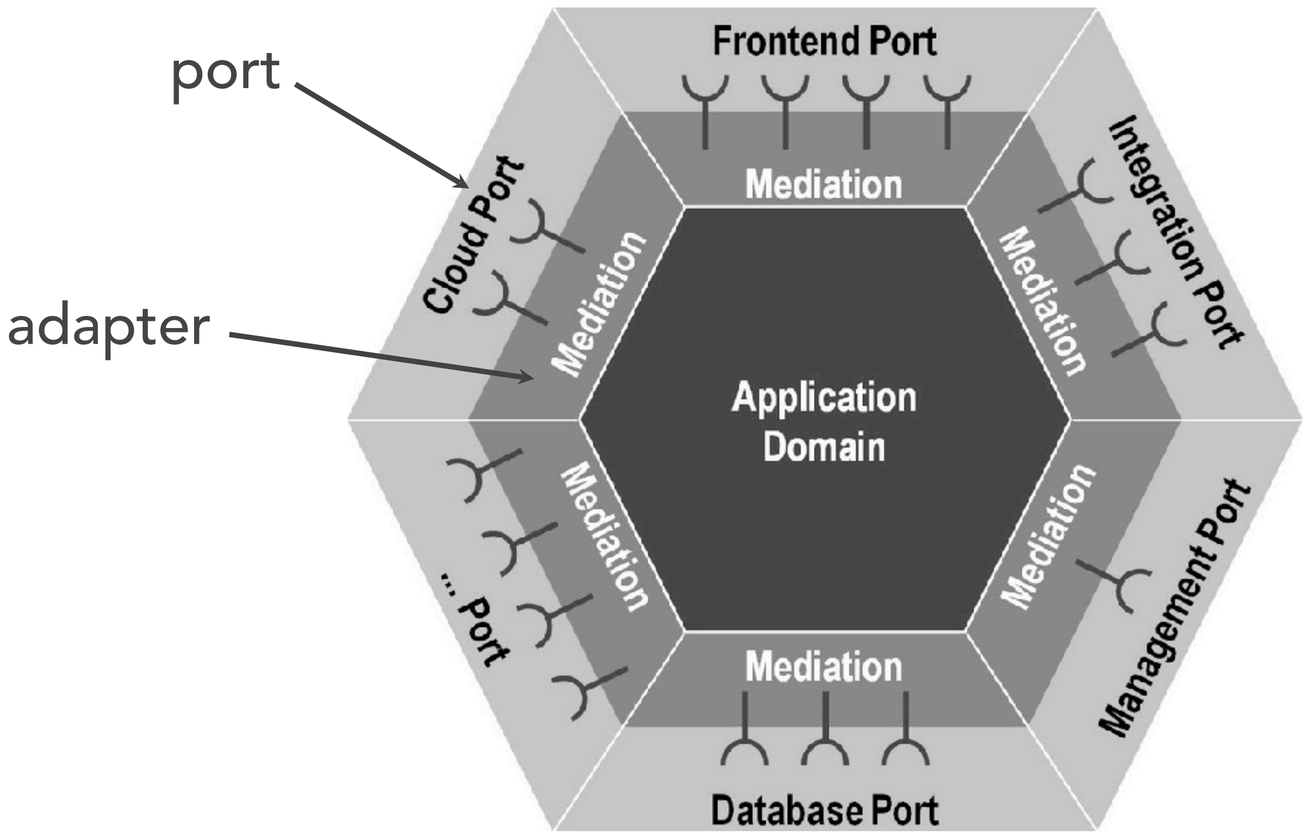
service design

micro-hexagonal design **to support** protocol agnostic processing



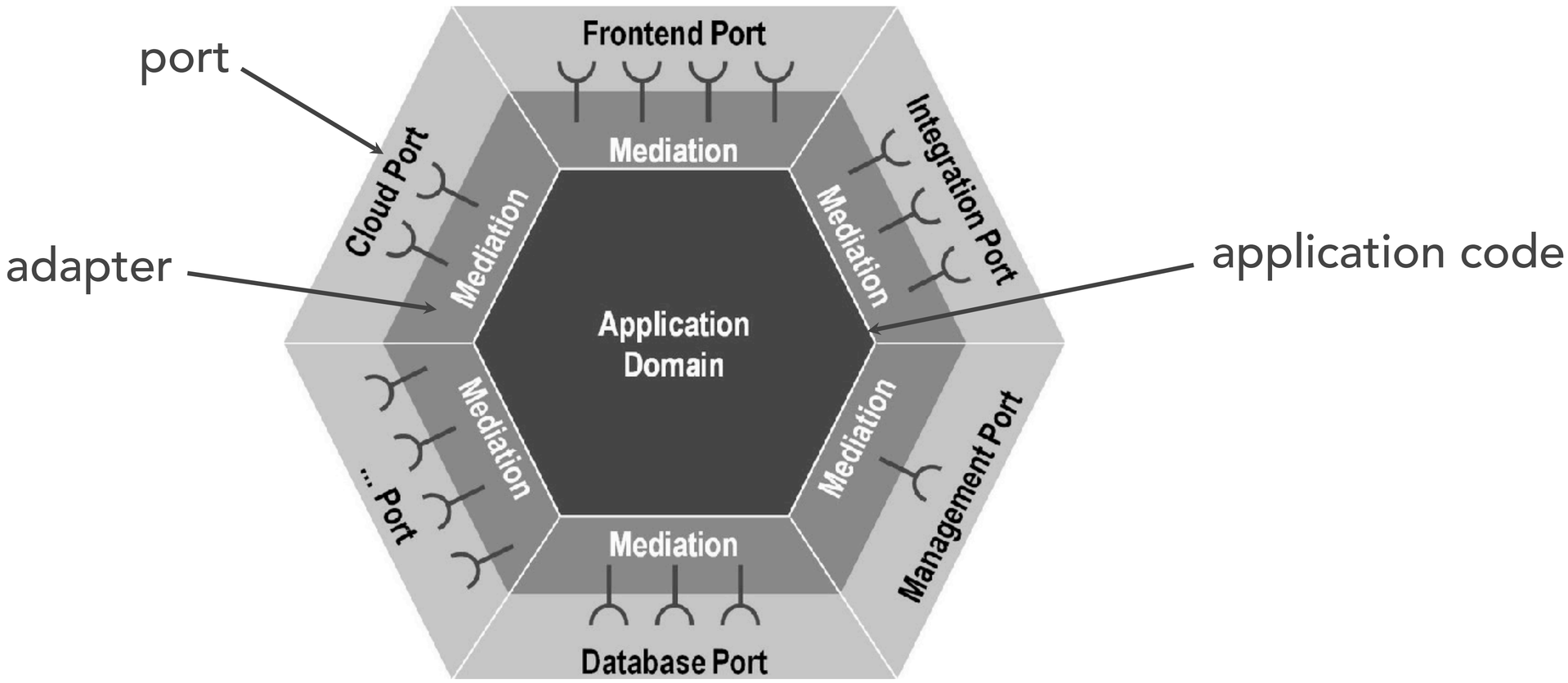
service design

micro-hexagonal design **to support** protocol agnostic processing



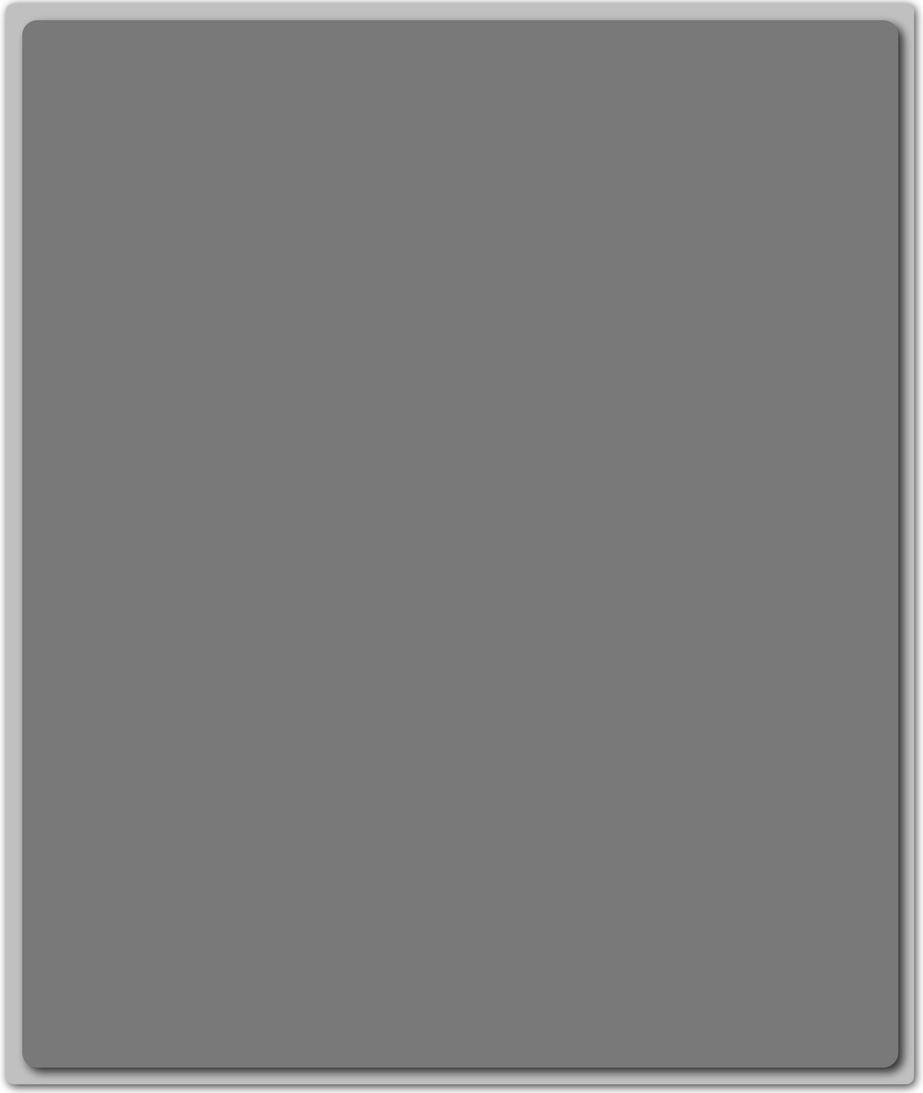
service design

micro-hexagonal design **to support** protocol agnostic processing



service design

micro-hexagonal design **to support** protocol agnostic processing



service design

micro-hexagonal design **to support** protocol agnostic processing

service design

micro-hexagonal design **to support** protocol agnostic processing

service design

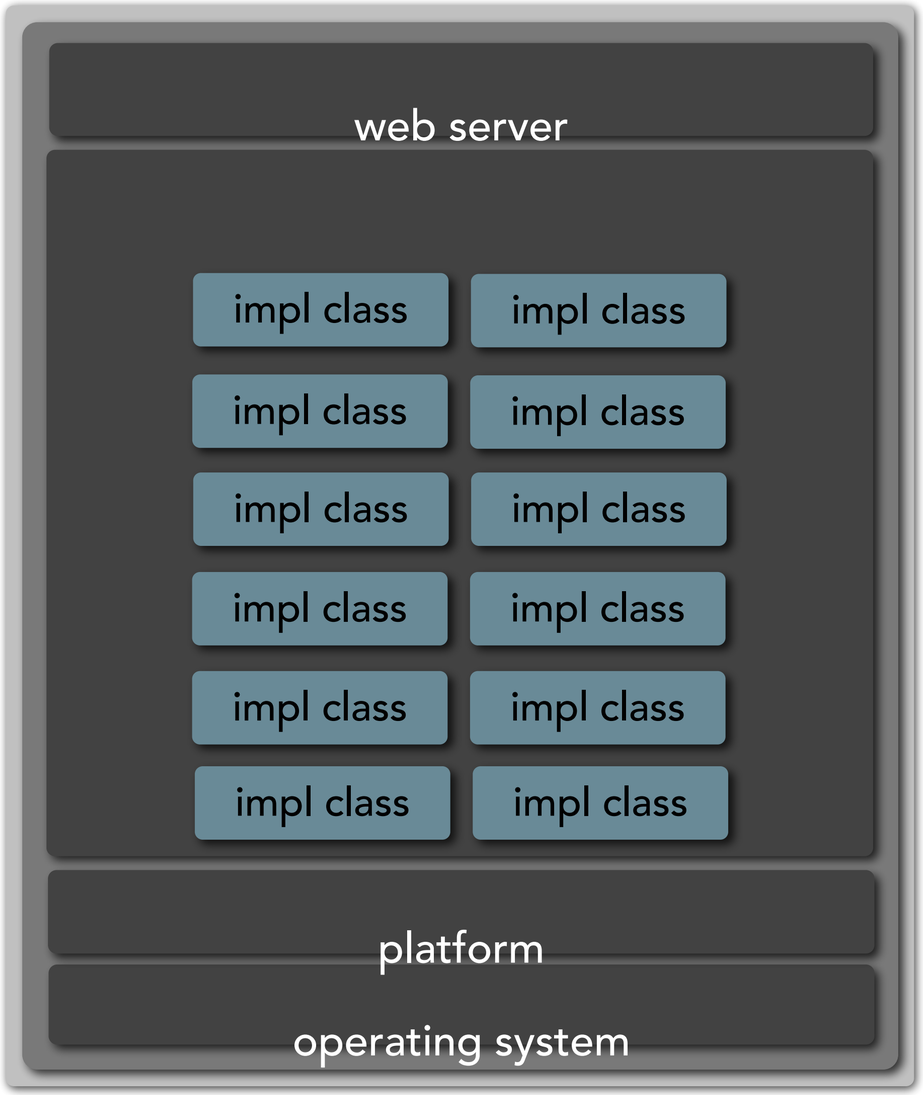
micro-hexagonal design **to support** protocol agnostic processing

service design

micro-hexagonal design **to support** protocol agnostic processing

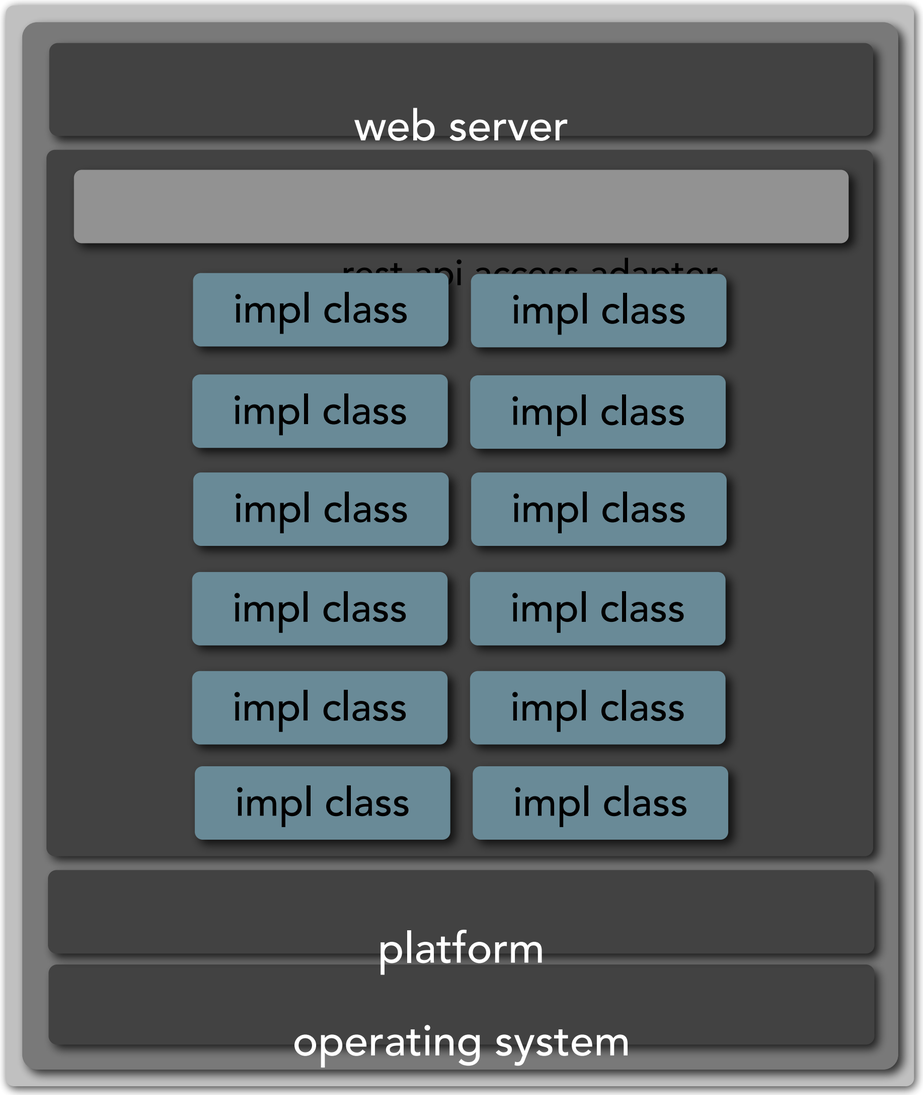
service design

micro-hexagonal design **to support** protocol agnostic processing



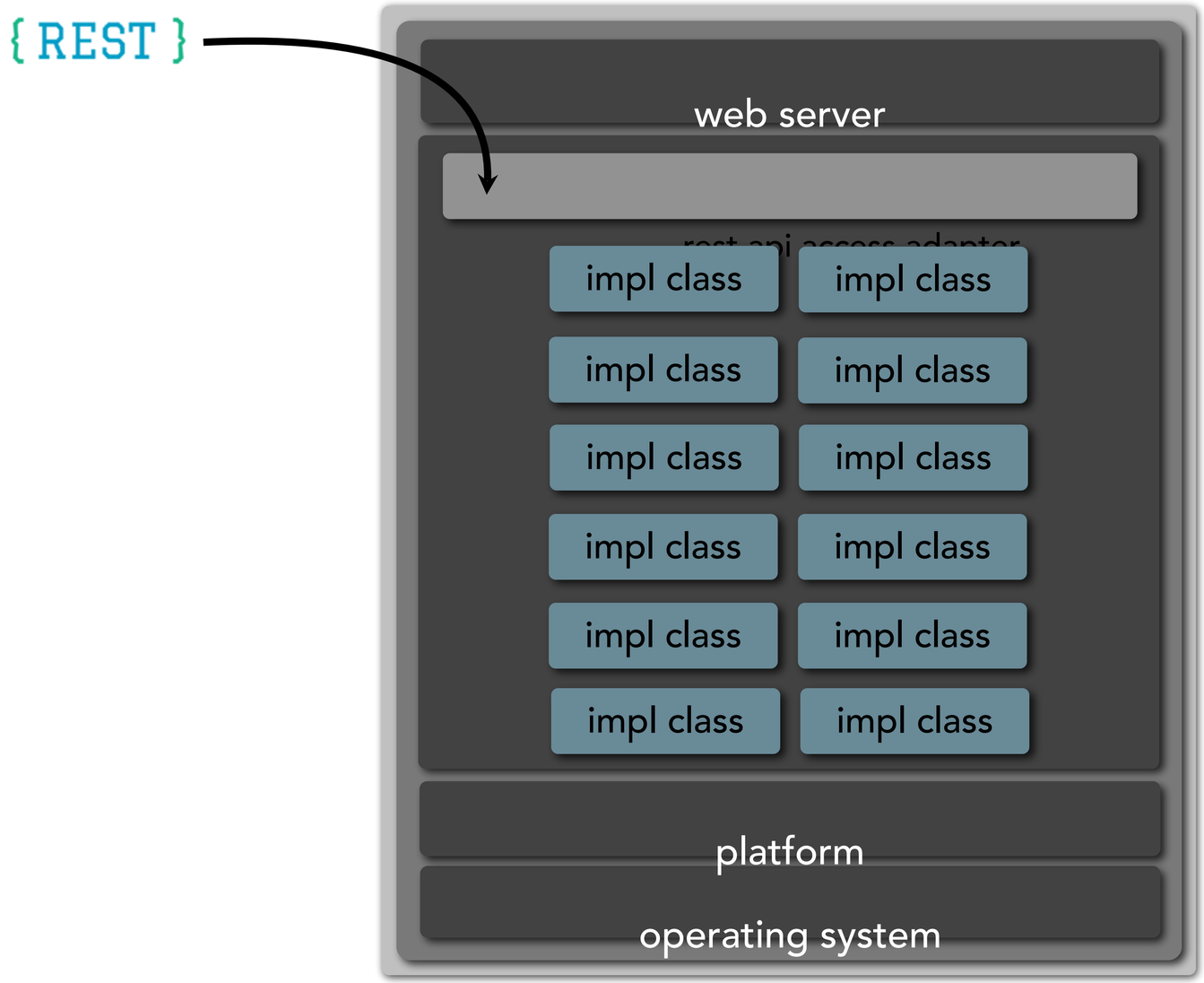
service design

micro-hexagonal design **to support** protocol agnostic processing



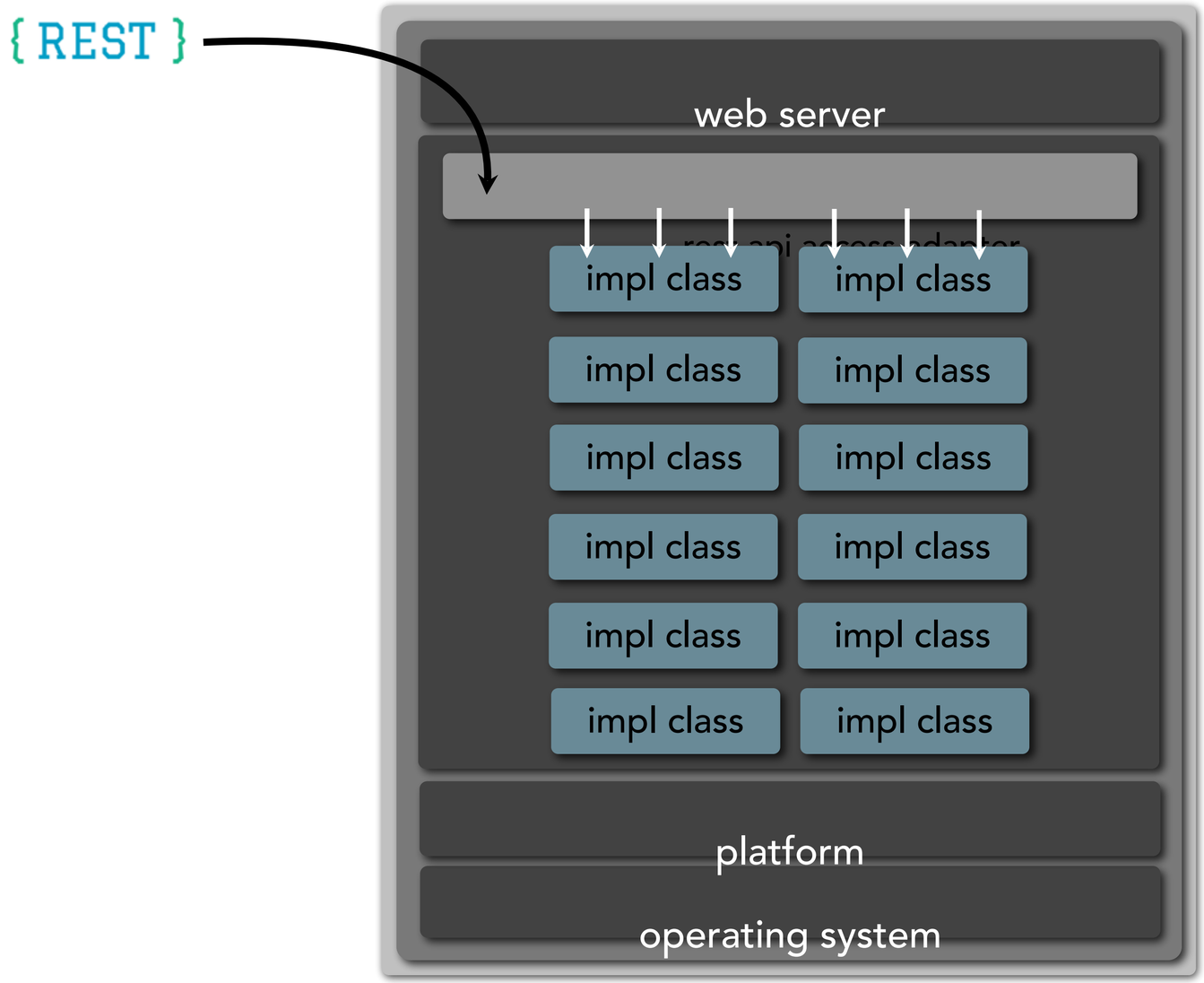
service design

micro-hexagonal design **to support** protocol agnostic processing



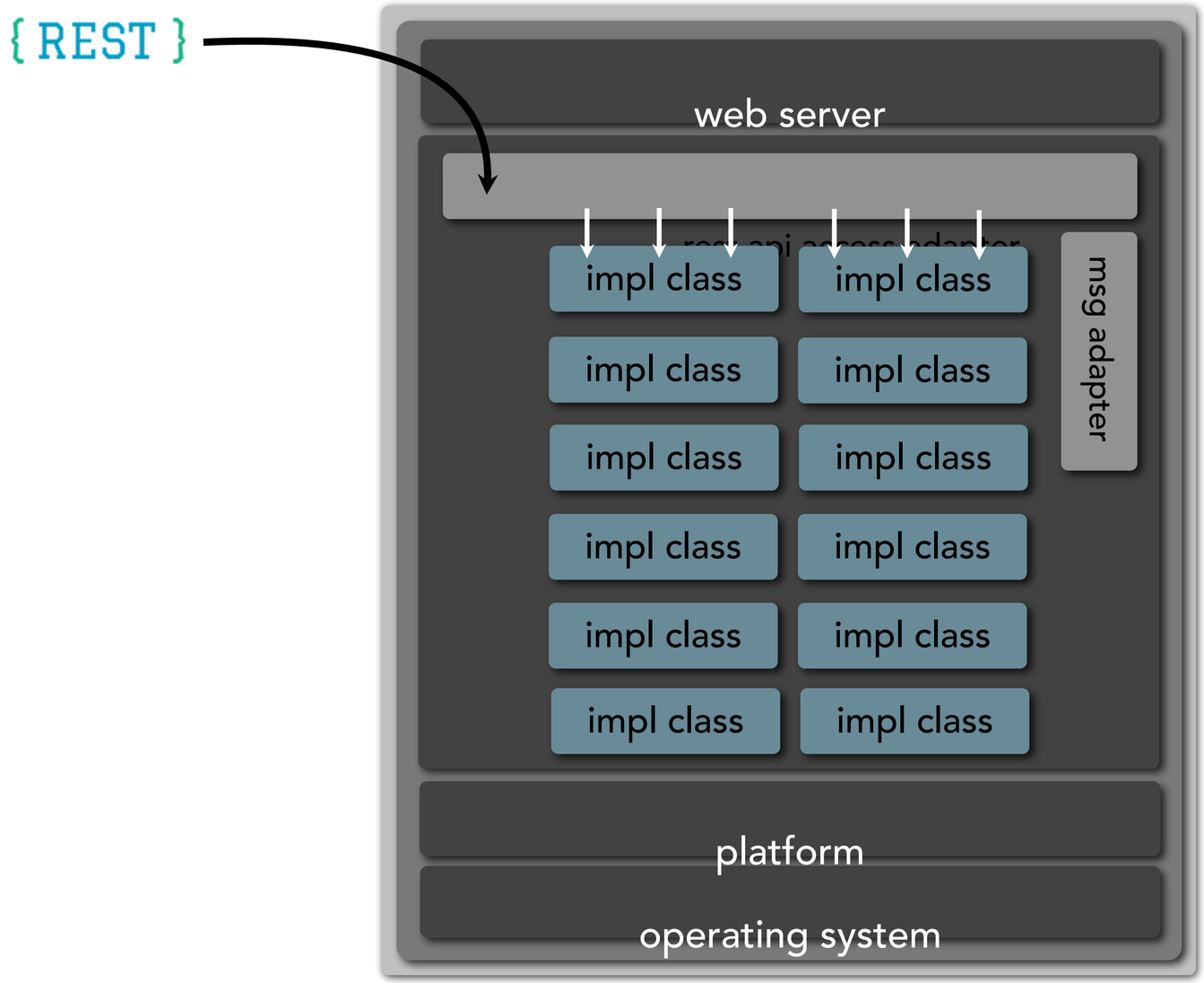
service design

micro-hexagonal design **to support** protocol agnostic processing



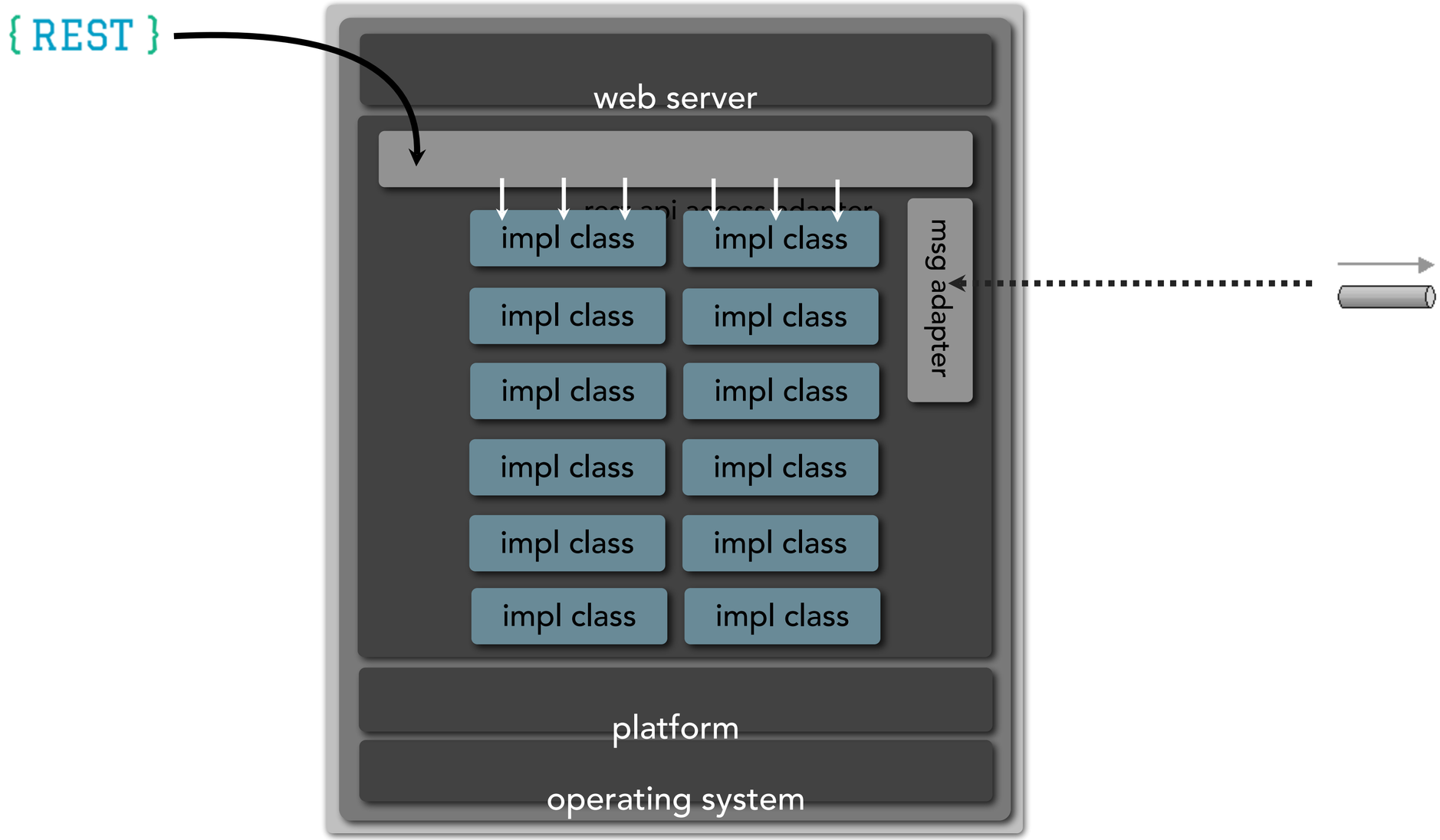
service design

micro-hexagonal design **to support** protocol agnostic processing



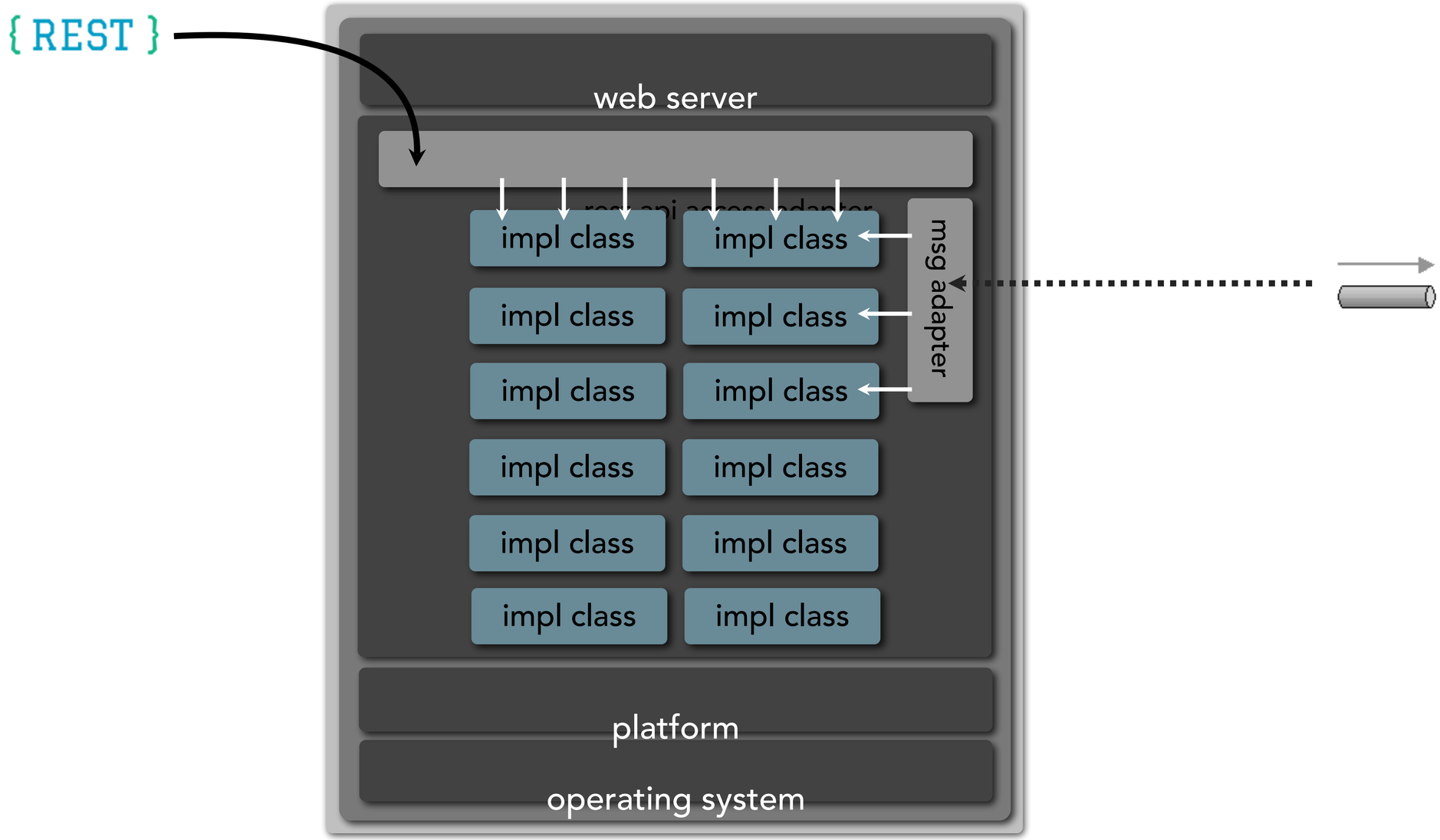
service design

micro-hexagonal design **to support** protocol agnostic processing



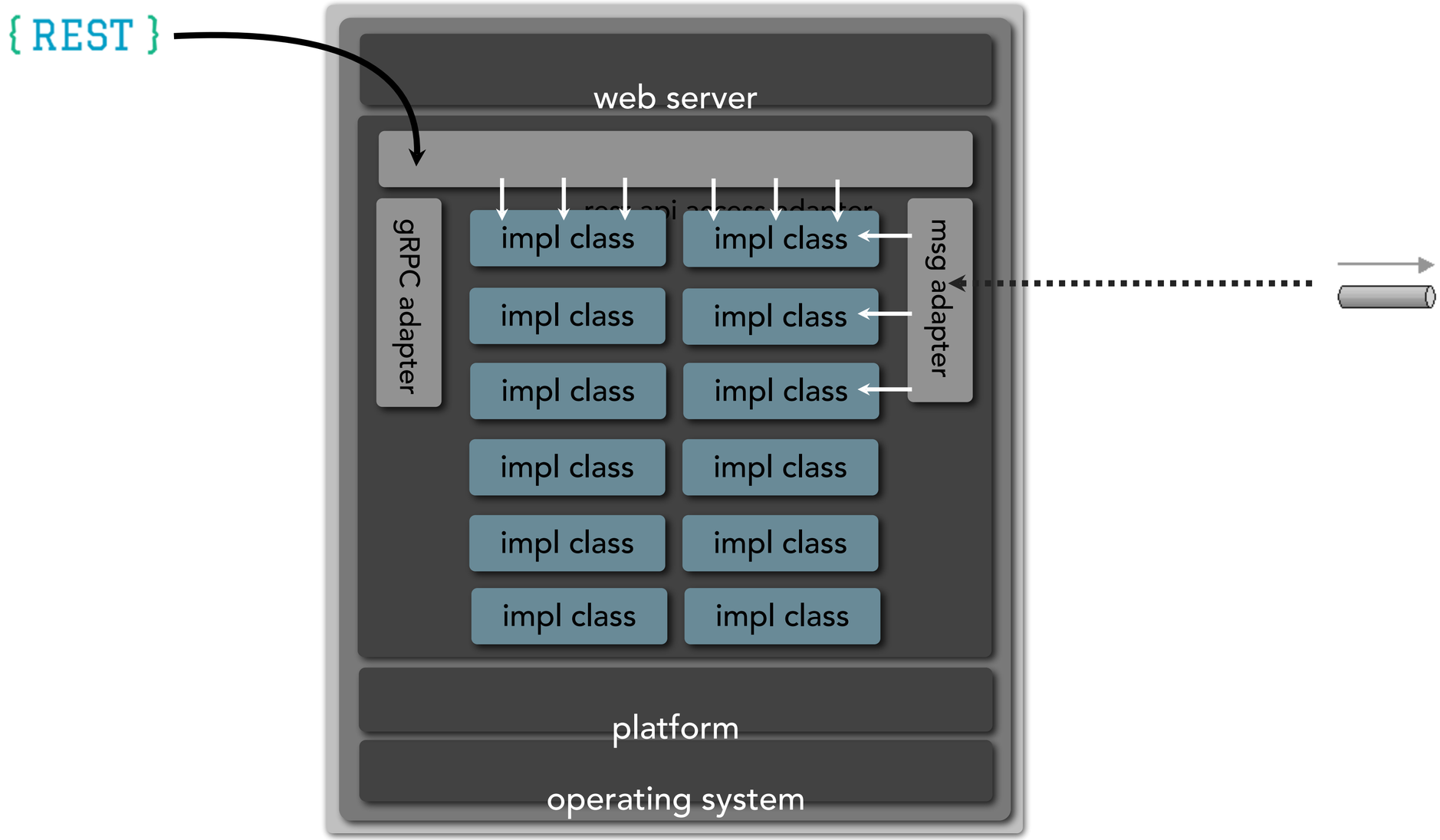
service design

micro-hexagonal design **to support** protocol agnostic processing



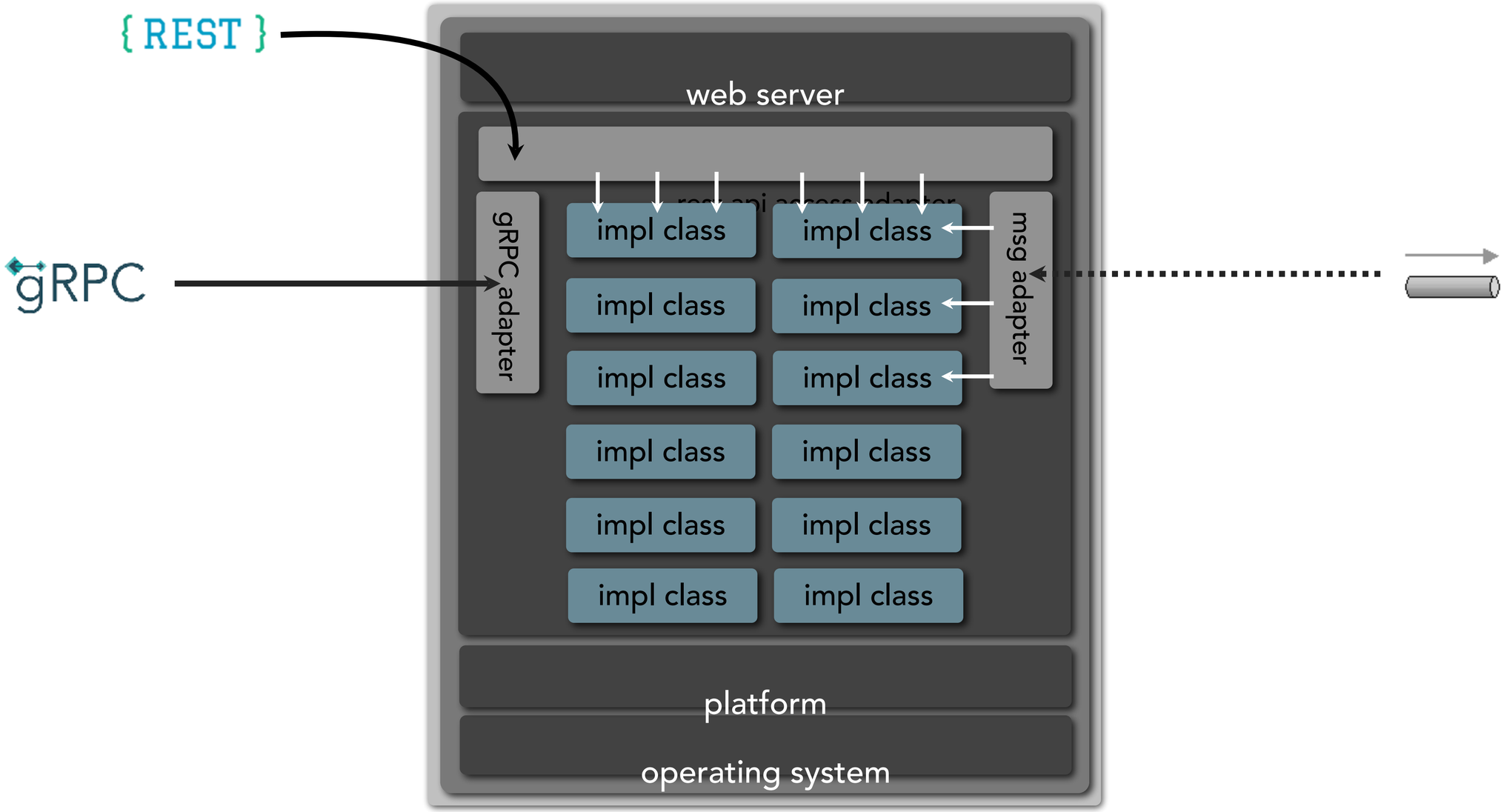
service design

micro-hexagonal design **to support** protocol agnostic processing



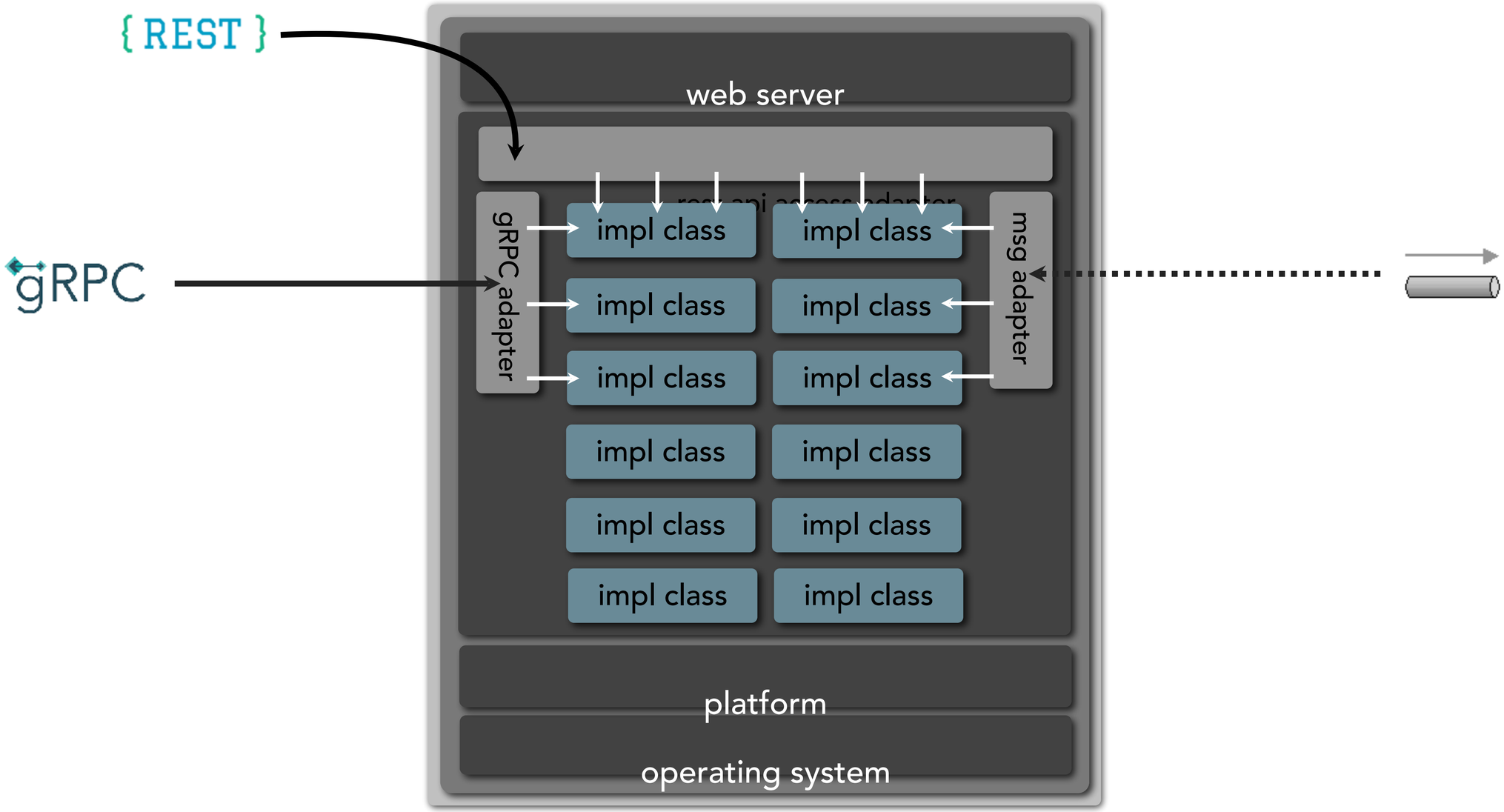
service design

micro-hexagonal design **to support** protocol agnostic processing



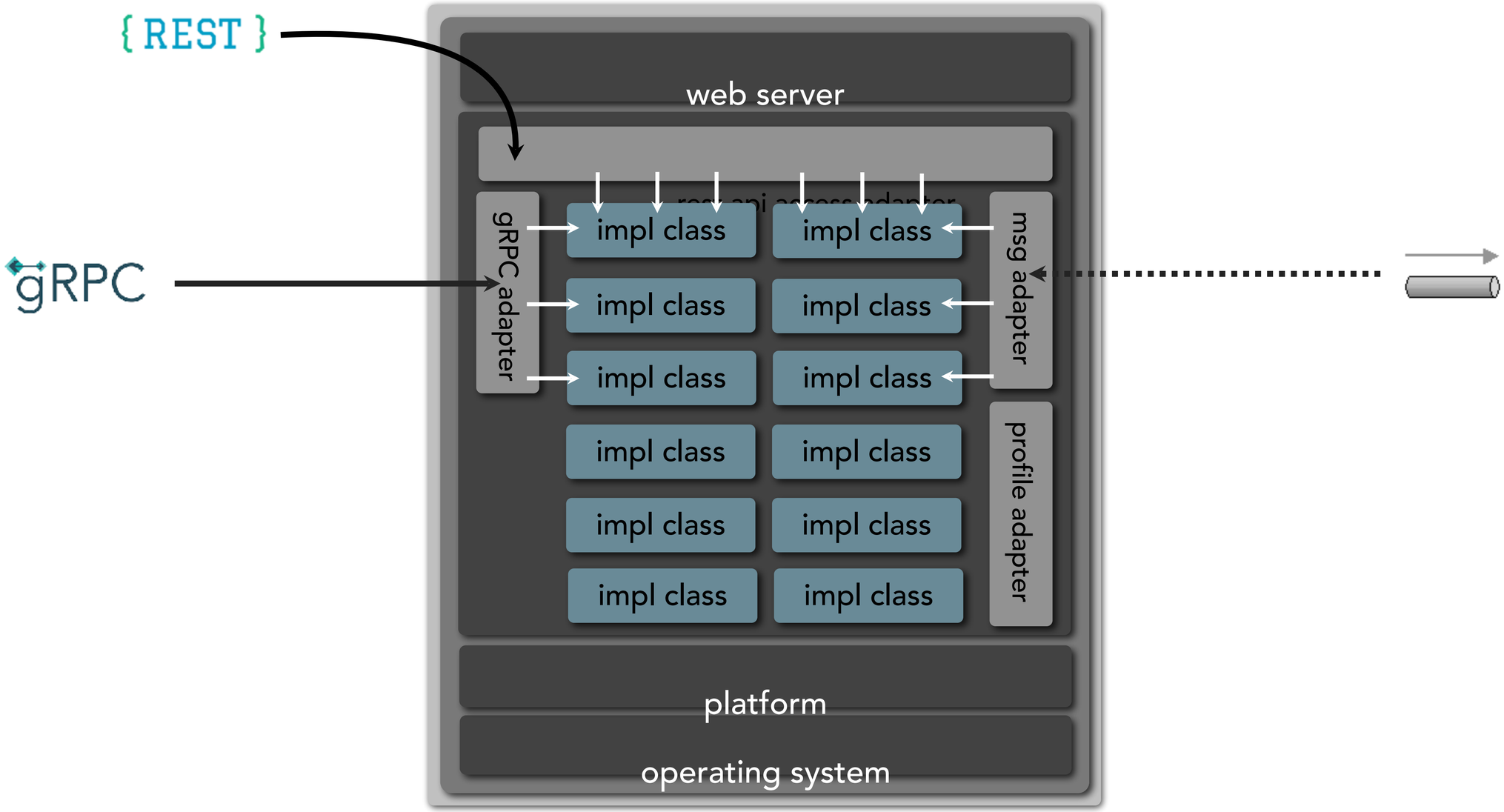
service design

micro-hexagonal design **to support** protocol agnostic processing



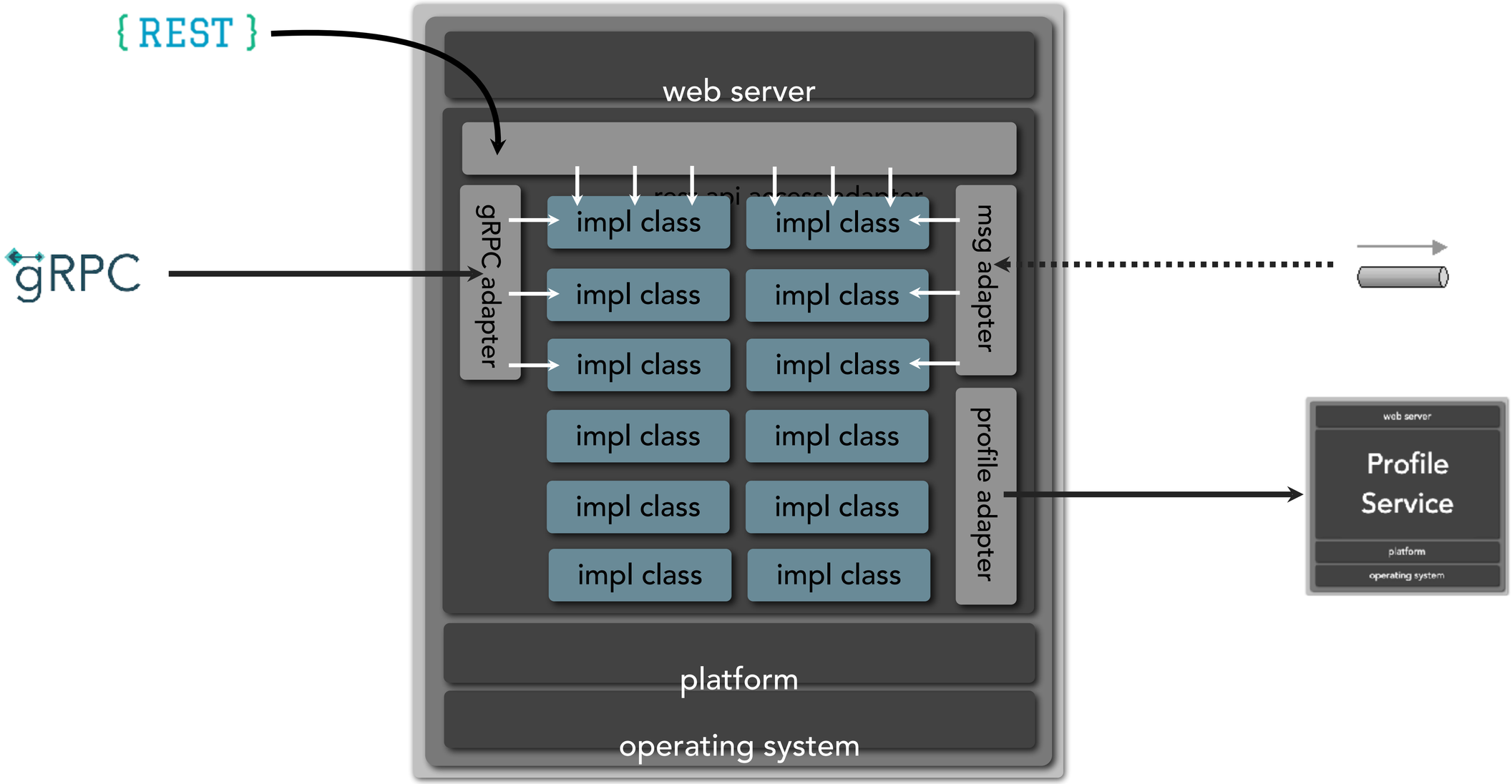
service design

micro-hexagonal design **to support** protocol agnostic processing



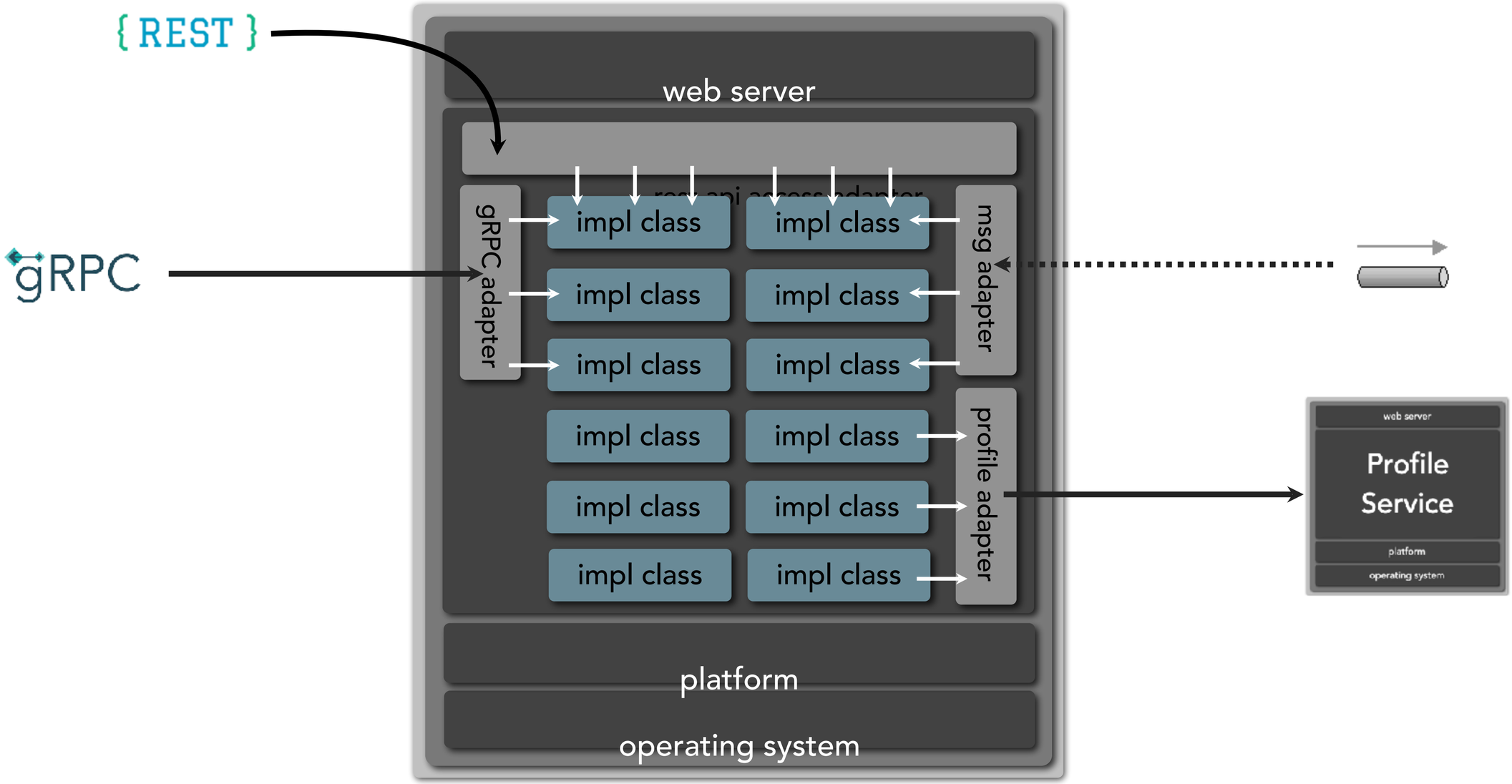
service design

micro-hexagonal design **to support** protocol agnostic processing



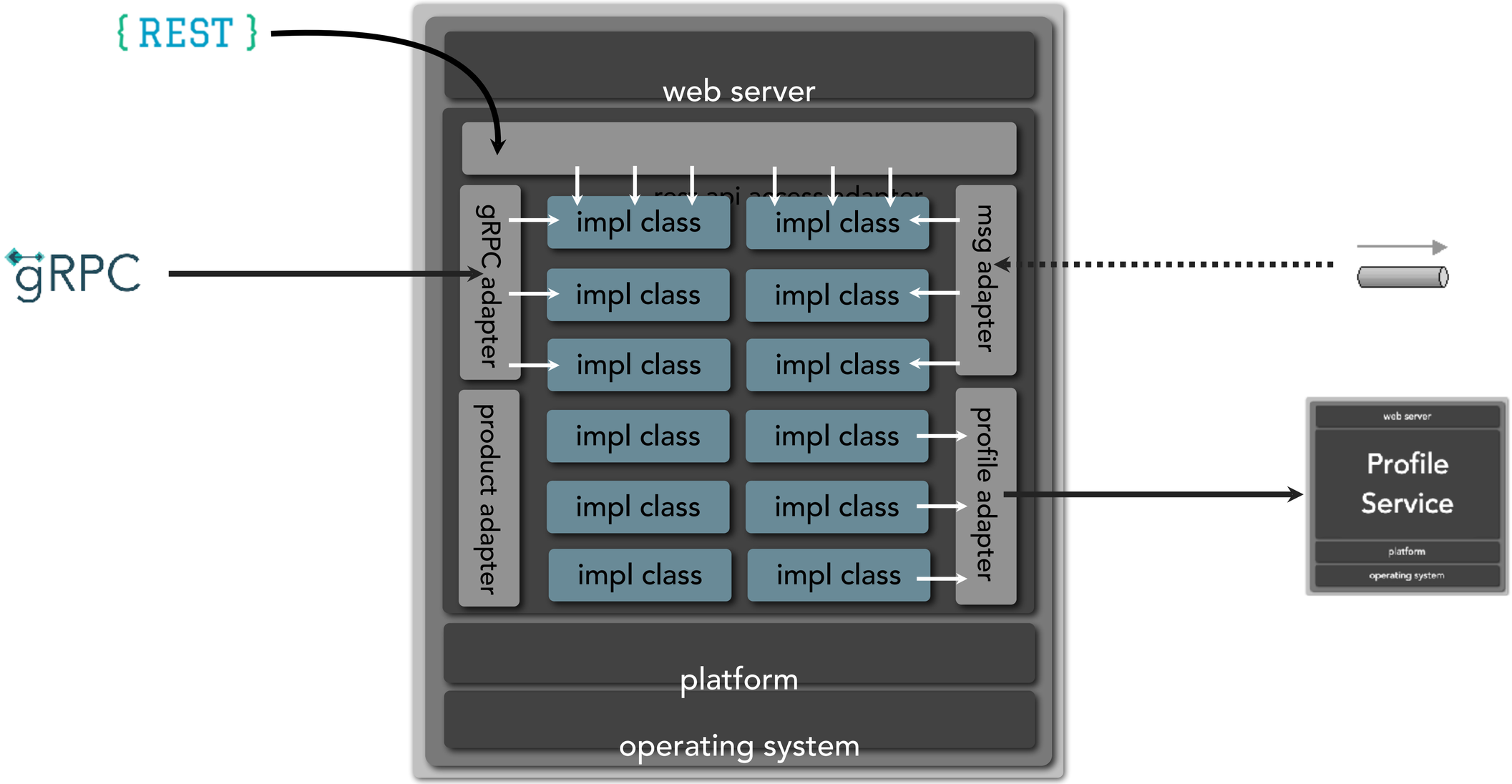
service design

micro-hexagonal design **to support** protocol agnostic processing



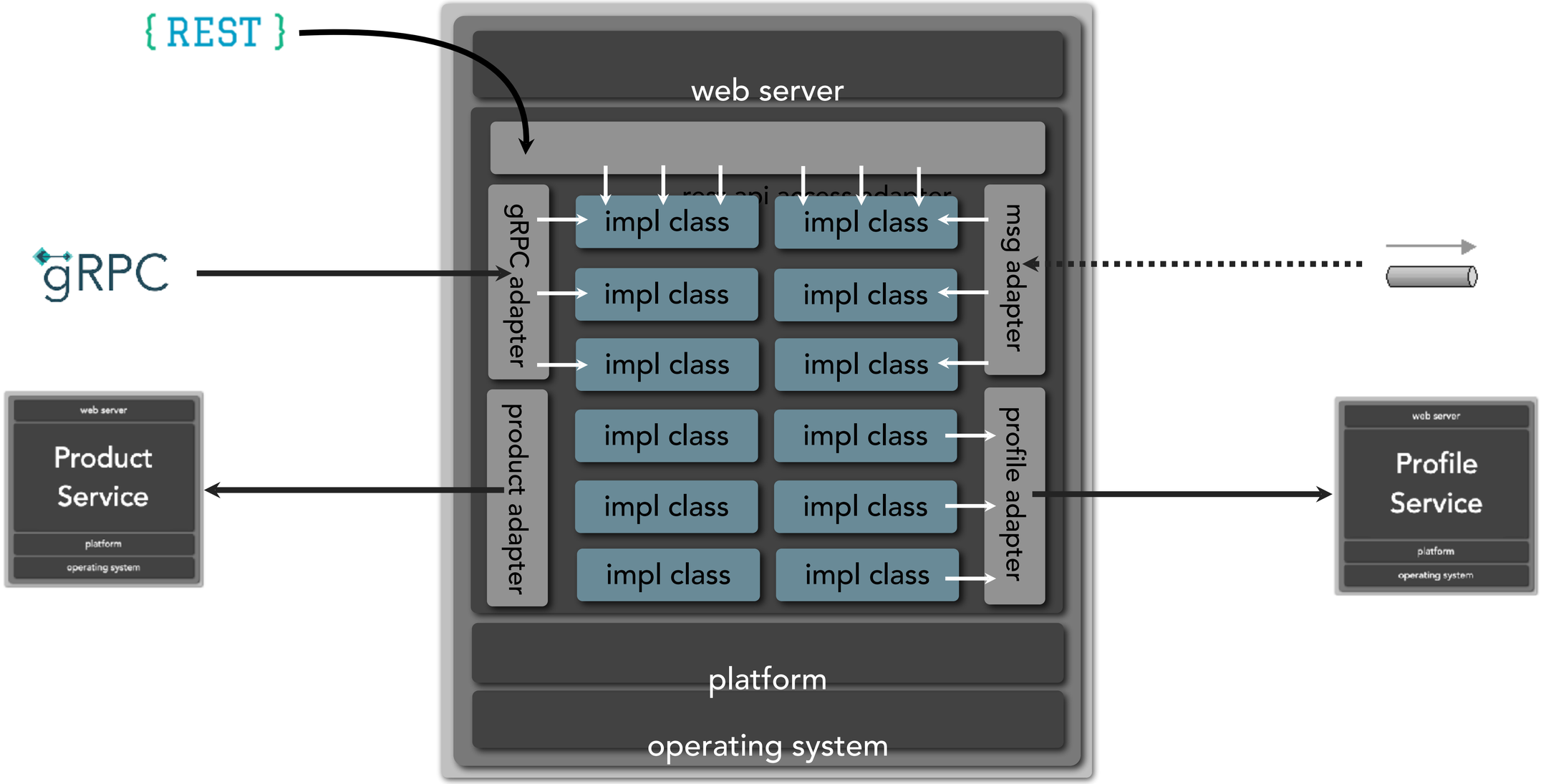
service design

micro-hexagonal design **to support** protocol agnostic processing



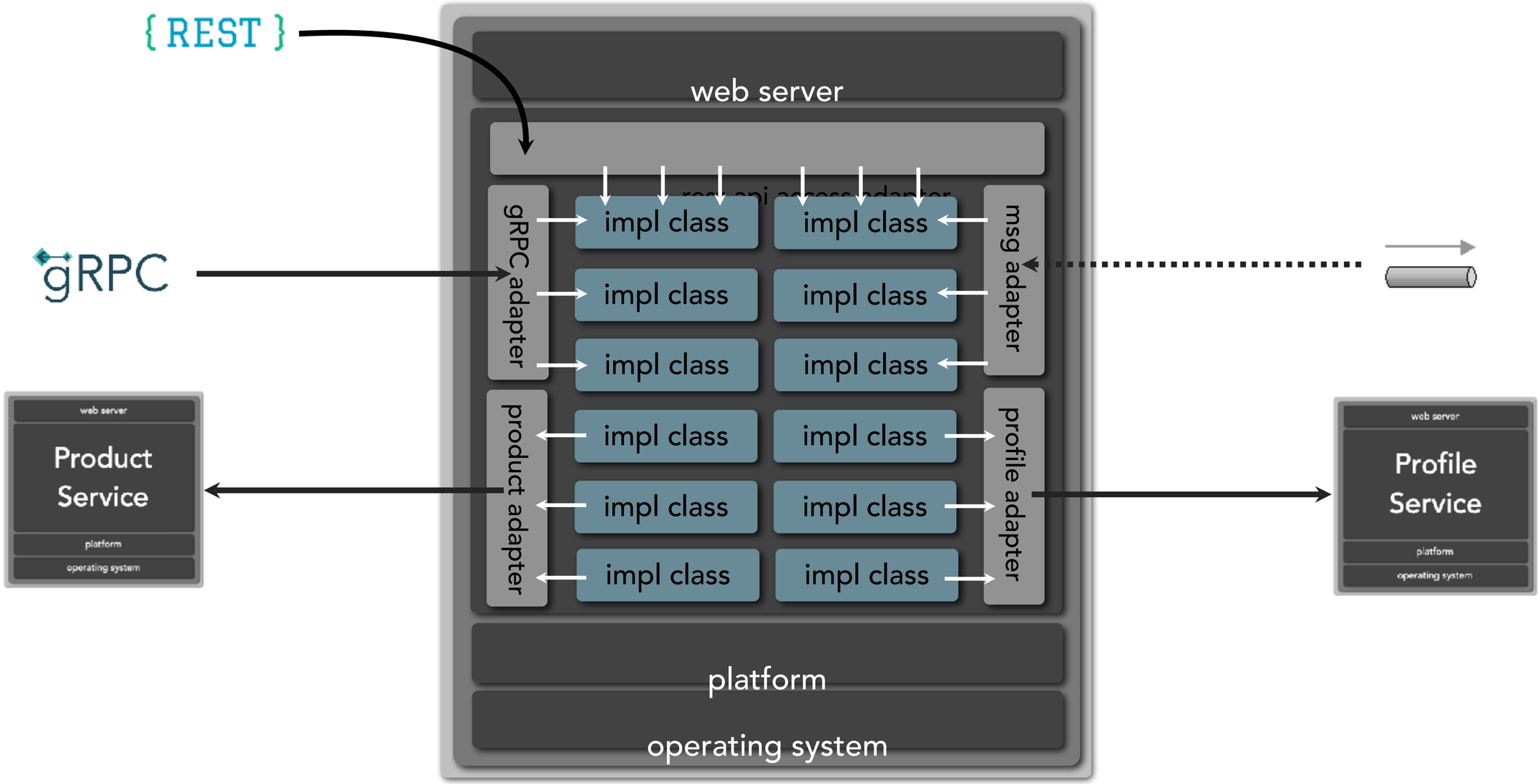
service design

micro-hexagonal design **to support** protocol agnostic processing



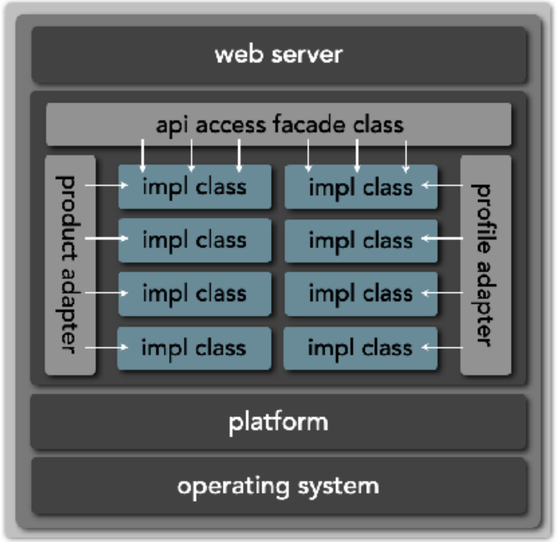
service design

micro-hexagonal design **to support** protocol agnostic processing



service design

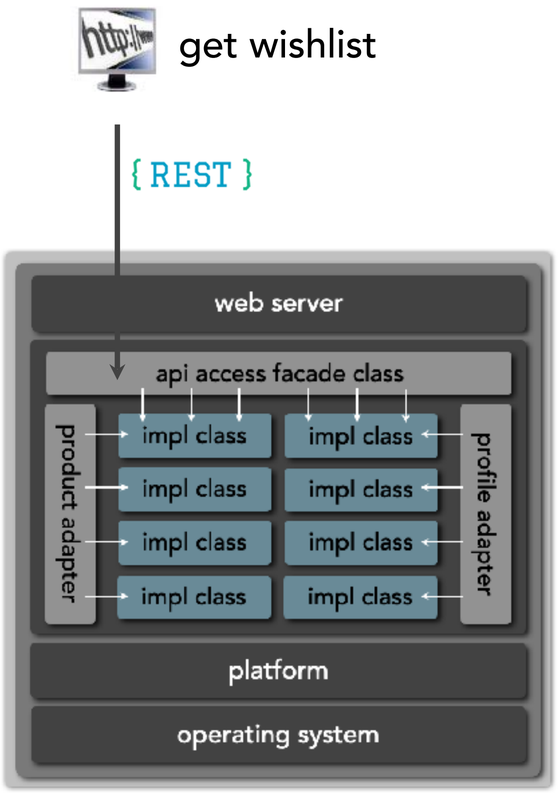
micro-hexagonal design **to support** protocol agnostic processing



Wishlist Service

service design

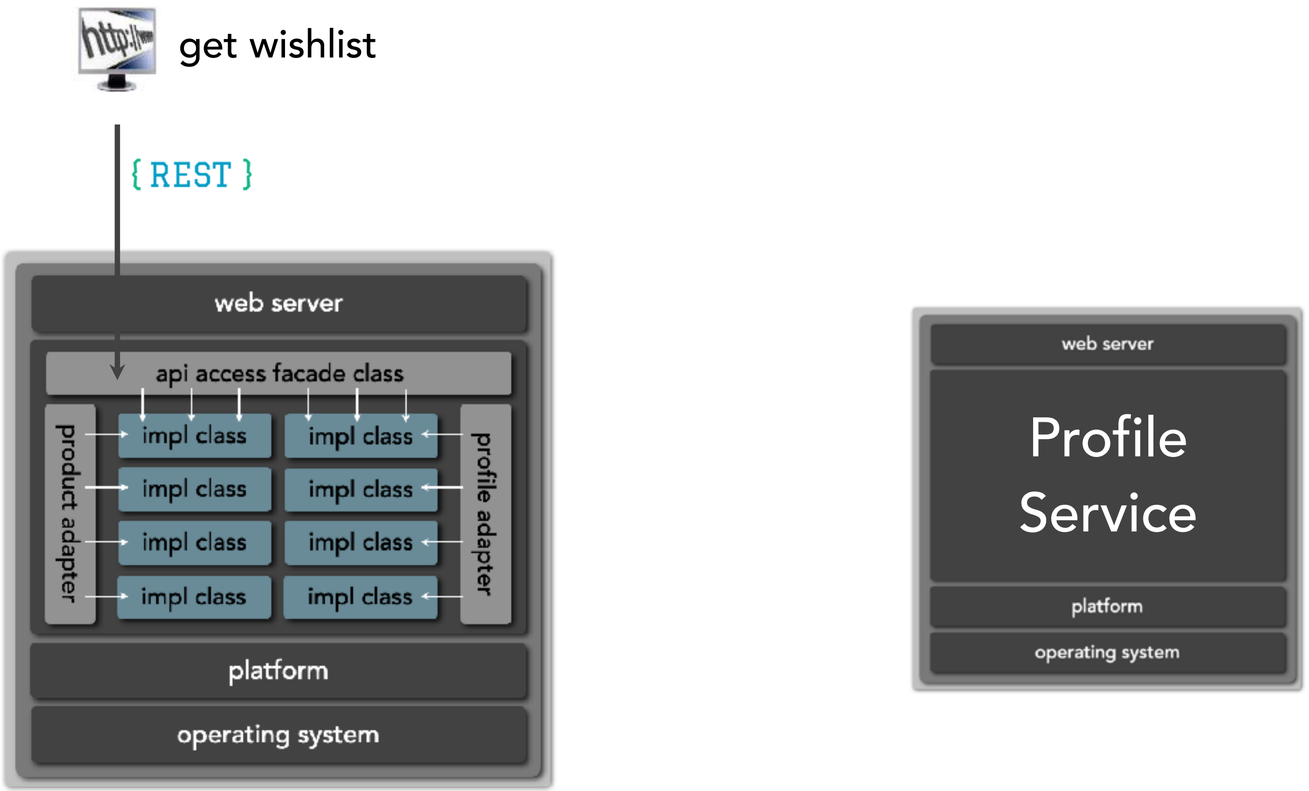
micro-hexagonal design **to support** protocol agnostic processing



Wishlist Service

service design

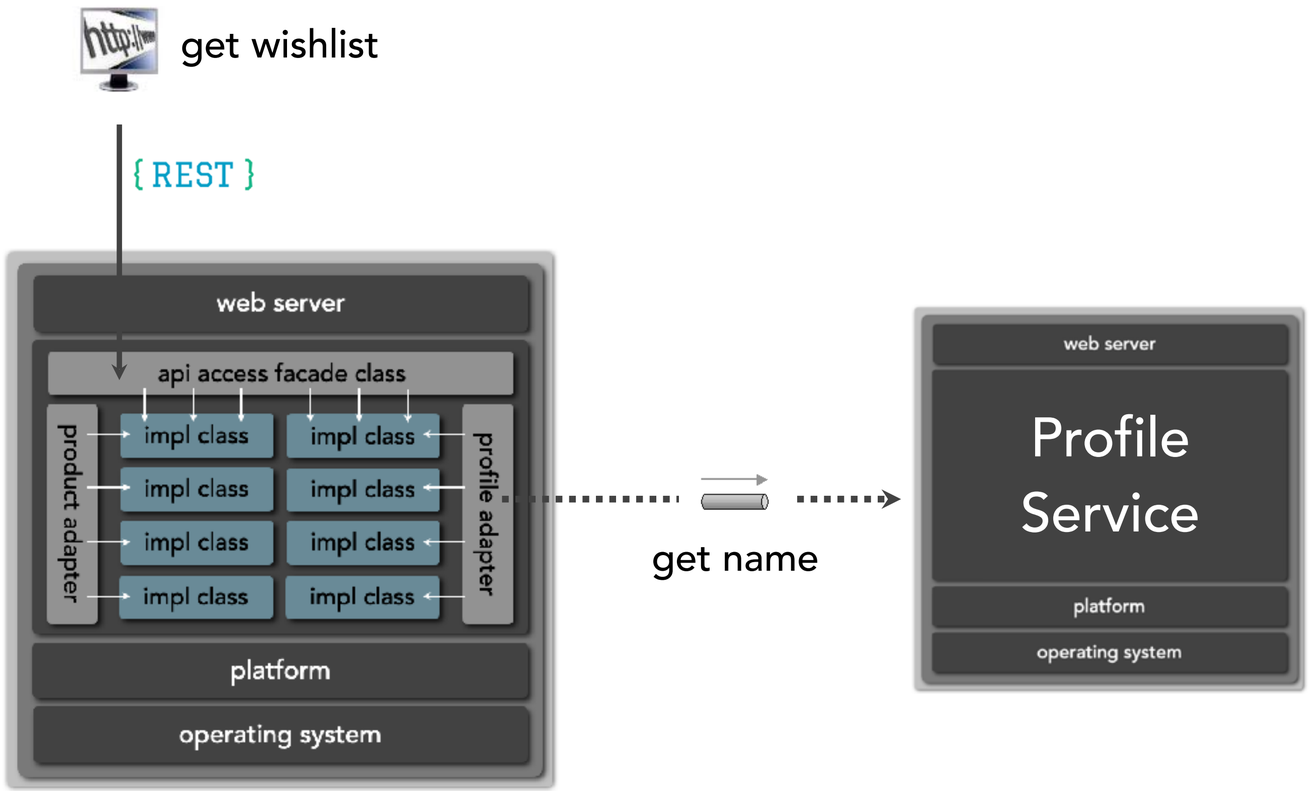
micro-hexagonal design **to support** protocol agnostic processing



Wishlist Service

service design

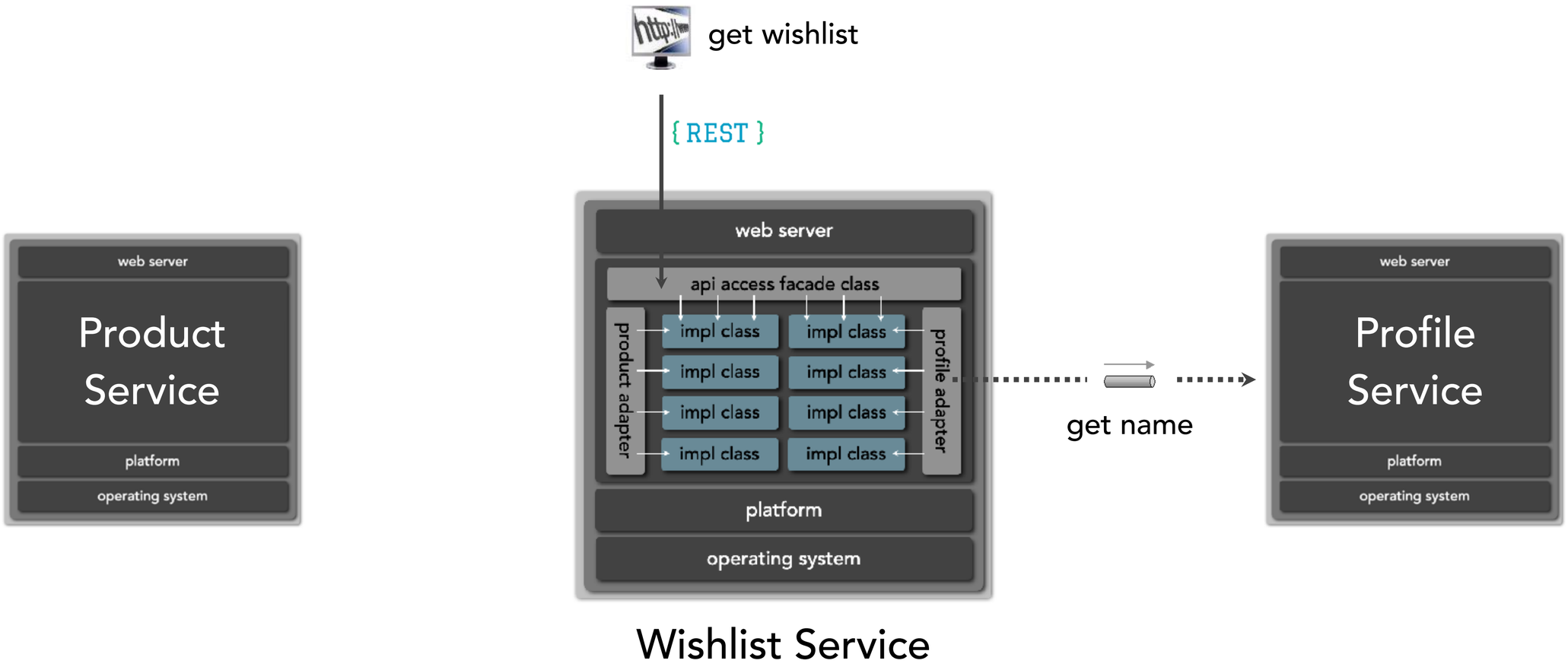
micro-hexagonal design **to support** protocol agnostic processing



Wishlist Service

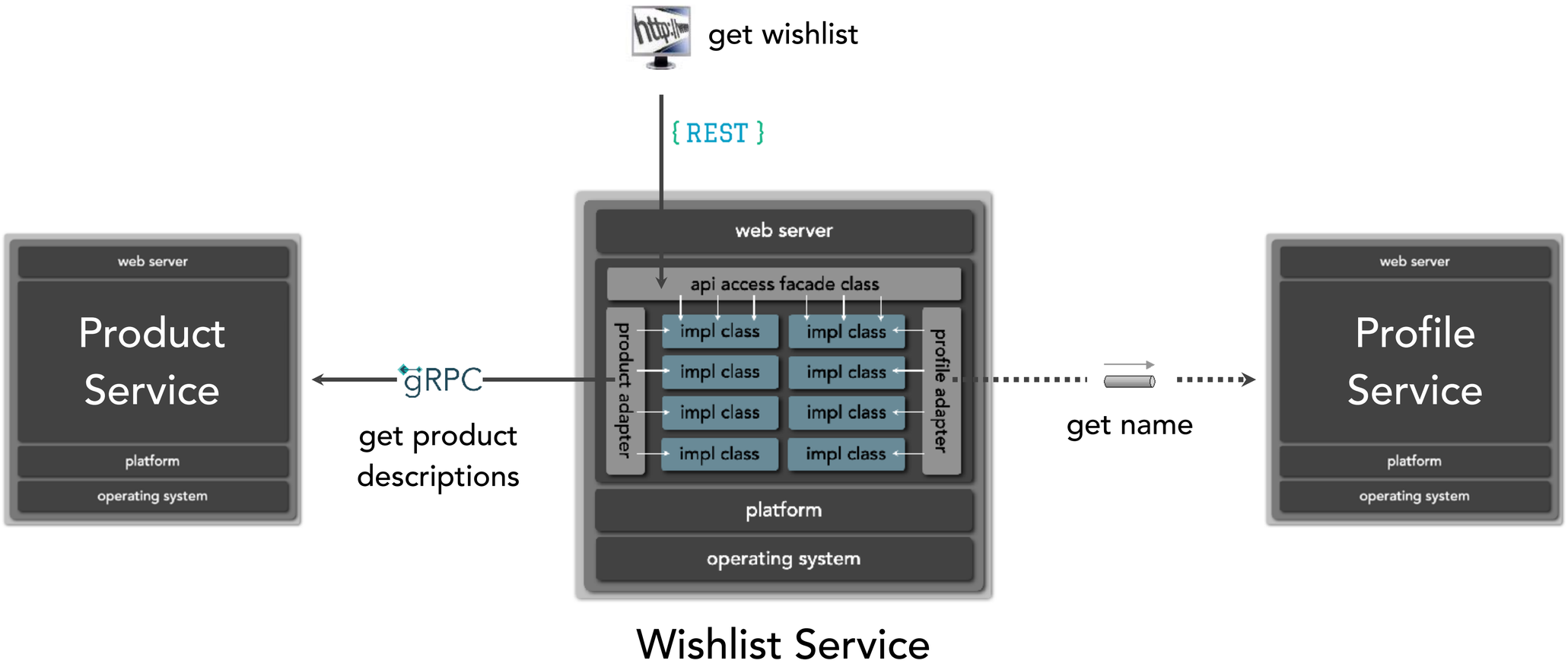
service design

micro-hexagonal design **to support** protocol agnostic processing



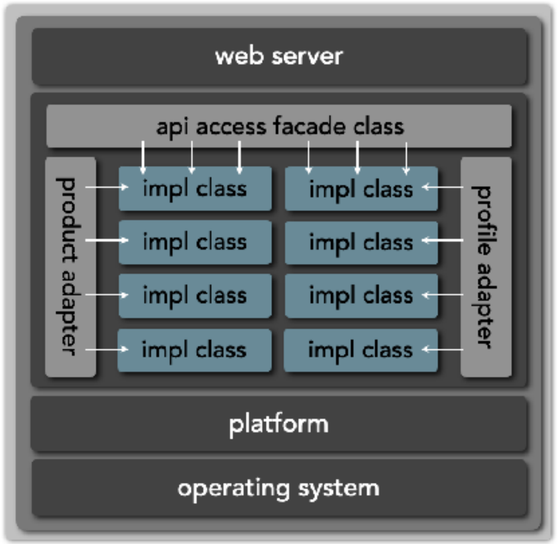
service design

micro-hexagonal design **to support** protocol agnostic processing



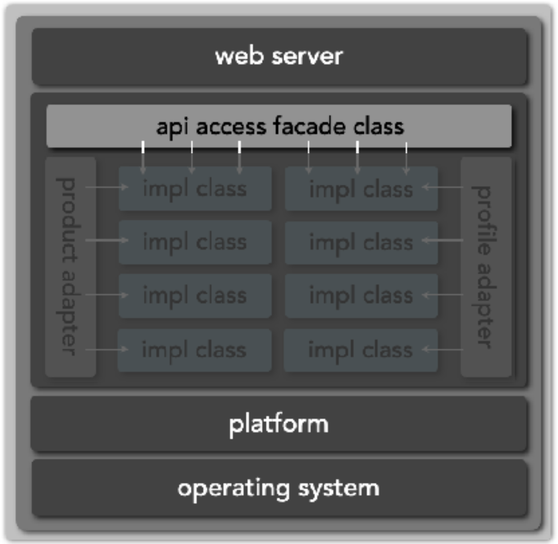
service design

micro-hexagonal design **to support** protocol agnostic processing



service design

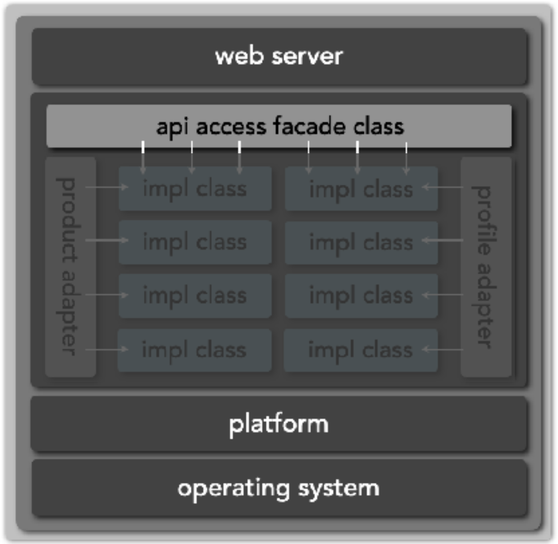
micro-hexagonal design **to support** protocol agnostic processing



API REST Access Adapter Class

service design

micro-hexagonal design **to support** protocol agnostic processing

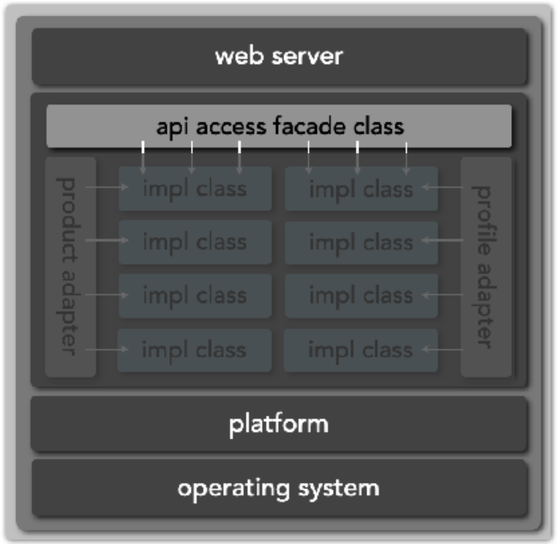


API REST Access Adapter Class

```
@Path("/wishlist")
@Produces(MediaType.APPLICATION_JSON)
@GET
public String getWishlist(long customerId) {
    WishlistImpl wishlist = new WishlistImpl();
    WishlistItems wishlistItems =
        wishlist.getWishlistItems(customerId);
    return convertToJson(wishlistItems);
}
```

service design

micro-hexagonal design **to support** protocol agnostic processing

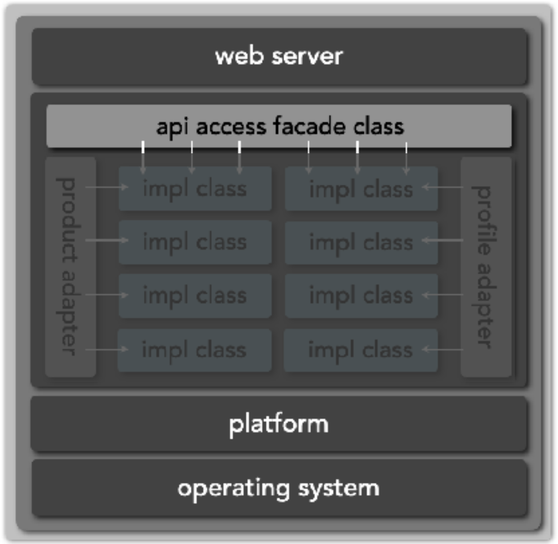


API REST Access Adapter Class

```
@Path("/wishlist")
@Produces(MediaType.APPLICATION_JSON)
@GET
public String getWishlist(long customerId) {
    WishlistImpl wishlist = new WishlistImpl();
    WishlistItems wishlistItems =
        wishlist.getWishlistItems(customerId);
    return convertToJson(wishlistItems);
}
```

service design

micro-hexagonal design **to support** protocol agnostic processing

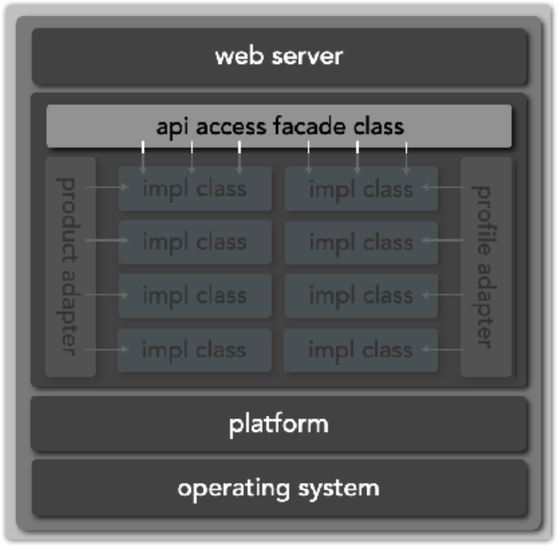


API REST Access Adapter Class

```
@Path("/wishlist")
@Produces(MediaType.APPLICATION_JSON)
@GET
public String getWishlist(long customerId) {
    WishlistImpl wishlist = new WishlistImpl();
    WishlistItems wishlistItems =
        wishlist.getWishlistItems(customerId);
    return convertToJson(wishlistItems);
}
```

service design

micro-hexagonal design **to support** protocol agnostic processing



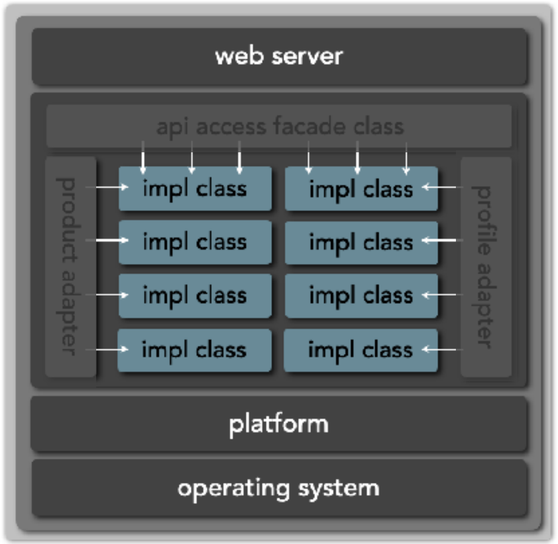
API REST Access Adapter Class

```
@Path("/wishlist")
@Produces(MediaType.APPLICATION_JSON)
@GET
public String getWishlist(long customerId) {
    WishlistImpl wishlist = new WishlistImpl();
    WishlistItems wishlistItems =
        wishlist.getWishlistItems(customerId);
    return convertToJson(wishlistItems);
}
```

service design

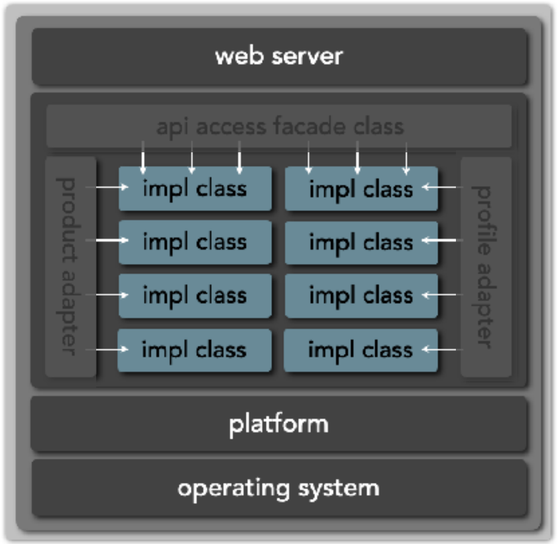
micro-hexagonal design **to support** protocol agnostic processing

Implementation Class



service design

micro-hexagonal design **to support** protocol agnostic processing

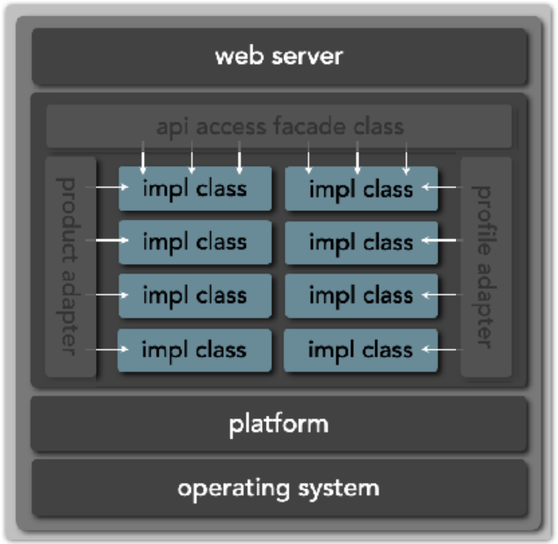


Implementation Class

```
public WishlistItems getWishlistItems(long customerId) {
    List<String> itemsIds = getWishlistItemsFromDB(customerId);
    String customerName = profileAdapter.getCustomerName(customerId);
    Map<String, String> itemDescriptions =
        productAdapter.getProductDescs(itemIds);
    return new WishlistItems(customerName, itemDescriptions);
}
```

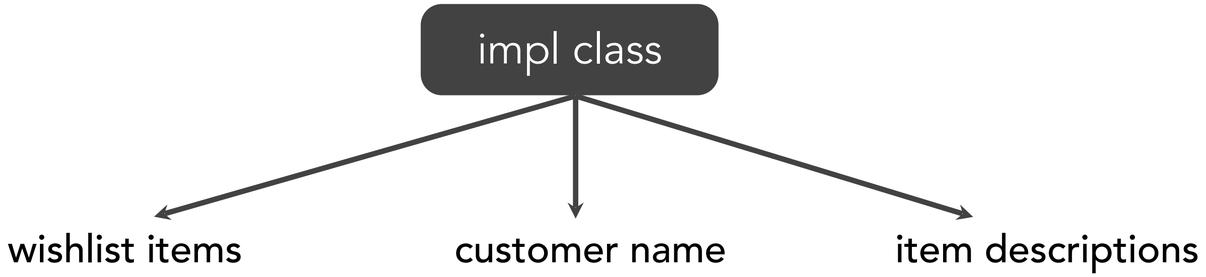
service design

micro-hexagonal design **to support** protocol agnostic processing



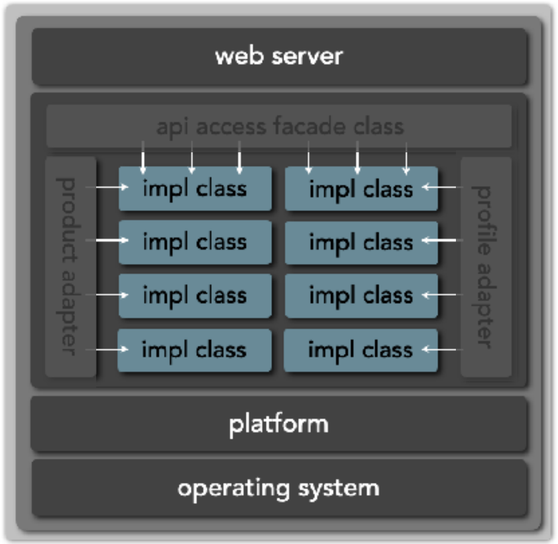
Implementation Class

```
public WishlistItems getWishlistItems(long customerId) {
    List<String> itemsIds = getWishlistItemsFromDB(customerId);
    String customerName = profileAdapter.getCustomerName(customerId);
    Map<String, String> itemDescriptions =
         productAdapter.getProductDescs(itemIds);
    return new WishlistItems(customerName, itemDescriptions);
}
```



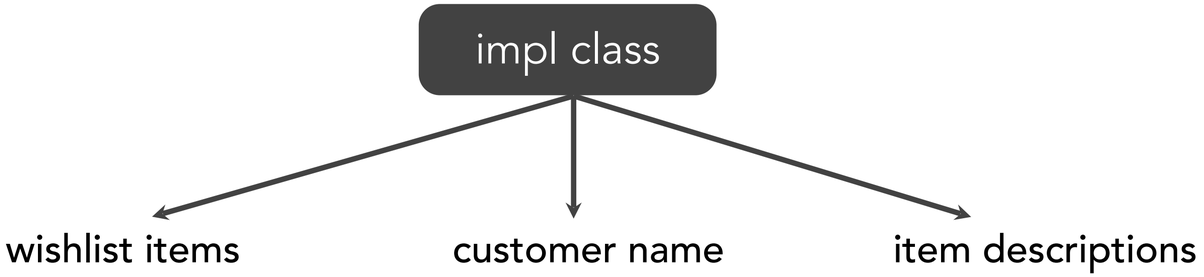
service design

micro-hexagonal design **to support** protocol agnostic processing



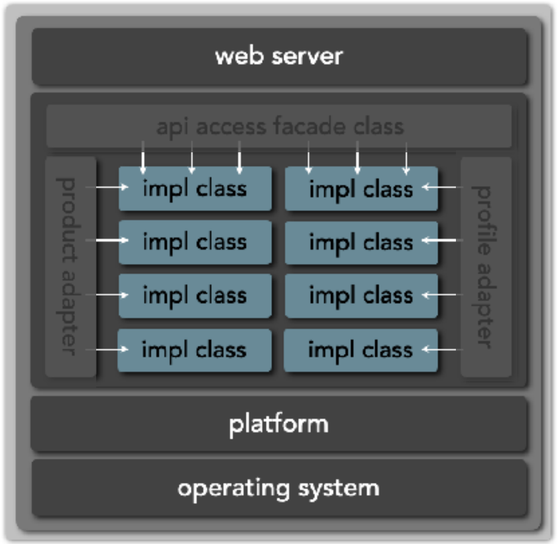
Implementation Class

```
public WishlistItems getWishlistItems(long customerId) {
    List<String> itemsIds = getWishlistItemsFromDB(customerId);
    String customerName = profileAdapter.getCustomerName(customerId);
    Map<String, String> itemDescriptions =
         productAdapter.getProductDescs(itemIds);
    return new WishlistItems(customerName, itemDescriptions);
}
```



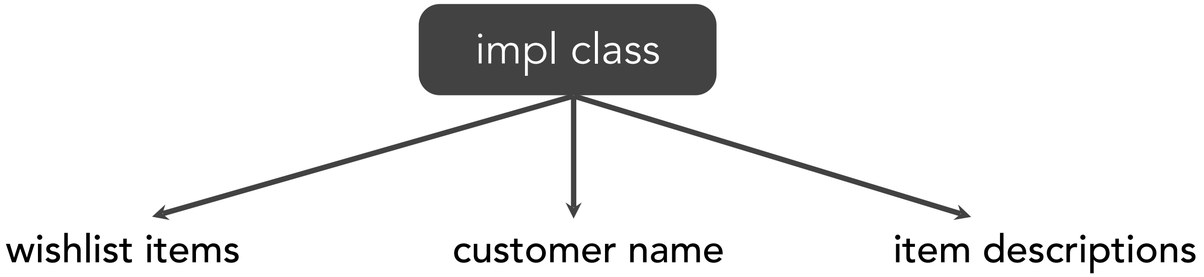
service design

micro-hexagonal design **to support** protocol agnostic processing



Implementation Class

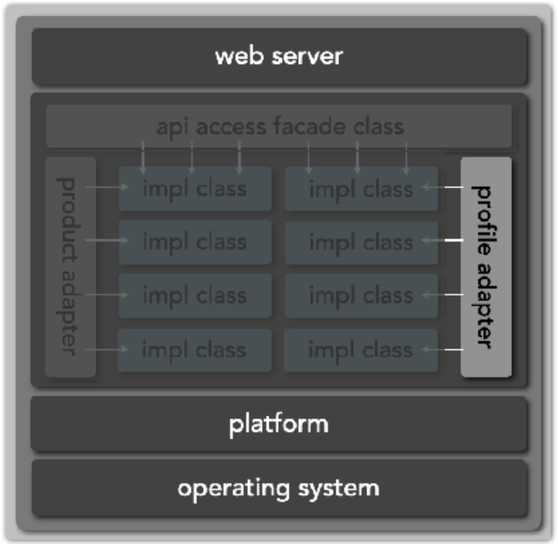
```
public WishlistItems getWishlistItems(long customerId) {
    List<String> itemsIds = getWishlistItemsFromDB(customerId);
    String customerName = profileAdapter.getCustomerName(customerId);
    Map<String, String> itemDescriptions =
        productAdapter.getProductDescs(itemIds);
    return new WishlistItems(customerName, itemDescriptions);
}
```



service design

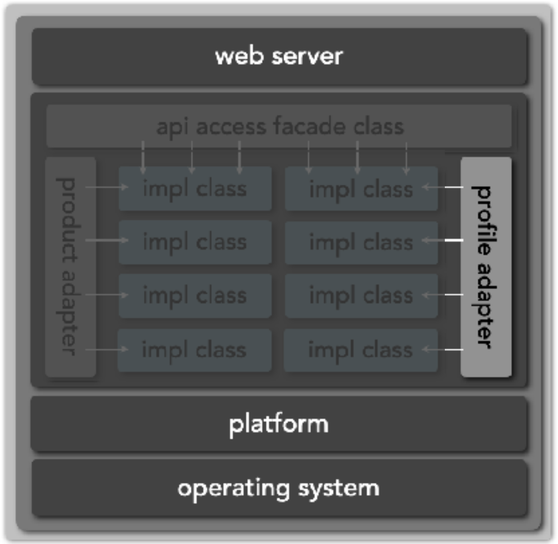
micro-hexagonal design **to support** protocol agnostic processing

Profile Adapter Class (AMQP)



service design

micro-hexagonal design **to support** protocol agnostic processing

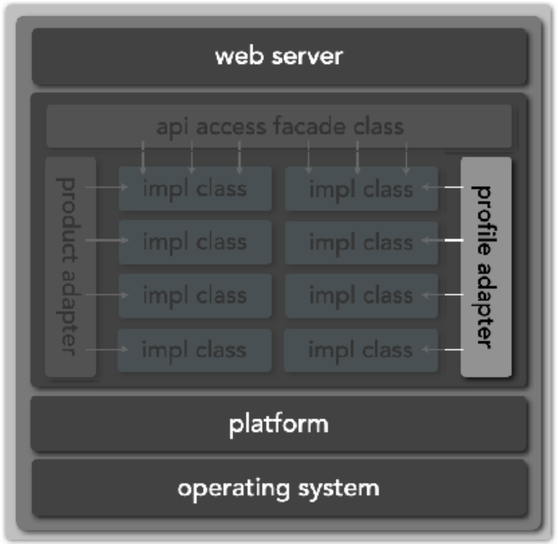


Profile Adapter Class (AMQP)

```
public String getCustomerName(long customerId) {
   Channel channel = connection.createChannel();
   byte[] message = new Long(customerId).toString().getBytes();
   BasicProperties props = new BasicProperties
      .Builder().replyTo("profile.getname.response.q").build();
   channel.basicPublish("", "profile.getname.request.q", props, message);

   final BlockingQueue<String> response = new ArrayBlockingQueue<String>(1);
   channel.basicConsume("profile.getname.response.q", true,
      new DefaultConsumer(channel) {
      @Override
      public void handleDelivery(String consumerTag, Envelope envelope,
          AMQP.BasicProperties properties, byte[] body) throws IOException {
        response.offer(new String(body, "UTF-8"));
      }
  });
  String name = new String(response.take());
  channel.close();
  return name;
}
```

service design

micro-hexagonal design **to support** protocol agnostic processing

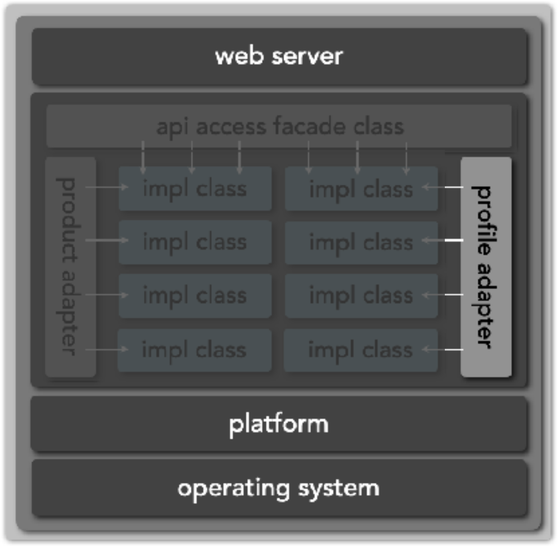


Profile Adapter Class (AMQP)

```
public String getCustomerName(long customerId) {
   Channel channel = connection.createChannel();
   byte[] message = new Long(customerId).toString().getBytes();
   BasicProperties props = new BasicProperties
      .Builder().replyTo("profile.getname.response.q").build();
   channel.basicPublish("", "profile.getname.request.q", props, message);

   final BlockingQueue<String> response = new ArrayBlockingQueue<String>(1);
   channel.basicConsume("profile.getname.response.q", true,
      new DefaultConsumer(channel) {
      @Override
      public void handleDelivery(String consumerTag, Envelope envelope,
          AMQP.BasicProperties properties, byte[] body) throws IOException {
        response.offer(new String(body, "UTF-8"));
      }
  });
  String name = new String(response.take());
  channel.close();
  return name;
}
```

service design

micro-hexagonal design **to support** protocol agnostic processing



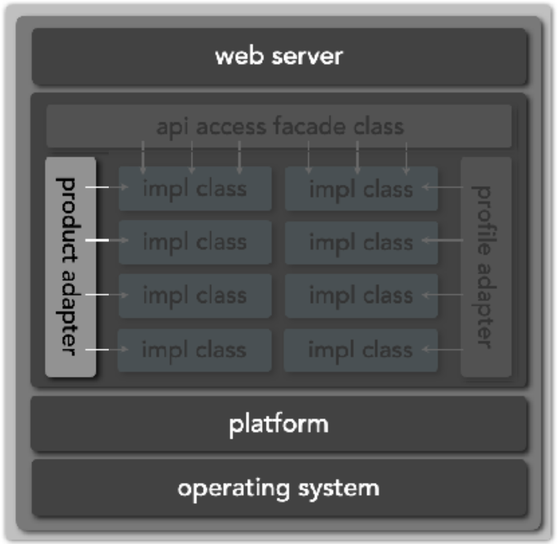
Profile Adapter Class (AMQP)

```
public String getCustomerName(long customerId) {
   Channel channel = connection.createChannel();
   byte[] message = new Long(customerId).toString().getBytes();
   BasicProperties props = new BasicProperties
      .Builder().replyTo("profile.getname.response.q").build();
   channel.basicPublish("", "profile.getname.request.q", props, message);

   final BlockingQueue<String> response = new ArrayBlockingQueue<String>(1);
   channel.basicConsume("profile.getname.response.q", true,
      new DefaultConsumer(channel) {
      @Override
      public void handleDelivery(String consumerTag, Envelope envelope,
          AMQP.BasicProperties properties, byte[] body) throws IOException {
        response.offer(new String(body, "UTF-8"));
      }
  });
  String name = new String(response.take());
  channel.close();
  return name;
}
```

service design

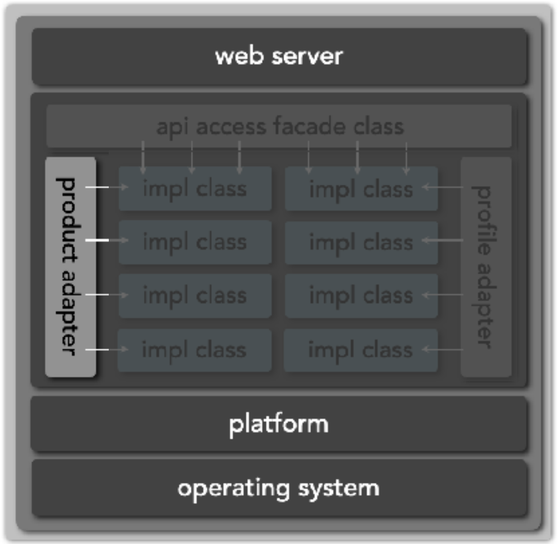
micro-hexagonal design **to support** protocol agnostic processing

Product Adapter Class (gRPC)



service design

micro-hexagonal design **to support** protocol agnostic processing

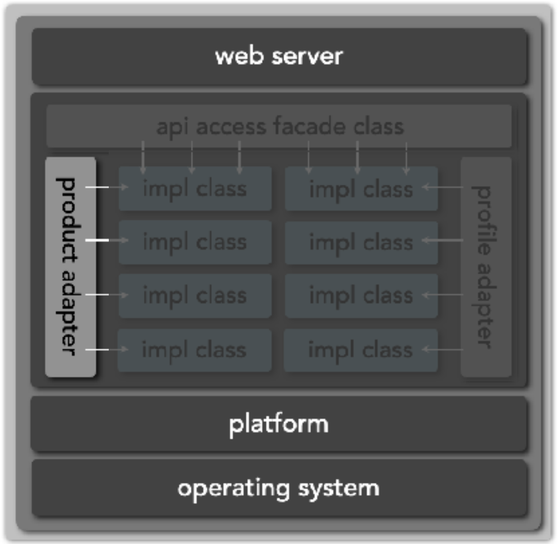


Product Adapter Class (gRPC)

```
public List<String> getProductDescs(List<String> productIds) {
  ProductRequest request = ProductRequest.newBuilder()
      .setProductIds(productIds).build();
  ProductReply response;
  try {
    response = blockingStub.getDescriptions(request);
  } catch (StatusRuntimeException e) {
    logger.log(Level.ERROR, "RPC call failed: {0}", e.getStatus());
    return new ArrayList<String>();
  }
  return response.getMessage();
}
```

service design

micro-hexagonal design **to support** protocol agnostic processing

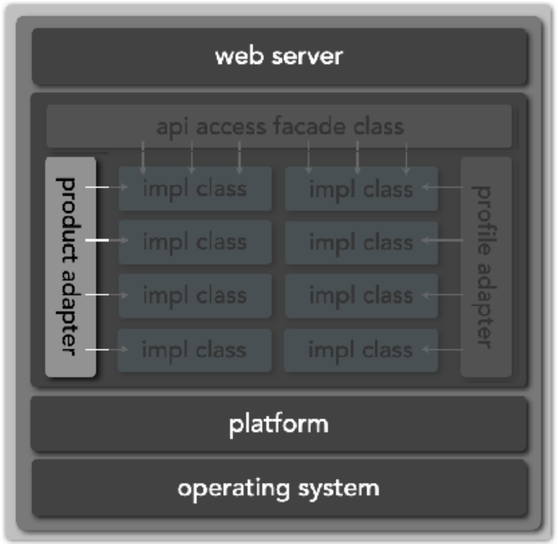


Product Adapter Class (gRPC)

```
public List<String> getProductDescs(List<String> productIds) {
  ProductRequest request = ProductRequest.newBuilder()
      .setProductIds(productIds).build();
  ProductReply response;
  try {
    response = blockingStub.getDescriptions(request);
  } catch (StatusRuntimeException e) {
    logger.log(Level.ERROR, "RPC call failed: {0}", e.getStatus());
    return new ArrayList<String>();
  }
  return response.getMessage();
}
```

service design

micro-hexagonal design **to support** protocol agnostic processing

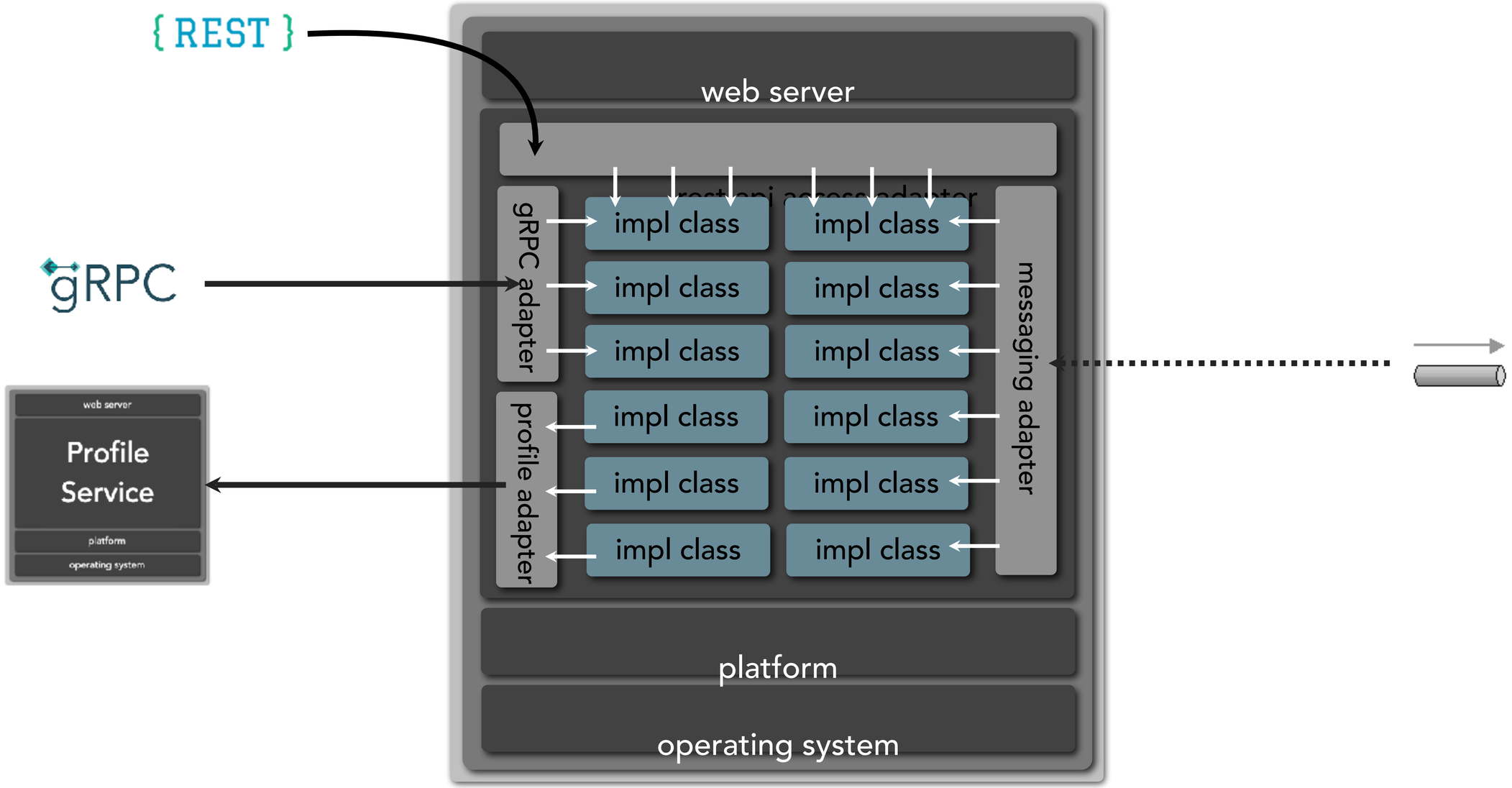


Product Adapter Class (gRPC)

```
public List<String> getProductDescs(List<String> productIds) {
  ProductRequest request = ProductRequest.newBuilder()
      .setProductIds(productIds).build();
  ProductReply response;
  try {
    response = blockingStub.getDescriptions(request);
  } catch (StatusRuntimeException e) {
    logger.log(Level.ERROR, "RPC call failed: {0}", e.getStatus());
    return new ArrayList<String>();
  }
  return response.getMessage();
}
```

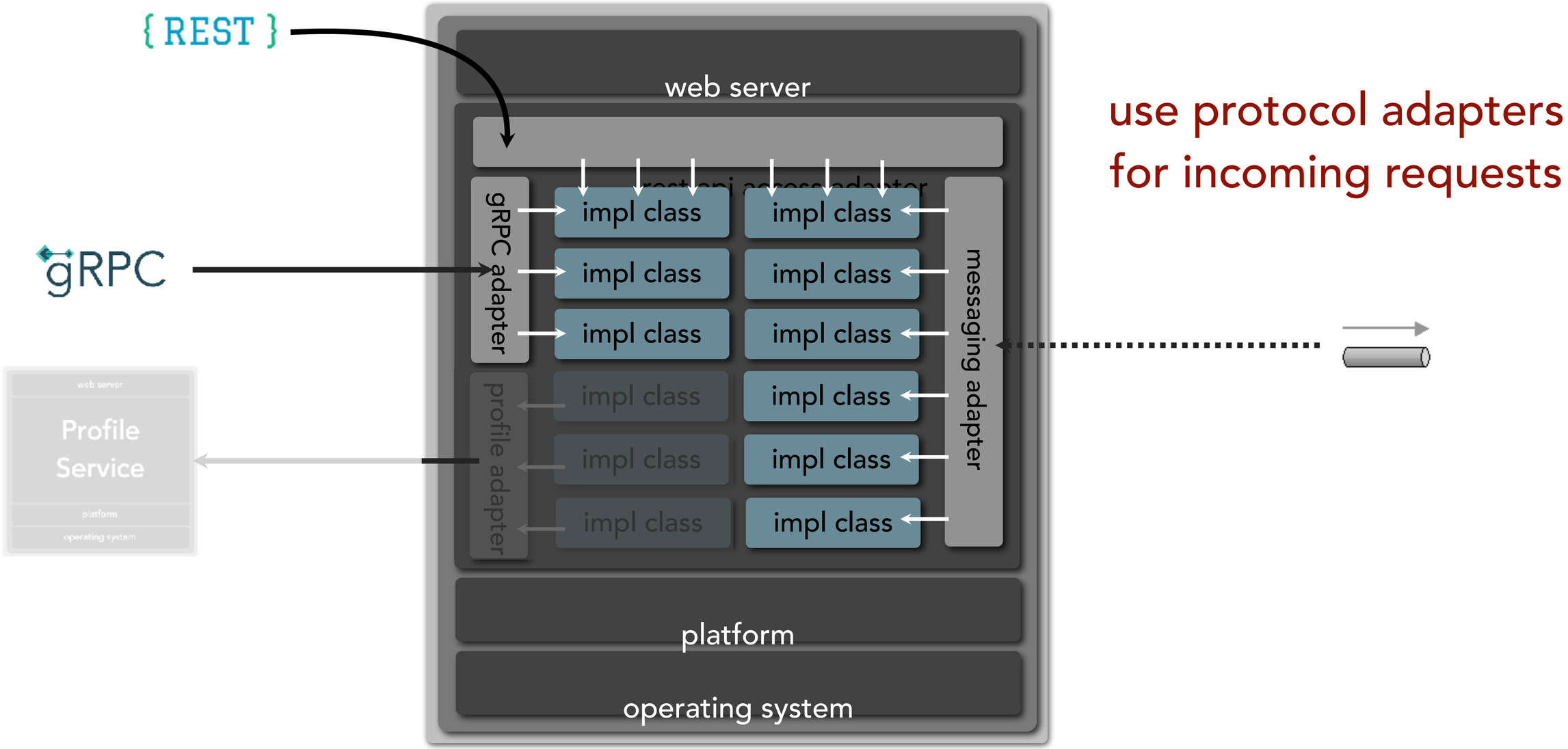
service design

micro-hexagonal design **to support** protocol agnostic processing



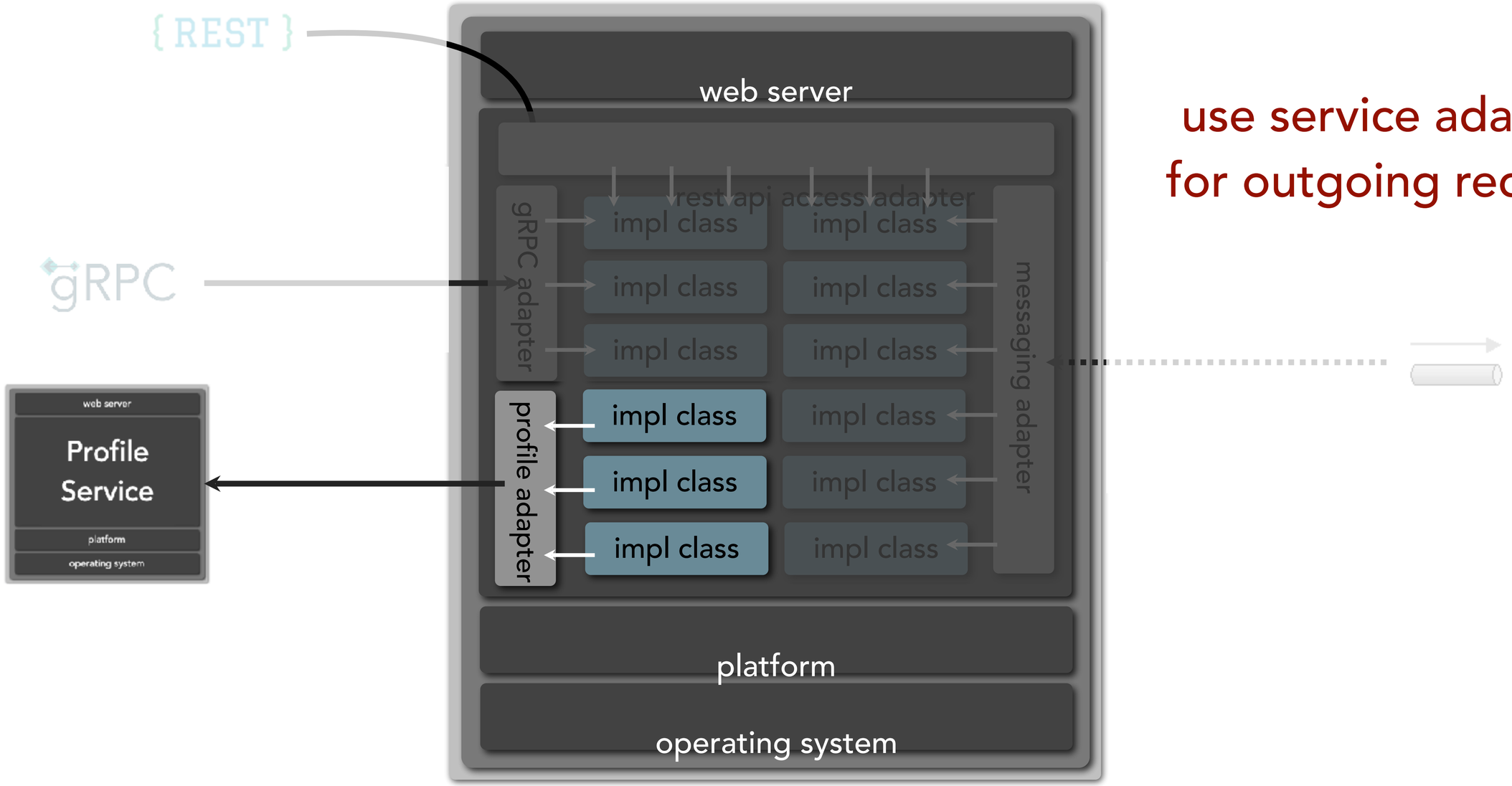
service design

micro-hexagonal design **to support** protocol agnostic processing



service design

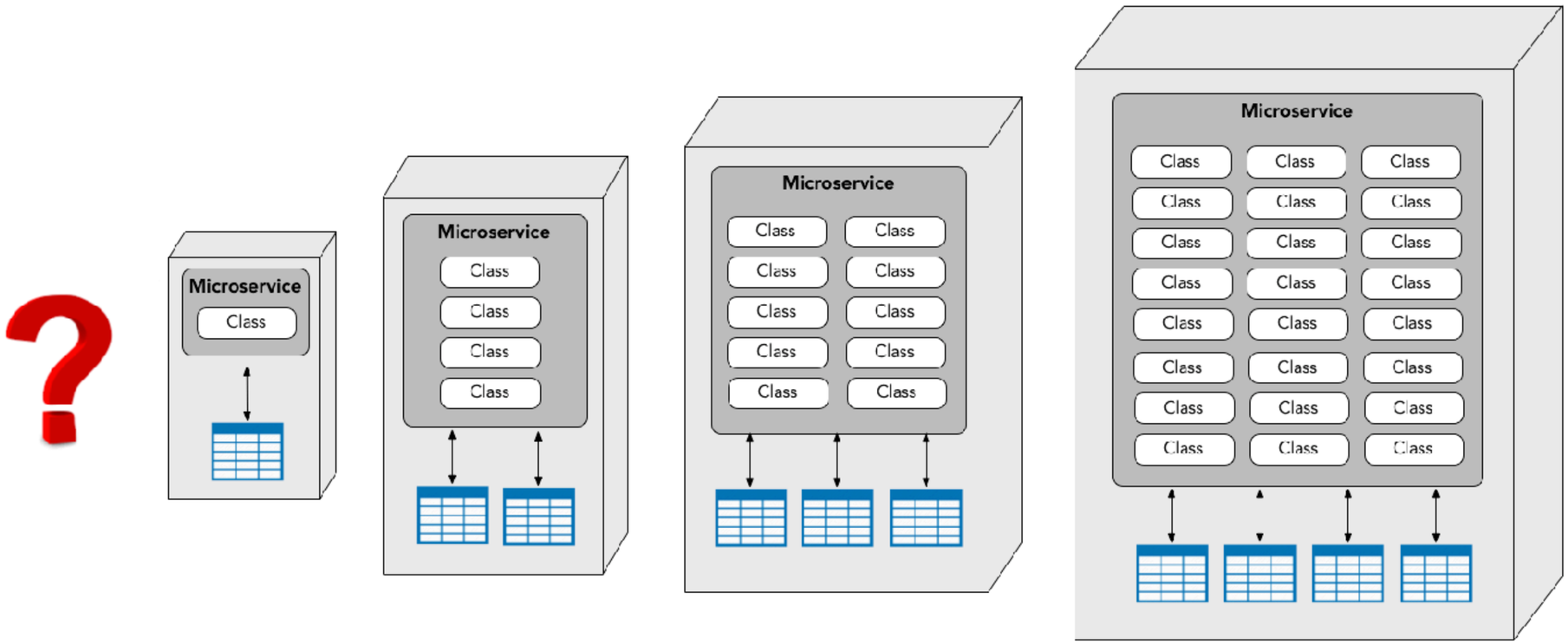
micro-hexagonal design **to support** protocol agnostic processing



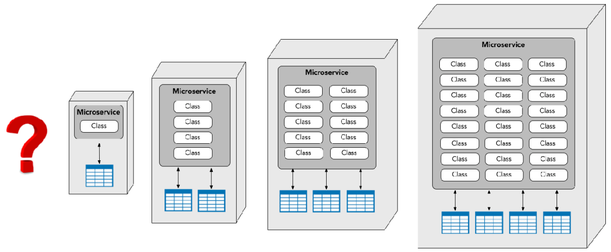
use service adapters for outgoing requests

Service Granularity

what is the right level of granularity for a service?



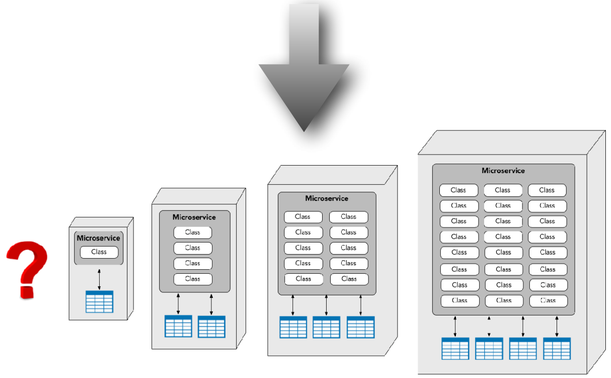
service granularity



service granularity

granularity drivers

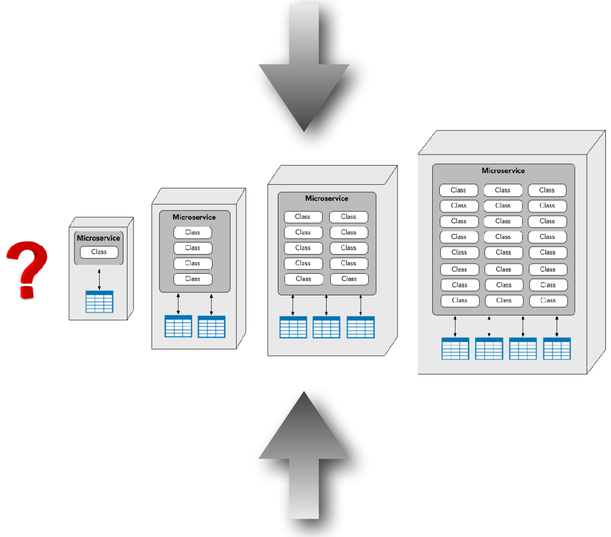
"when should I consider breaking apart a service?"



service granularity

granularity drivers

"when should I consider breaking apart a service?"



granularity factors

"what factors influence service granularity?"

service granularity

granularity drivers

"when should I consider breaking apart a service?"

service granularity

granularity drivers

"when should I consider breaking apart a service?"

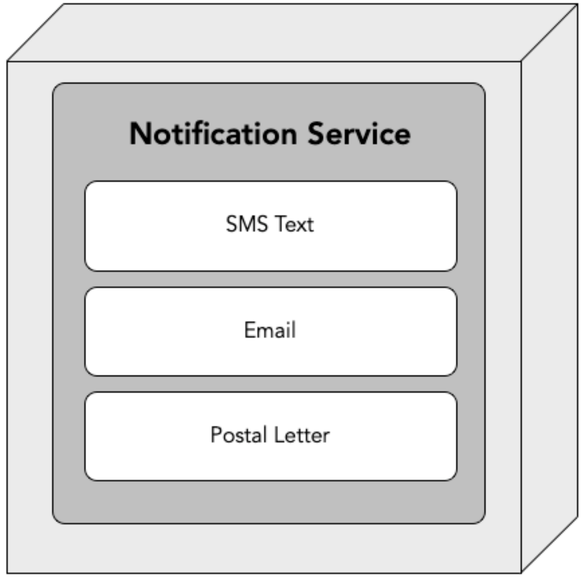
service
functionality

service granularity

granularity drivers

"when should I consider breaking apart a service?"

service functionality

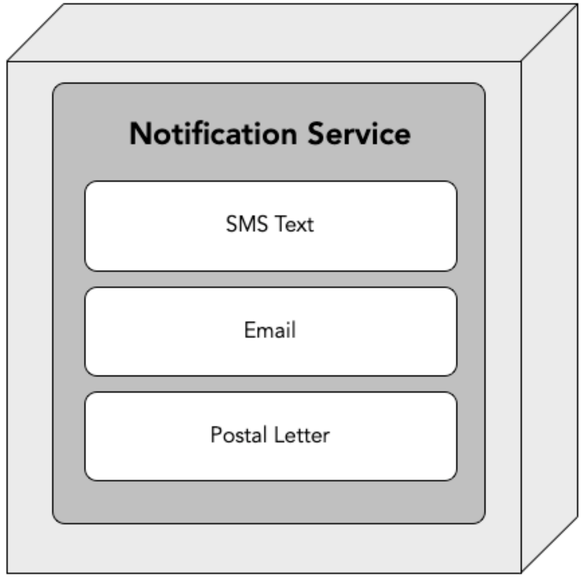


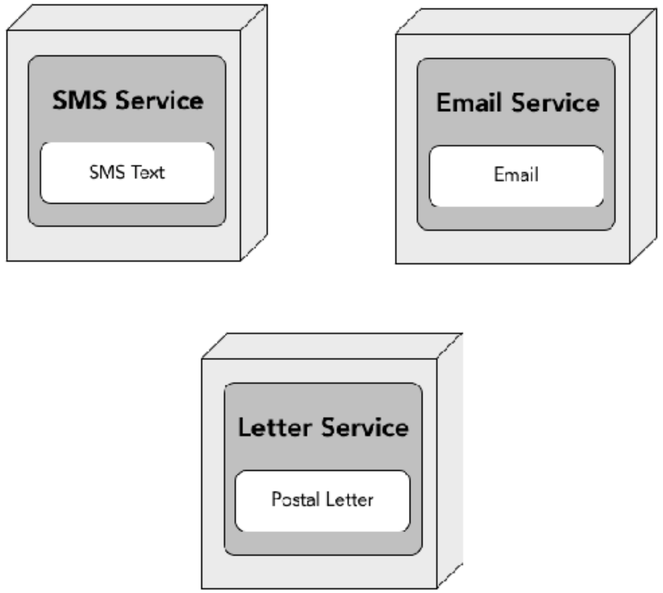
service granularity

granularity drivers

"when should I consider breaking apart a service?"

service functionality



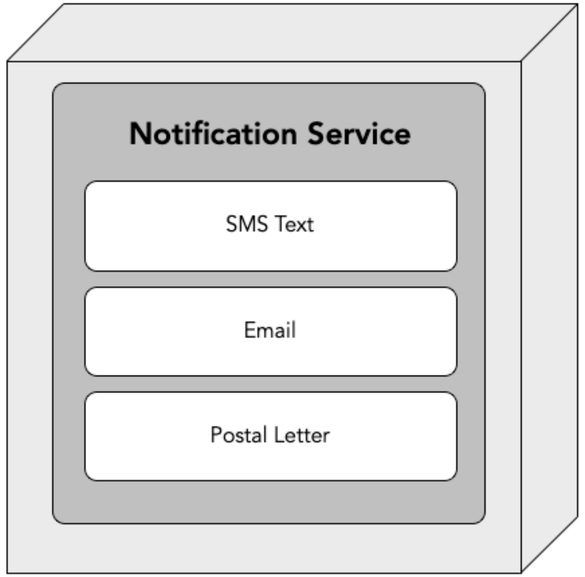


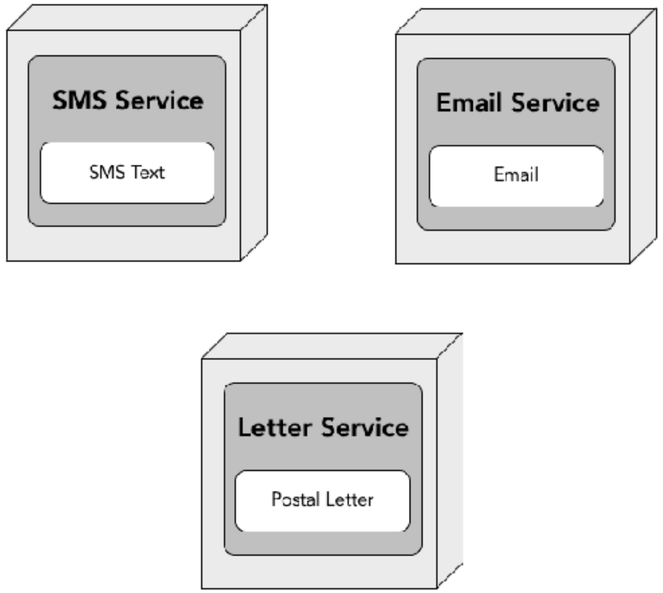
service granularity

granularity drivers

"when should I consider breaking apart a service?"

service functionality



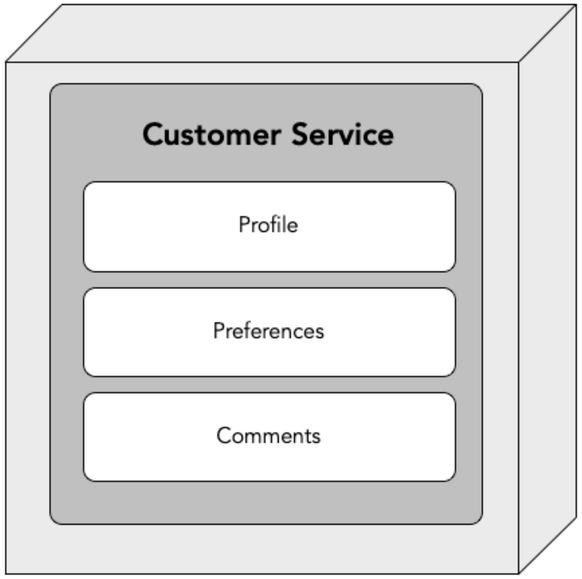


service granularity

granularity drivers

"when should I consider breaking apart a service?"

service functionality

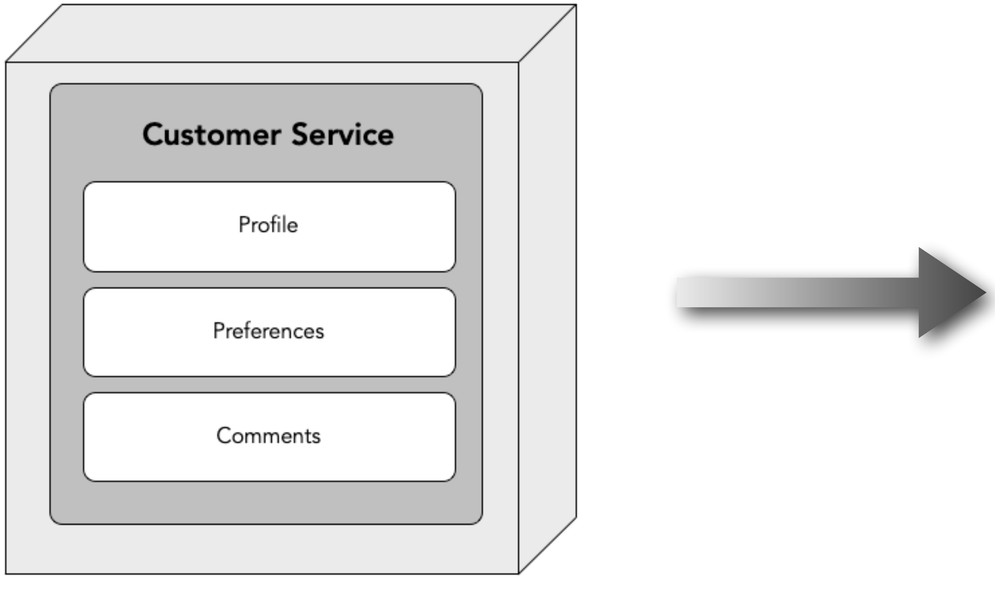


service granularity

granularity drivers

"when should I consider breaking apart a service?"

service functionality



service granularity

granularity drivers

"when should I consider breaking apart a service?"

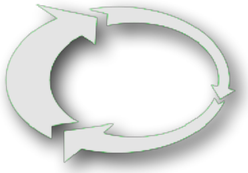
service
functionality

service granularity

granularity drivers

"when should I consider breaking apart a service?"

service
functionality

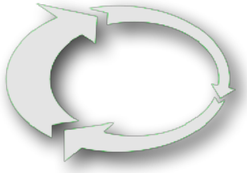


code
volatility

service granularity

granularity drivers

"when should I consider breaking apart a service?"

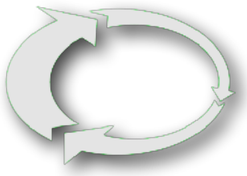


code
volatility

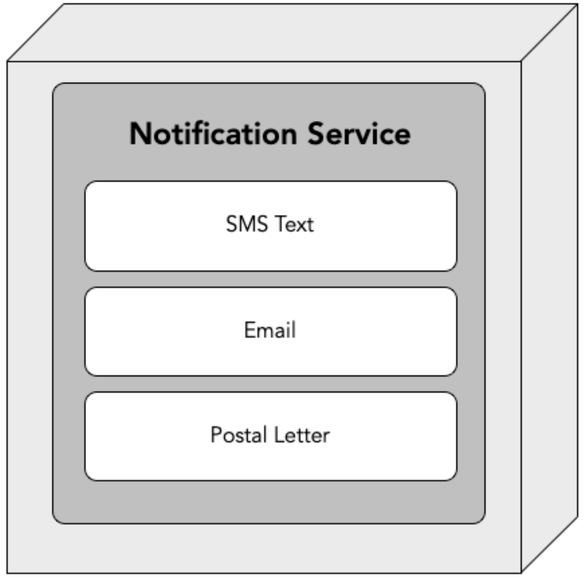
service granularity

granularity drivers

"when should I consider breaking apart a service?"



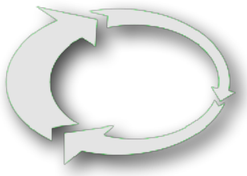
code
volatility



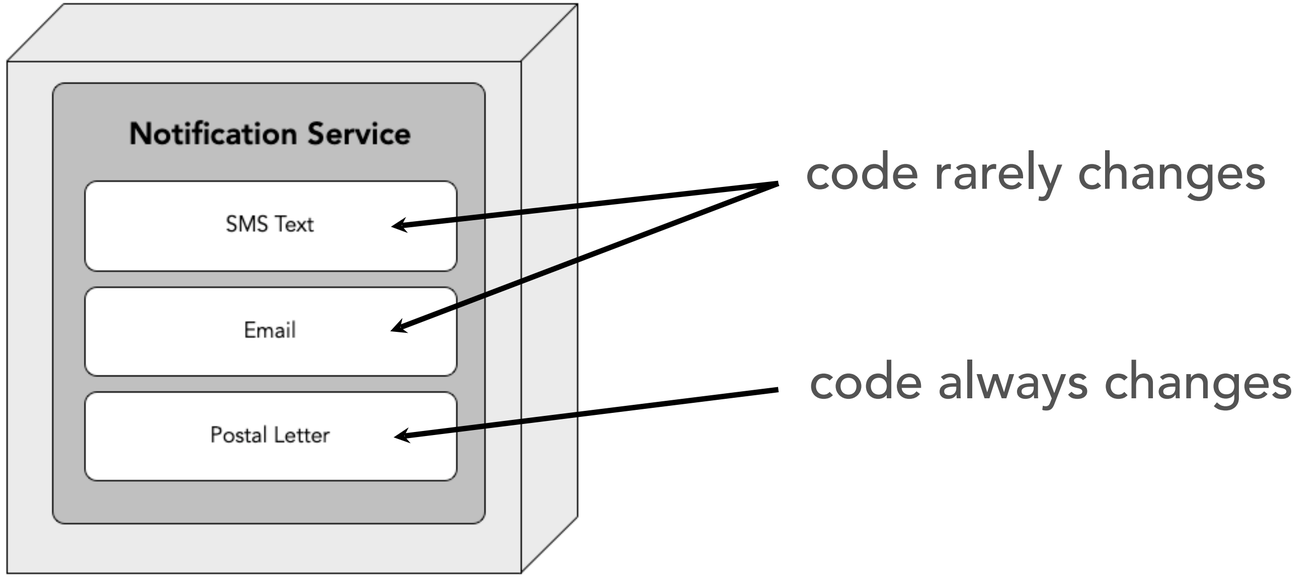
service granularity

granularity drivers

"when should I consider breaking apart a service?"



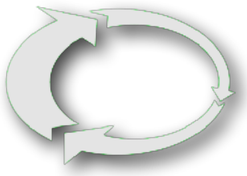
code
volatility



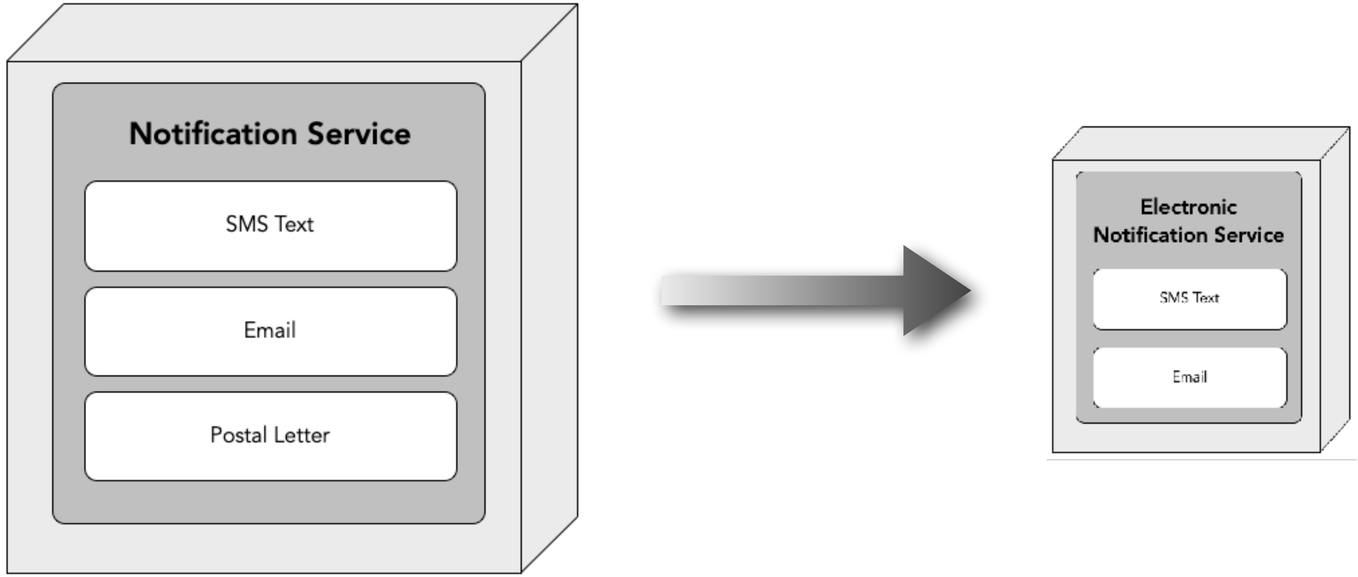
service granularity

granularity drivers

"when should I consider breaking apart a service?"



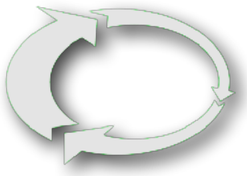
code volatility



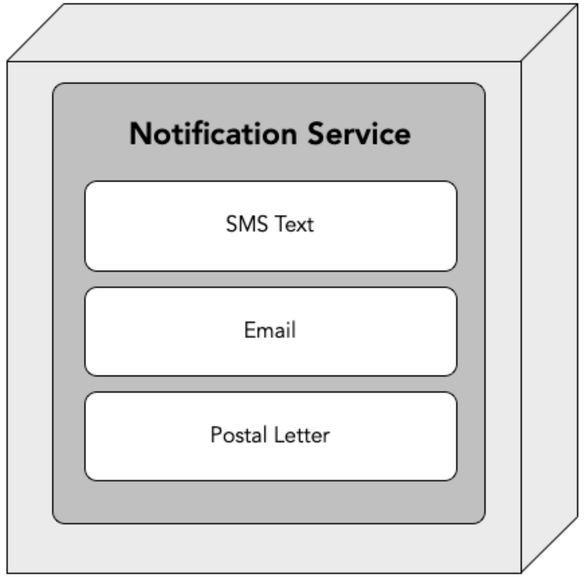
service granularity

granularity drivers

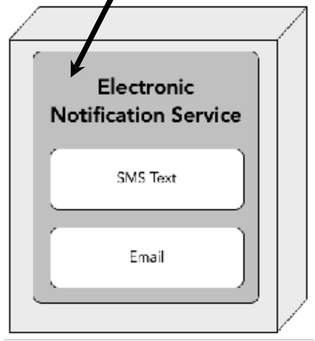
"when should I consider breaking apart a service?"

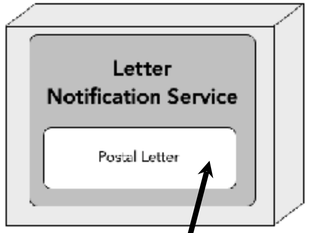


code volatility



code rarely changes





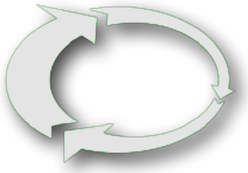
code always changes

service granularity

granularity drivers

"when should I consider breaking apart a service?"

service
functionality



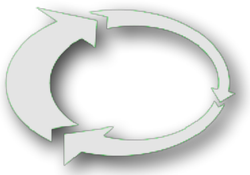
code
volatility

service granularity

granularity drivers

"when should I consider breaking apart a service?"

service functionality



code volatility

scalability and throughput

service granularity

granularity drivers

"when should I consider breaking apart a service?"

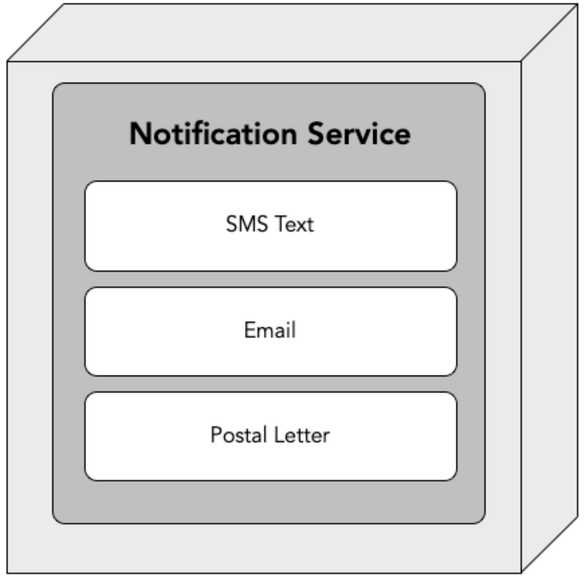
scalability and throughput

service granularity

granularity drivers

"when should I consider breaking apart a service?"

scalability and throughput

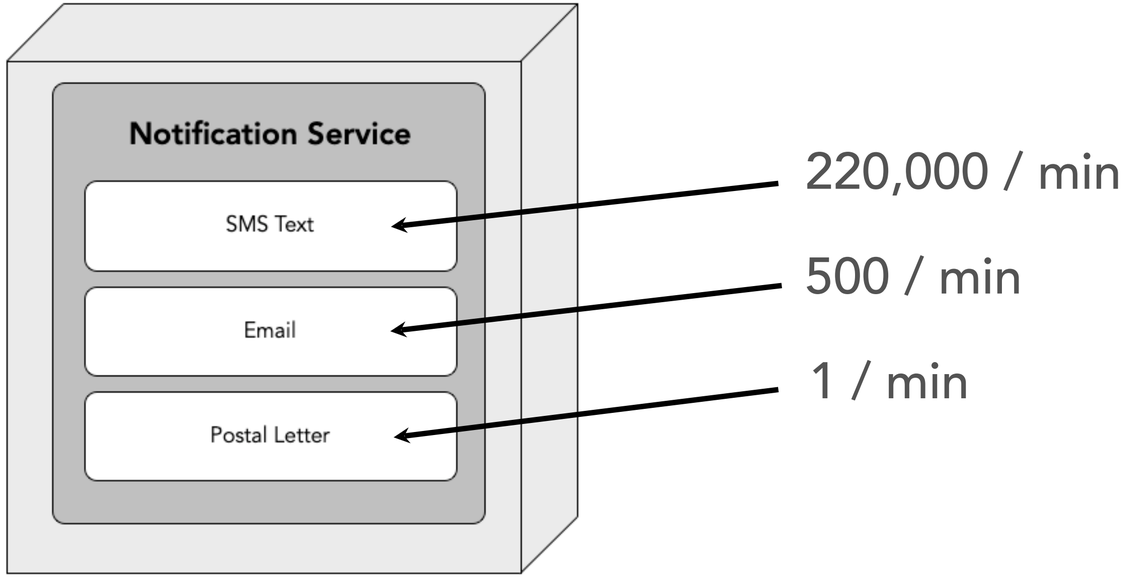


service granularity

granularity drivers

"when should I consider breaking apart a service?"

scalability and throughput

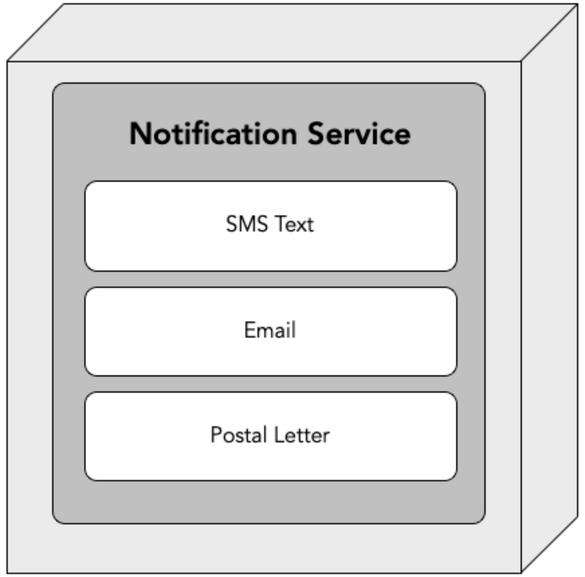


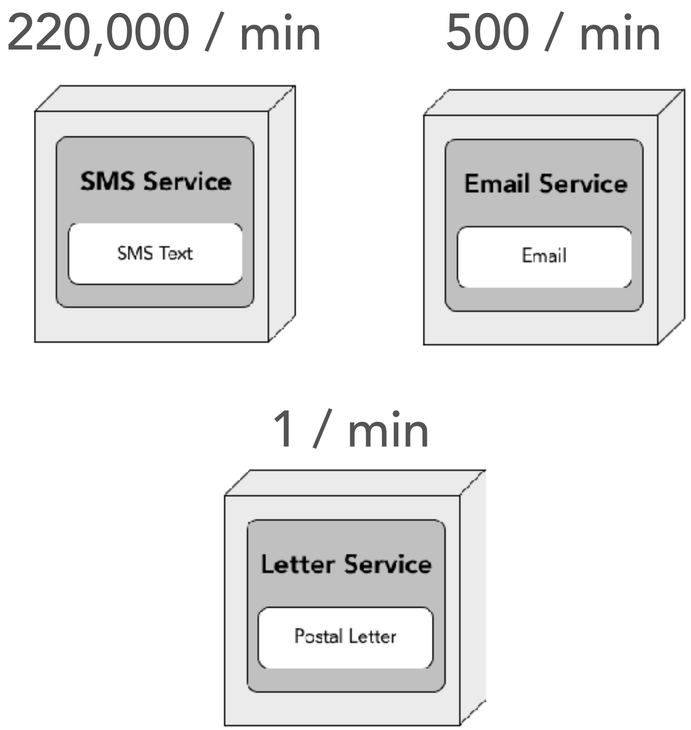
service granularity

granularity drivers

"when should I consider breaking apart a service?"

scalability and throughput



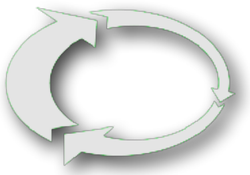


service granularity

granularity drivers

"when should I consider breaking apart a service?"

service functionality



code volatility

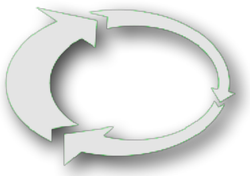
scalability and throughput

service granularity

granularity drivers

"when should I consider breaking apart a service?"

service functionality



code volatility

scalability and throughput

fault tolerance

service granularity

granularity drivers

"when should I consider breaking apart a service?"

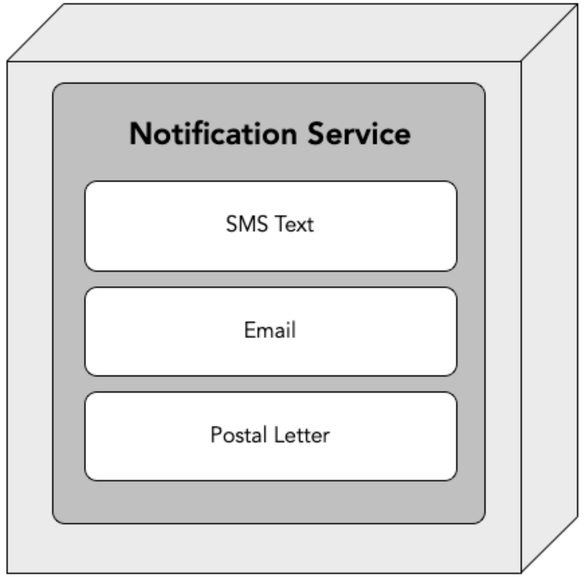
fault
tolerance

service granularity

granularity drivers

"when should I consider breaking apart a service?"

fault tolerance

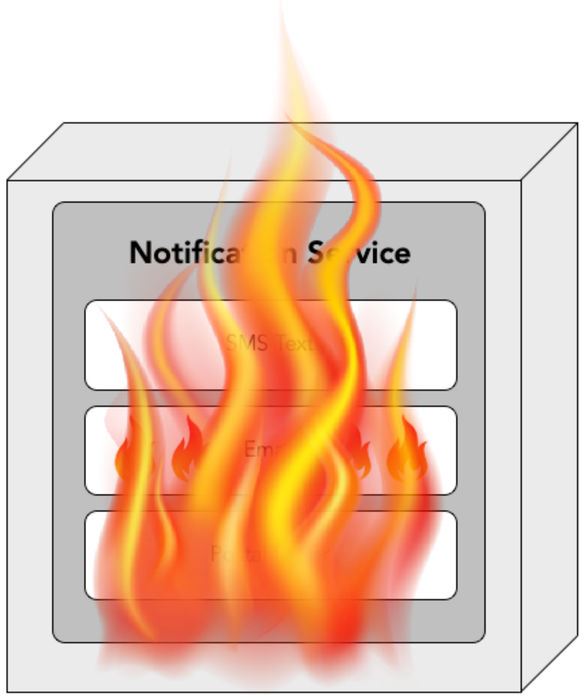


service granularity

granularity drivers

"when should I consider breaking apart a service?"

fault tolerance

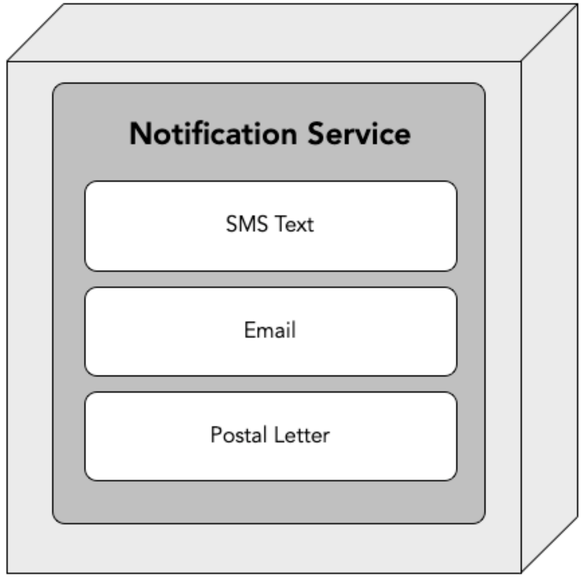


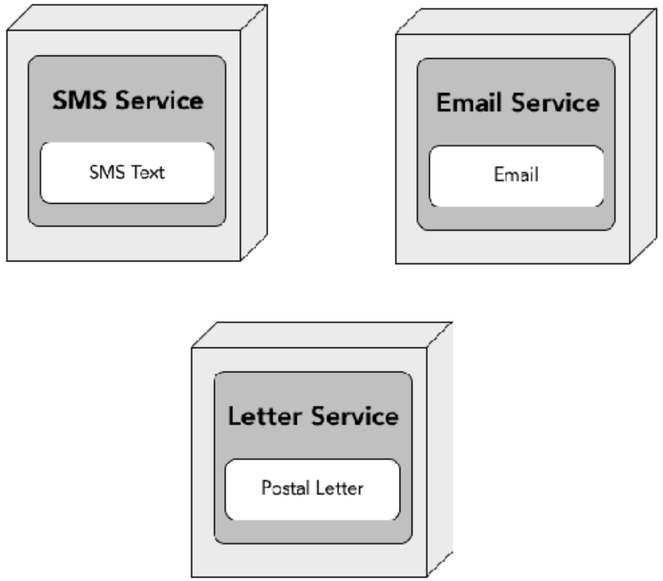
service granularity

granularity drivers

"when should I consider breaking apart a service?"

fault tolerance



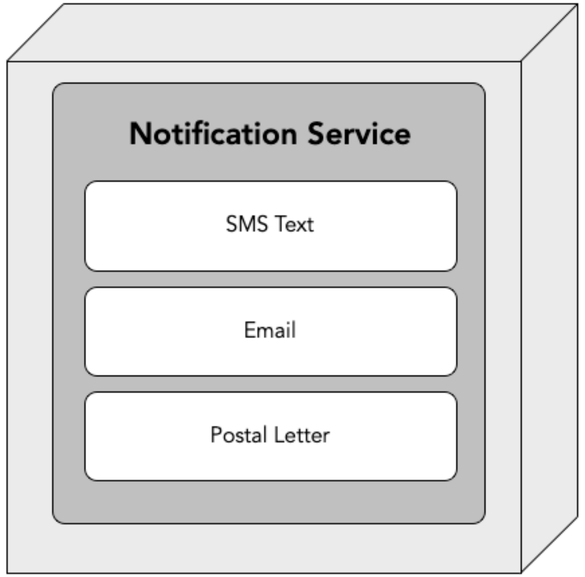


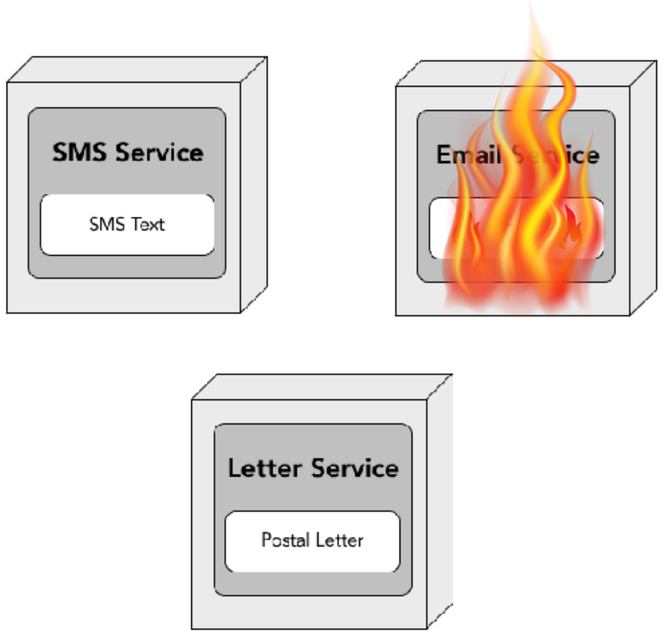
service granularity

granularity drivers

"when should I consider breaking apart a service?"

fault tolerance



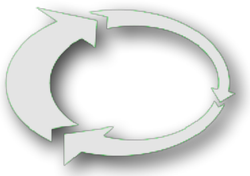


service granularity

granularity drivers

"when should I consider breaking apart a service?"

service functionality



code volatility

scalability and throughput

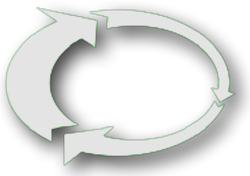
fault tolerance

service granularity

granularity drivers

"when should I consider breaking apart a service?"

service functionality



code volatility

scalability and throughput

fault tolerance

data security

service granularity

granularity drivers

"when should I consider breaking apart a service?"

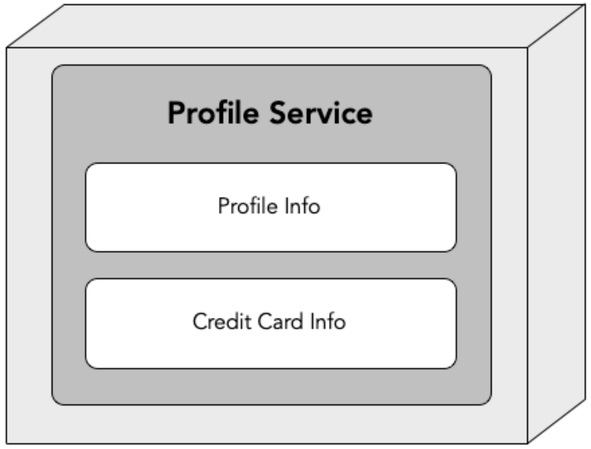
data
security

service granularity

granularity drivers

"when should I consider breaking apart a service?"

data security

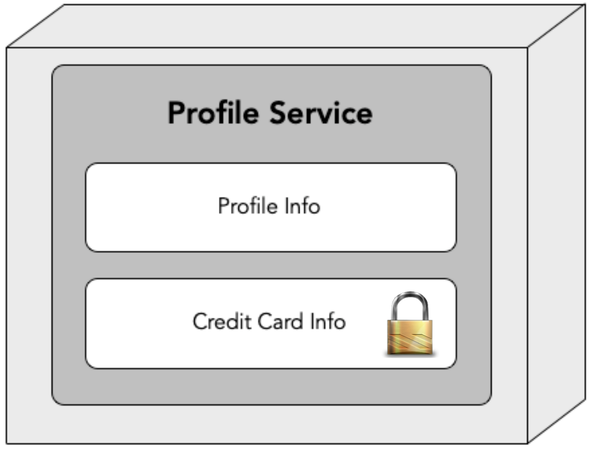


service granularity

granularity drivers

"when should I consider breaking apart a service?"

data security

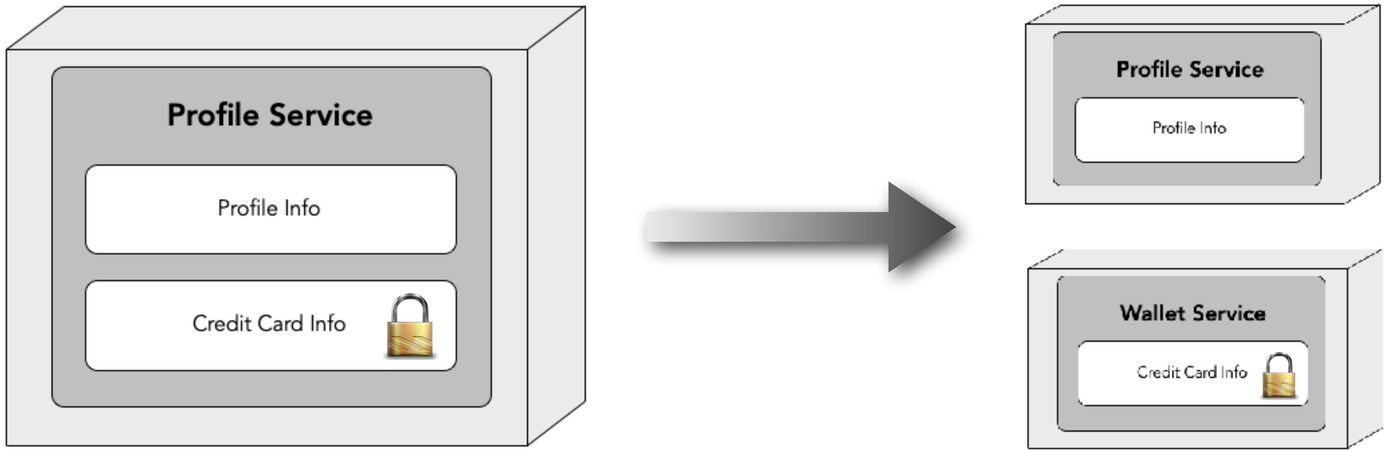


service granularity

granularity drivers

"when should I consider breaking apart a service?"

data security

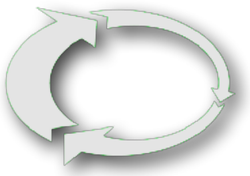


service granularity

granularity drivers

"when should I consider breaking apart a service?"

service functionality



code volatility

scalability and throughput

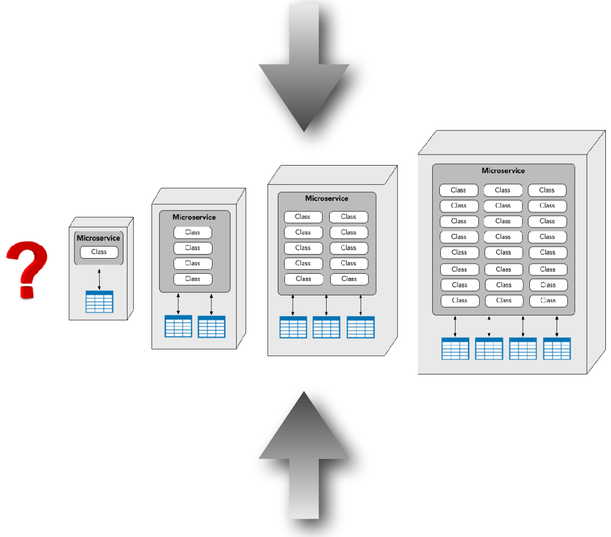
fault tolerance

data security

service granularity

granularity drivers

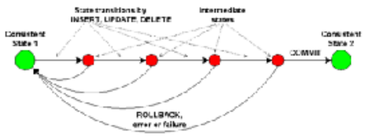
"when should I consider breaking apart a service?"



granularity factors

"what factors influence service granularity?"

service granularity

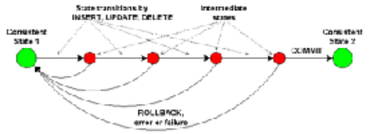


database
transactions

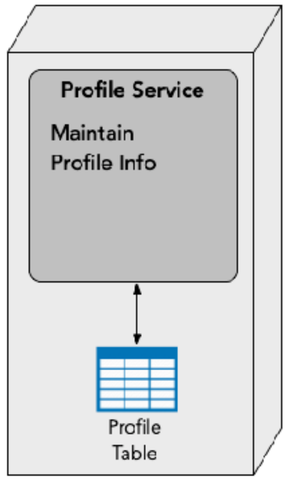
granularity factors

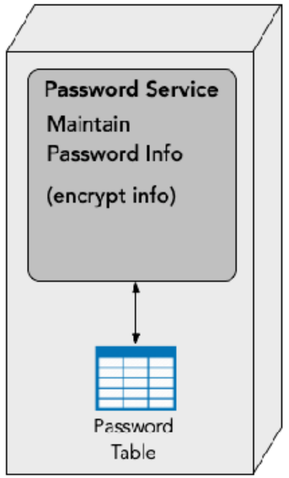
"what factors influence service granularity?"

service granularity

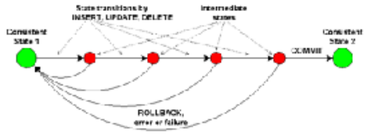


database
transactions

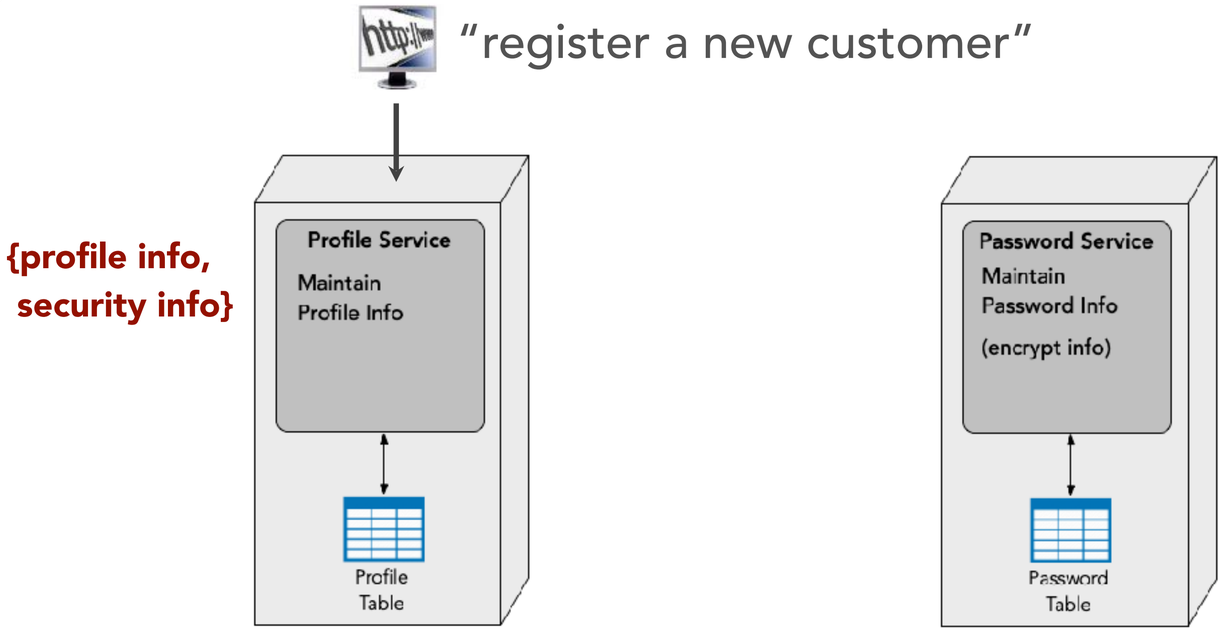




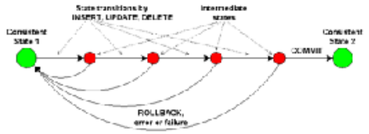
service granularity



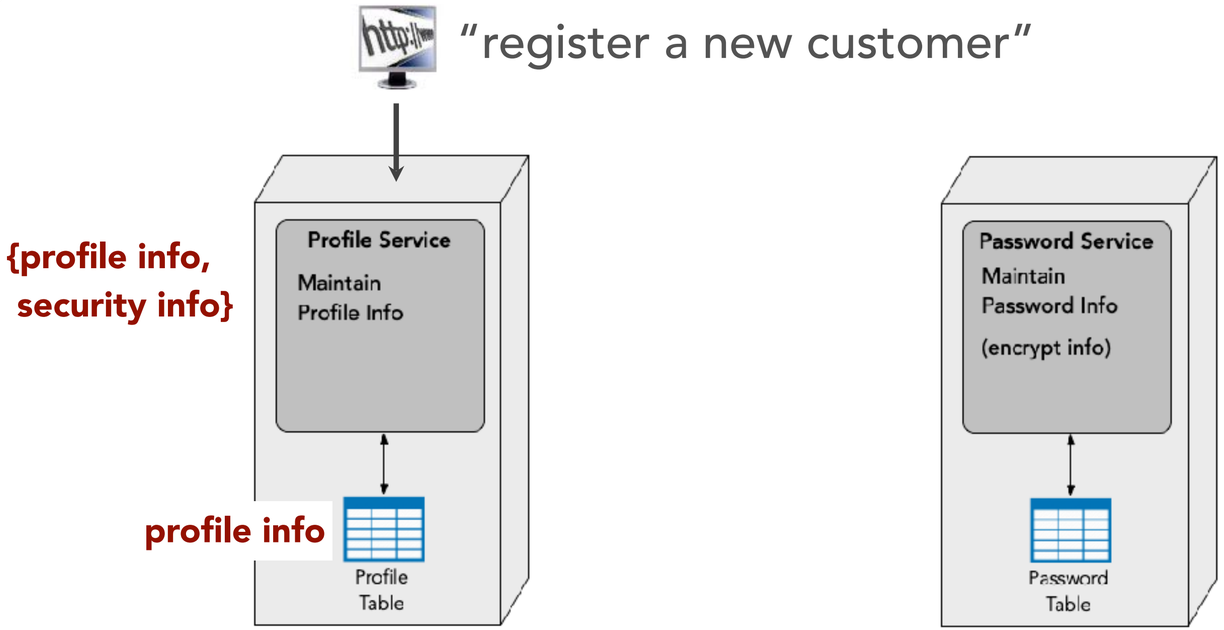
database
transactions



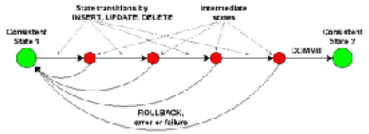
service granularity



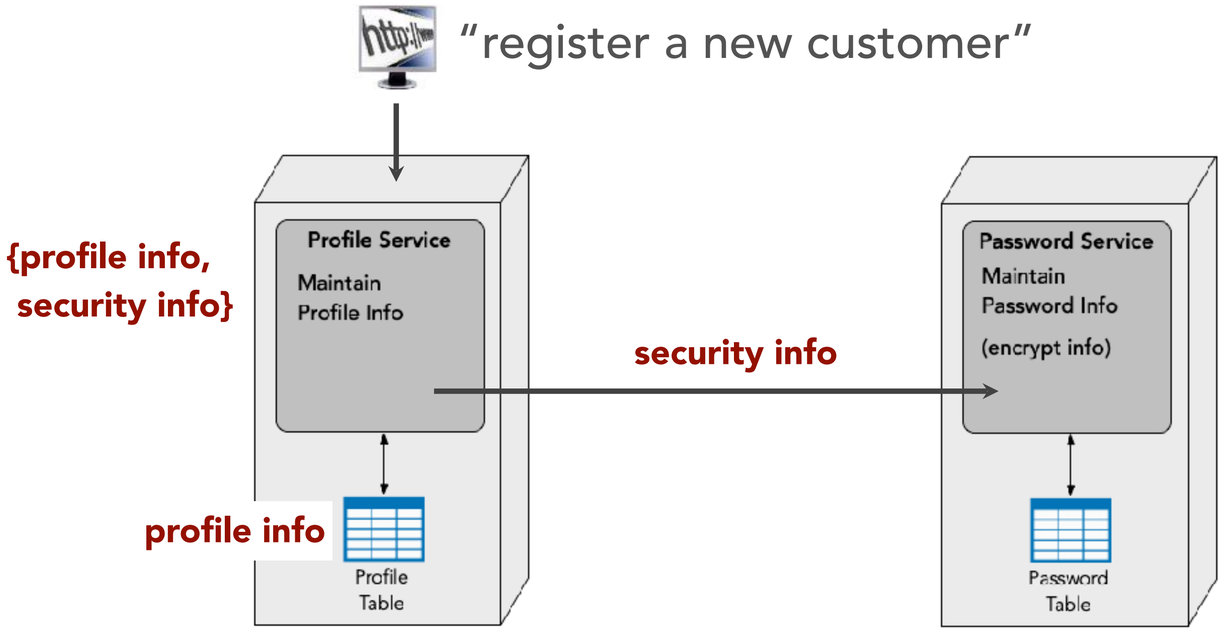
database
transactions



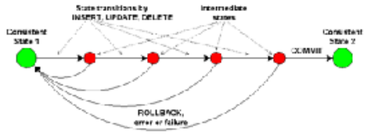
service granularity



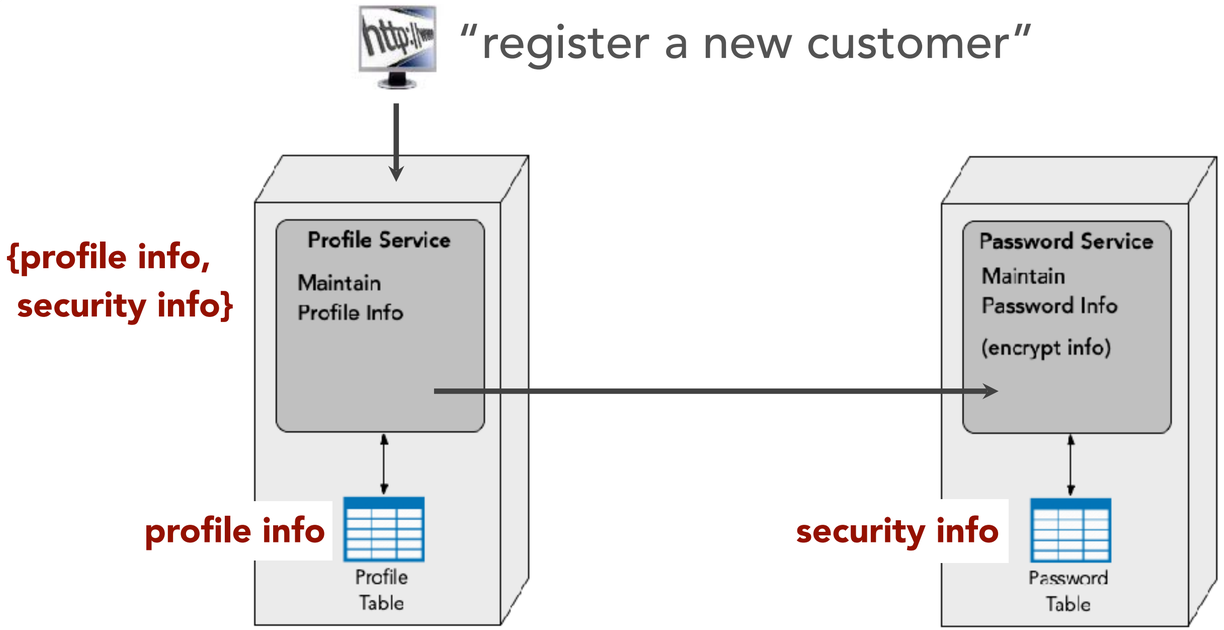
database
transactions



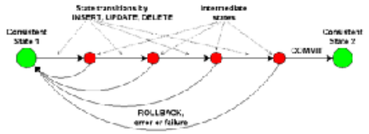
service granularity



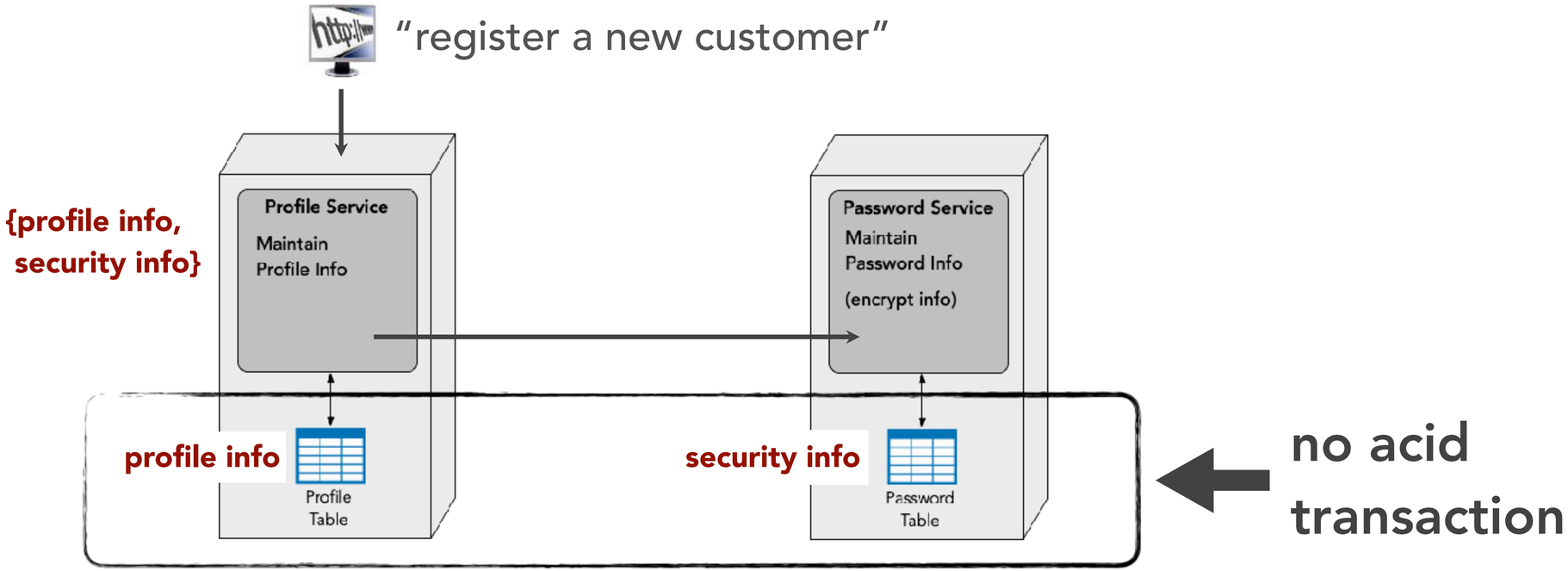
database
transactions



service granularity

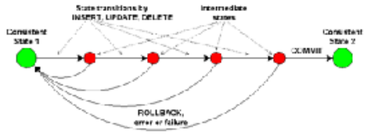


database
transactions

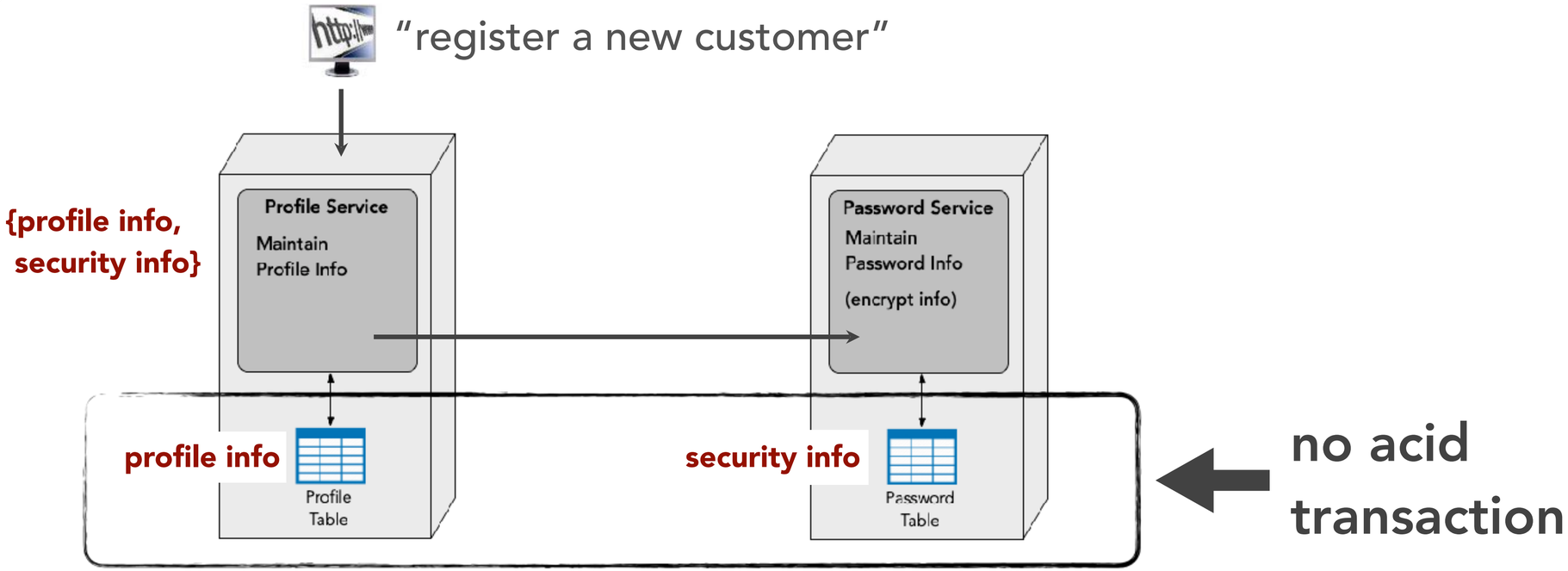


no acid
transaction

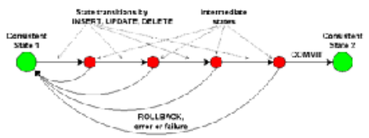
service granularity



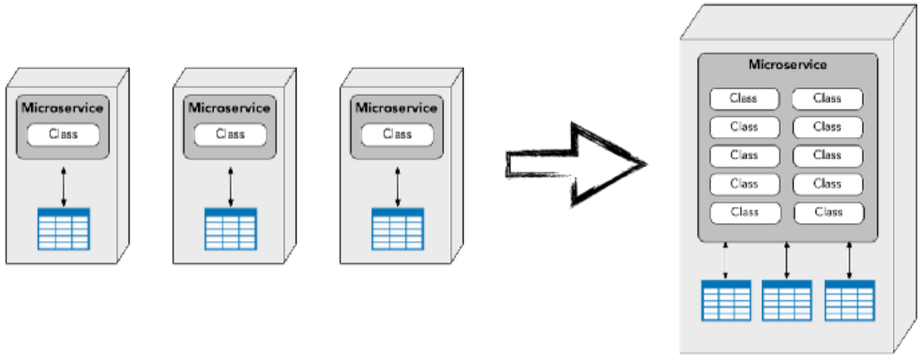
database transactions

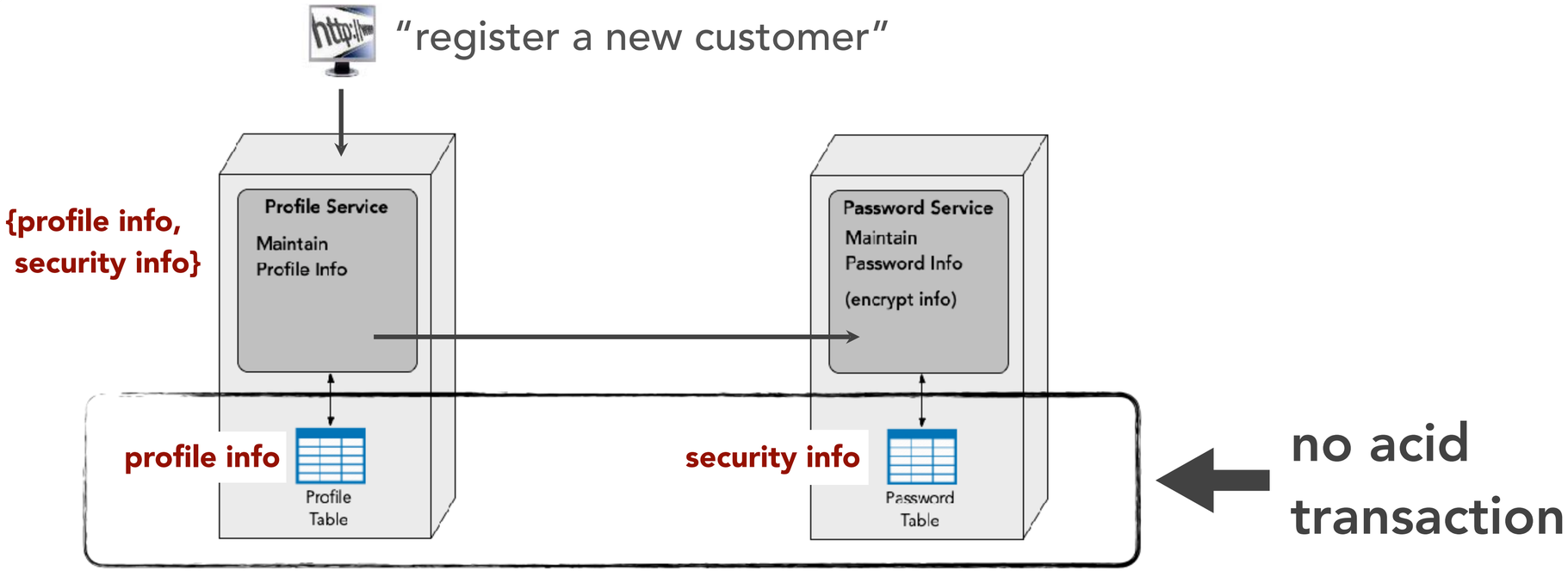


service granularity

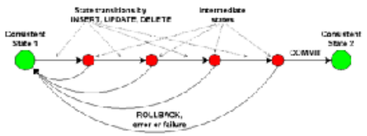


database
transactions





service granularity

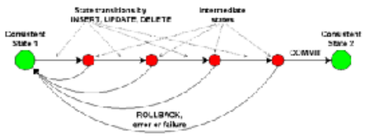


database
transactions

granularity factors

"what factors influence service granularity?"

service granularity



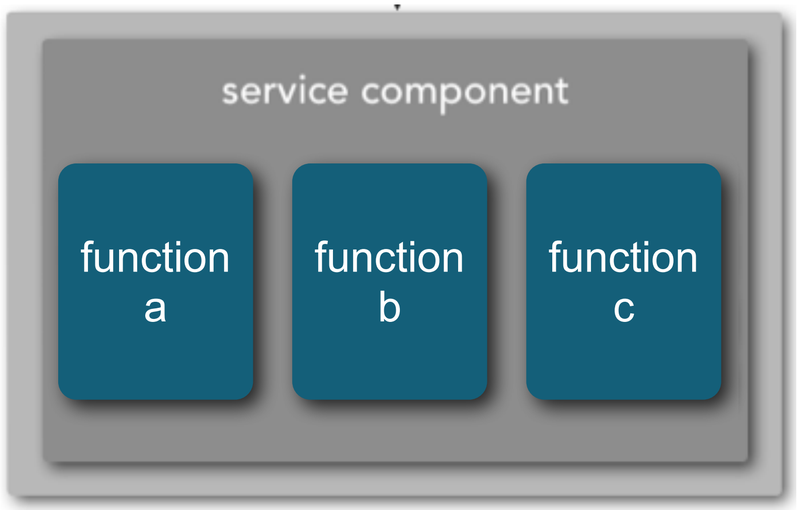
database
transactions

data
dependencies

granularity factors

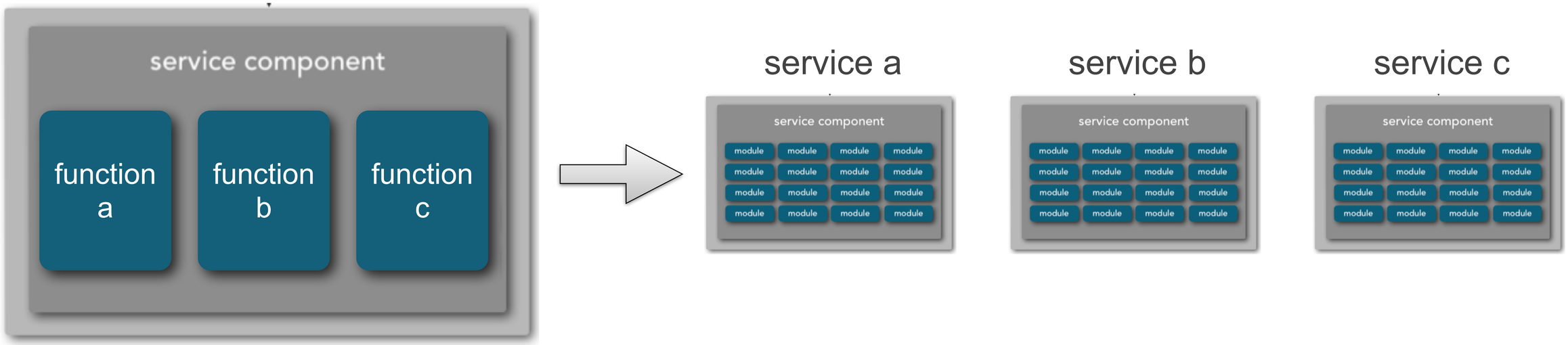
"what factors influence service granularity?"

service granularity



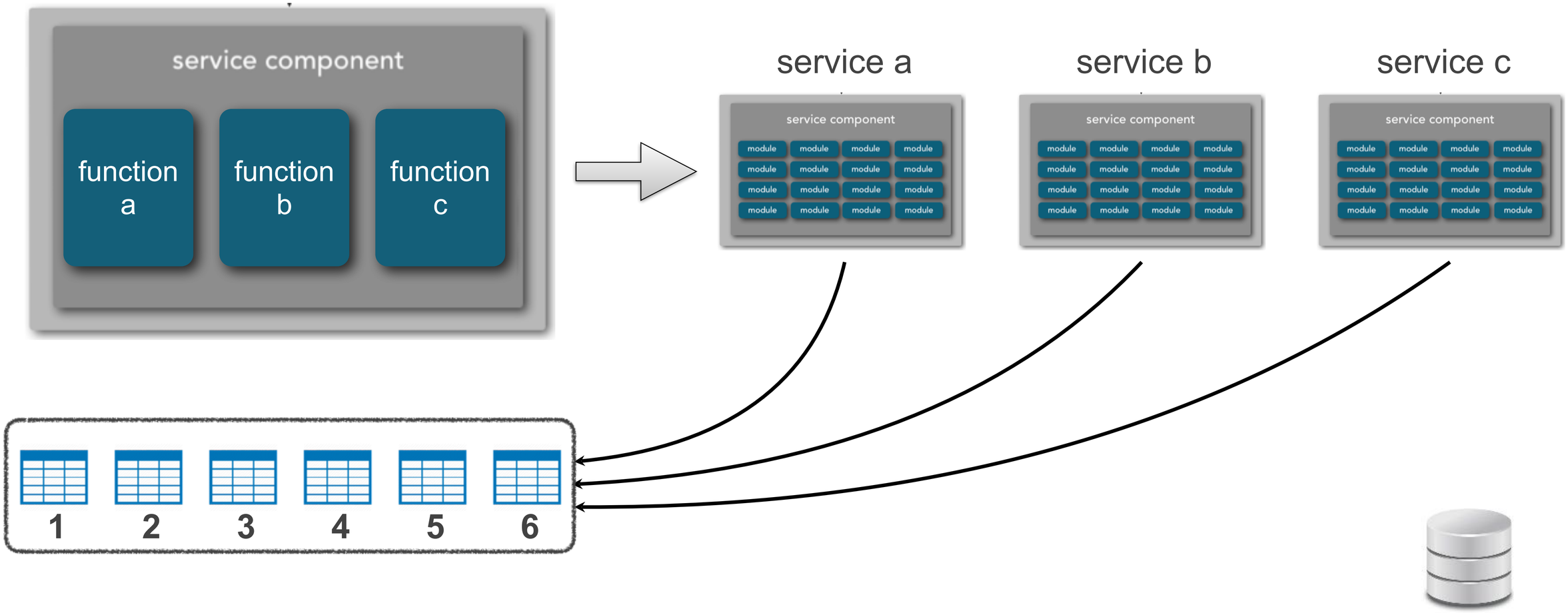
data dependencies

service granularity



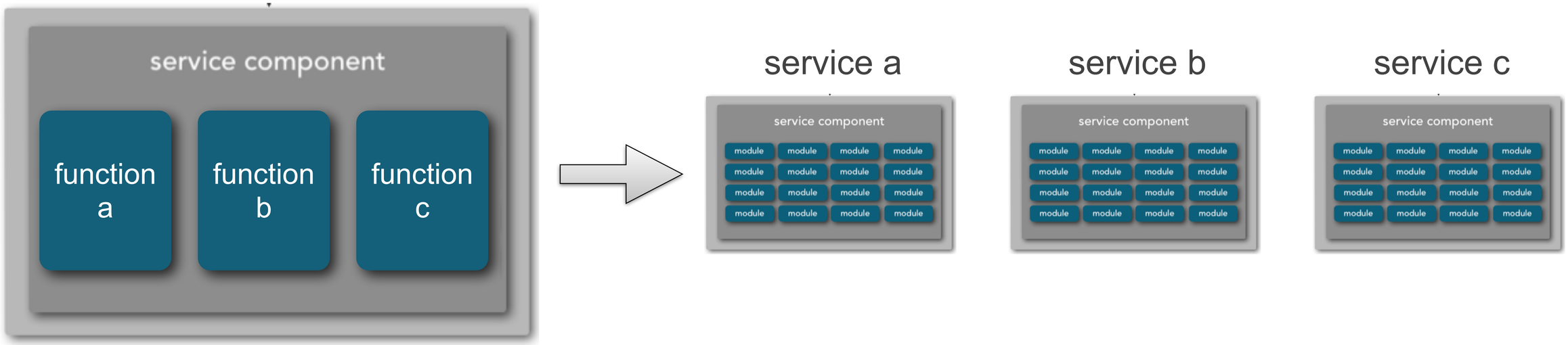
data dependencies

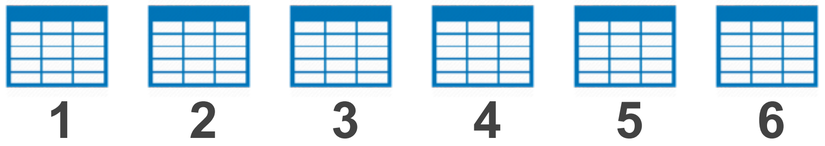
service granularity



data dependencies

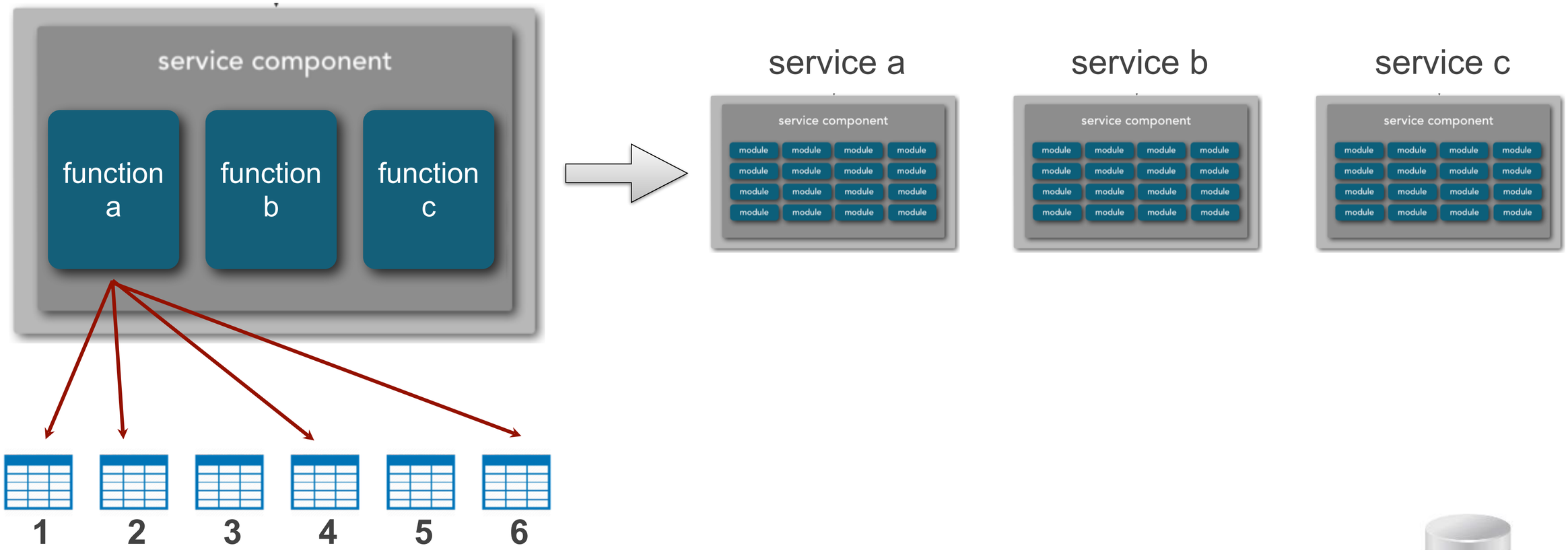
service granularity





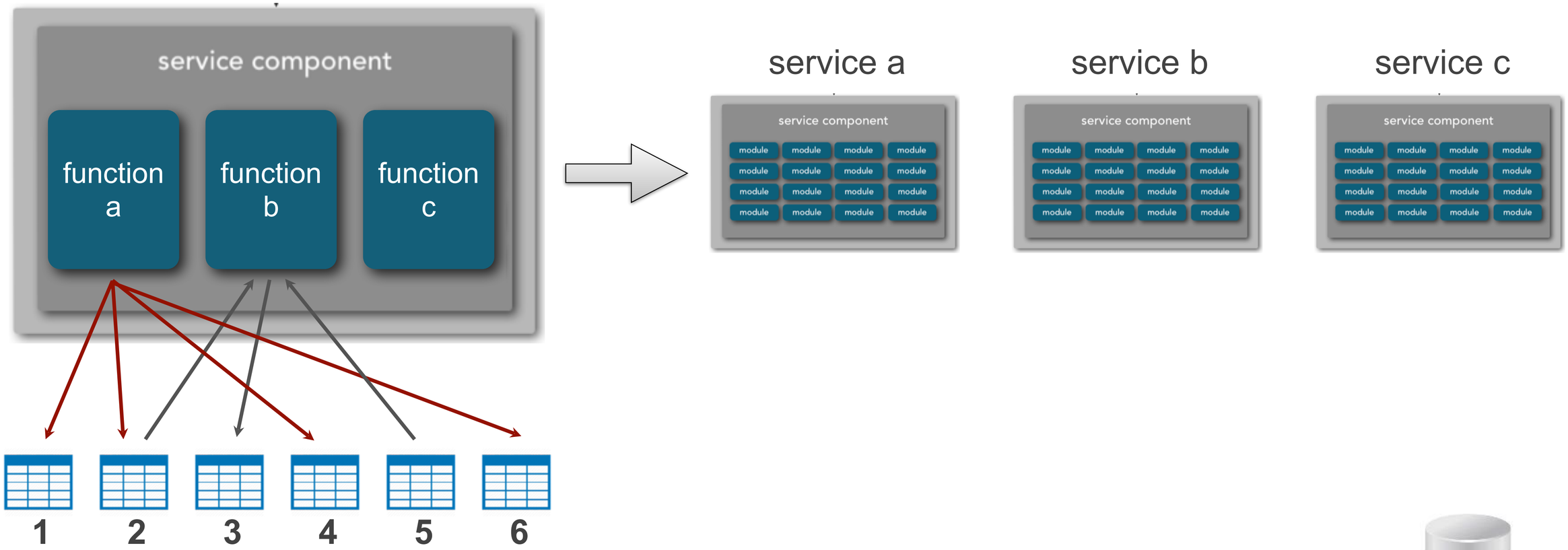
data dependencies

service granularity



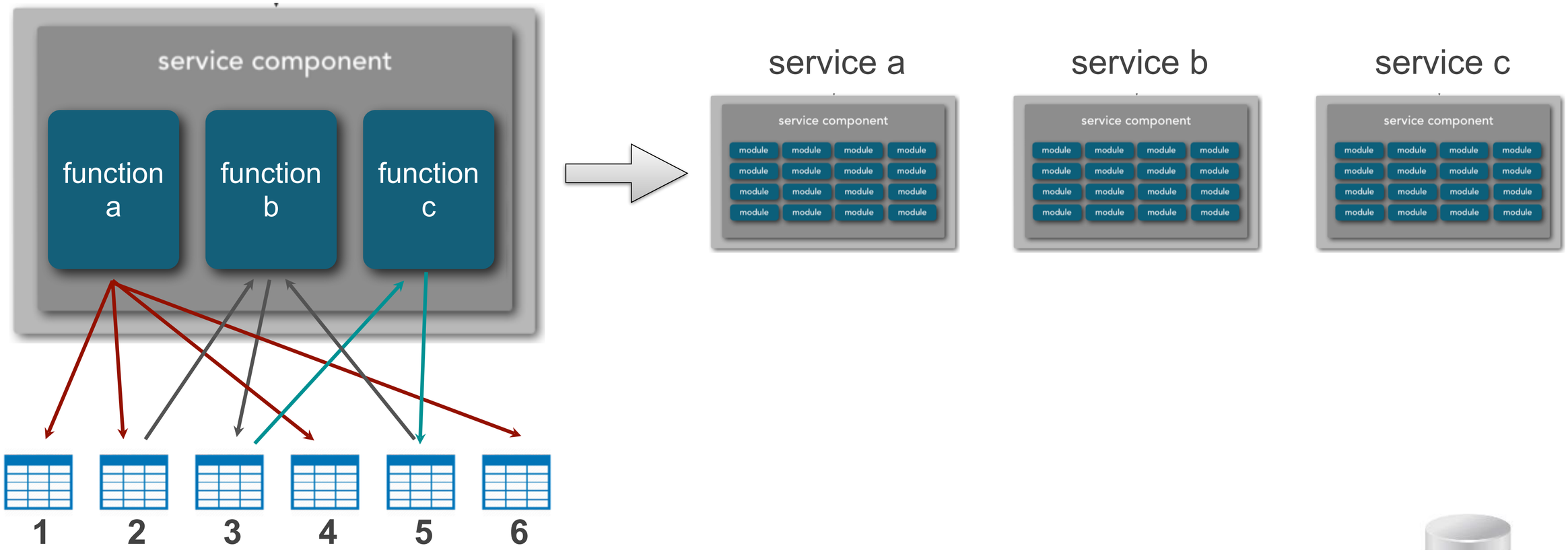
data dependencies

service granularity



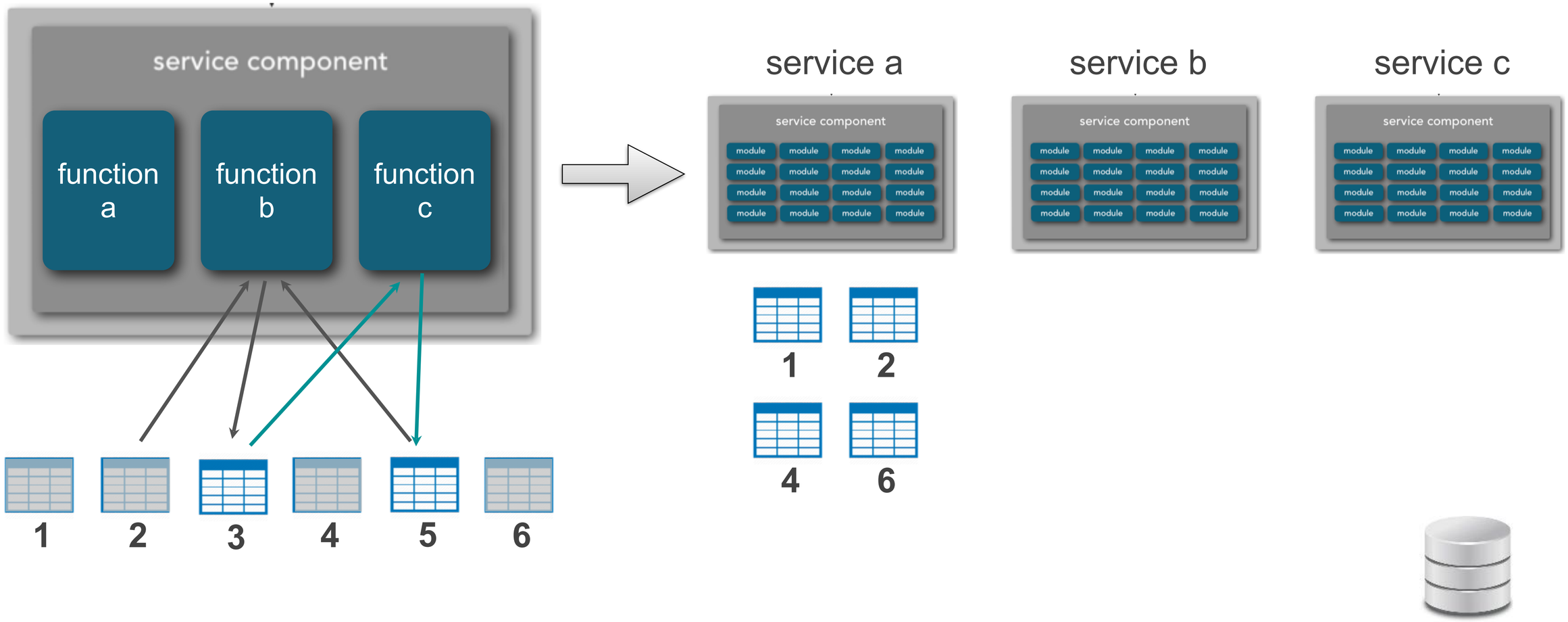
data dependencies

service granularity



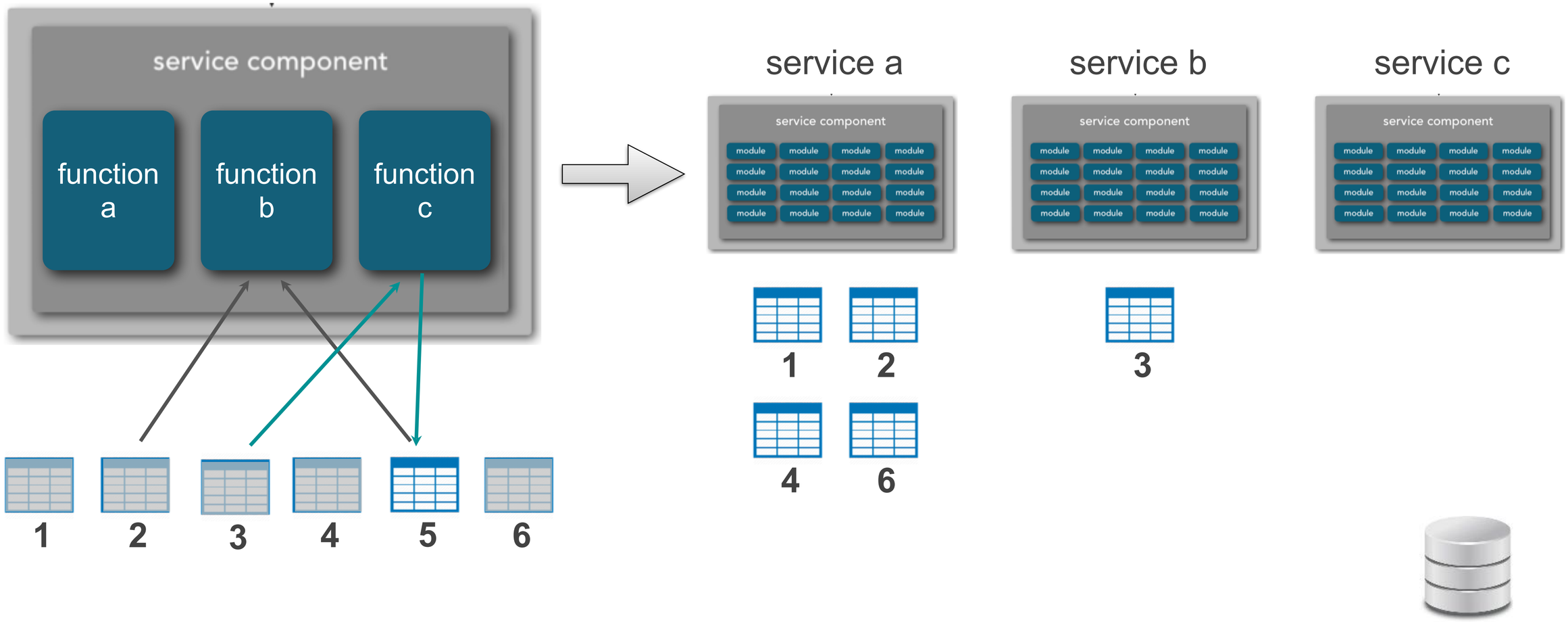
data dependencies

service granularity



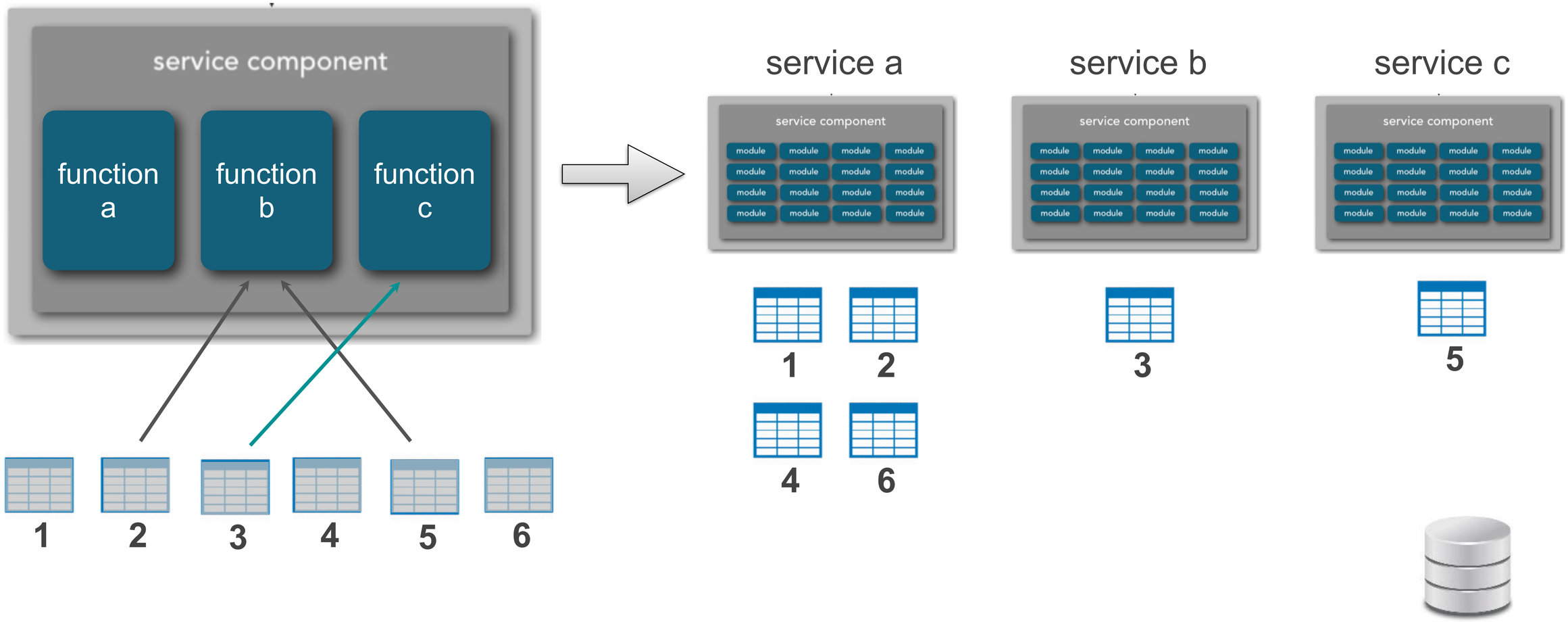
data dependencies

service granularity



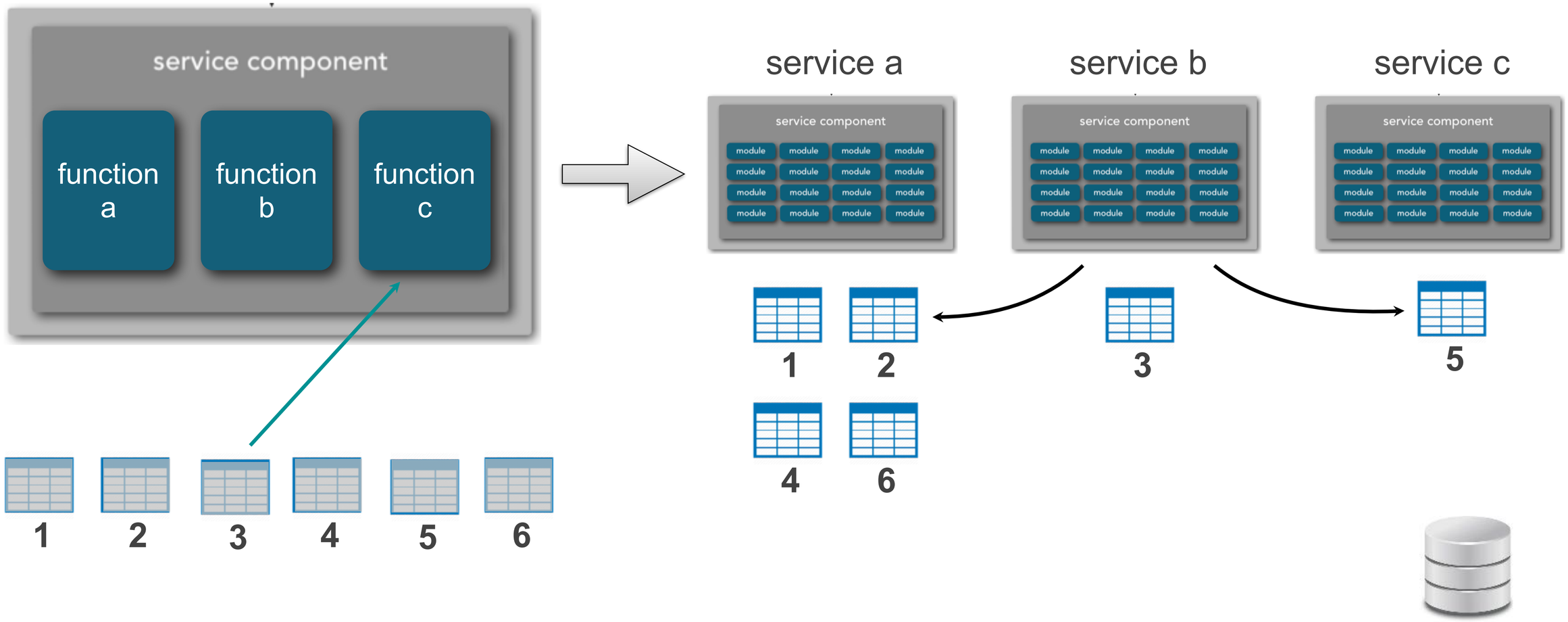
data dependencies

service granularity



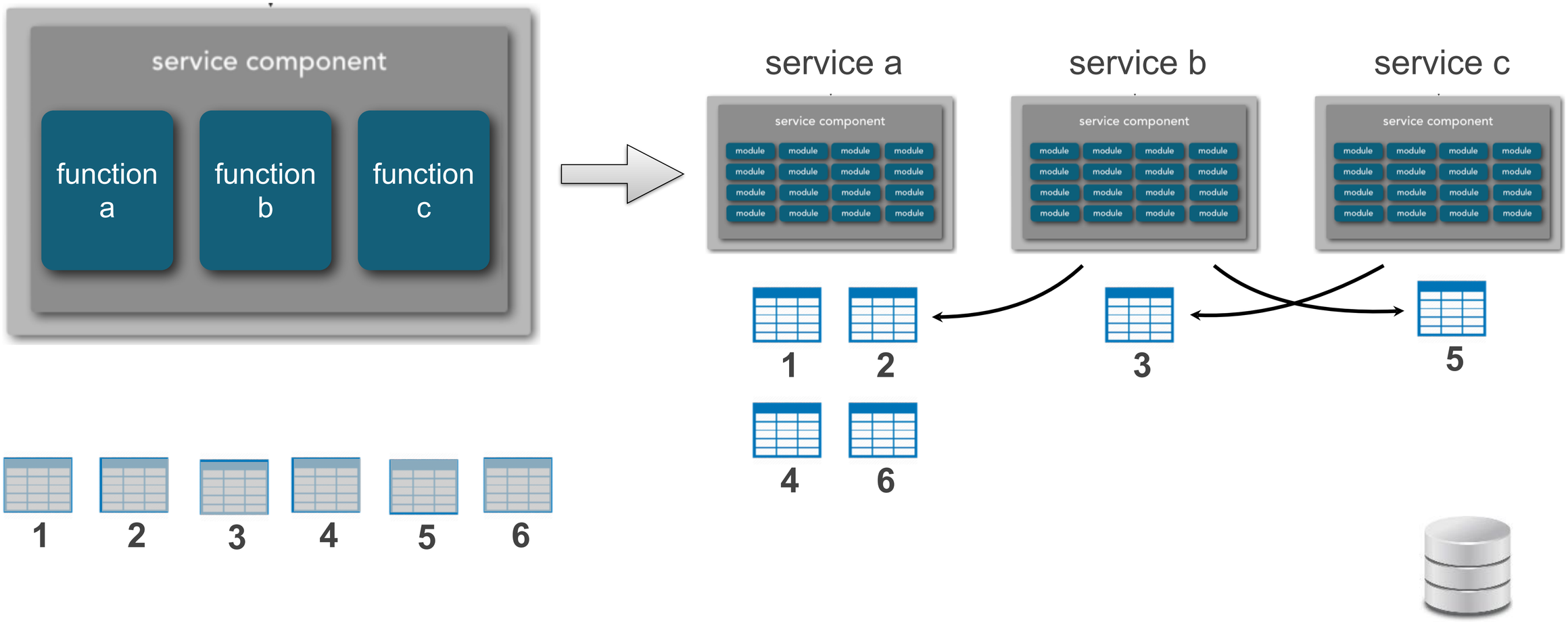
data dependencies

service granularity



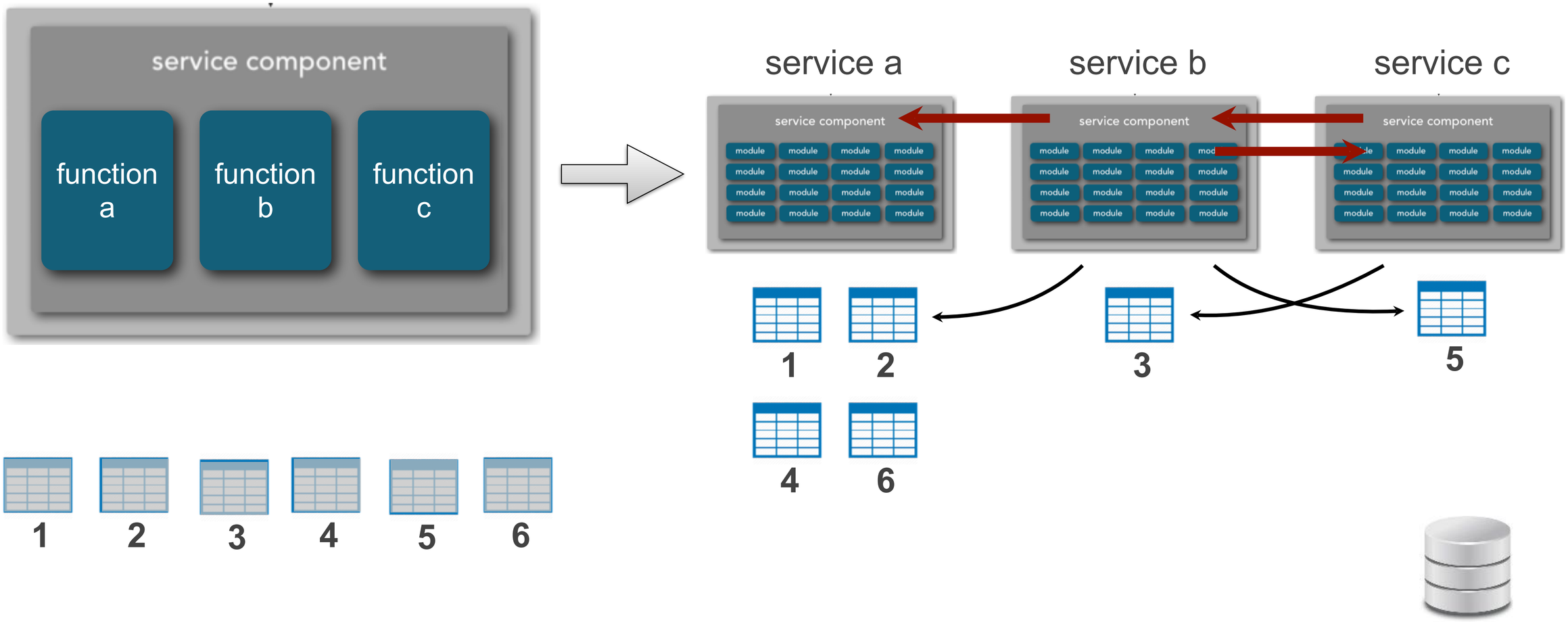
data dependencies

service granularity



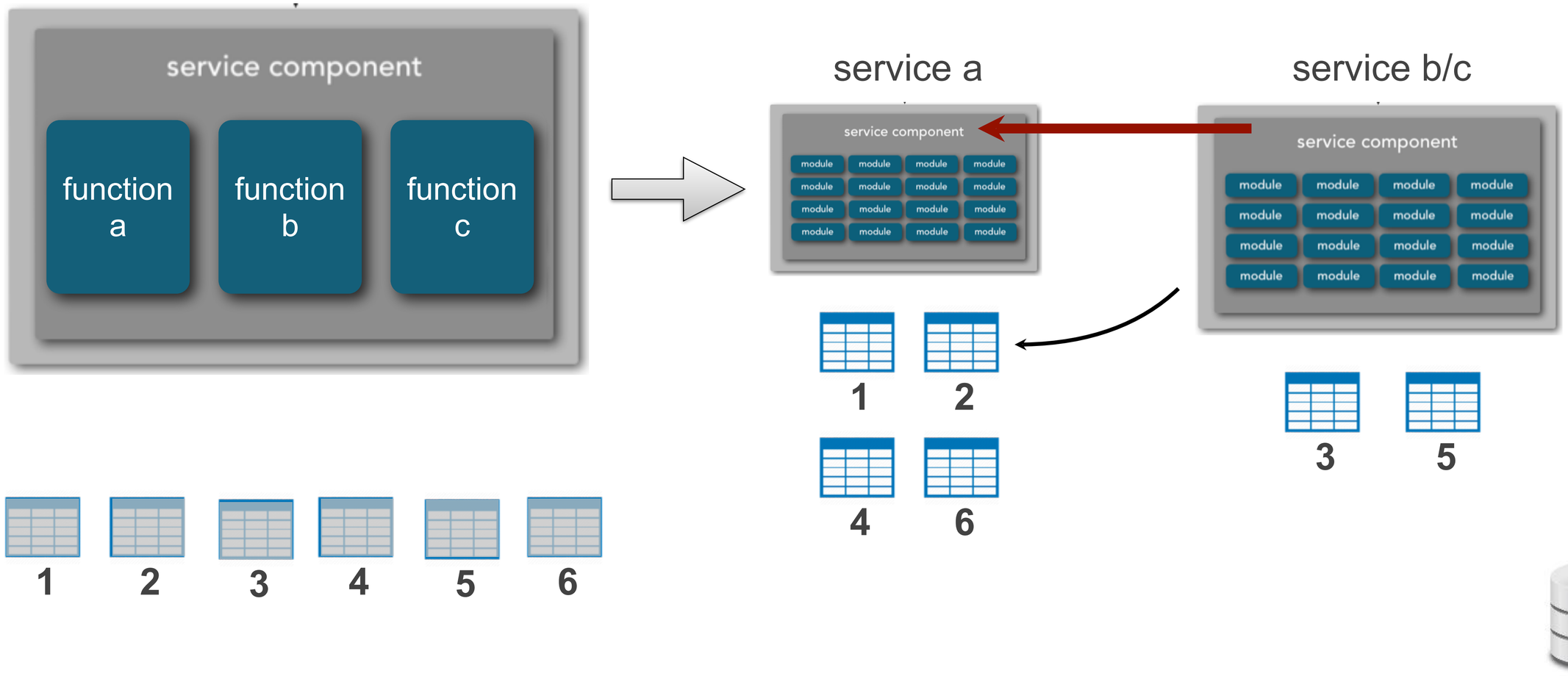
data dependencies

service granularity



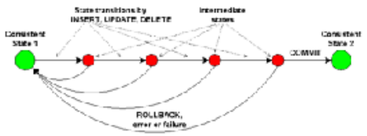
data dependencies

service granularity



data dependencies

service granularity



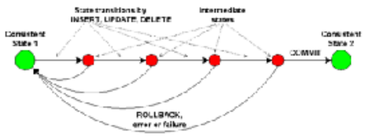
database
transactions

data
dependencies

granularity factors

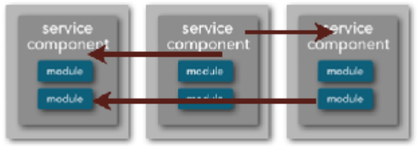
"what factors influence service granularity?"

service granularity



database transactions

data dependencies



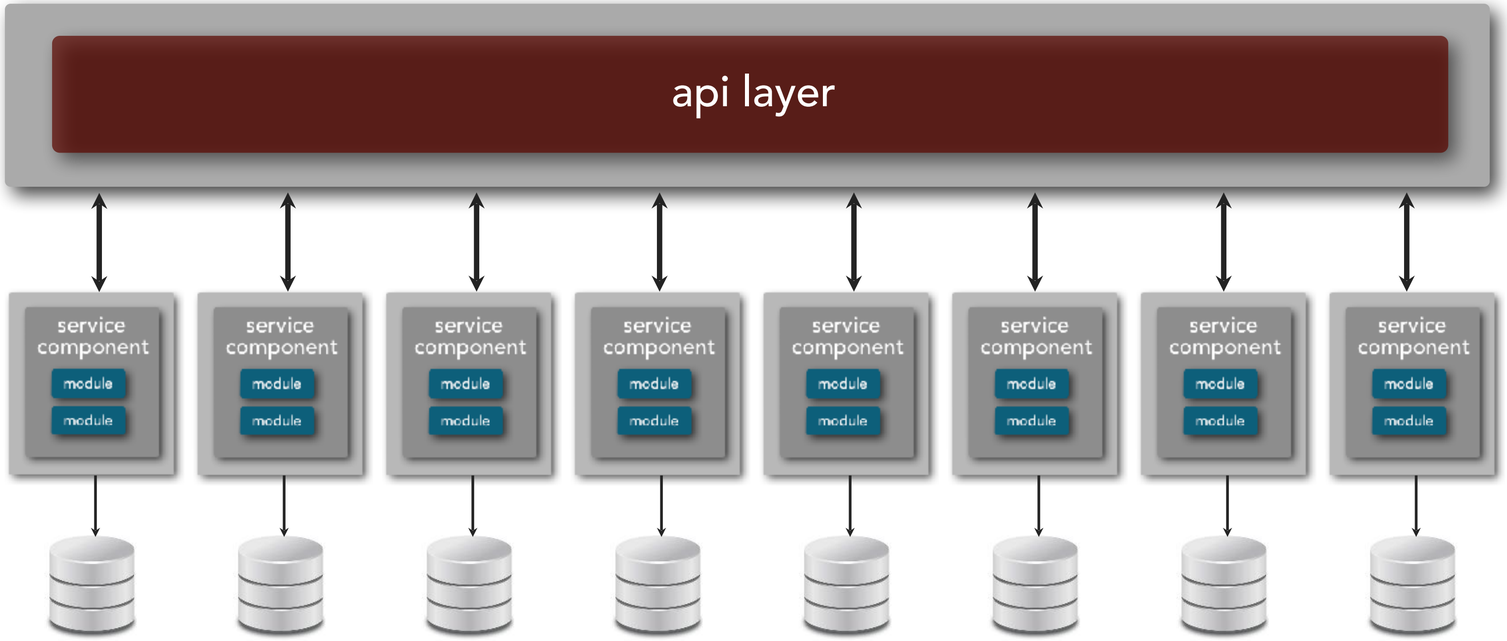
workflow and choreography

granularity factors

"what factors influence service granularity?"

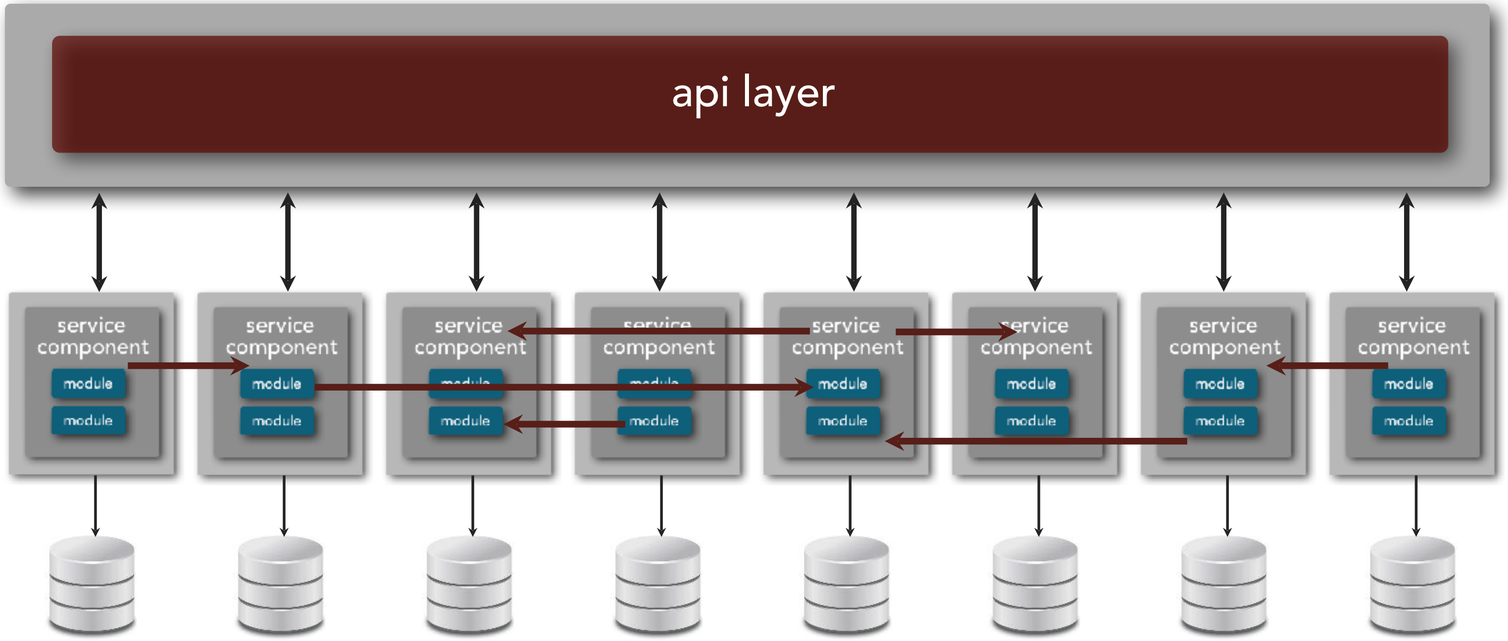
service granularity

workflow and choreography



service granularity

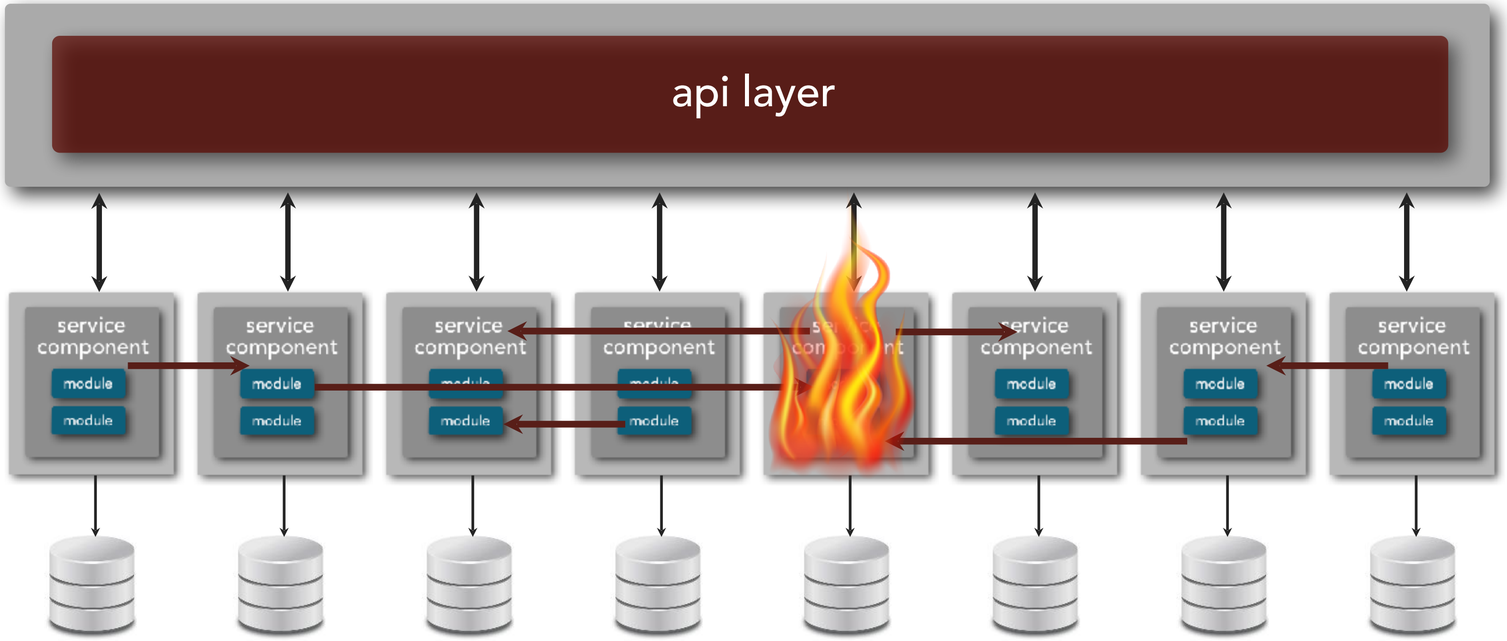
workflow and choreography



service granularity

fault tolerance (availability)

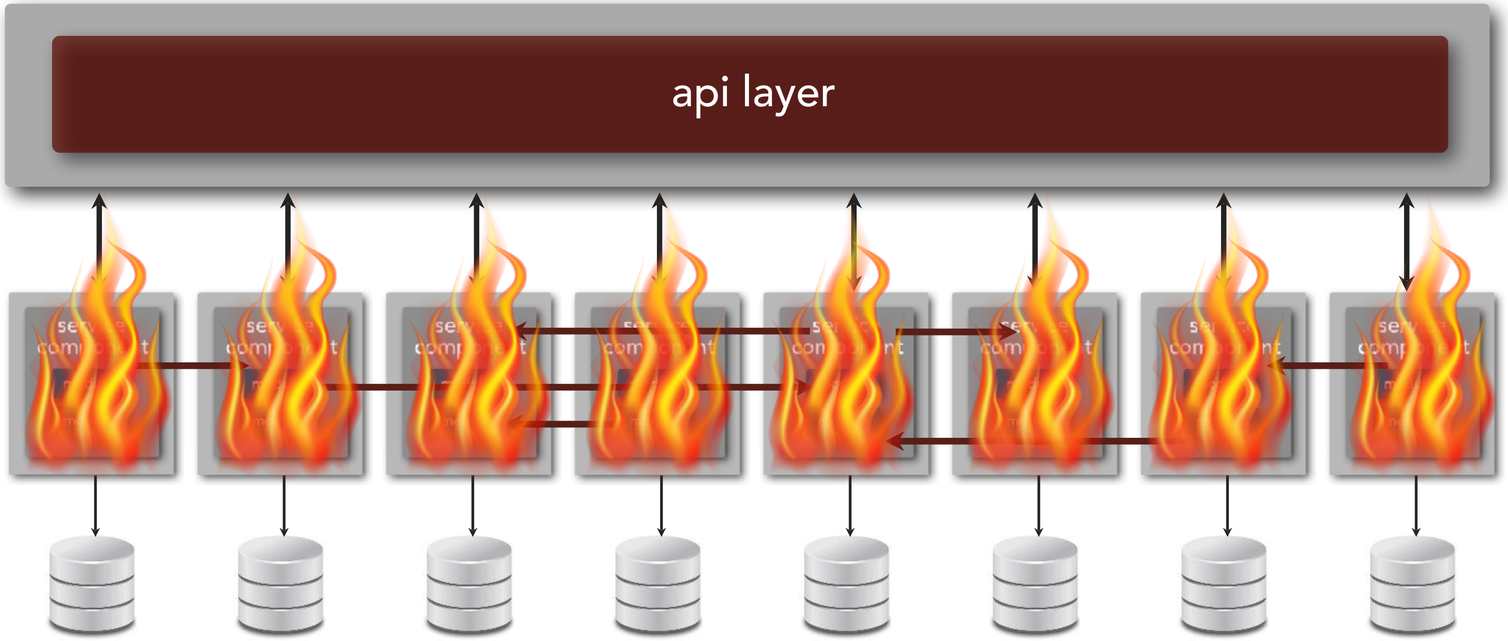
workflow and choreography



service granularity

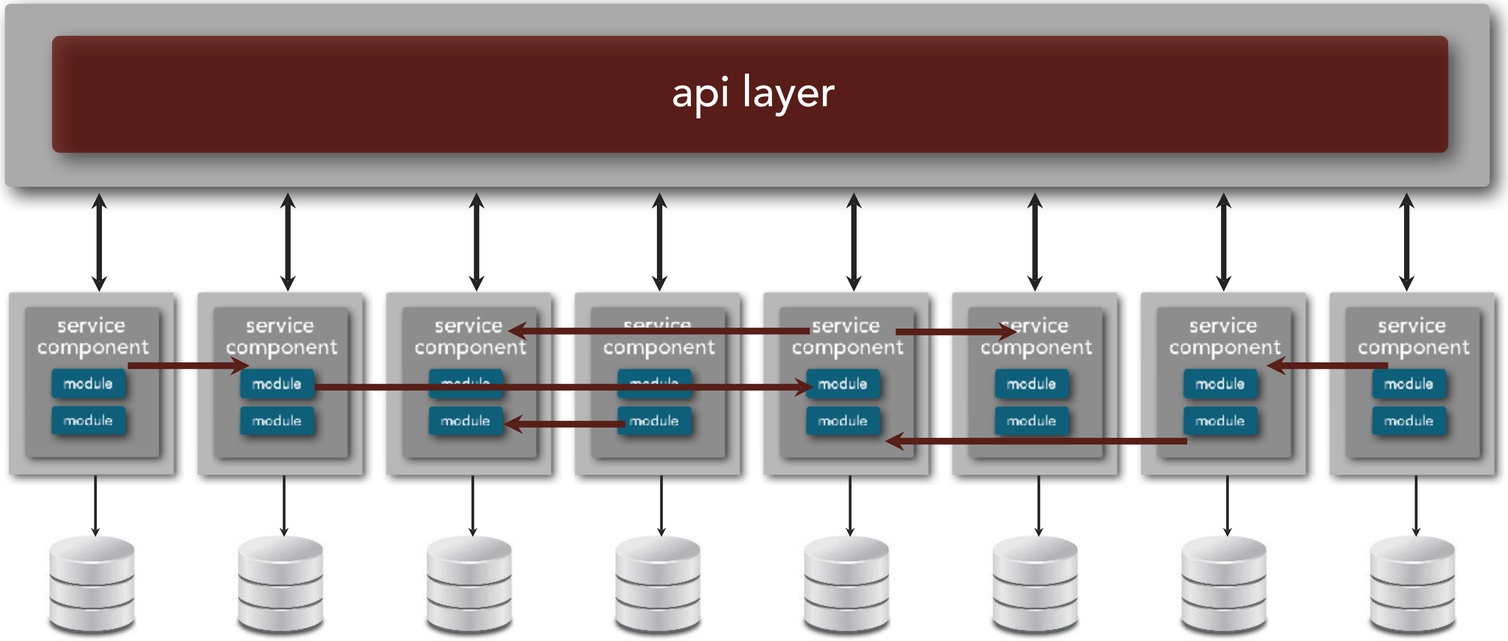
fault tolerance (availability)

workflow and choreography



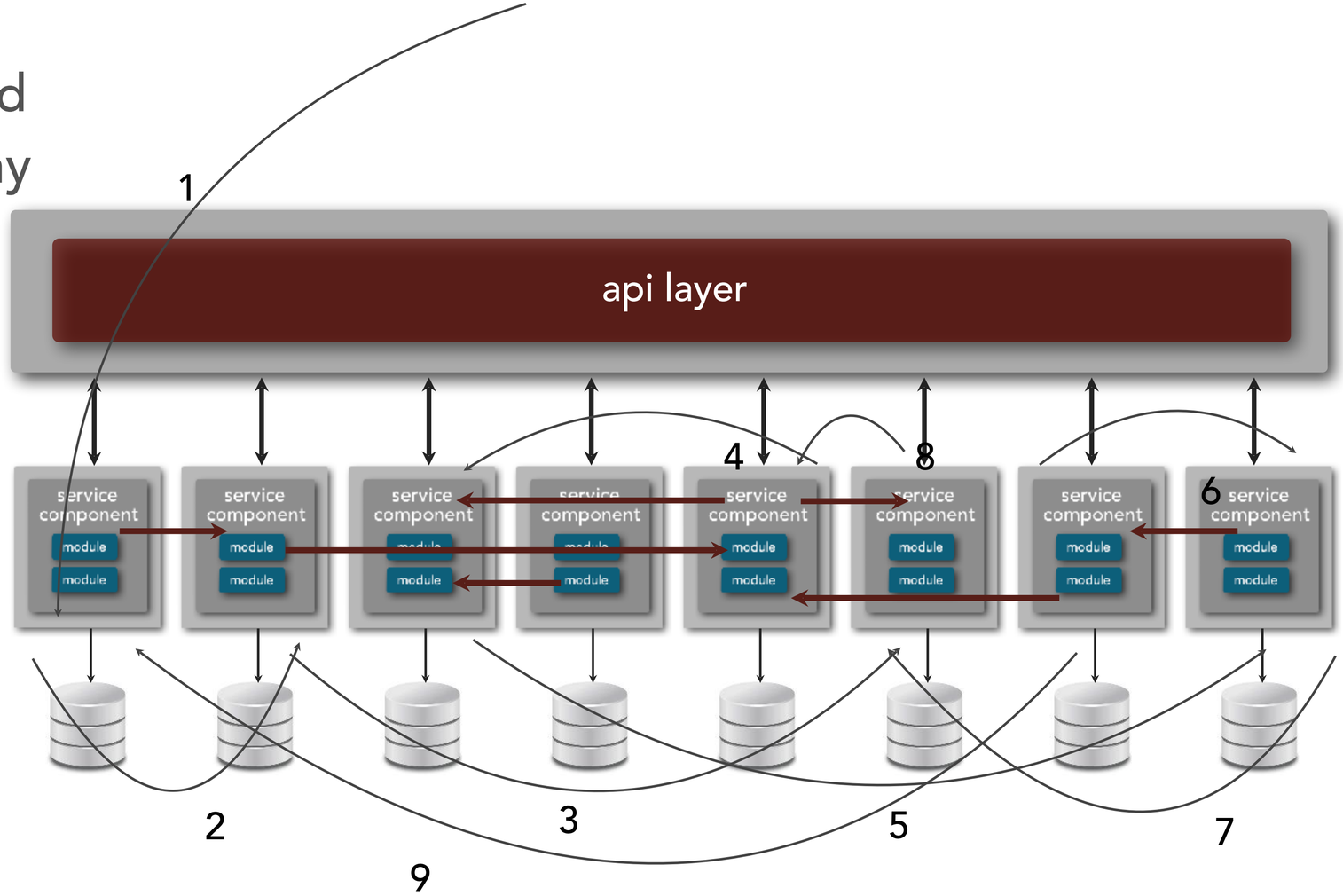
service granularity

workflow and choreography

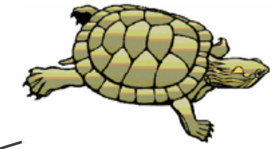


service granularity

workflow and choreography

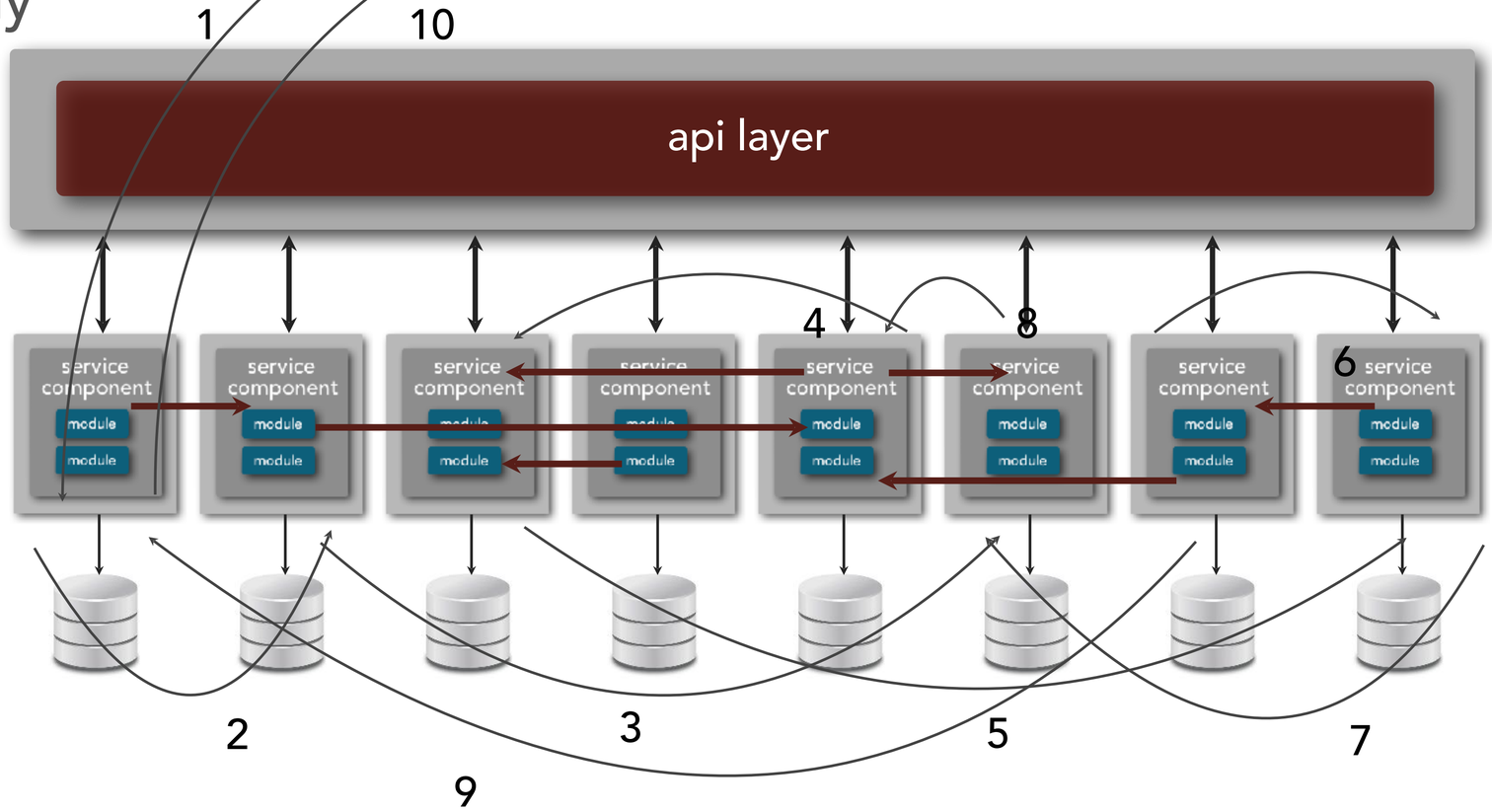


service granularity



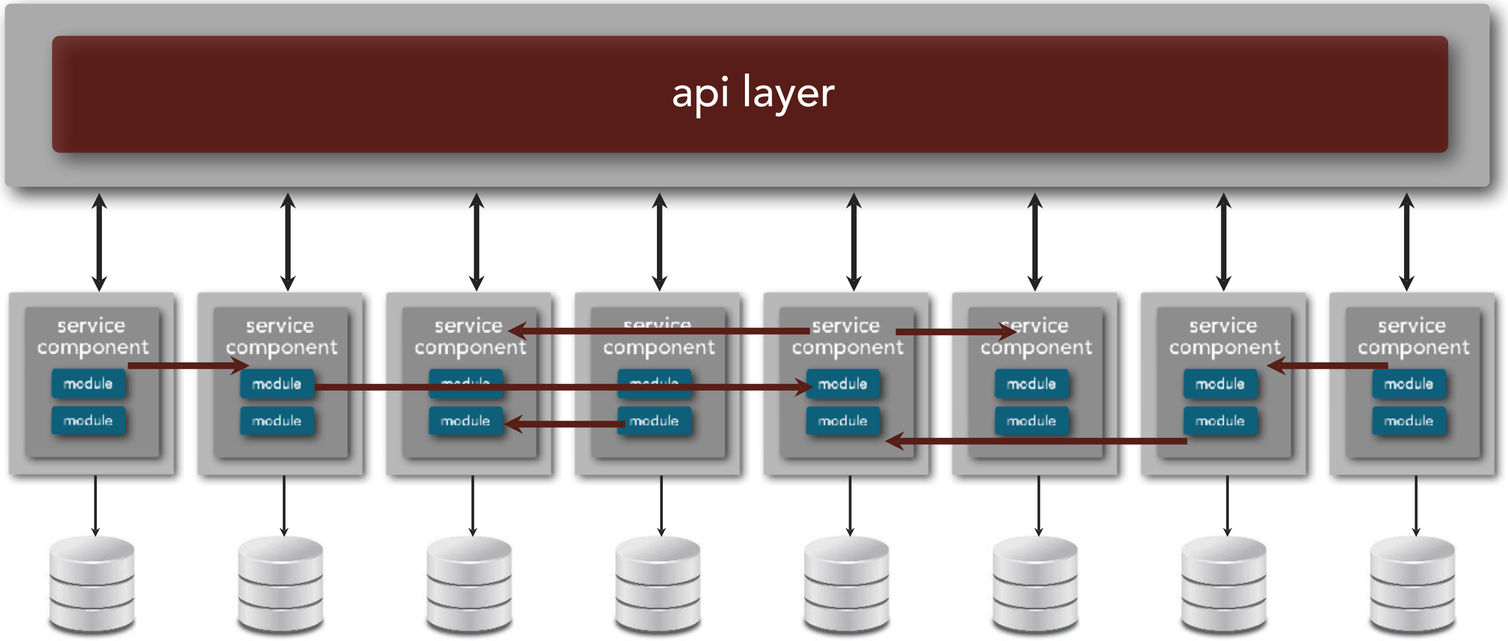
slow performance due to latency and security

workflow and choreography



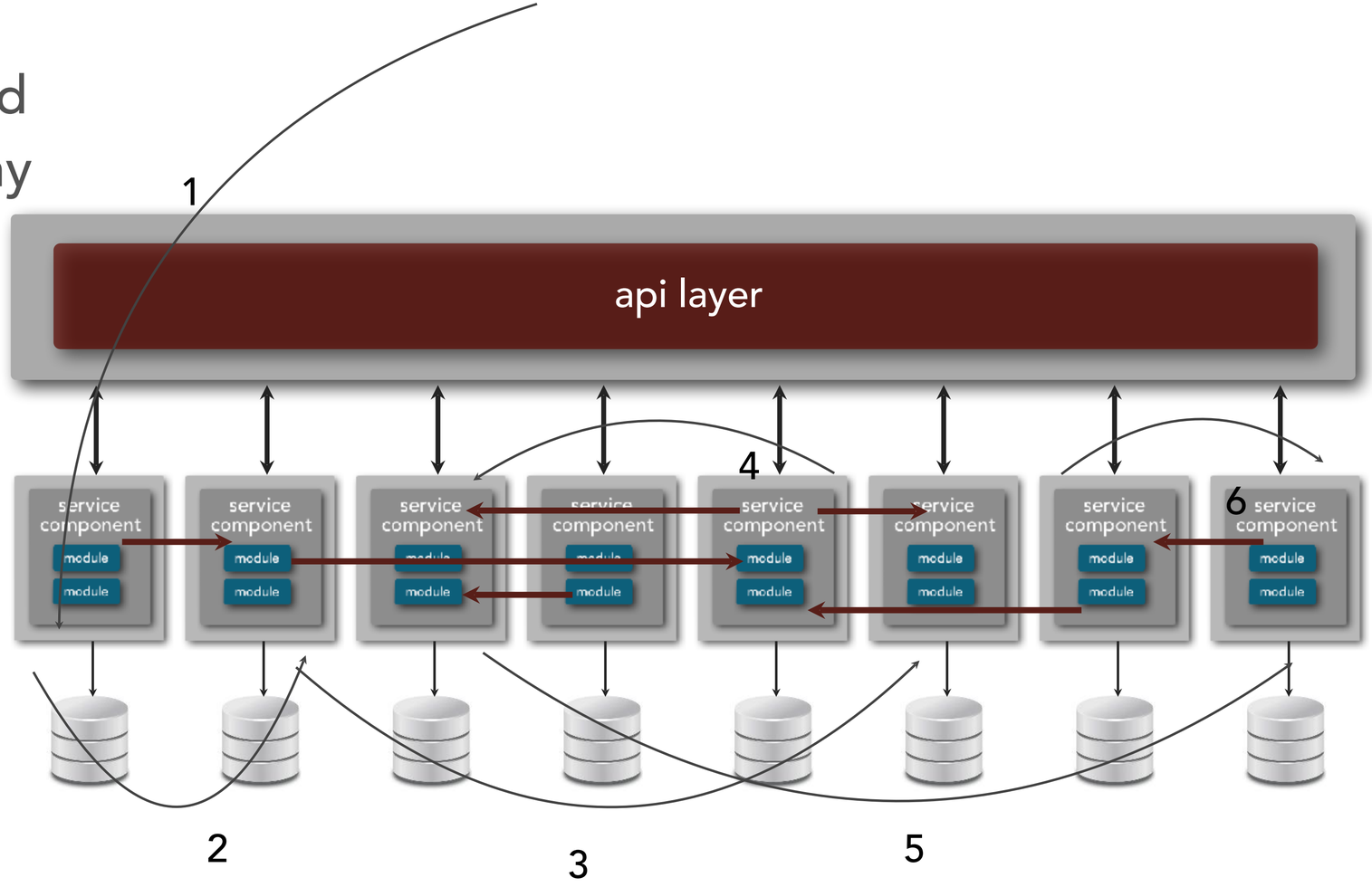
service granularity

workflow and choreography



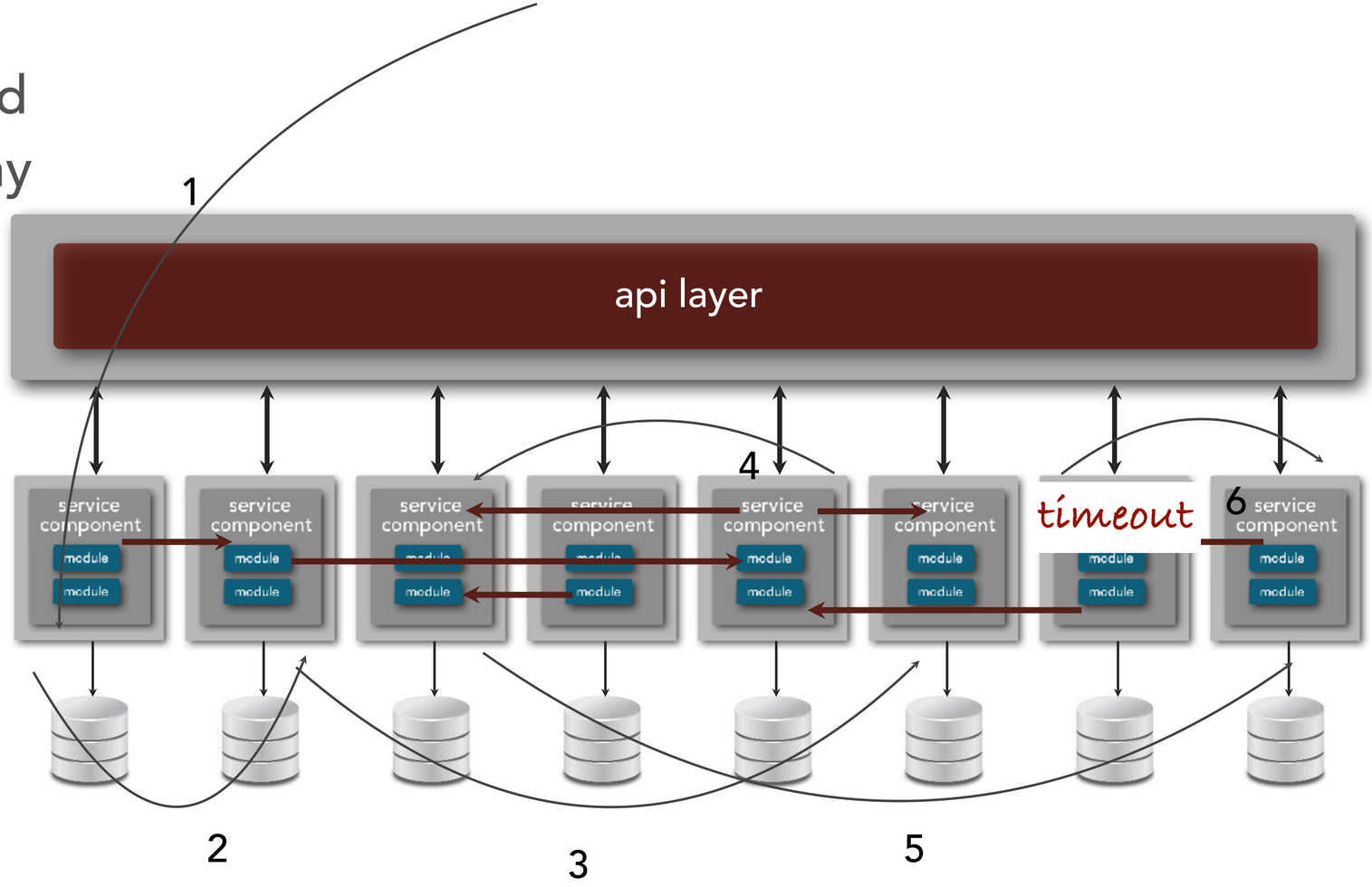
service granularity

workflow and choreography



service granularity

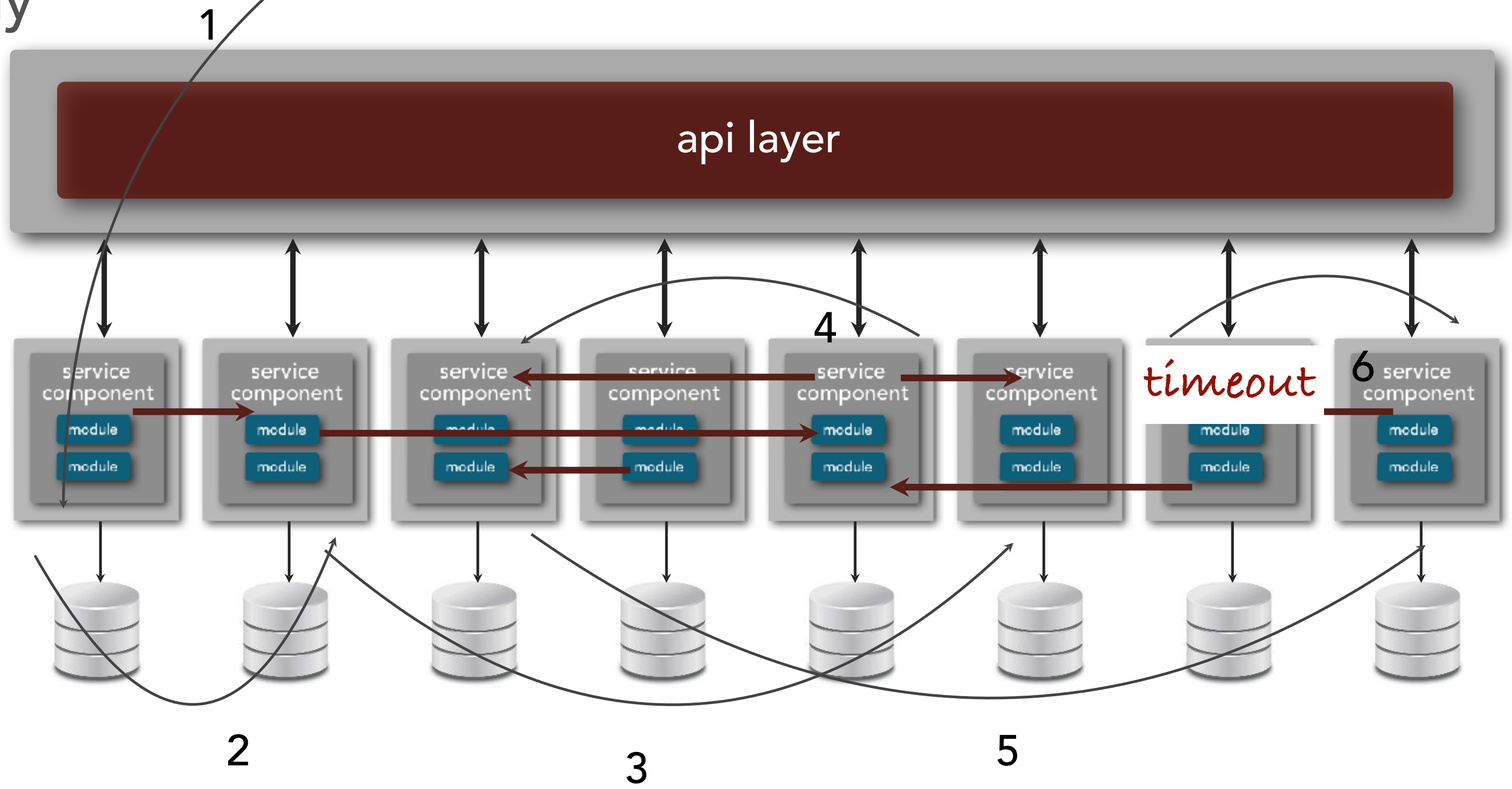
workflow and choreography



service granularity

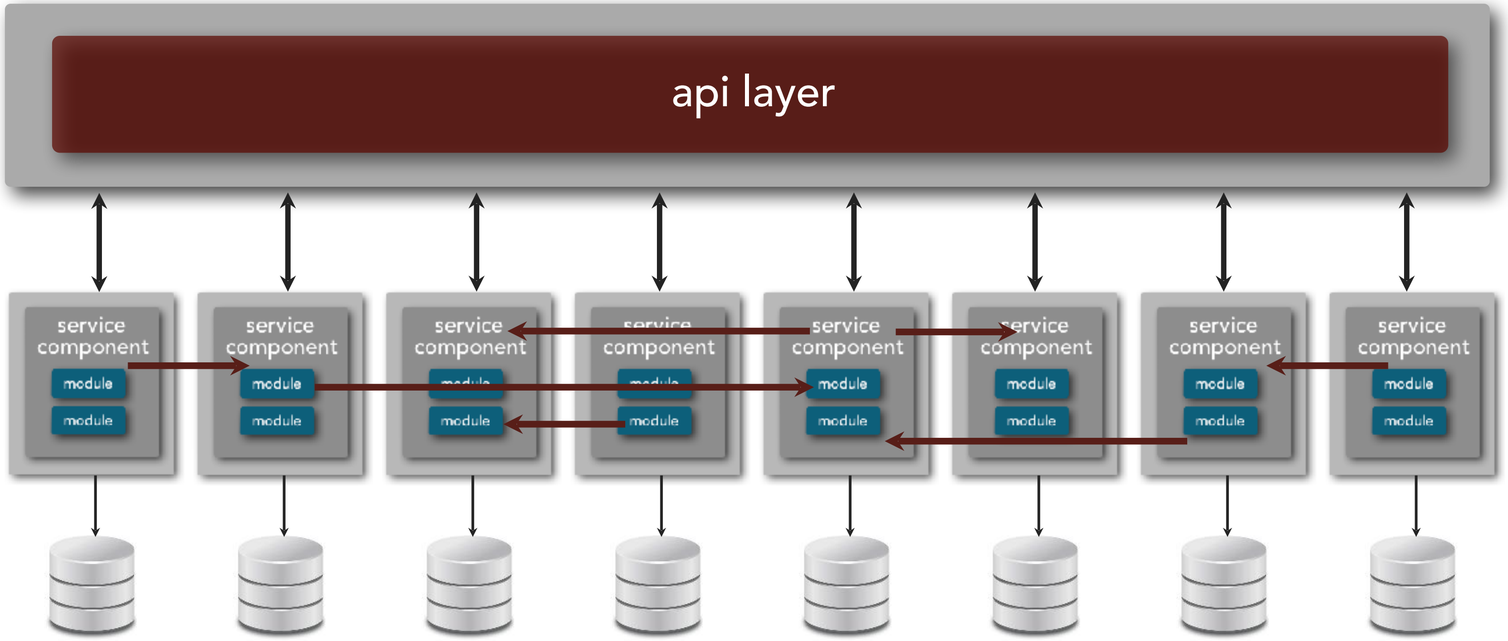
reliability and data consistency

workflow and choreography



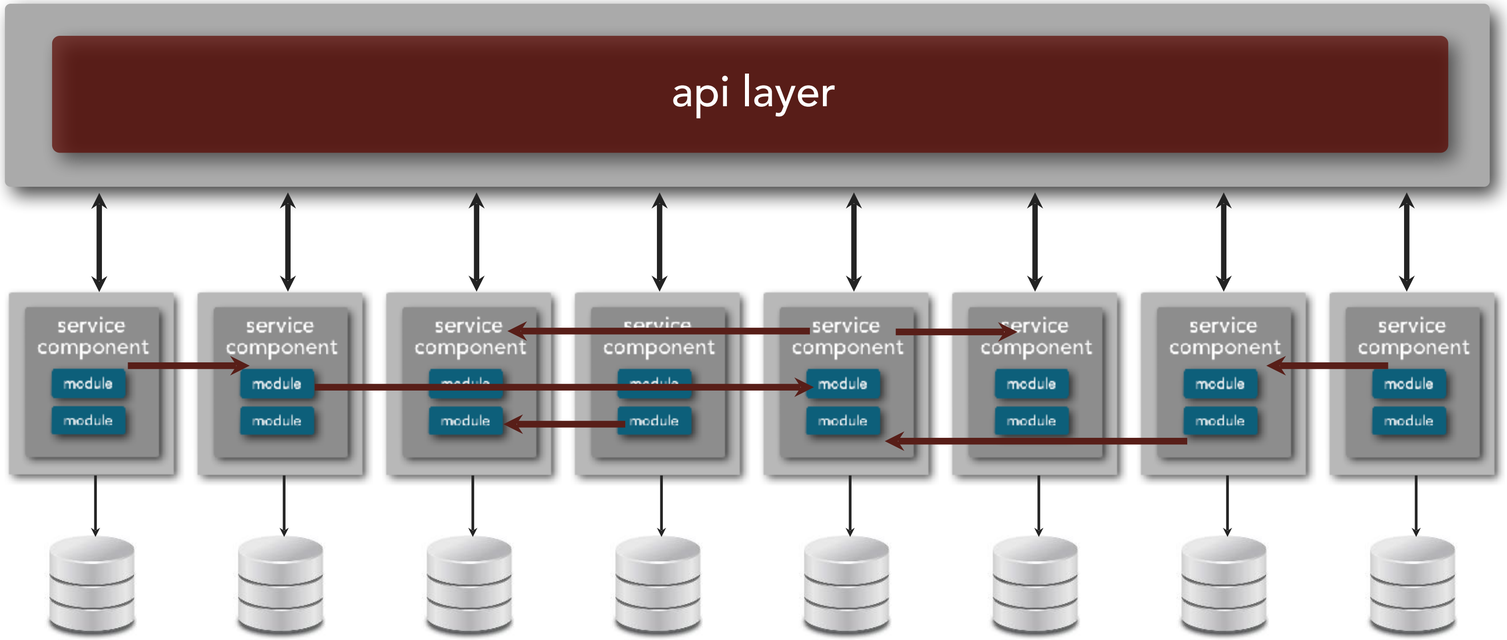
service granularity

workflow and choreography

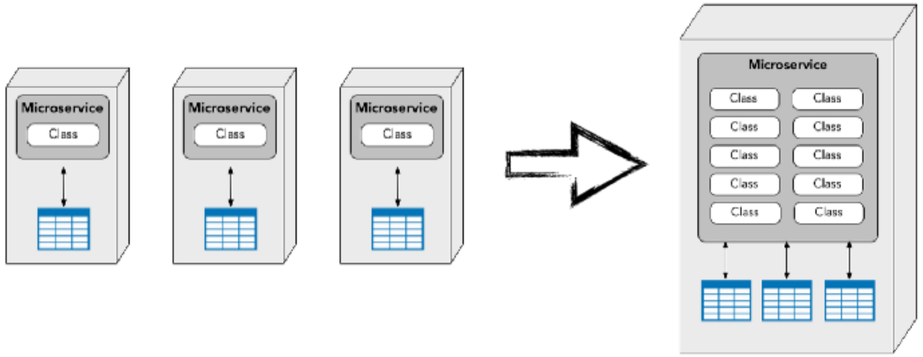


service granularity

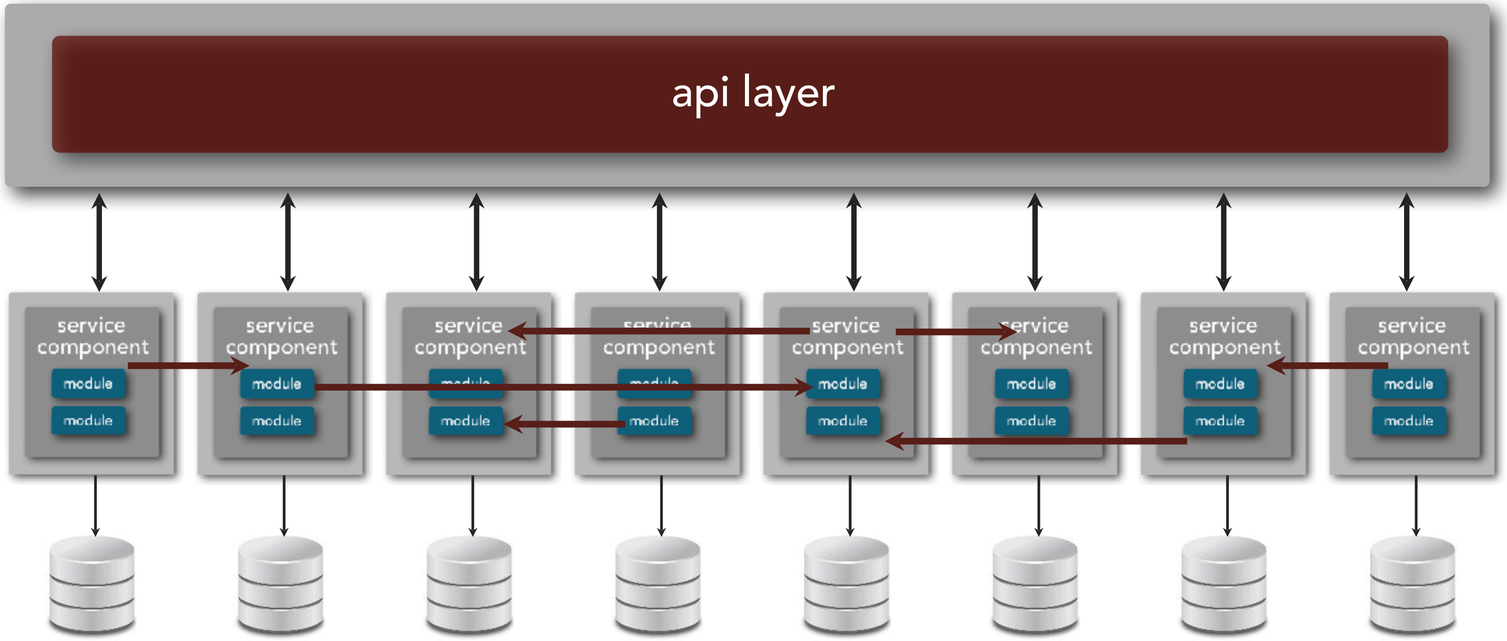
workflow and choreography



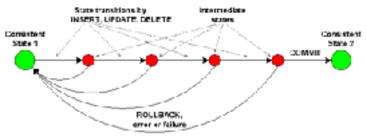
service granularity



workflow and choreography

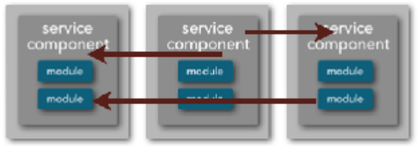


service granularity



database transactions

data dependencies



workflow and choreography

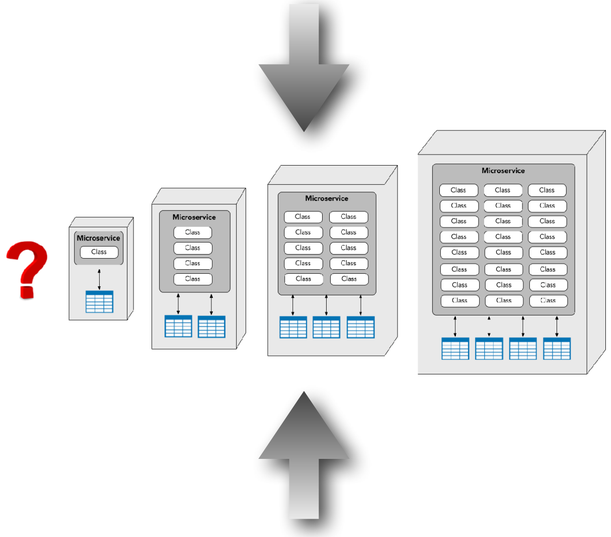
granularity factors

"what factors influence service granularity?"

service granularity

granularity drivers

"when should I consider breaking apart a service?"

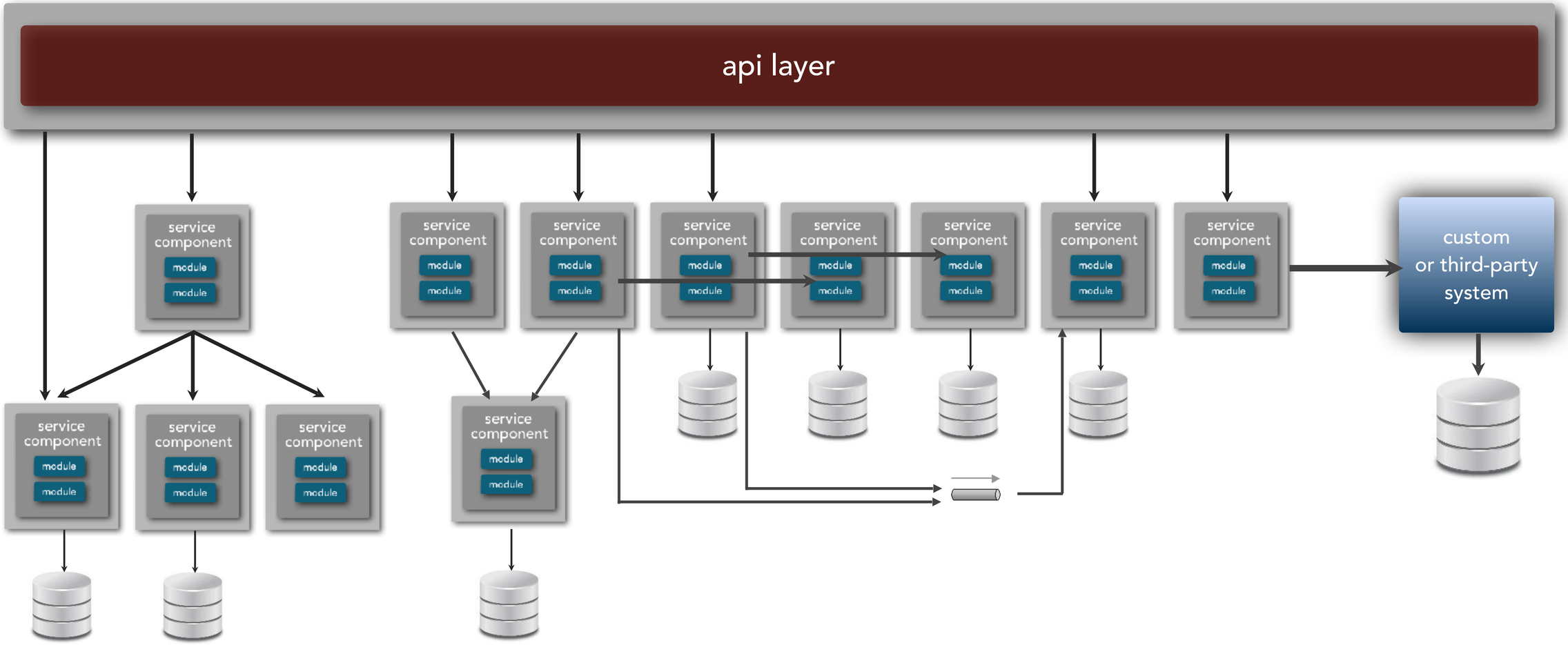


granularity factors

"what factors influence service granularity?"

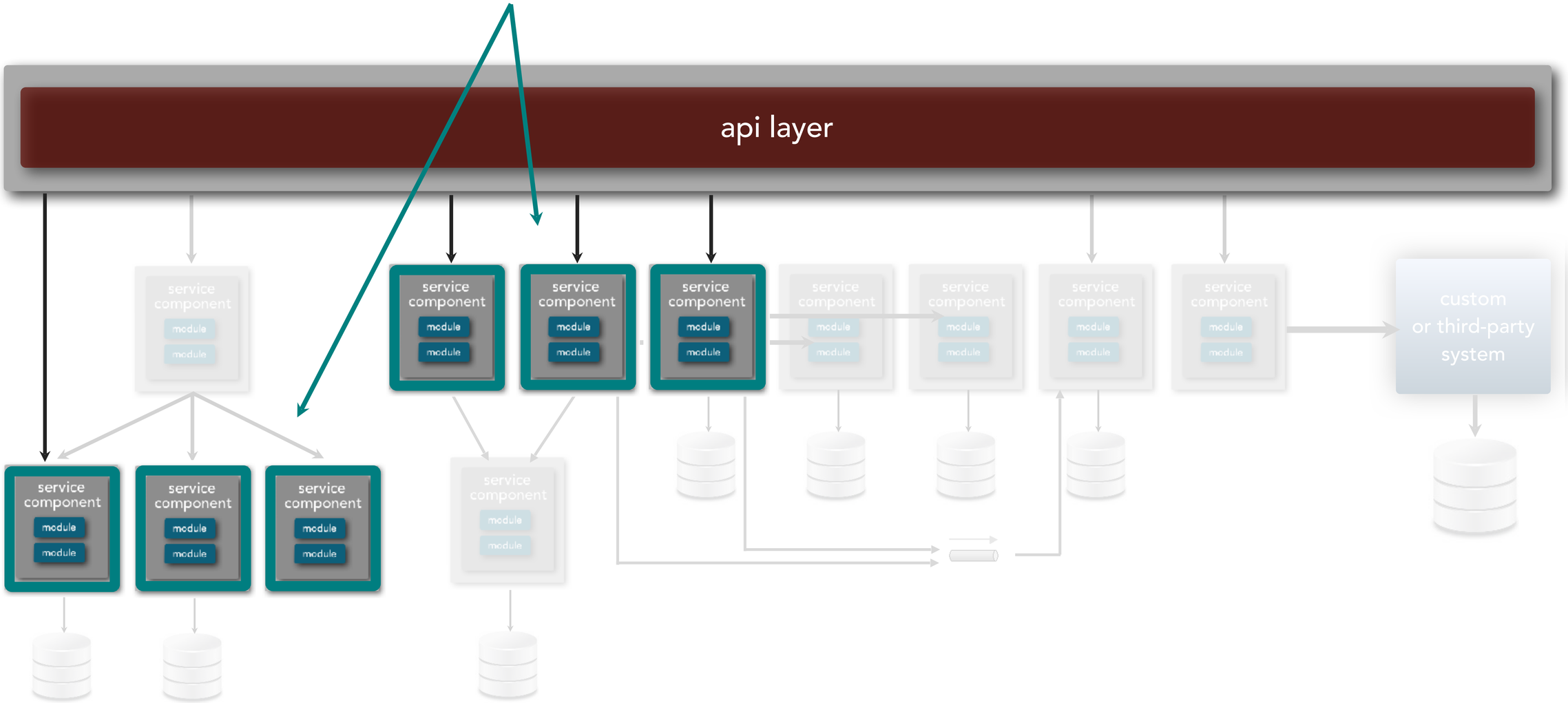
Service Types

service types



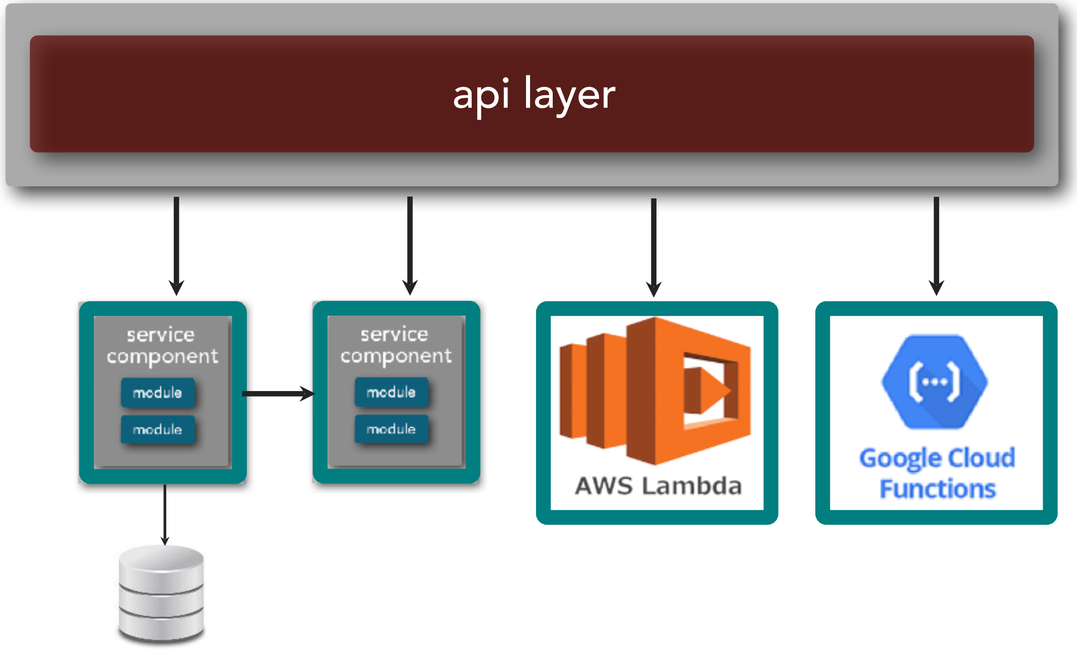
service types

functional services



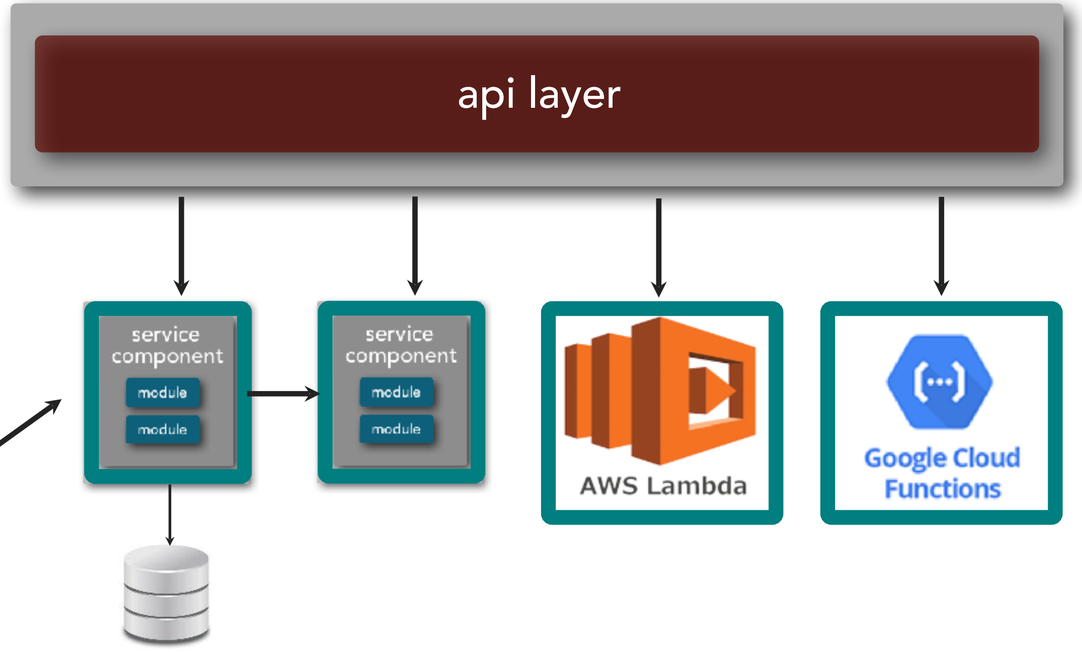
service types

functional services



service types

functional services



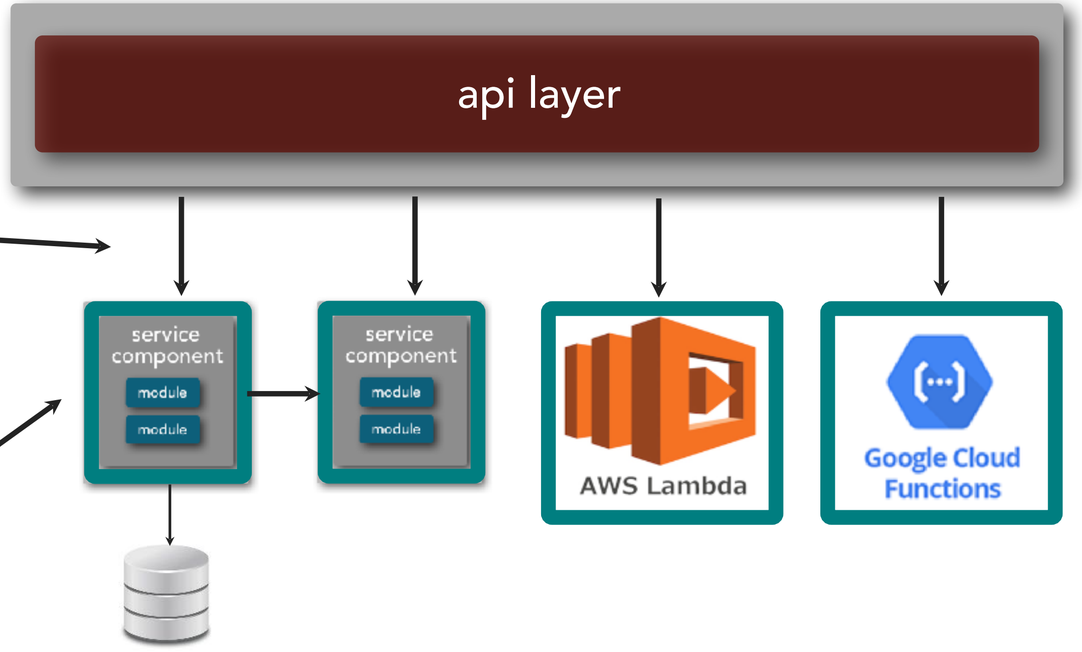
perform a particular business function

service types

functional services

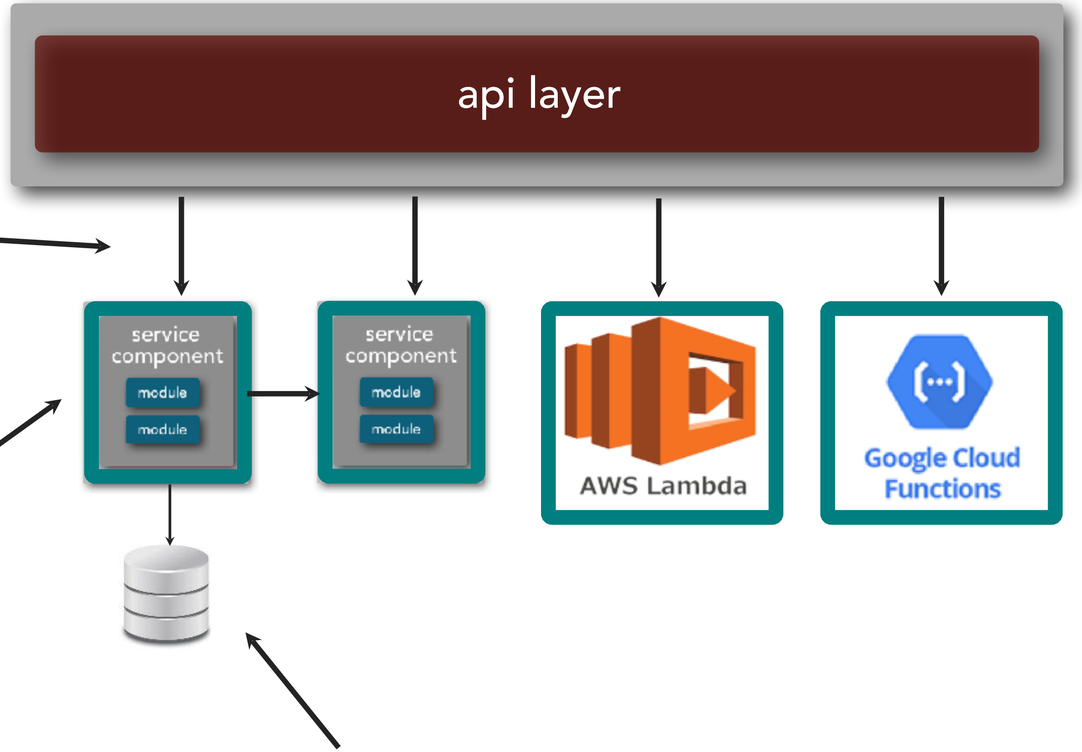
attached to a public API endpoint

perform a particular business function



service types

functional services



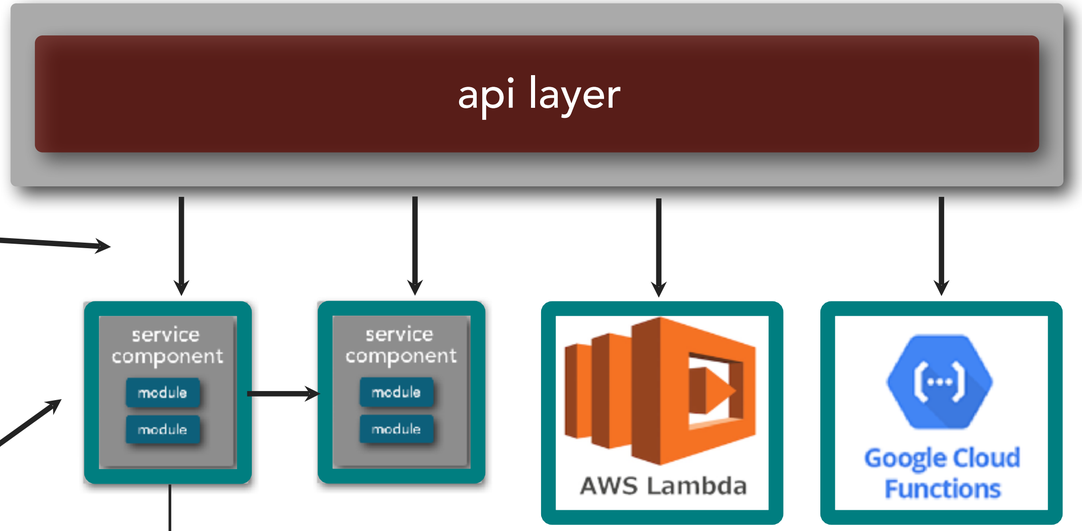
attached to a public API endpoint

perform a particular business function

usually contain data

service types

functional services



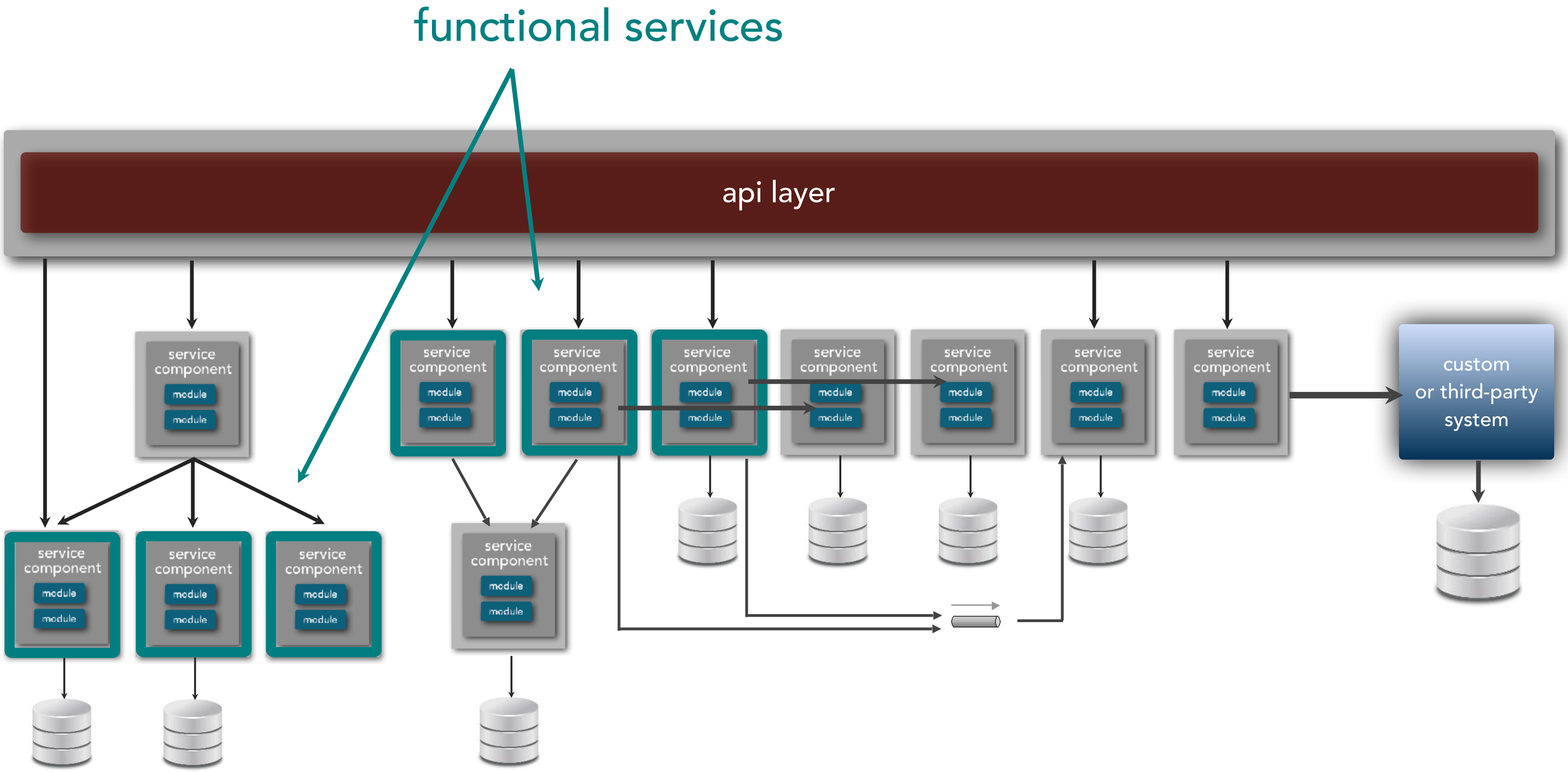
attached to a public API endpoint

perform a particular business function

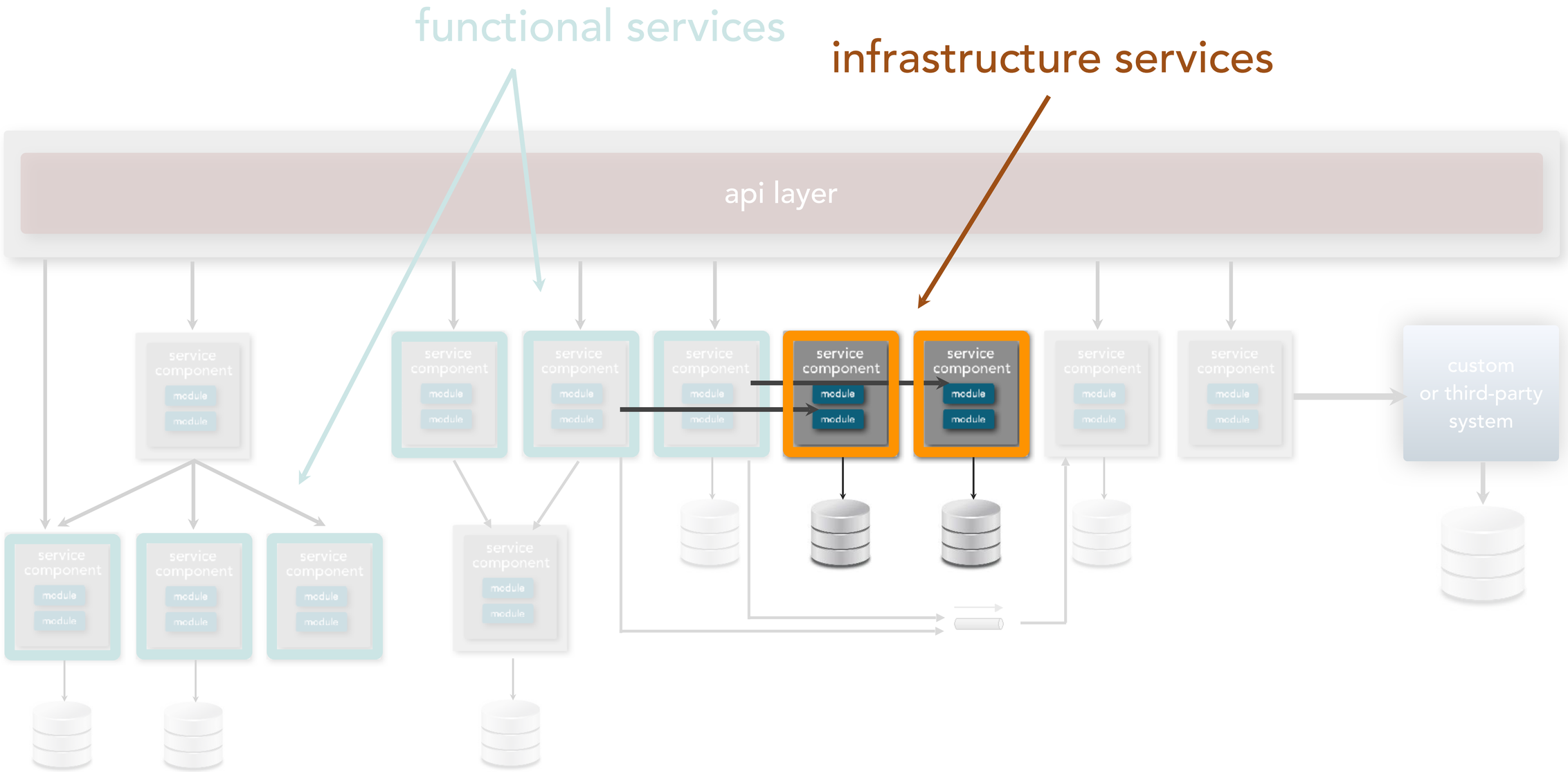
usually contain data

can be a containerized or serverless function

service types

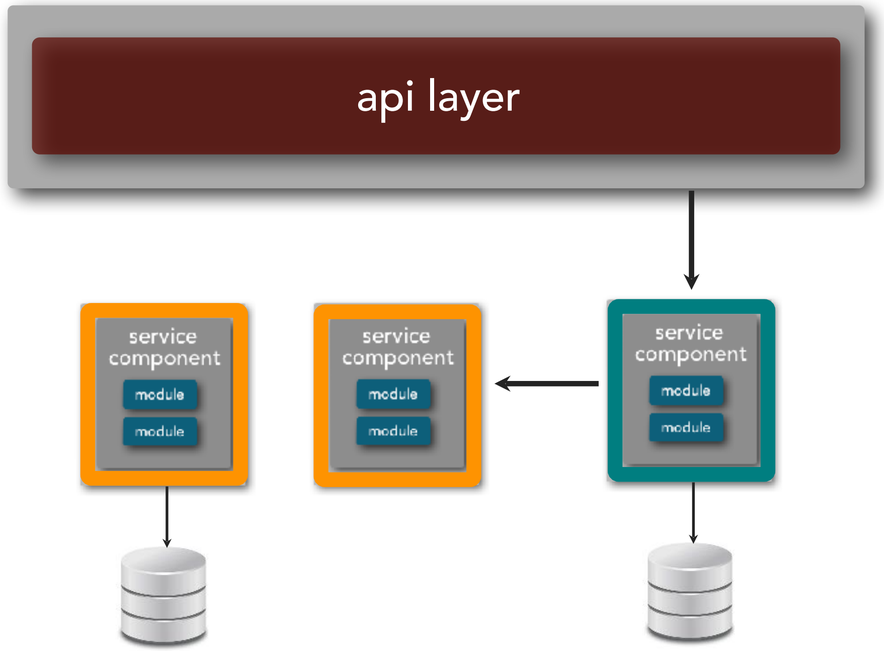


service types



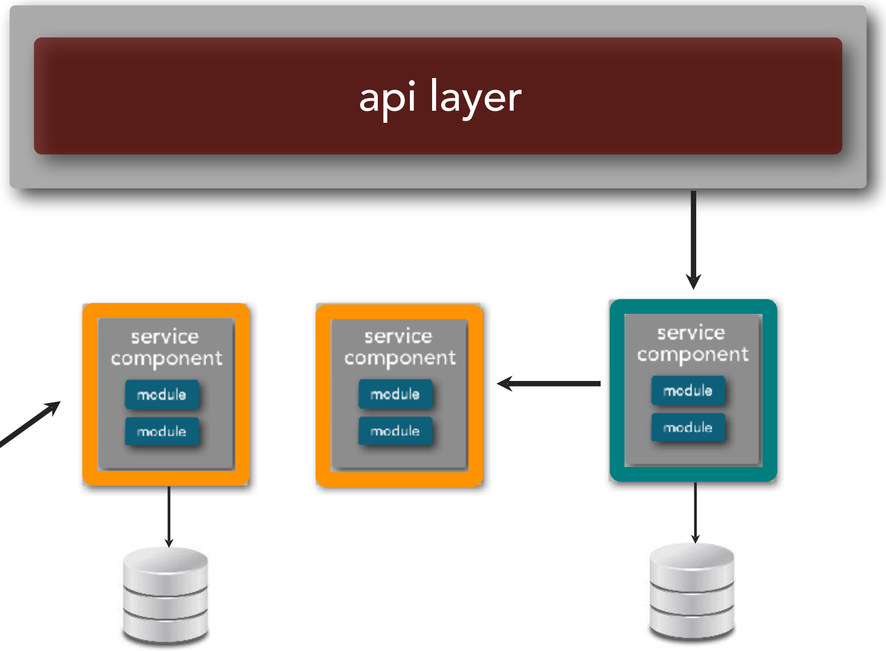
service types

infrastructure services



service types

infrastructure services

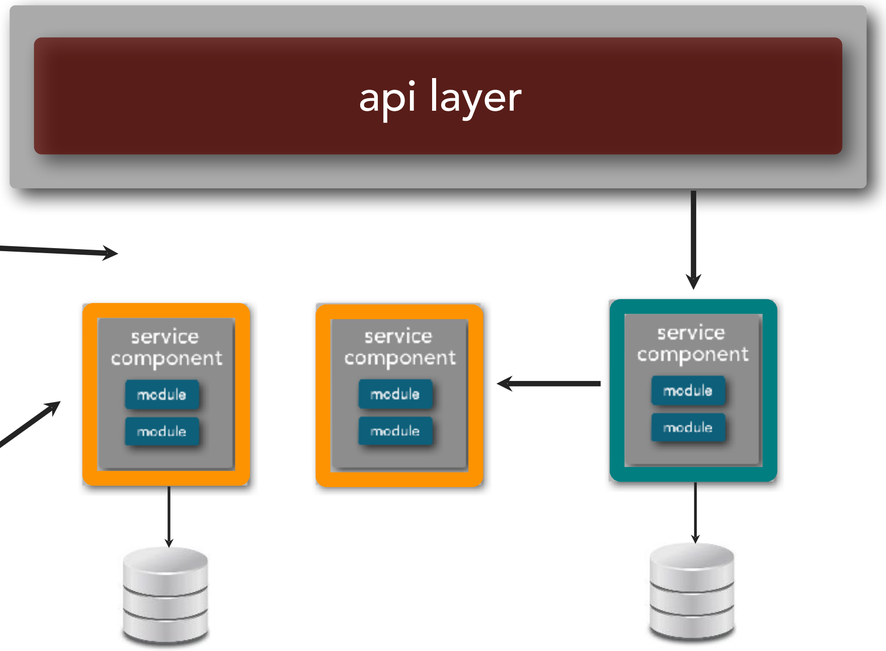


perform an internal or infrastructure function

service types

infrastructure services

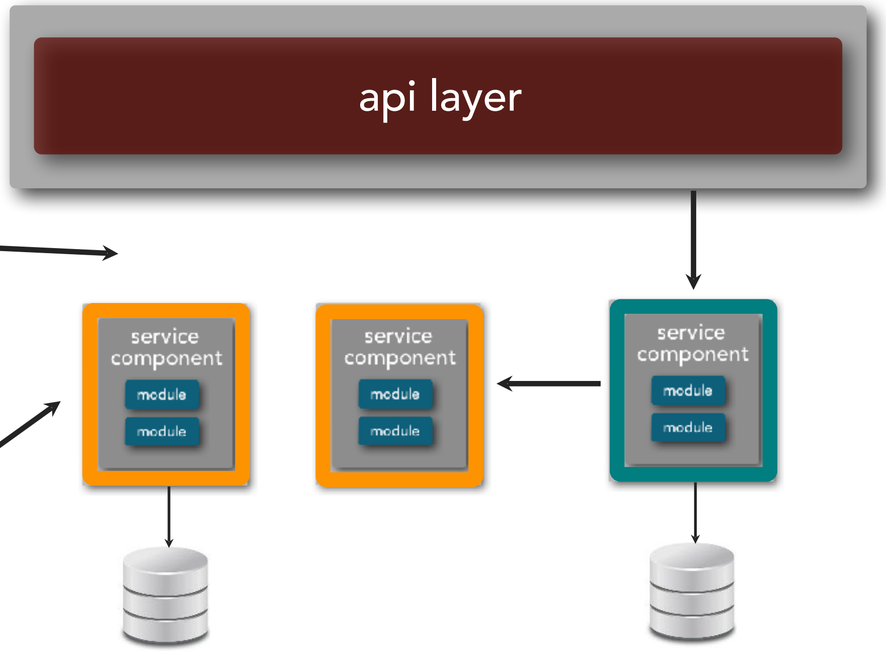
not exposed to
the public API



perform an internal
or infrastructure
function

service types

infrastructure services



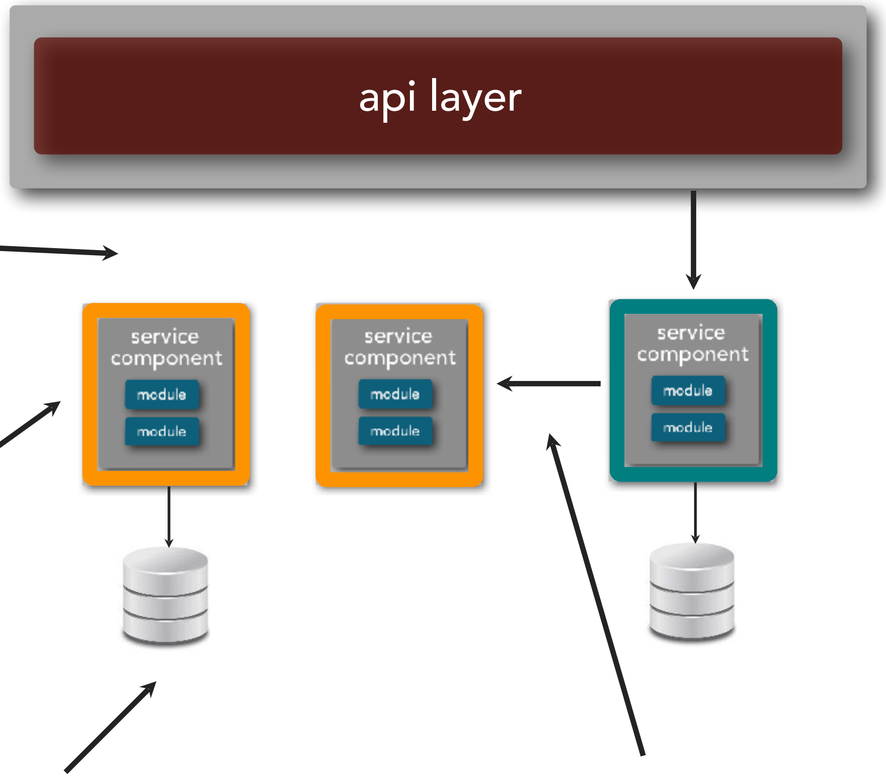
not exposed to the public API

perform an internal or infrastructure function

can contain data

service types

infrastructure services



not exposed to the public API

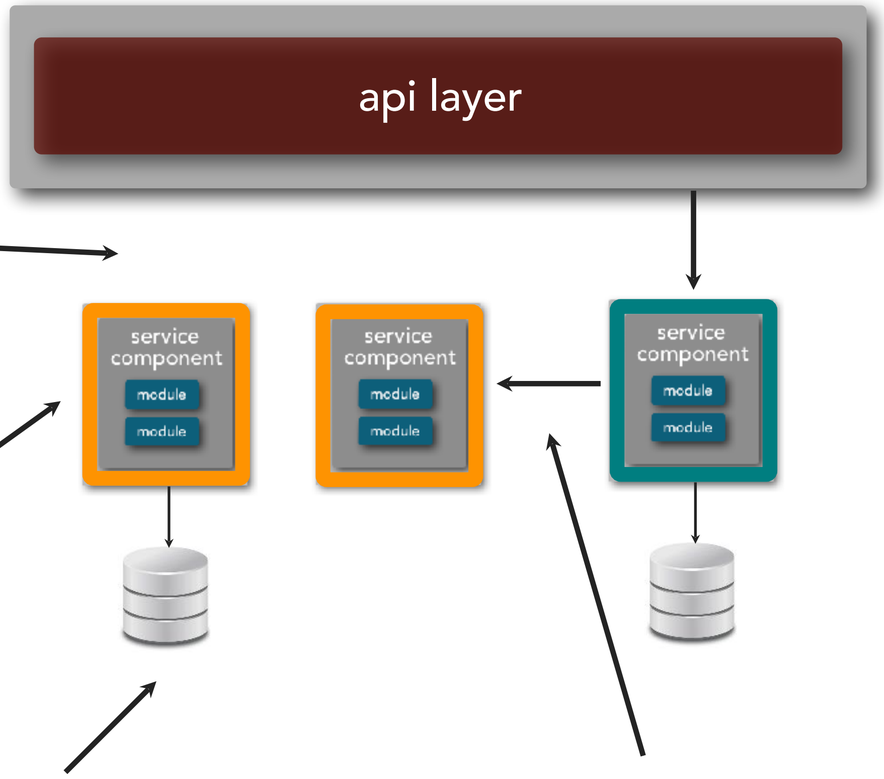
perform an internal or infrastructure function

can contain data

accessed by other services through inter-service communication

service types

infrastructure services



not exposed to the public API

perform an internal or infrastructure function

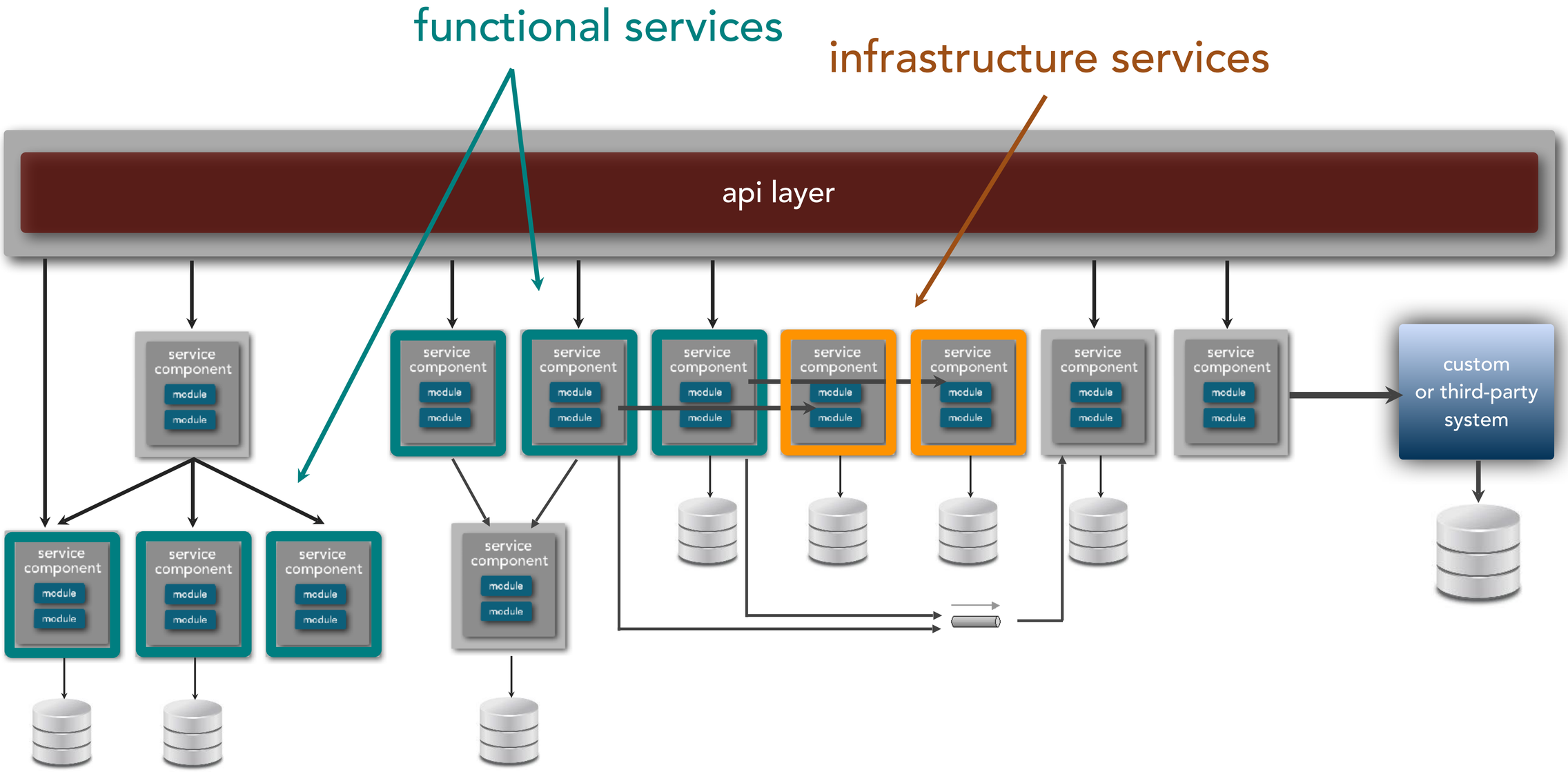
can contain data

accessed by other services through inter-service communication

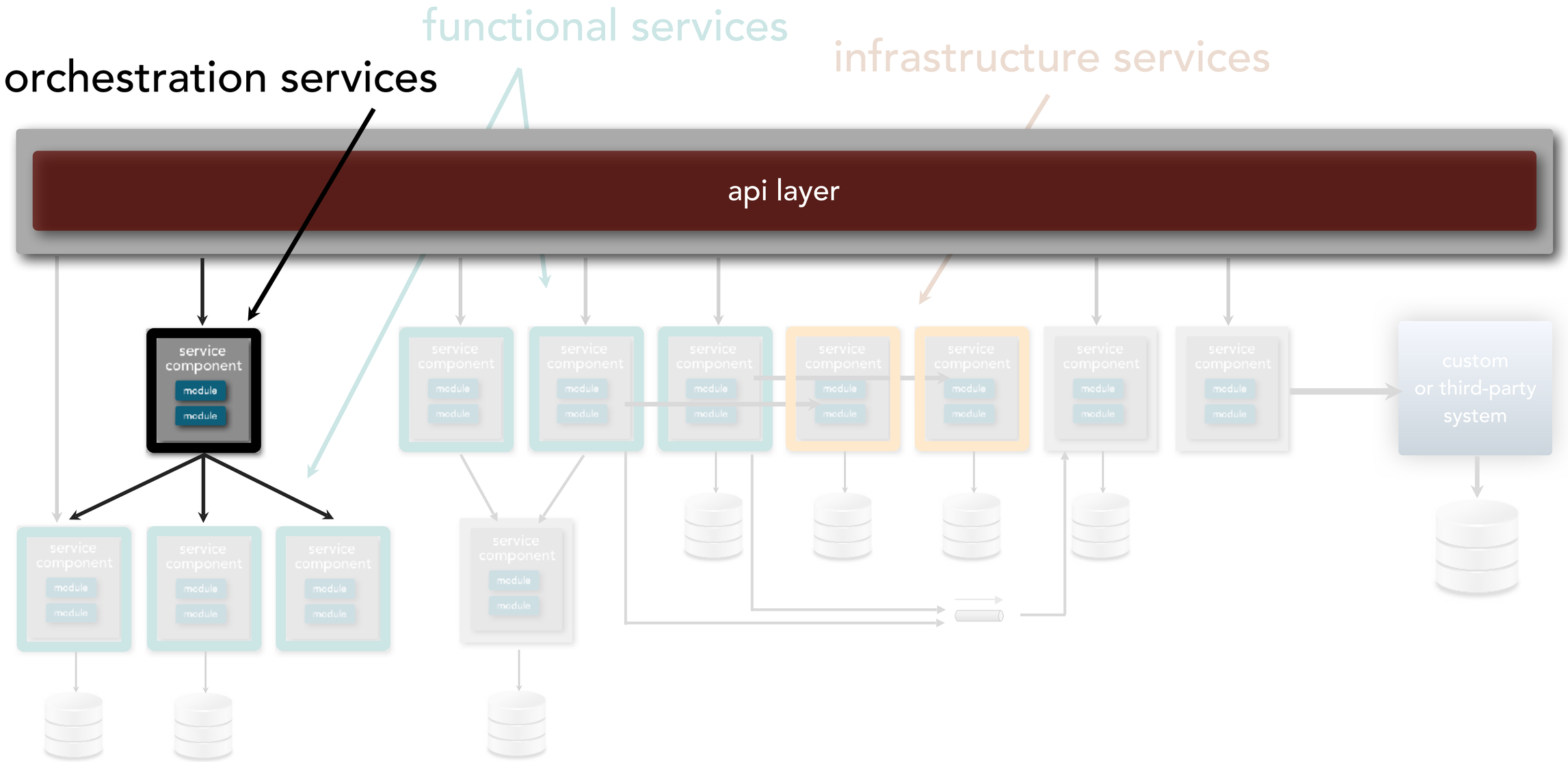
examples:

- auditing
- logging
- metrics gathering
- fitness functions
- internal calculators
- security

service types

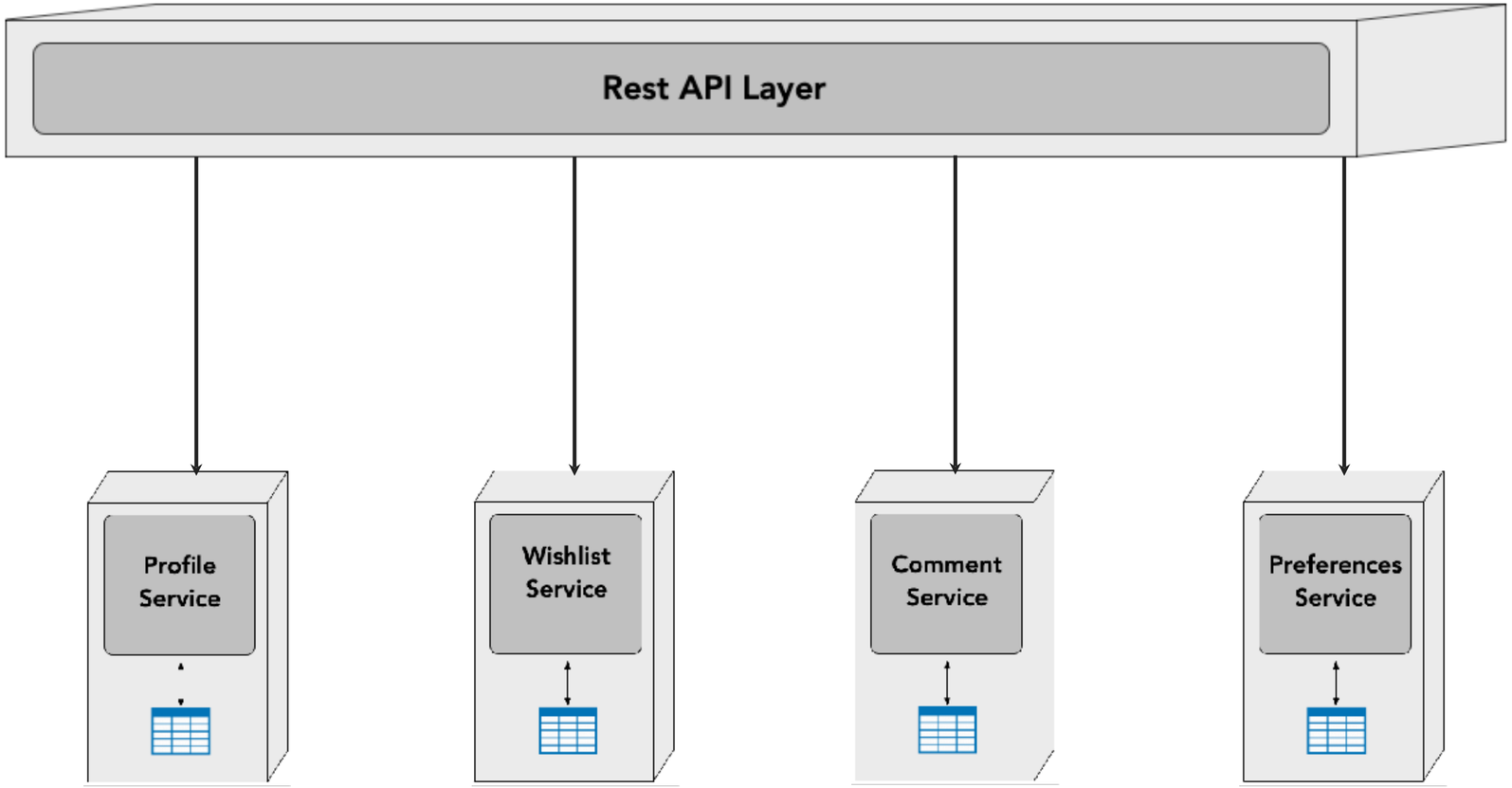


service types



service types

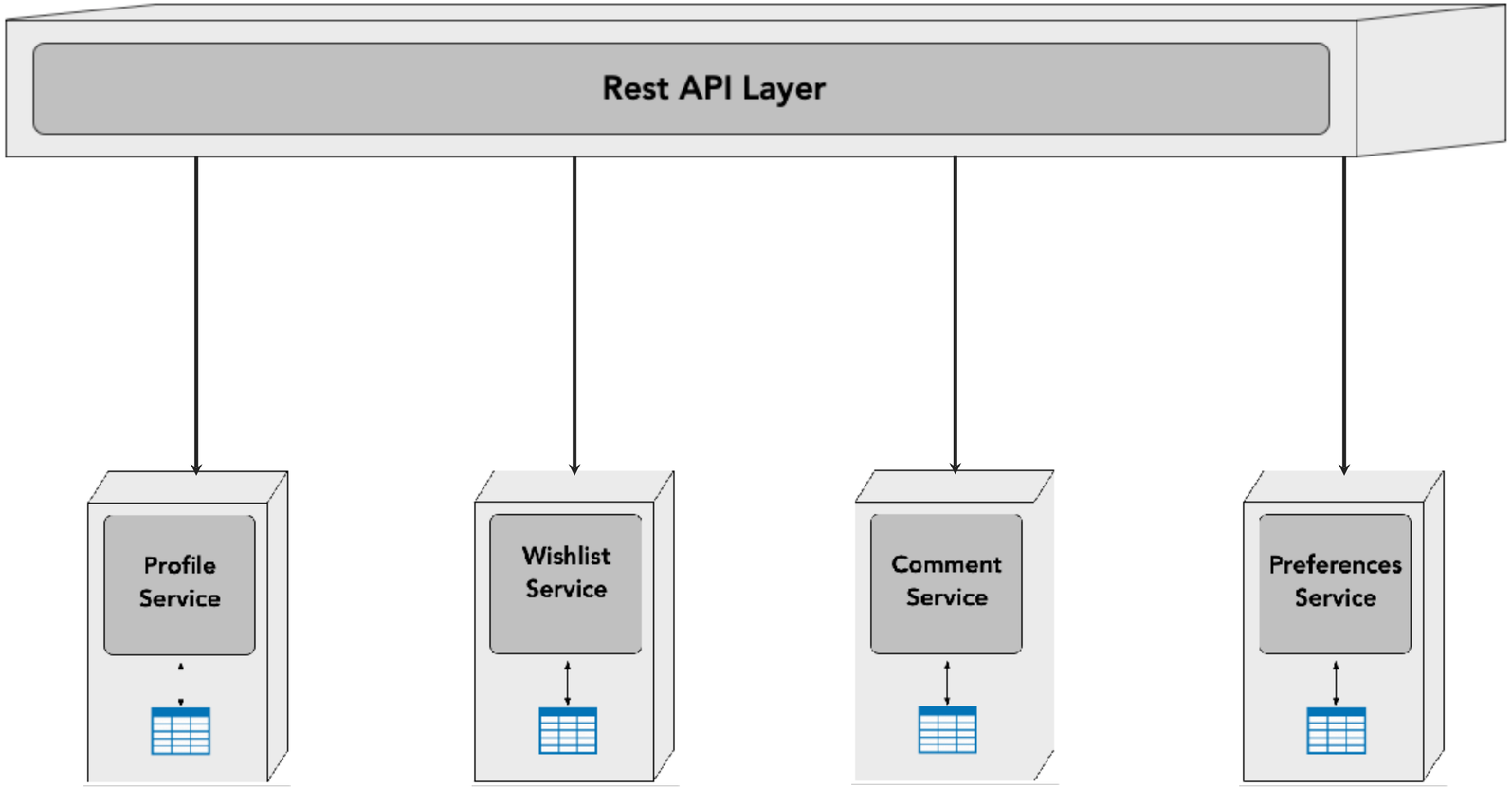
orchestration services



service types

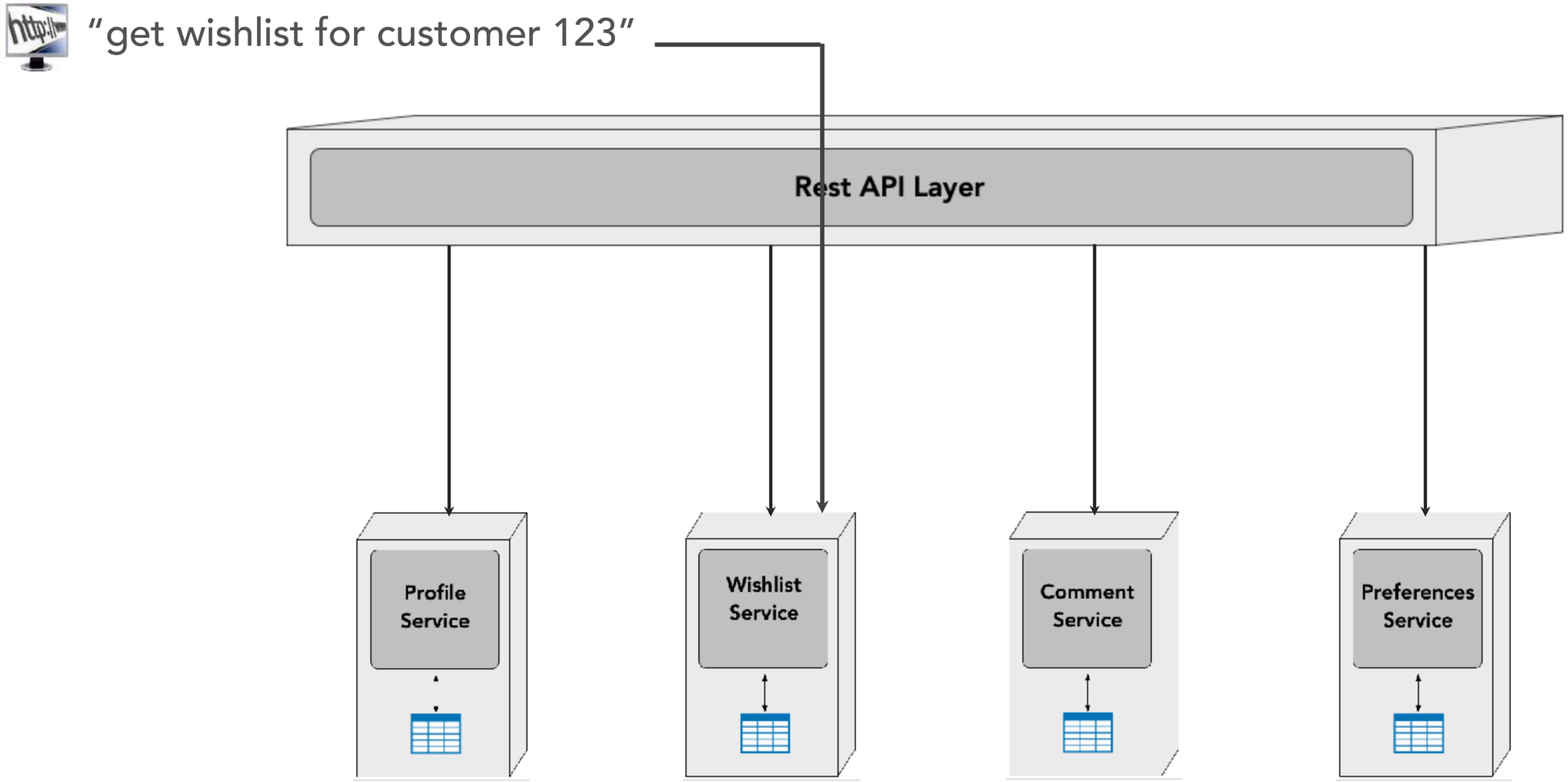
orchestration services

"get wishlist for customer 123"



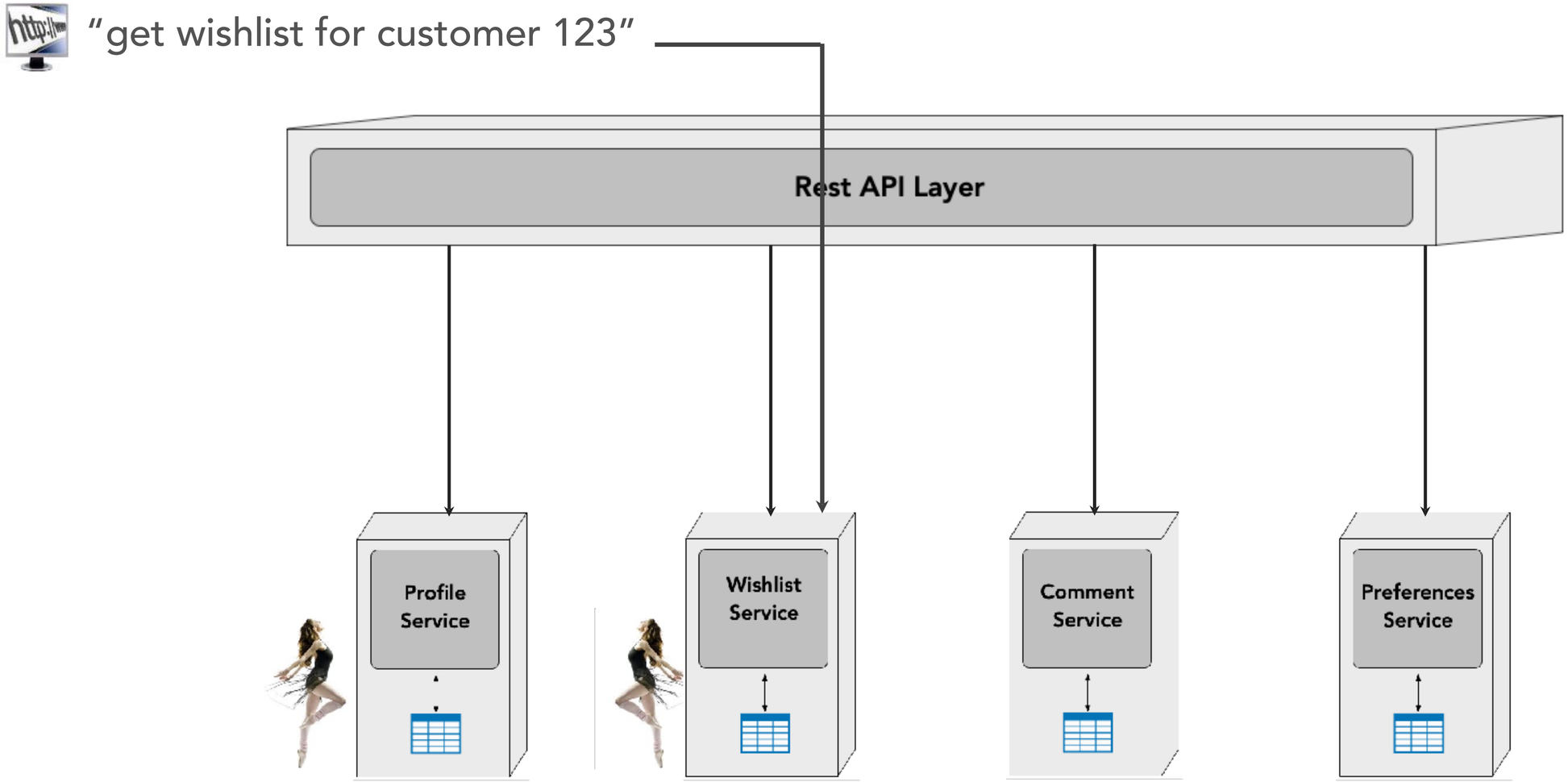
service types

orchestration services



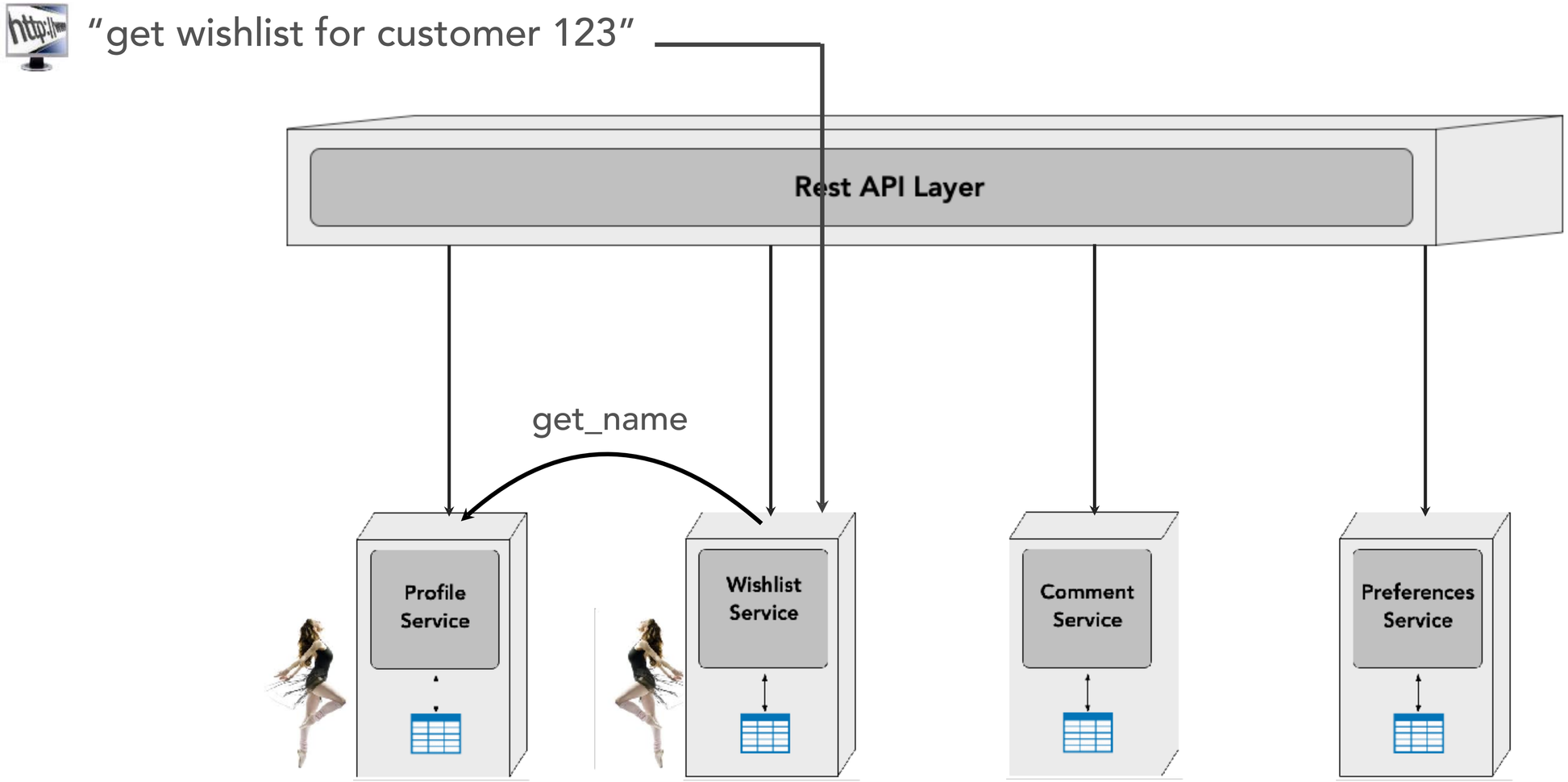
service types

orchestration services



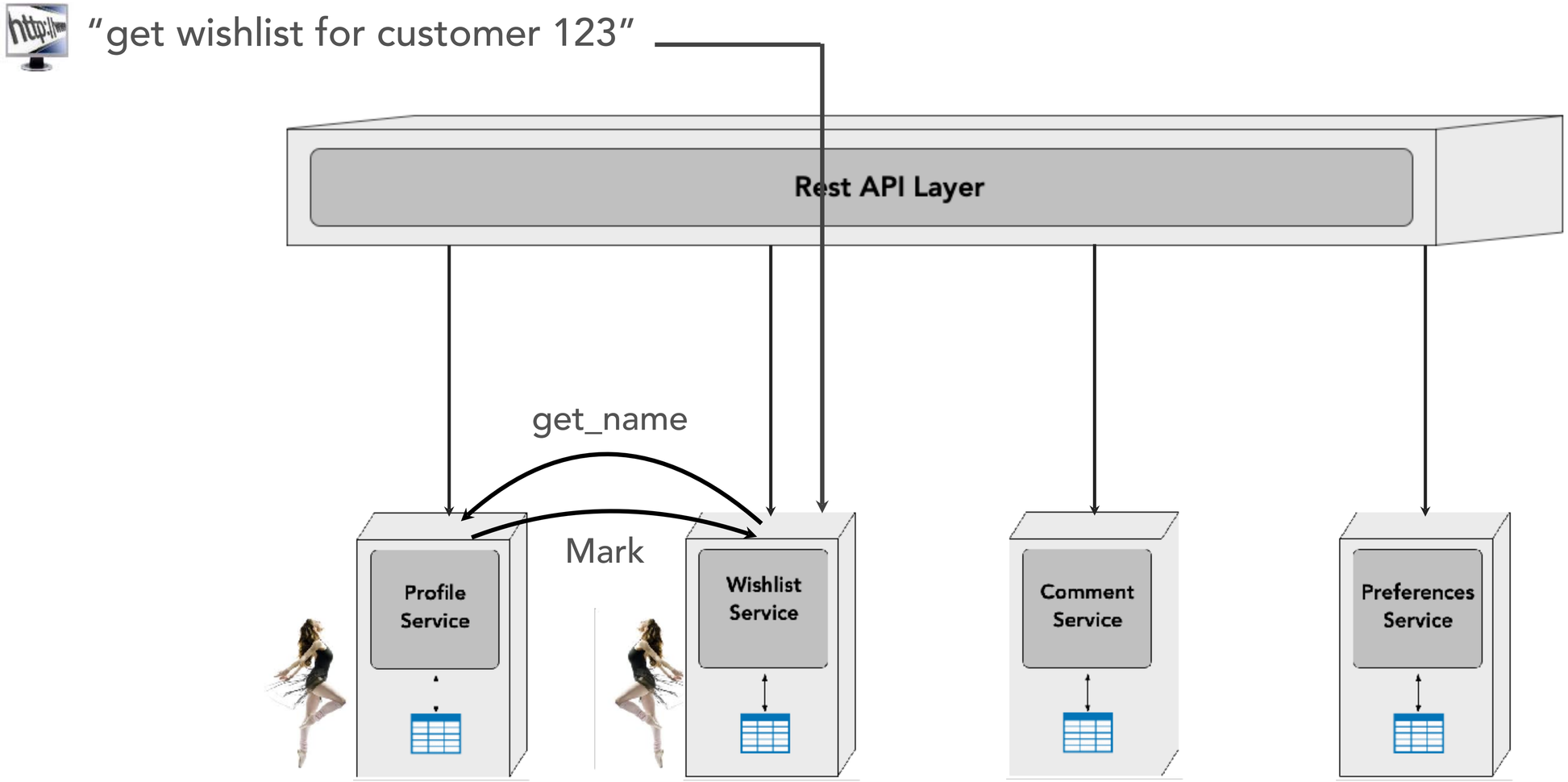
service types

orchestration services



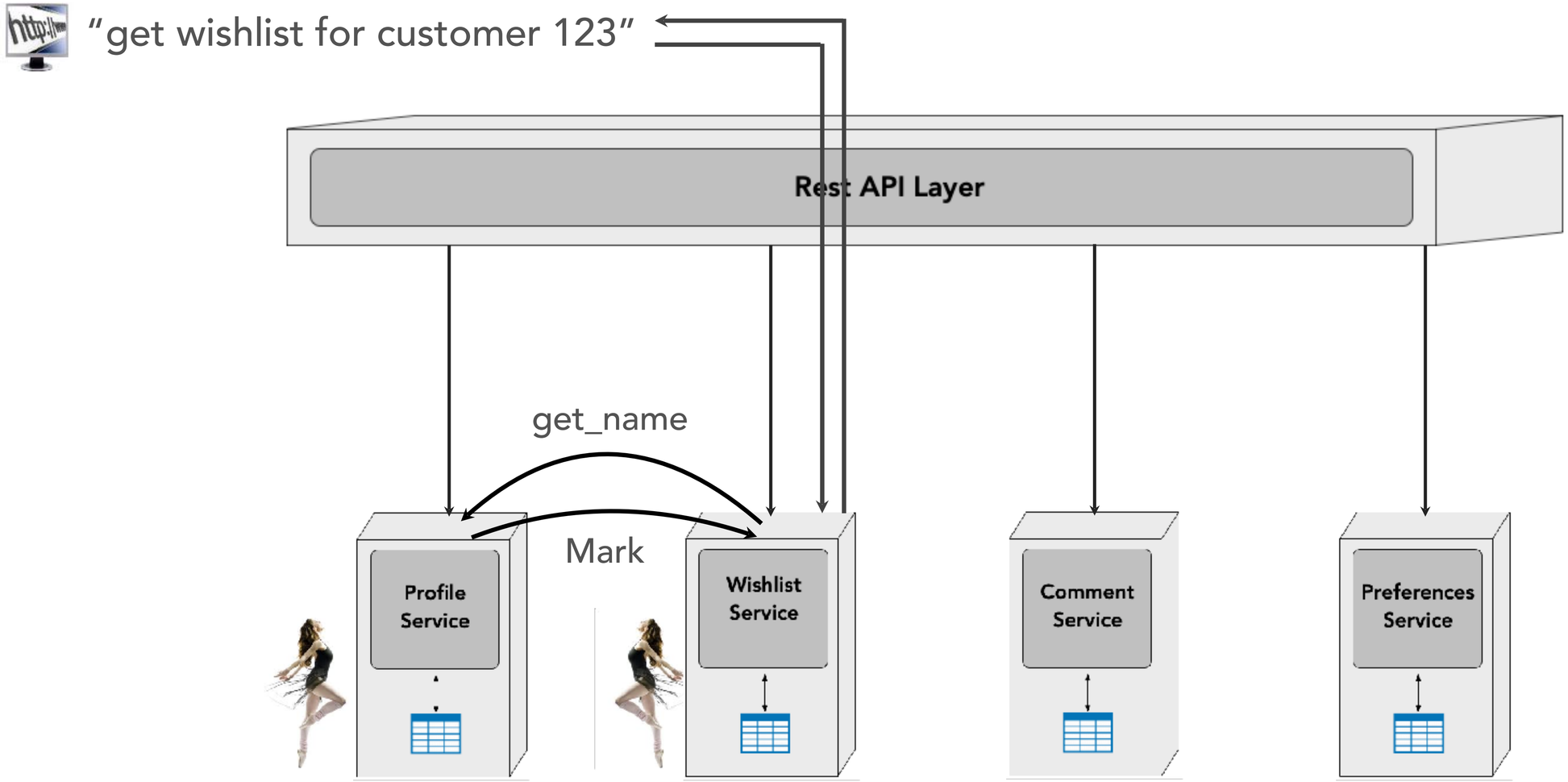
service types

orchestration services



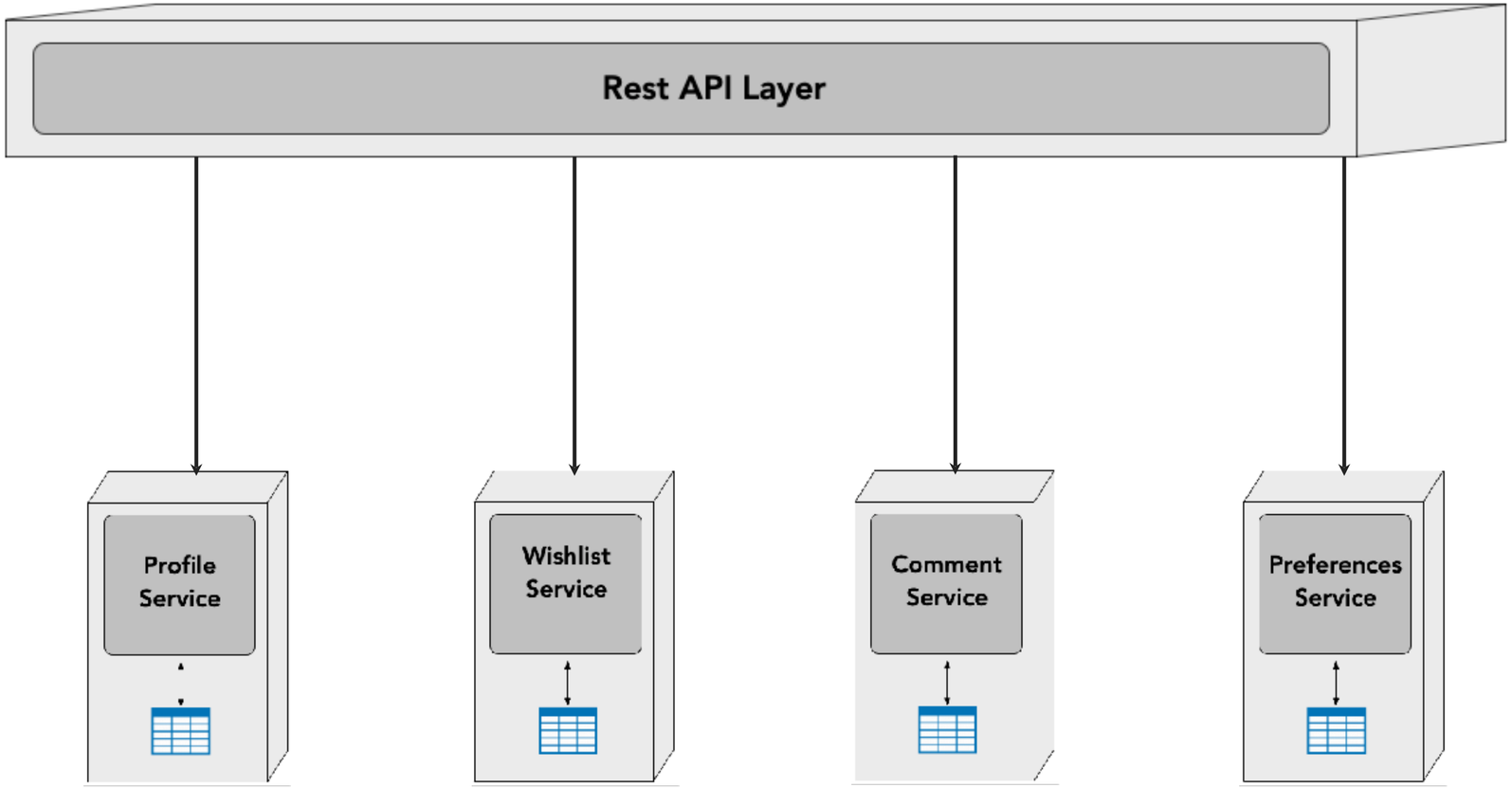
service types

orchestration services



service types

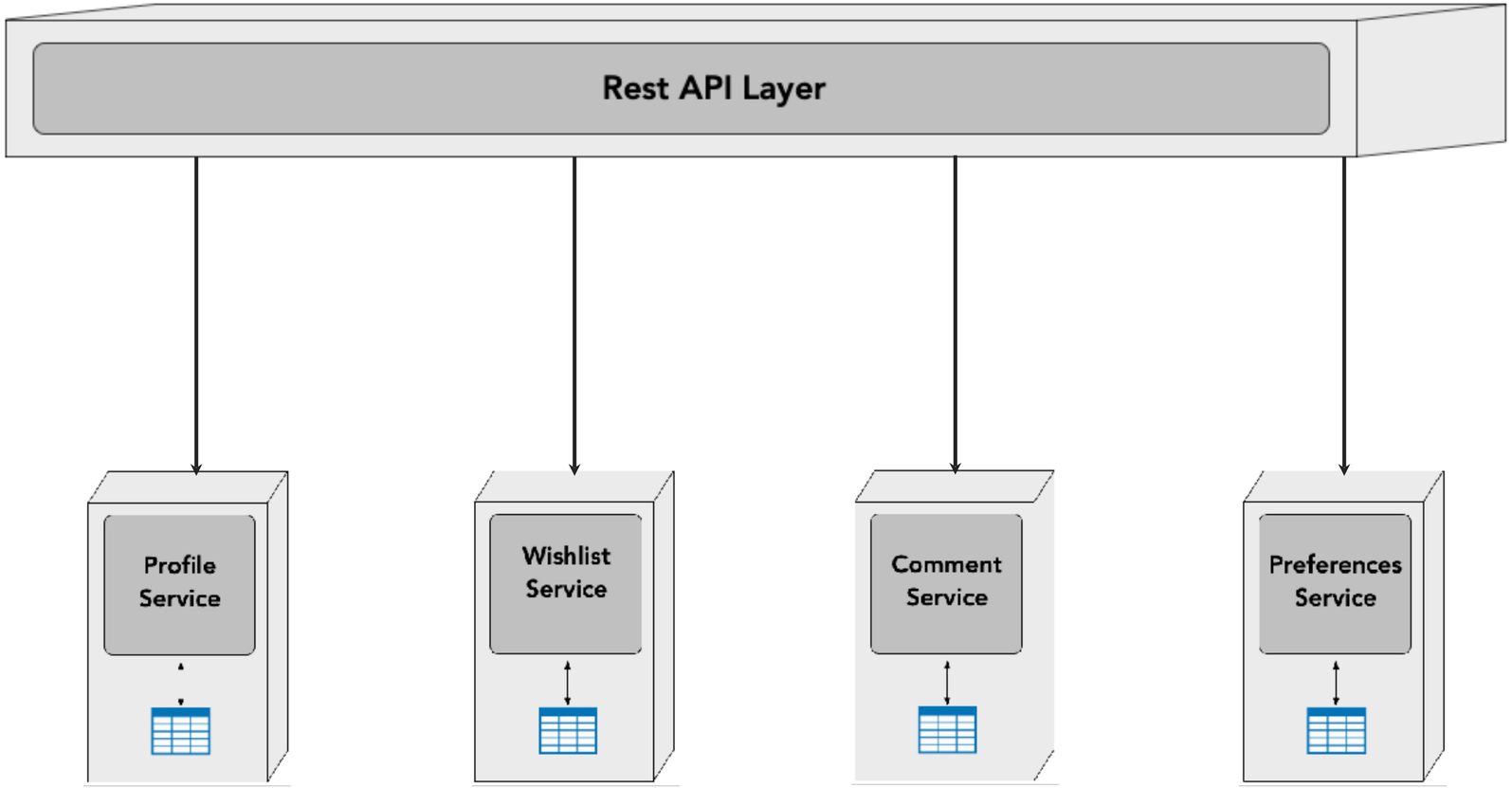
orchestration services



service types

orchestration services

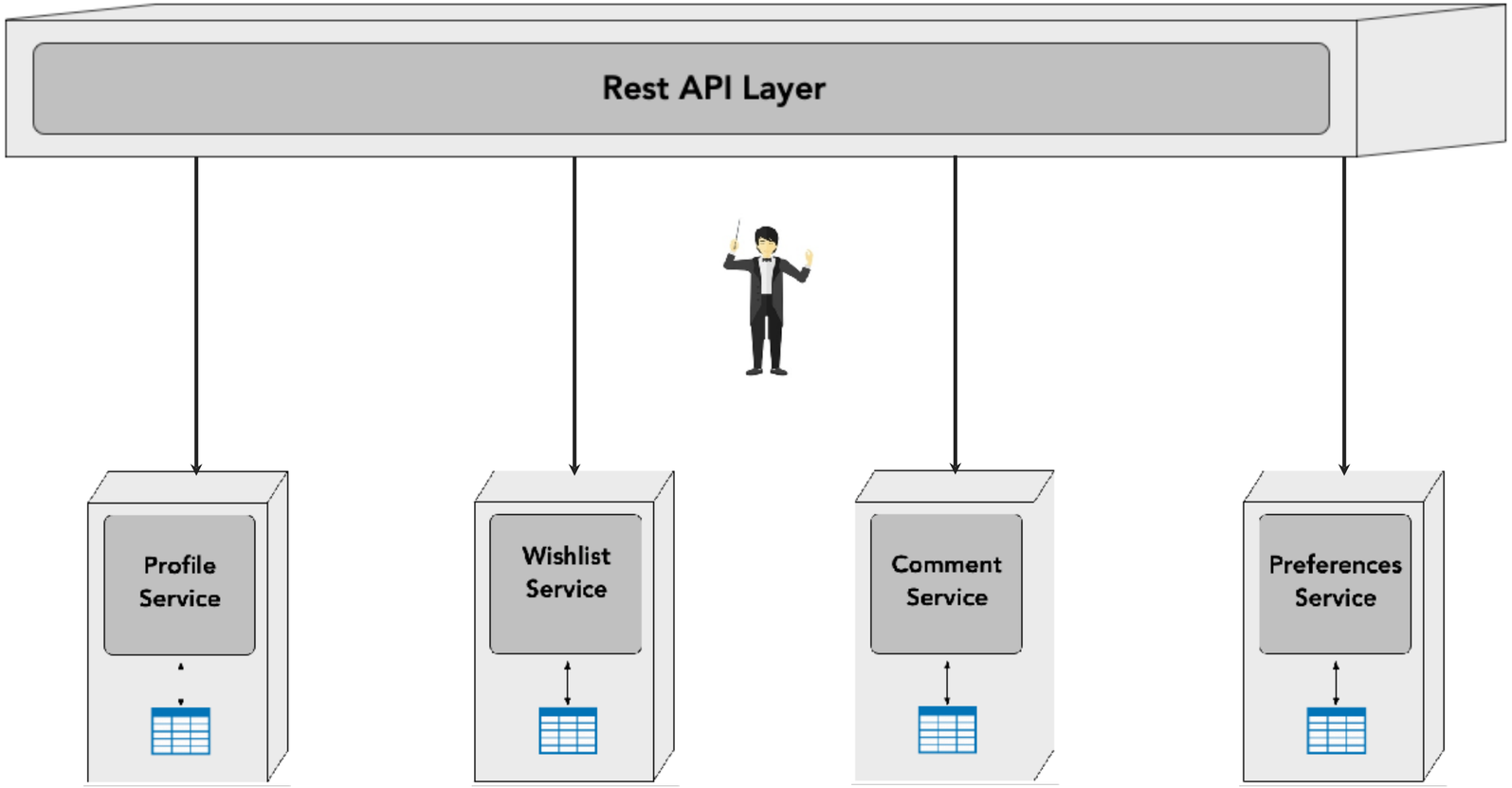
"get all data for customer 123"



service types

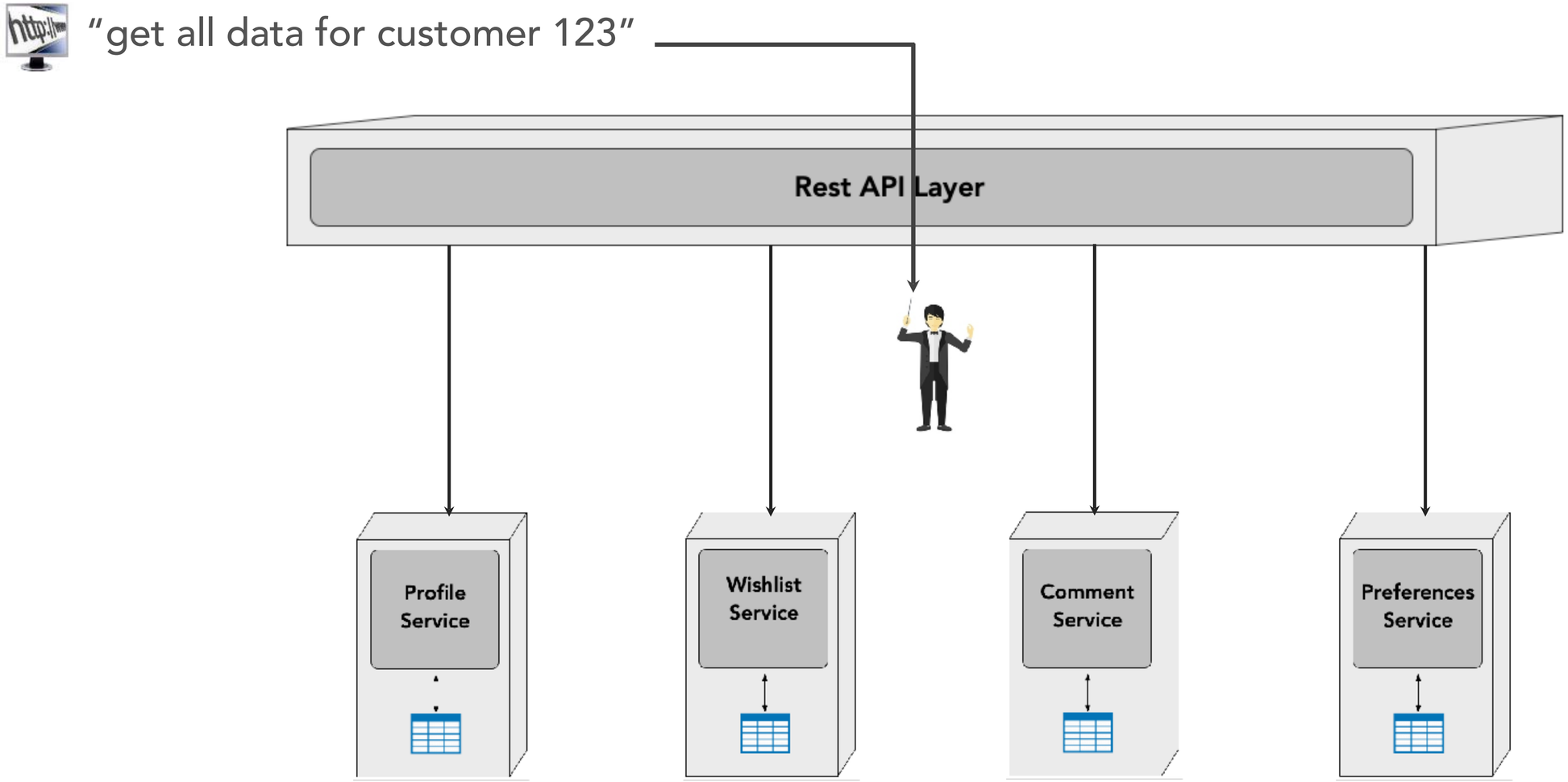
orchestration services

“get all data for customer 123”



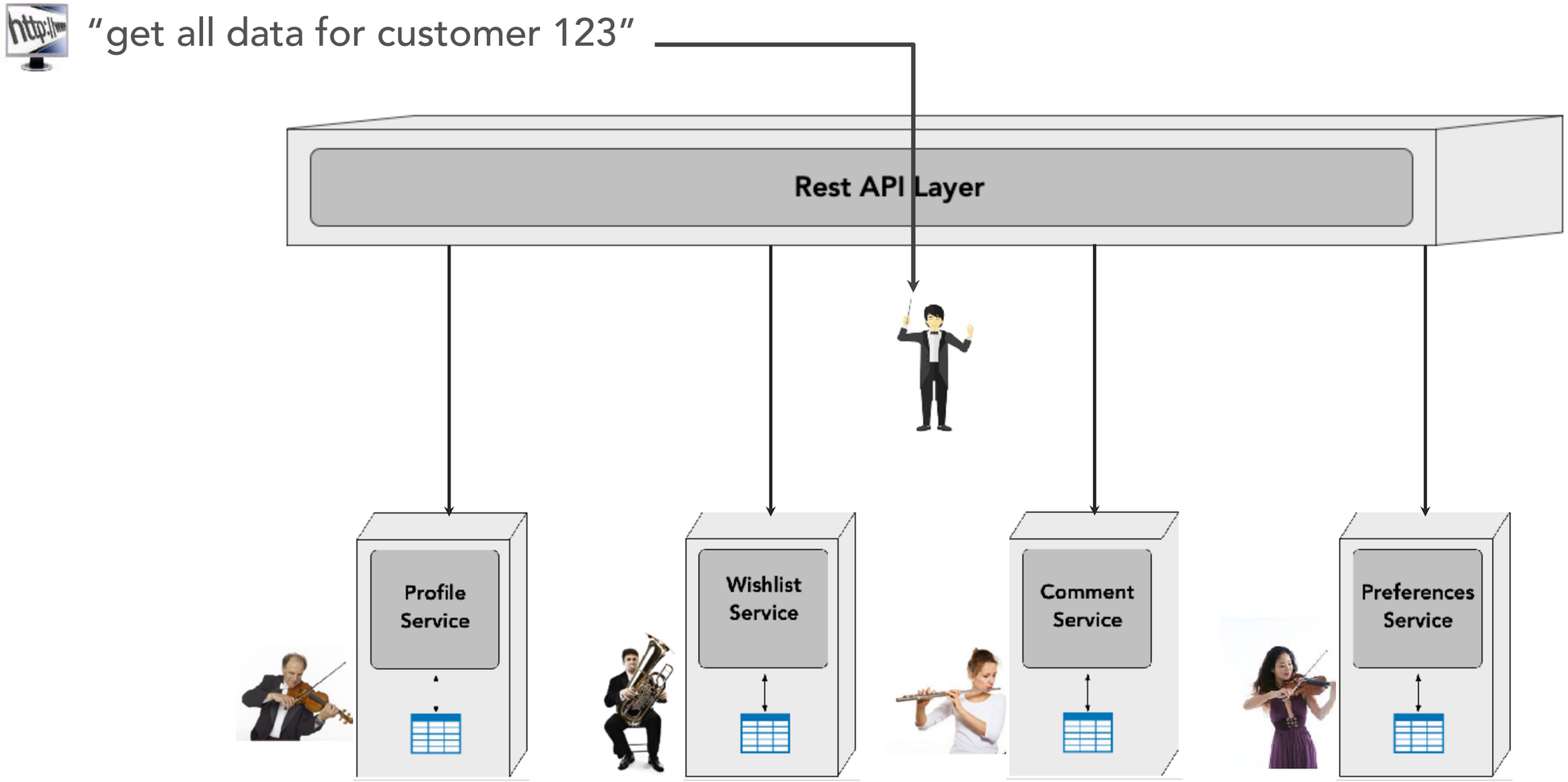
service types

orchestration services



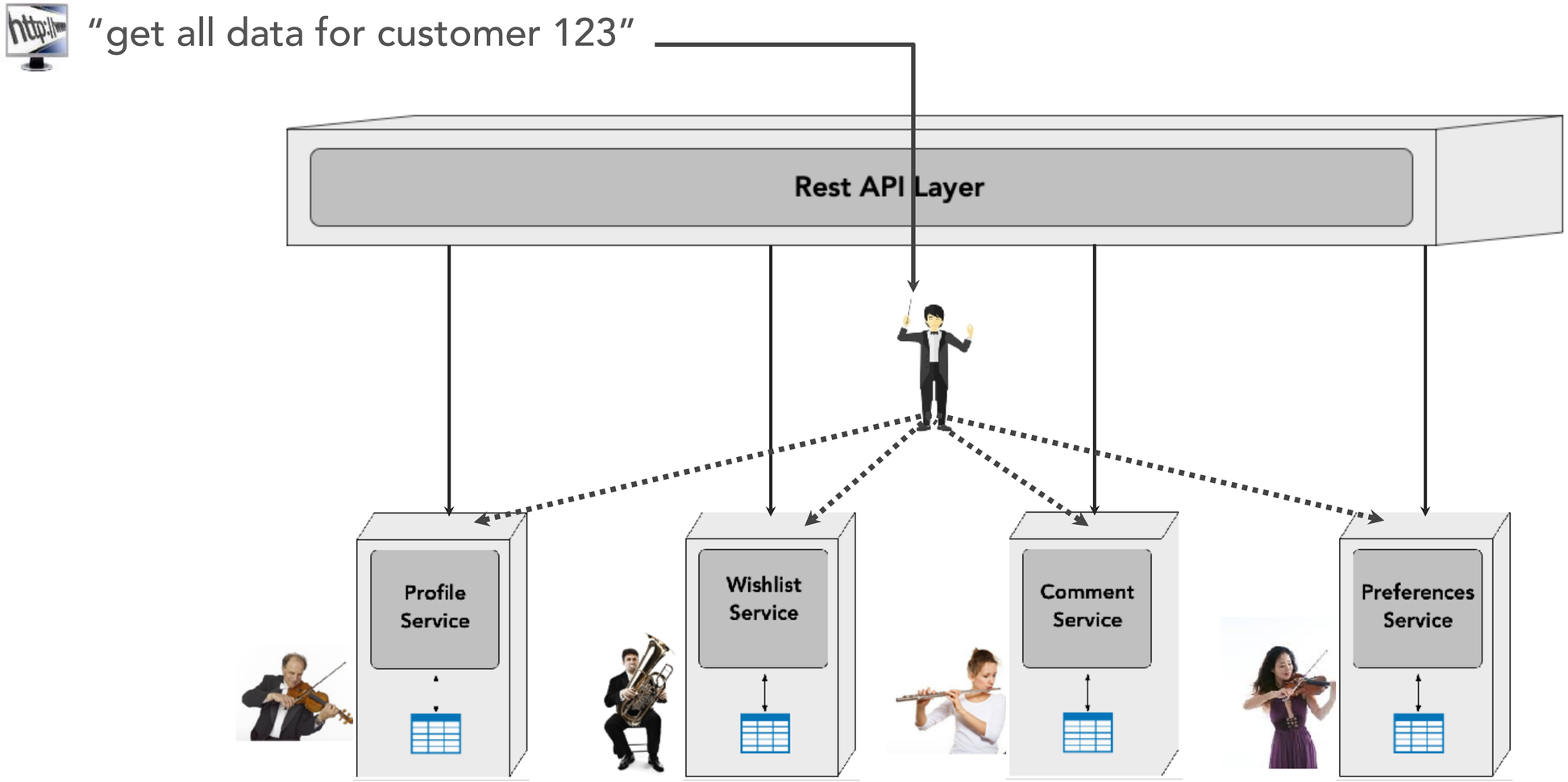
service types

orchestration services



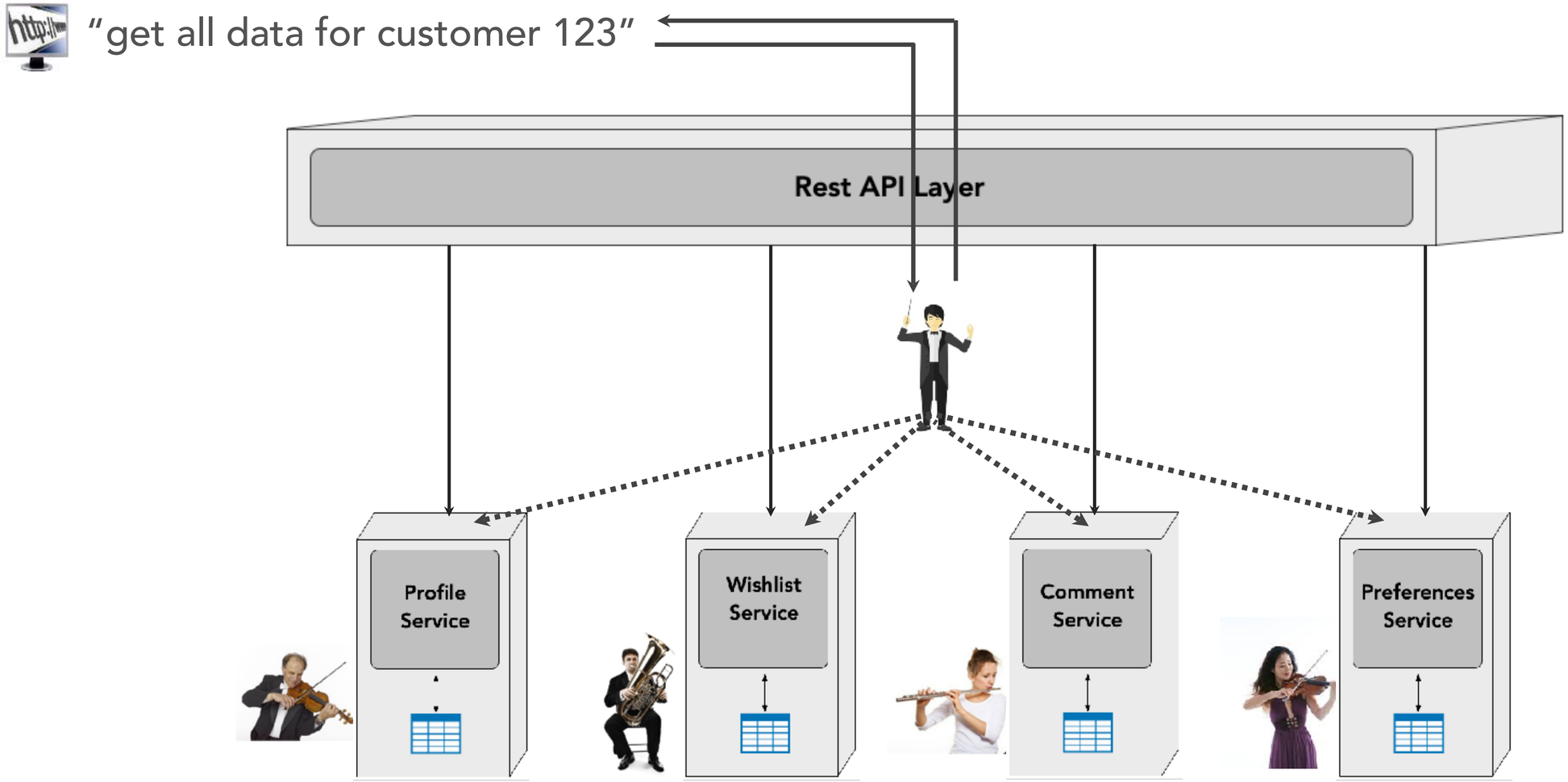
service types

orchestration services



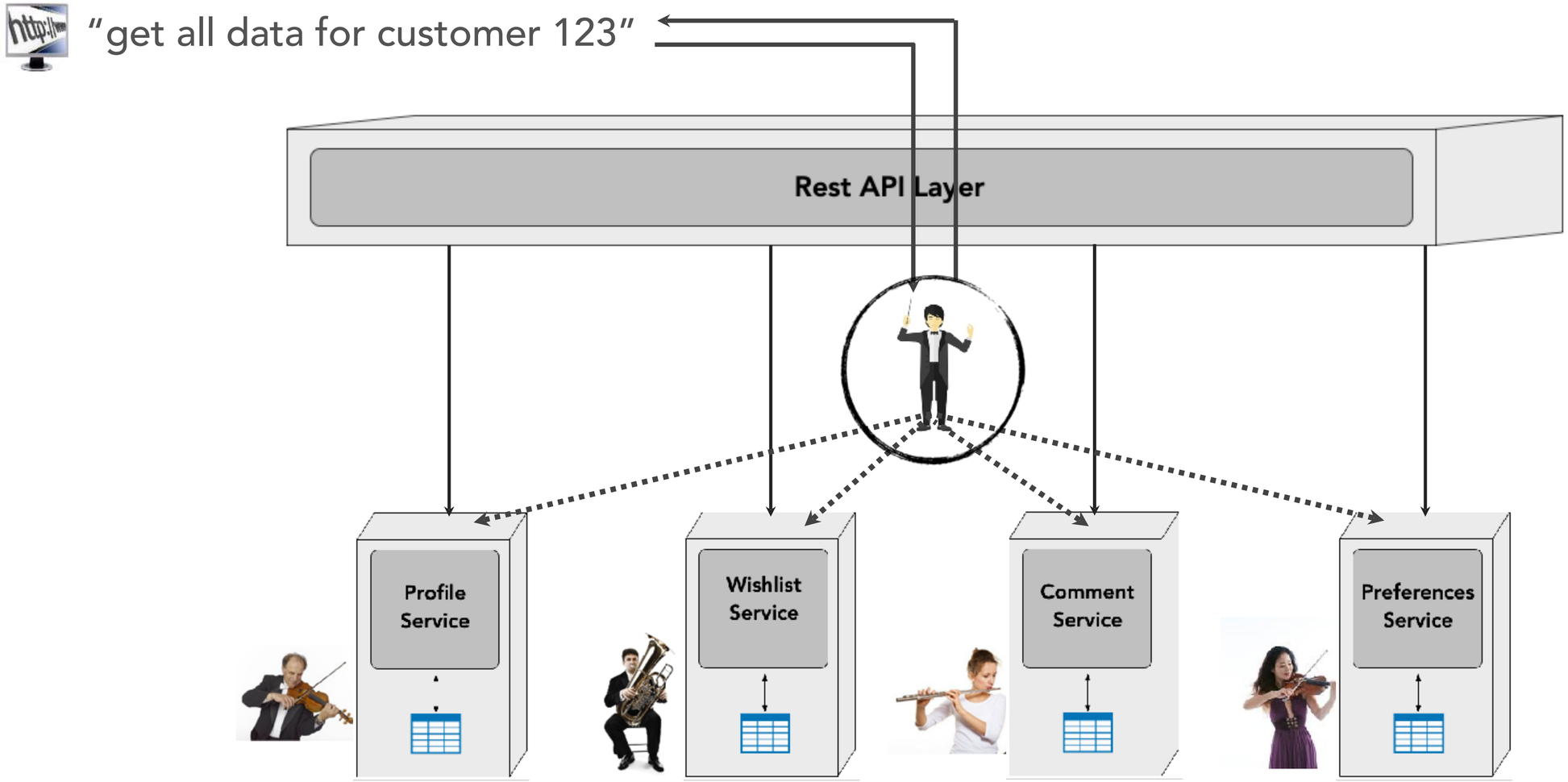
service types

orchestration services



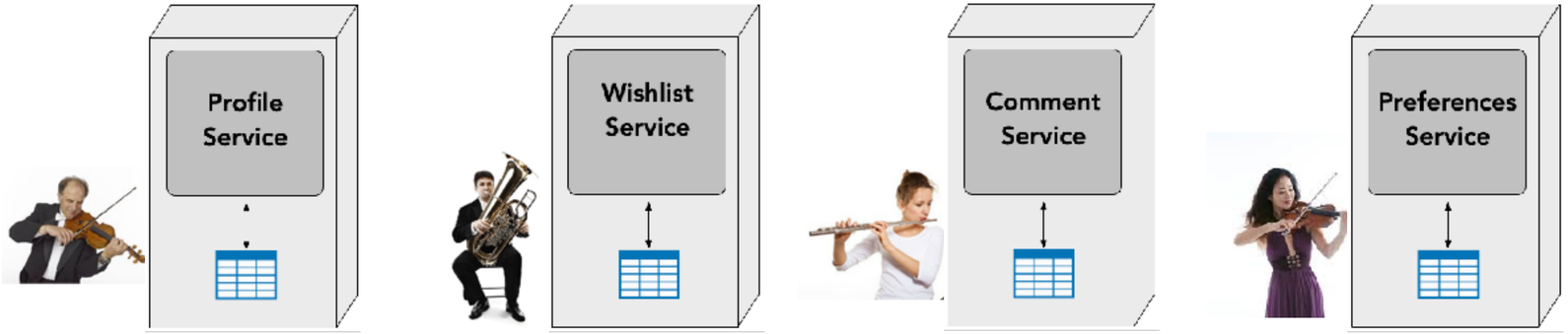
service types

orchestration services



service types

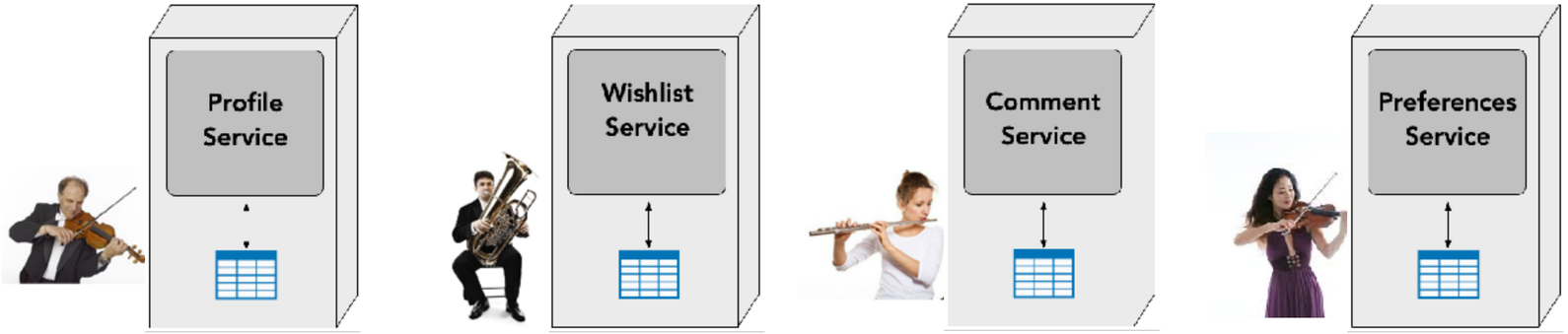
orchestration services



service types

orchestration services

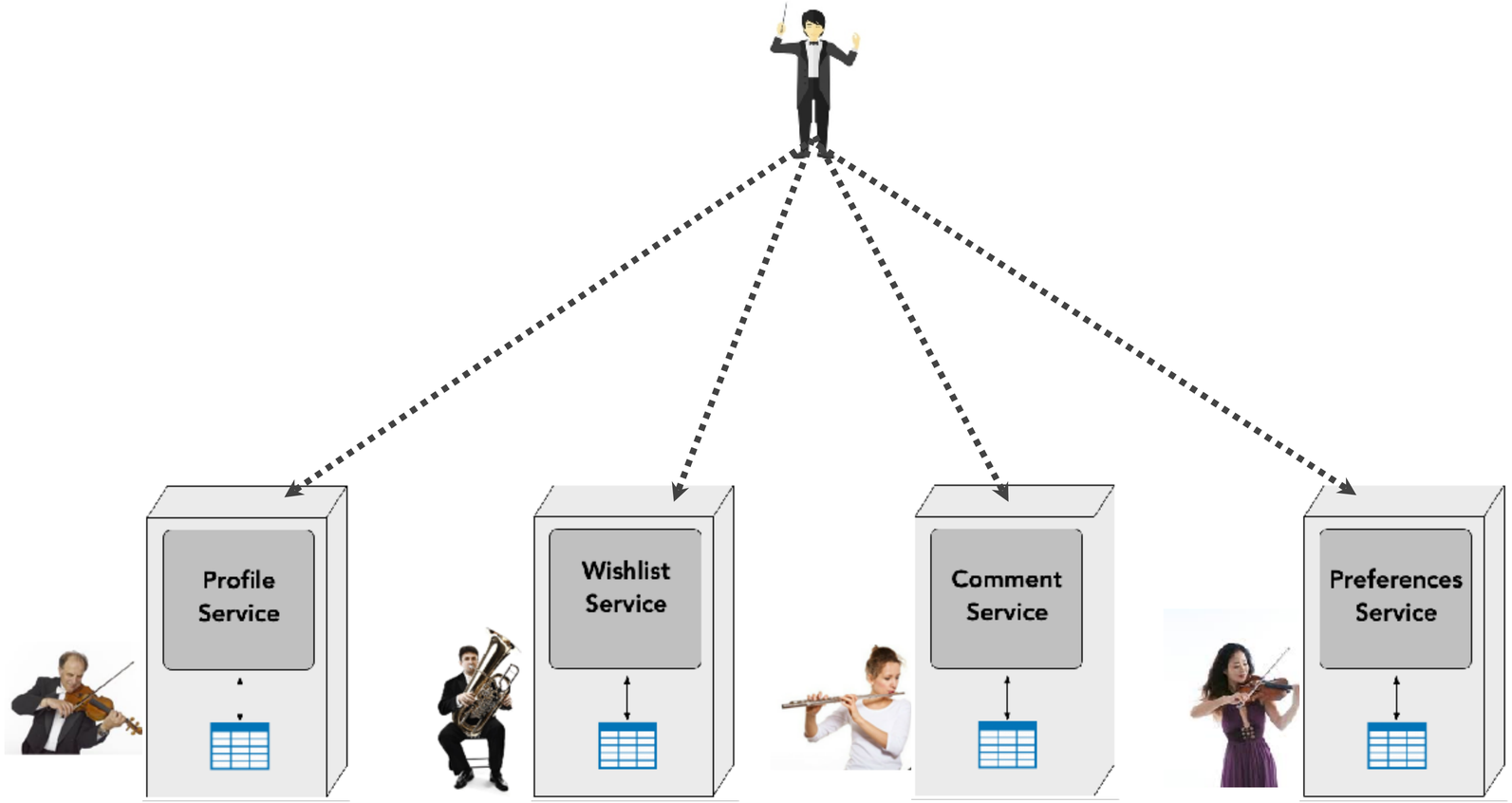
1. multicasting logic



service types

orchestration services

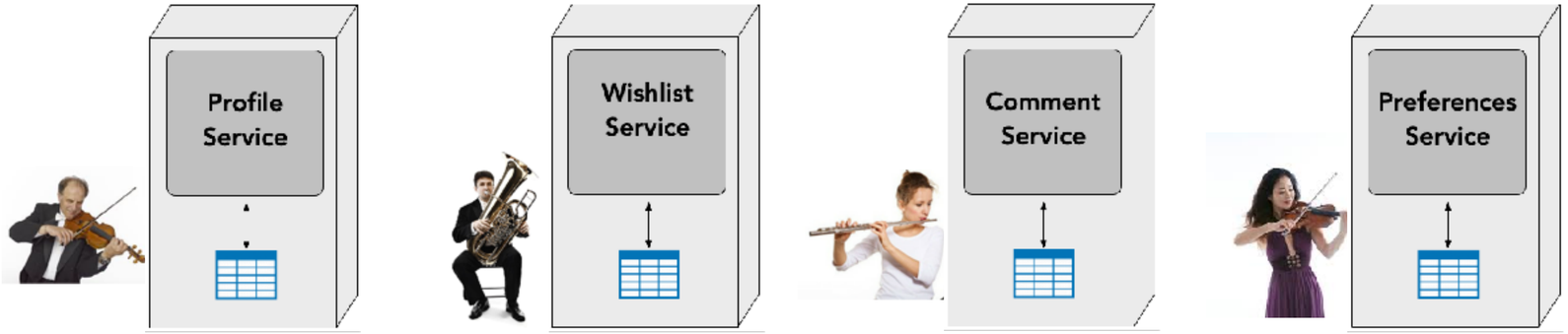
1. multicasting logic



service types

orchestration services

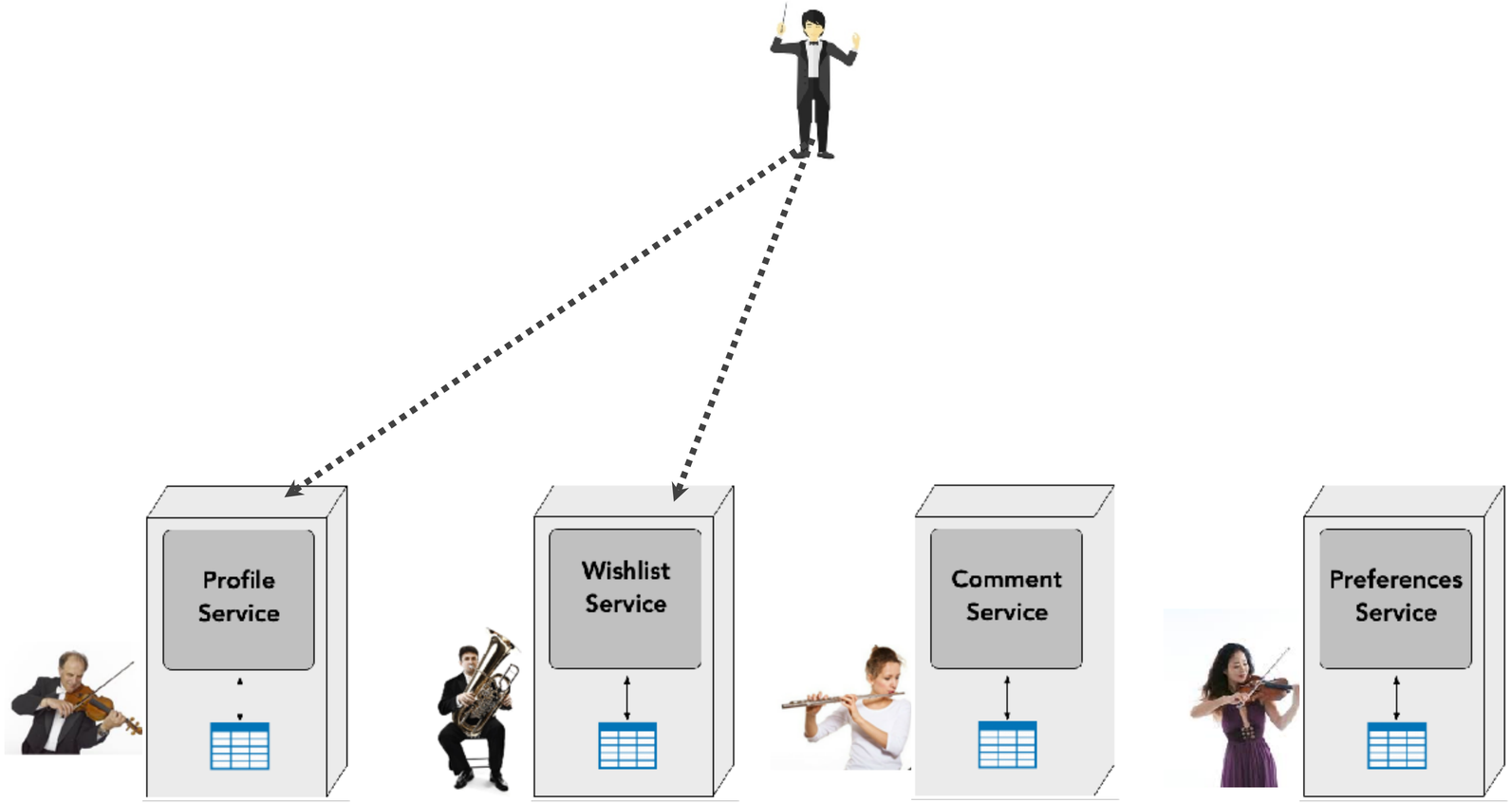
1. multicasting logic



service types

orchestration services

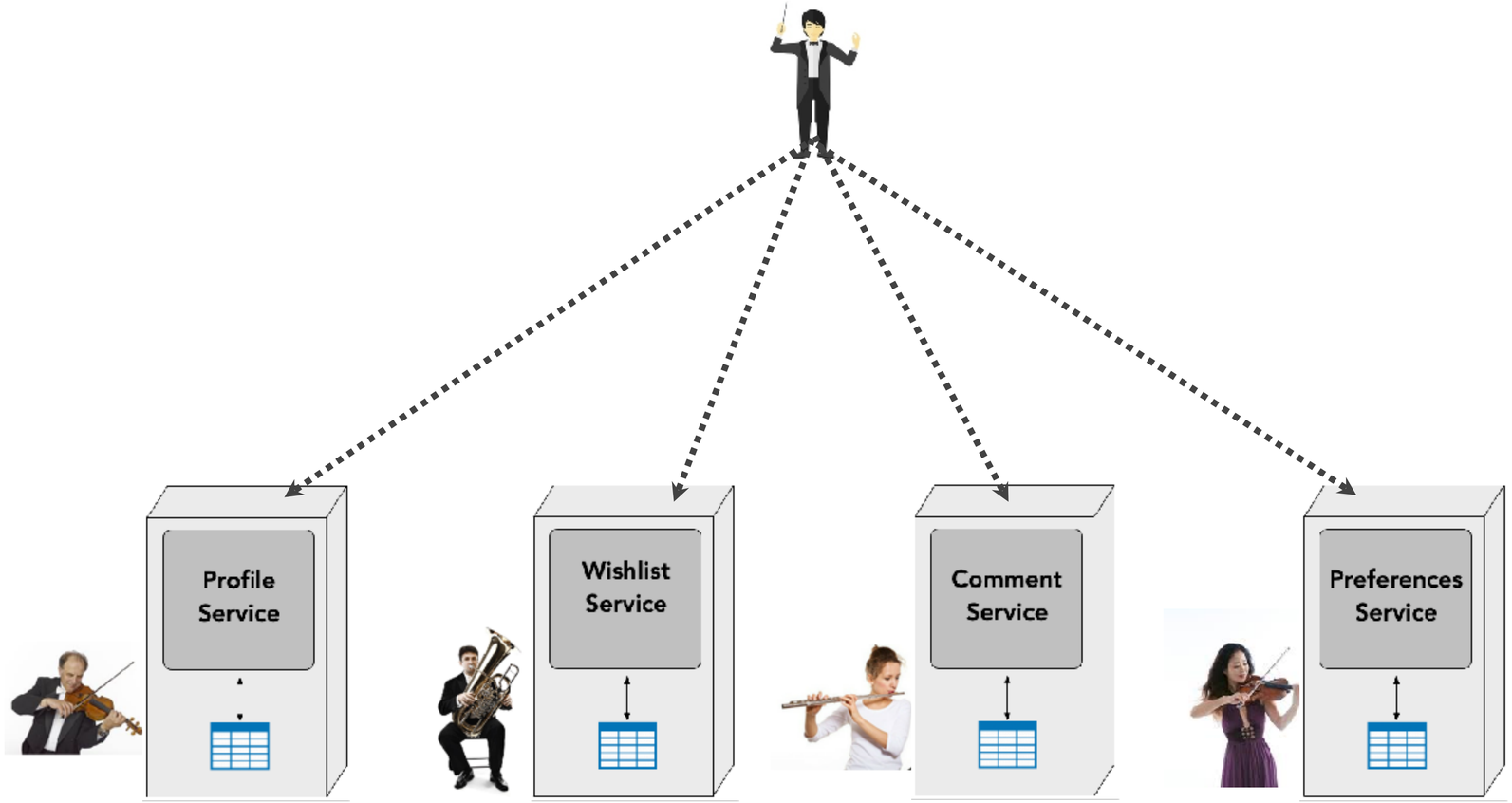
1. multicasting logic



service types

orchestration services

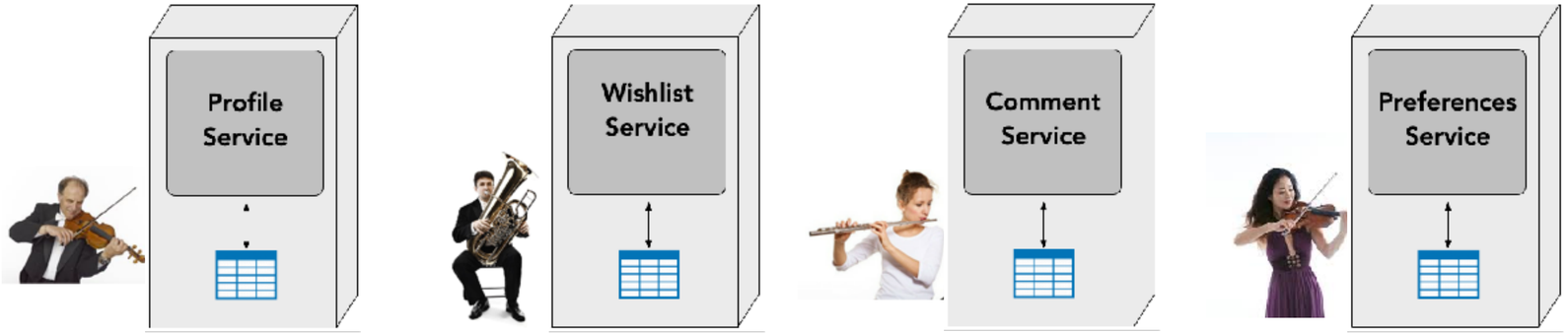
1. multicasting logic



service types

orchestration services

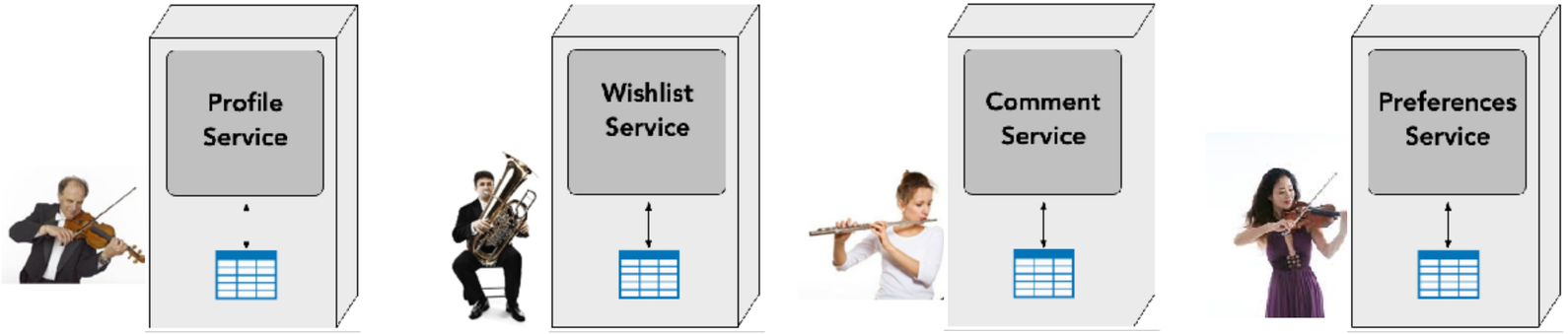
1. multicasting logic



service types

orchestration services

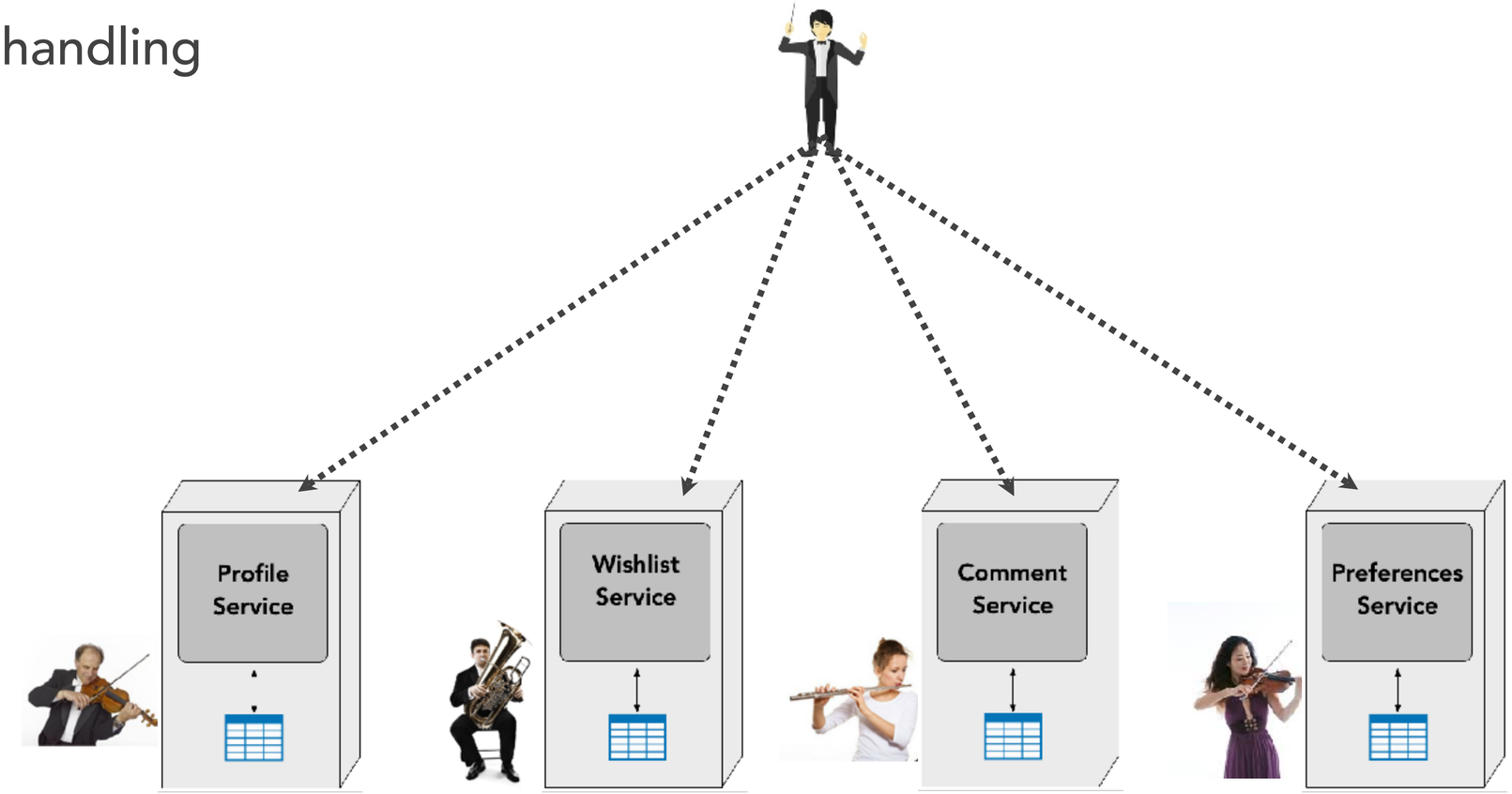
1. multicasting logic
2. error handling



service types

orchestration services

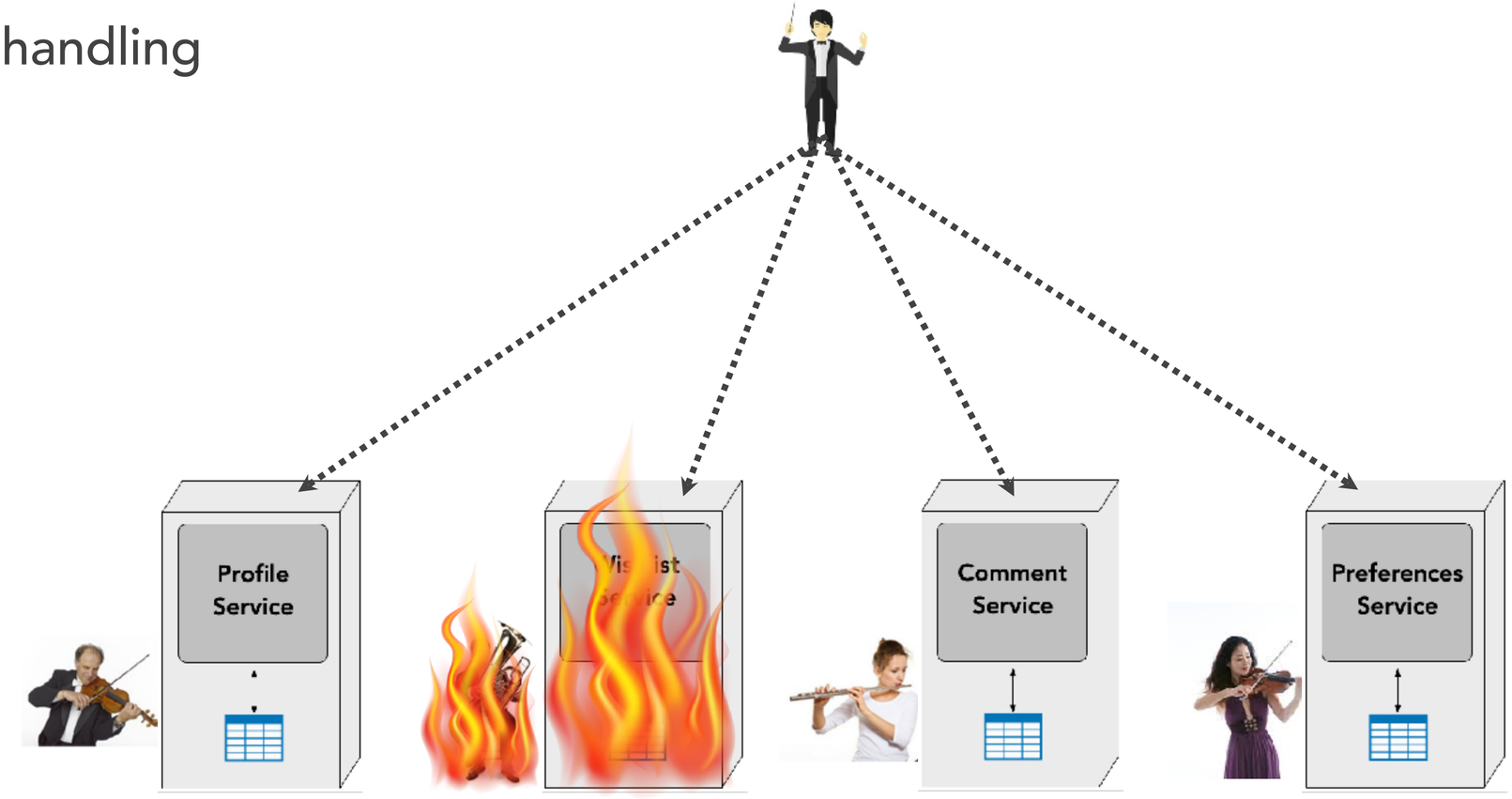
1. multicasting logic
2. error handling



service types

orchestration services

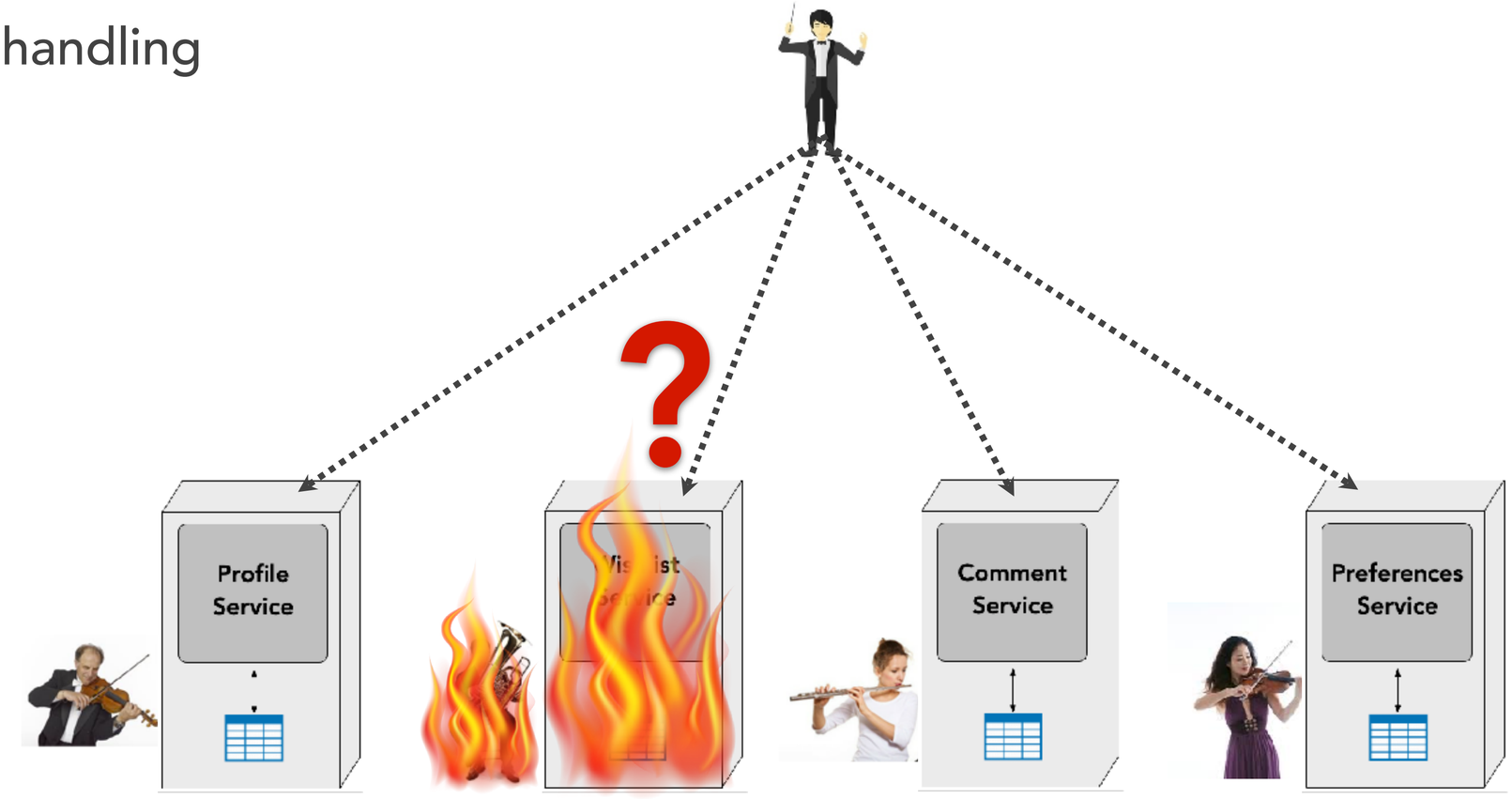
1. multicasting logic
2. error handling



service types

orchestration services

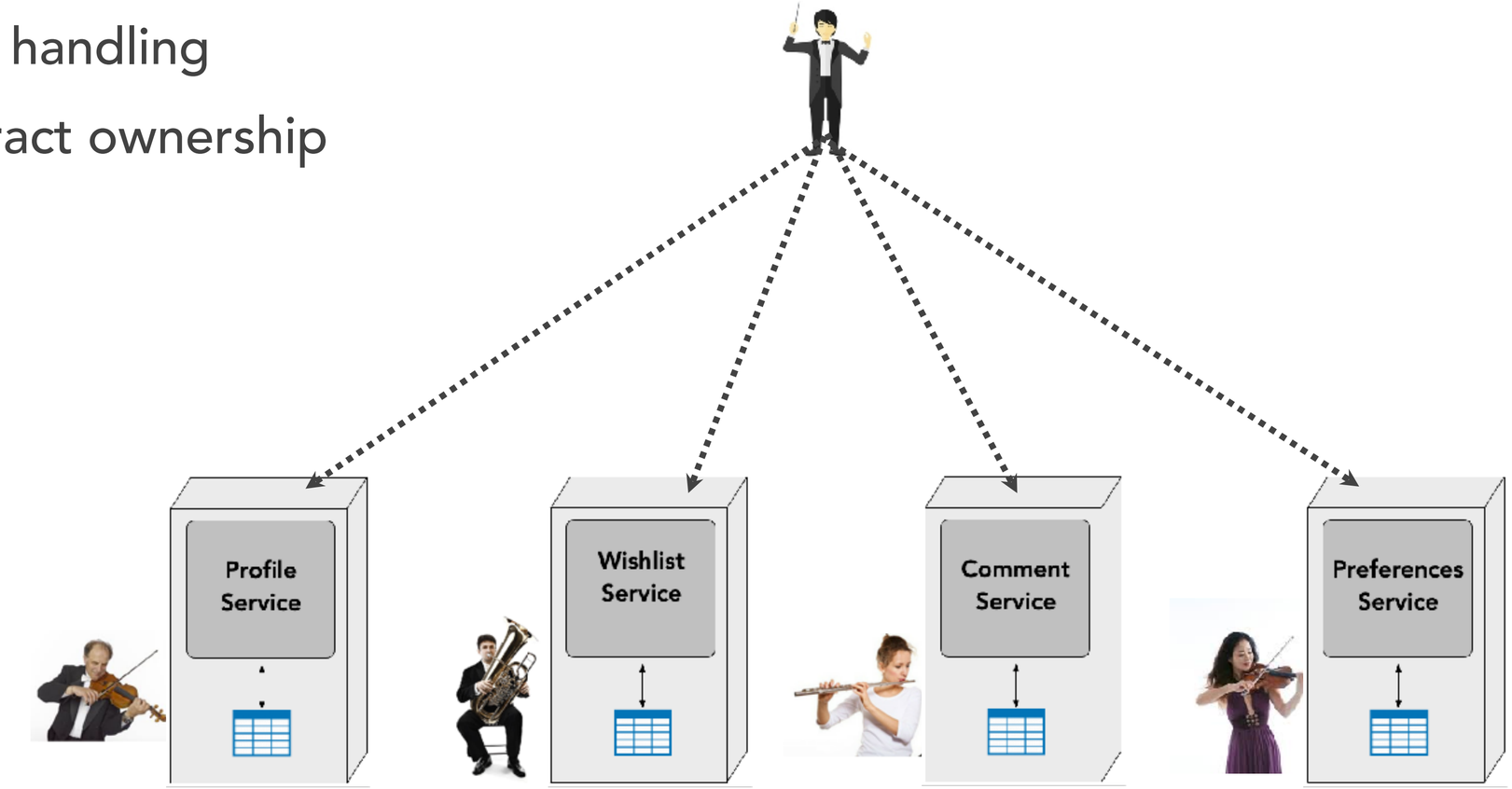
1. multicasting logic
2. error handling



service types

orchestration services

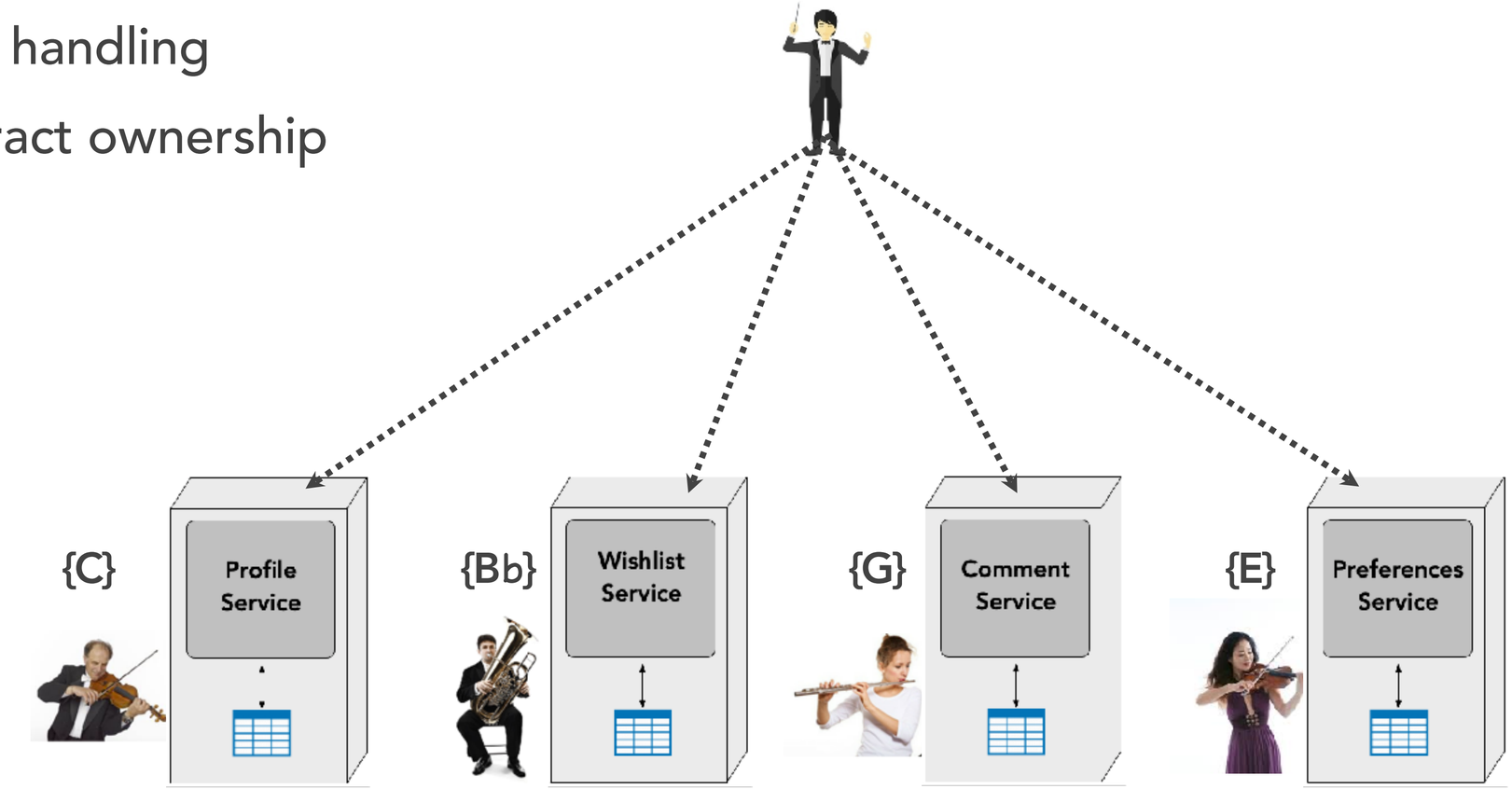
1. multicasting logic
2. error handling
3. contract ownership



service types

orchestration services

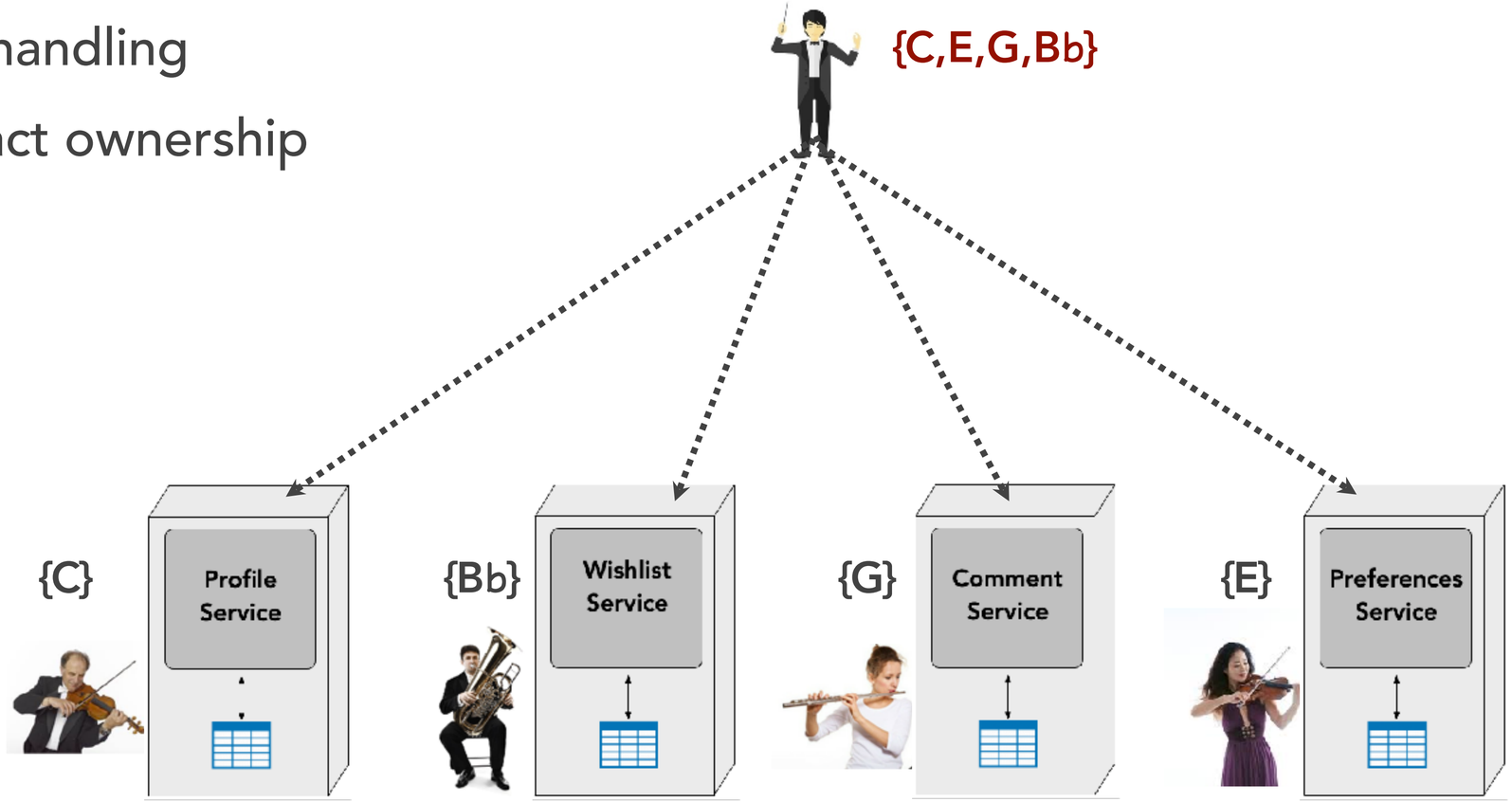
1. multicasting logic
2. error handling
3. contract ownership



service types

orchestration services

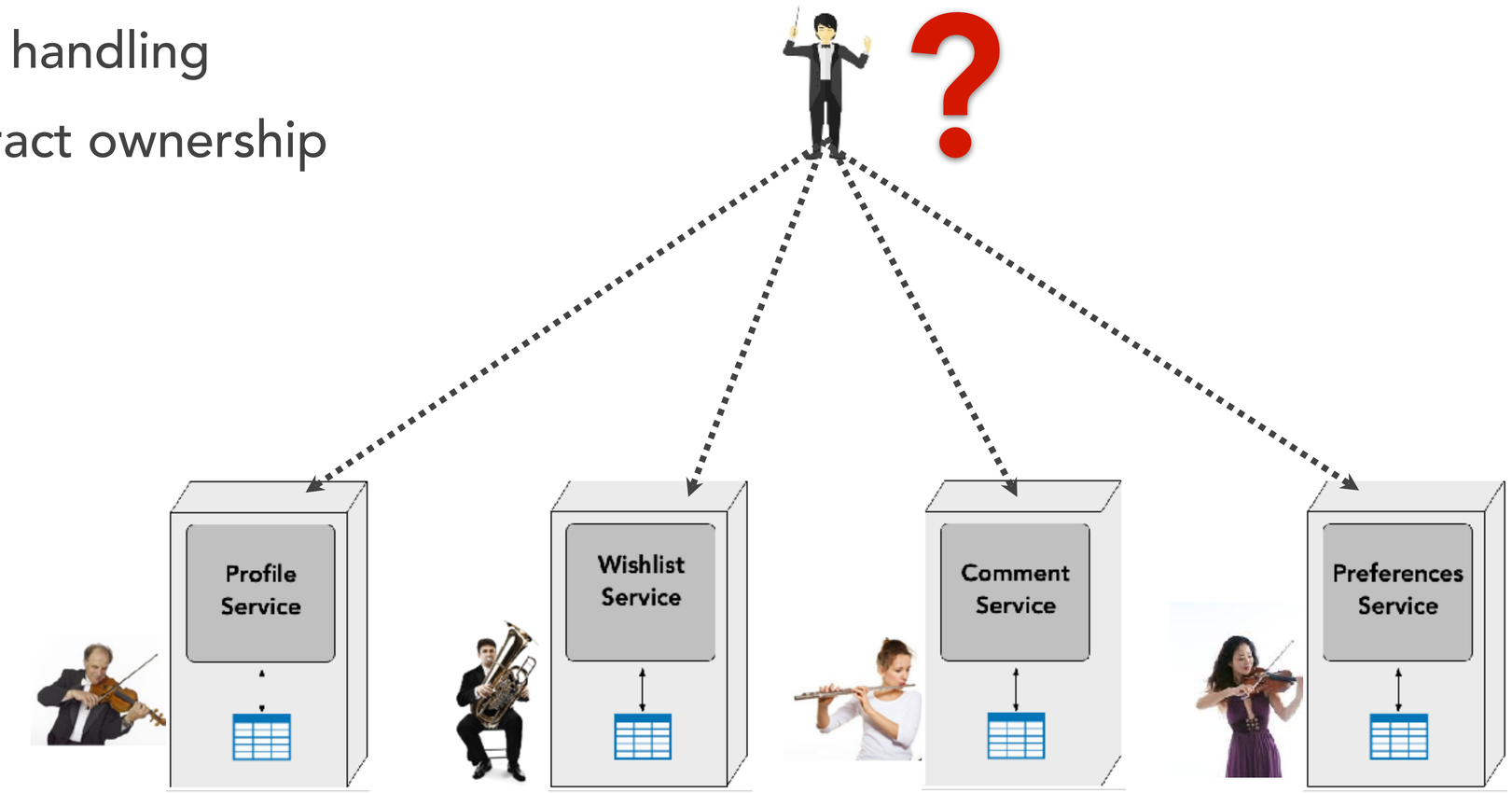
1. multicasting logic
2. error handling
3. contract ownership



service types

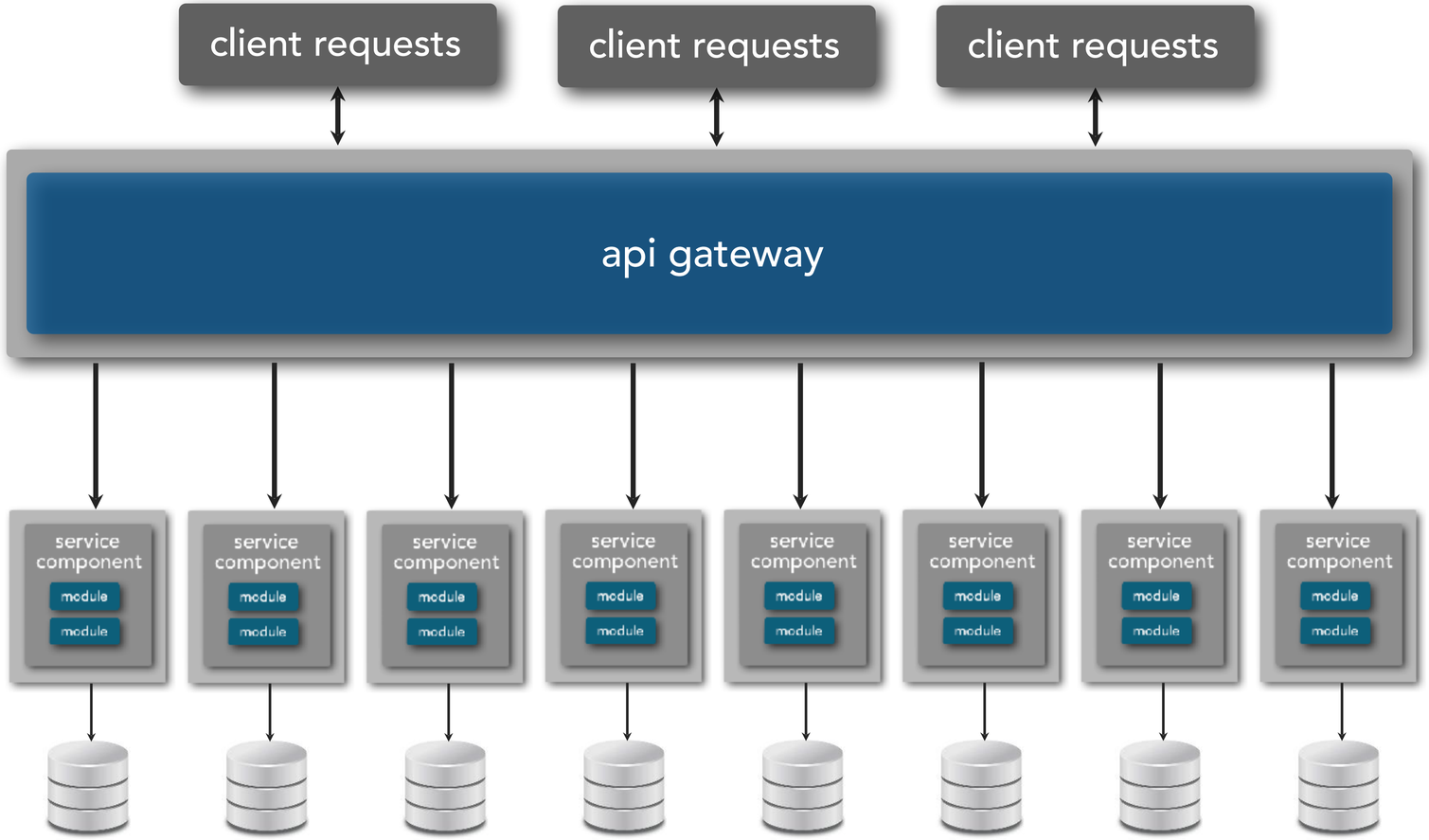
orchestration services

1. multicasting logic
2. error handling
3. contract ownership



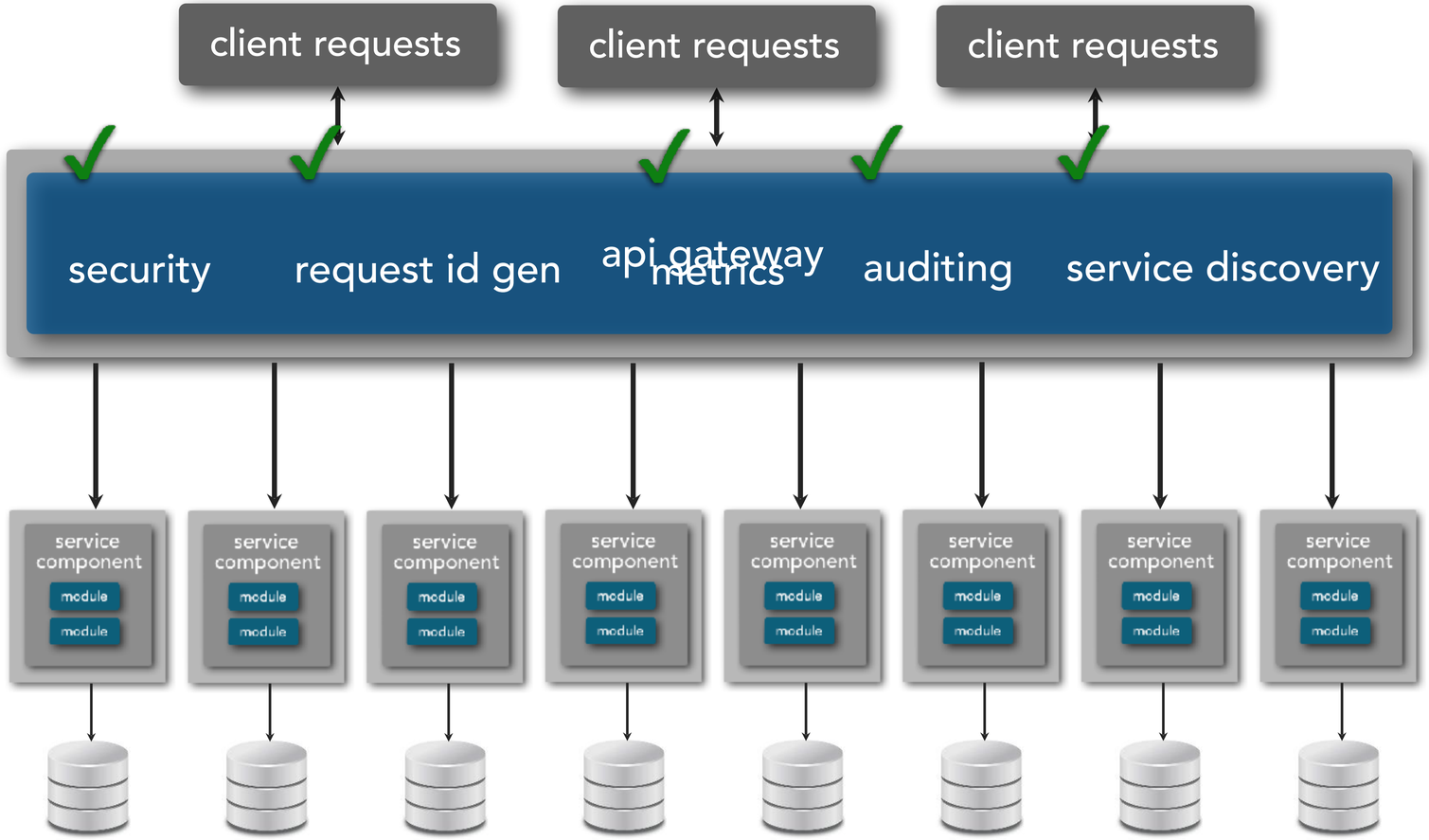
service types

orchestration services



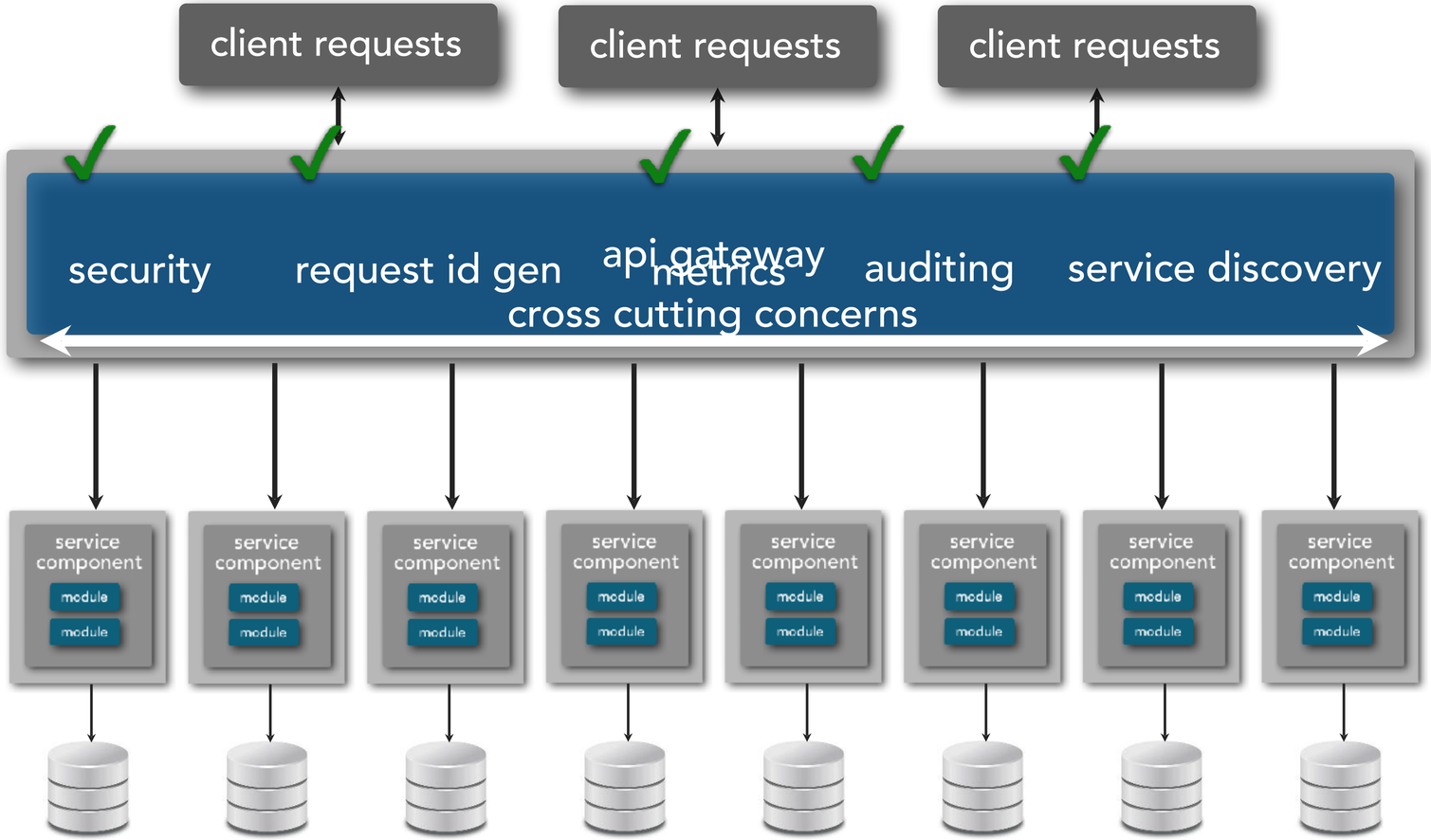
service types

orchestration services



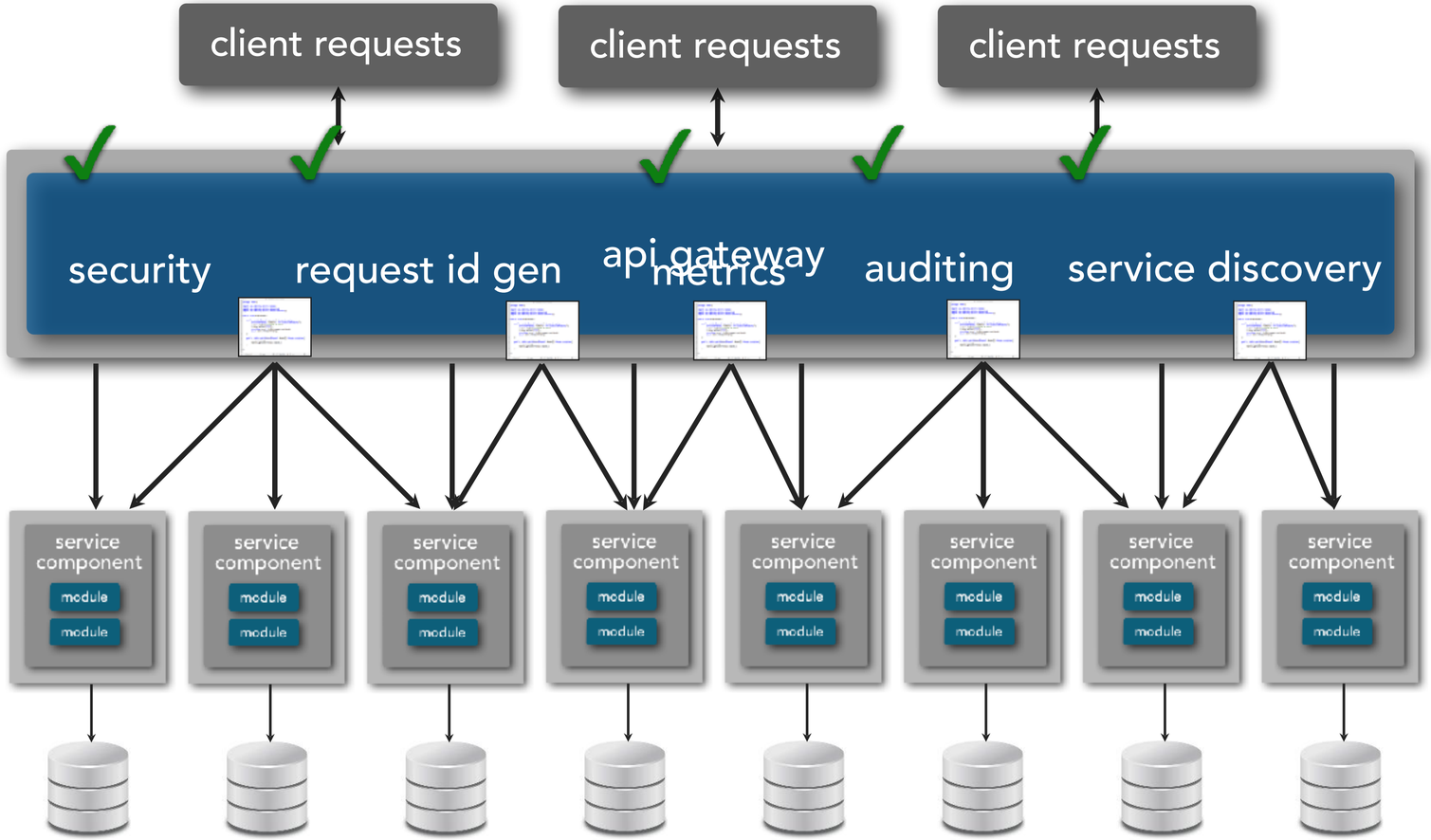
service types

orchestration services



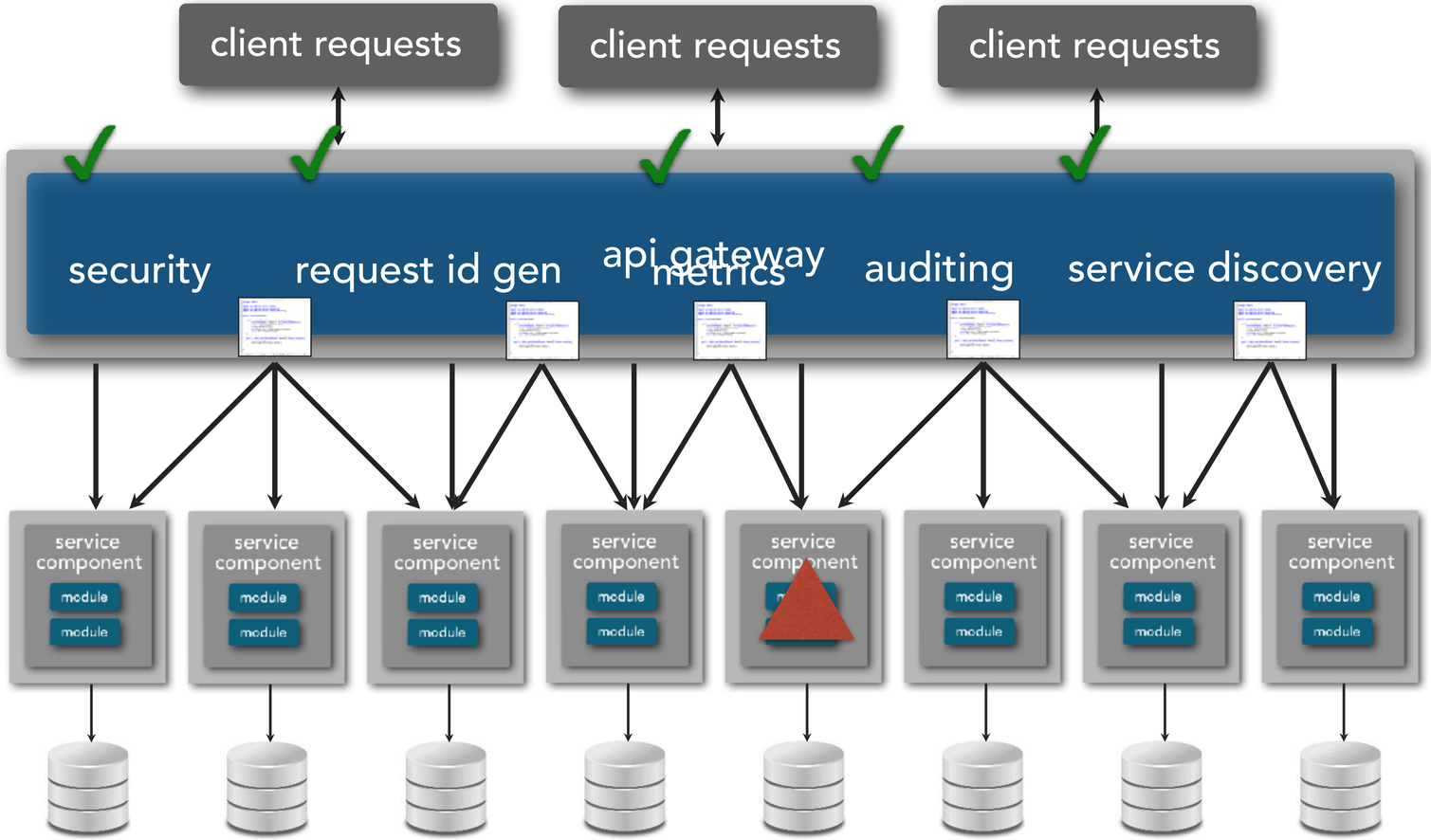
service types

orchestration services



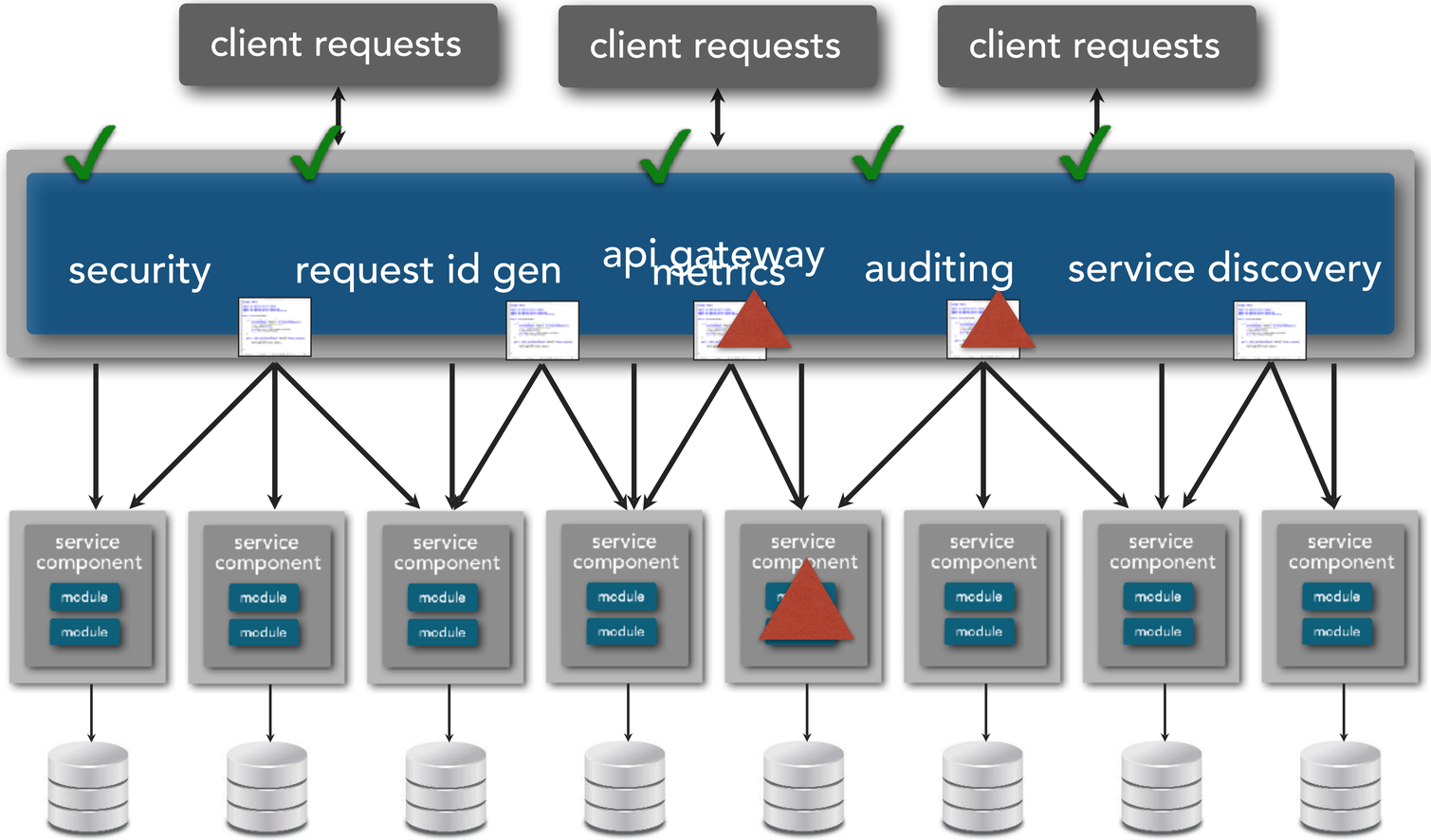
service types

orchestration services



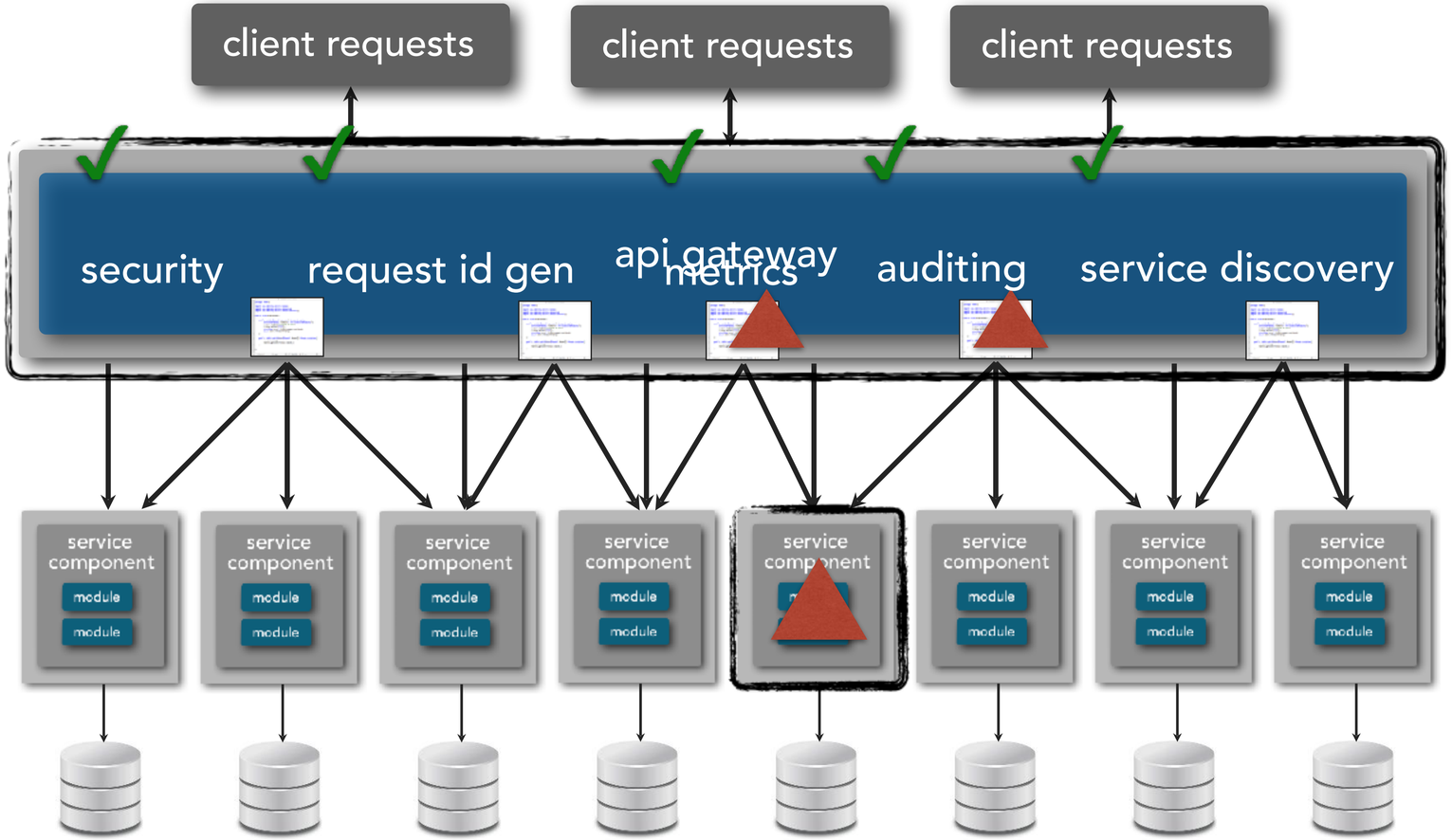
service types

orchestration services



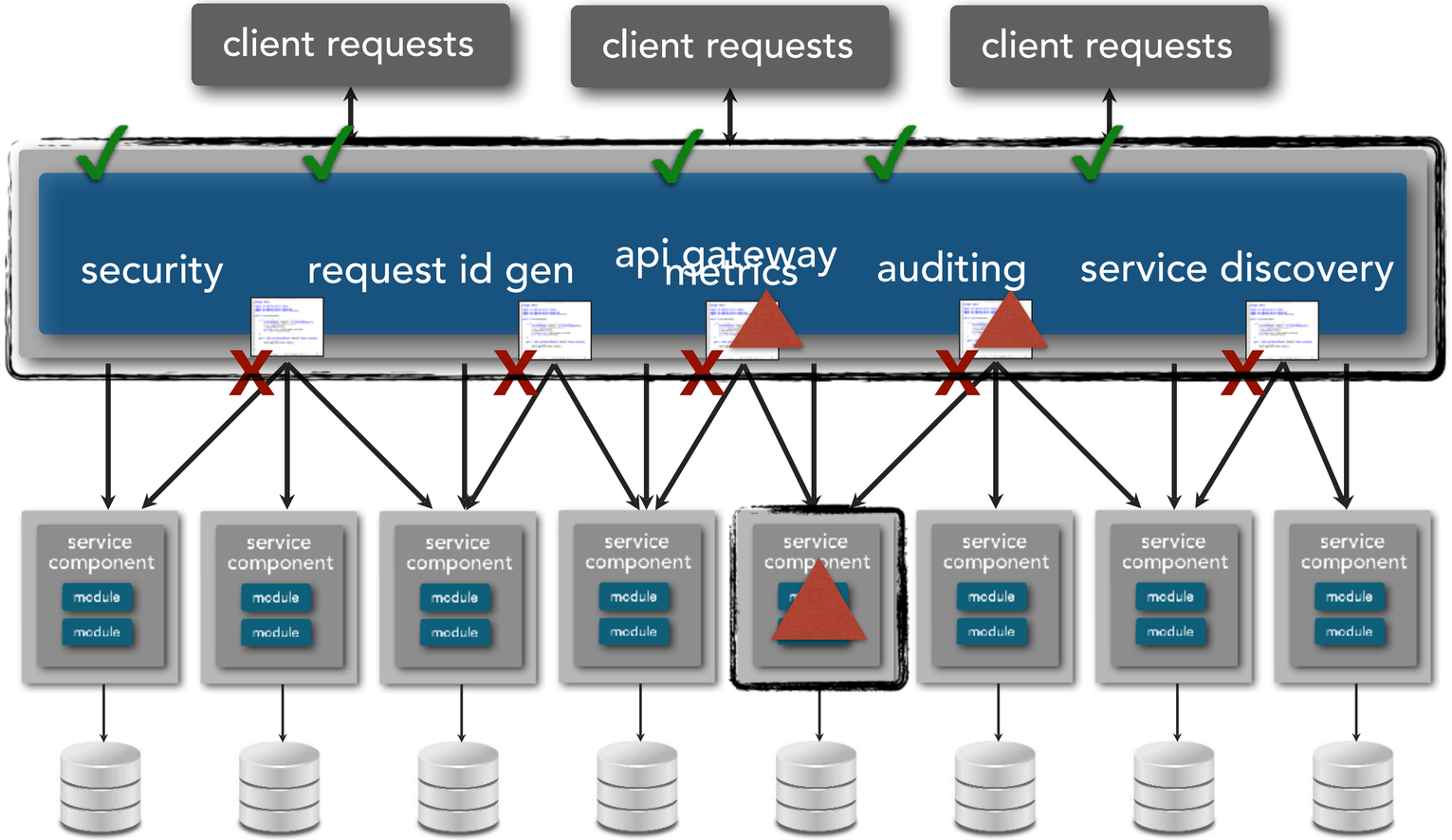
service types

orchestration services



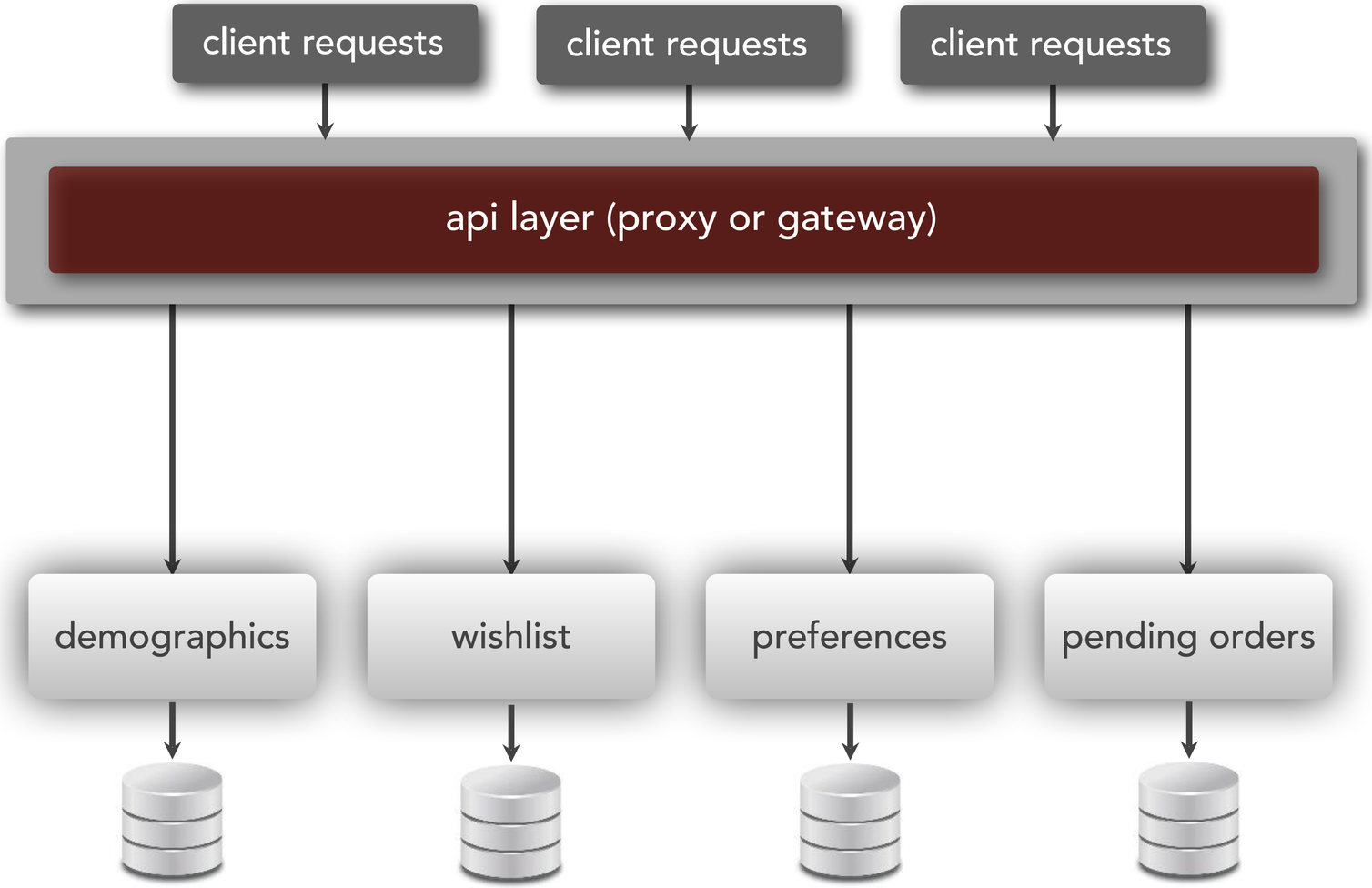
service types

orchestration services



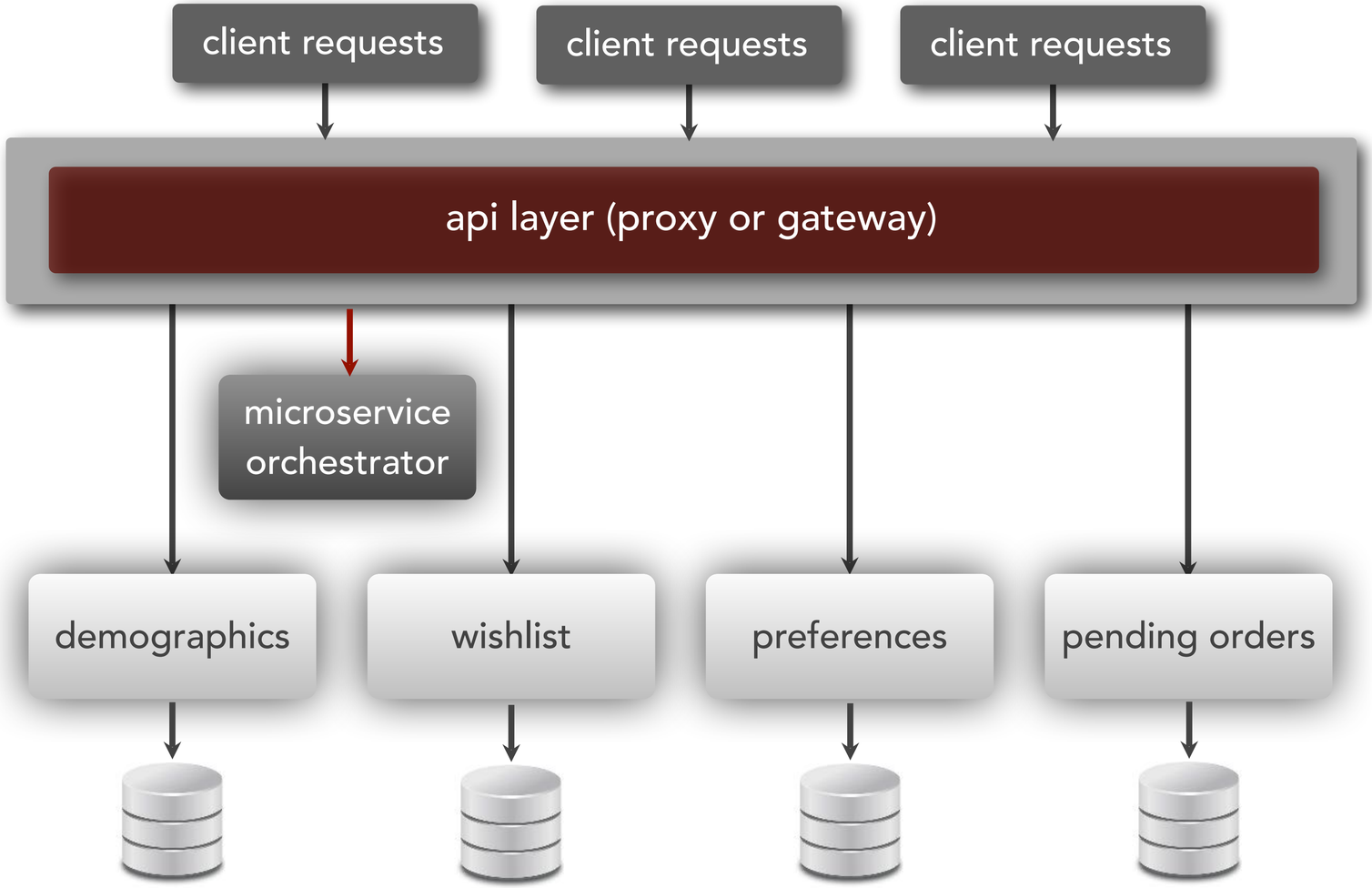
service types

orchestration services



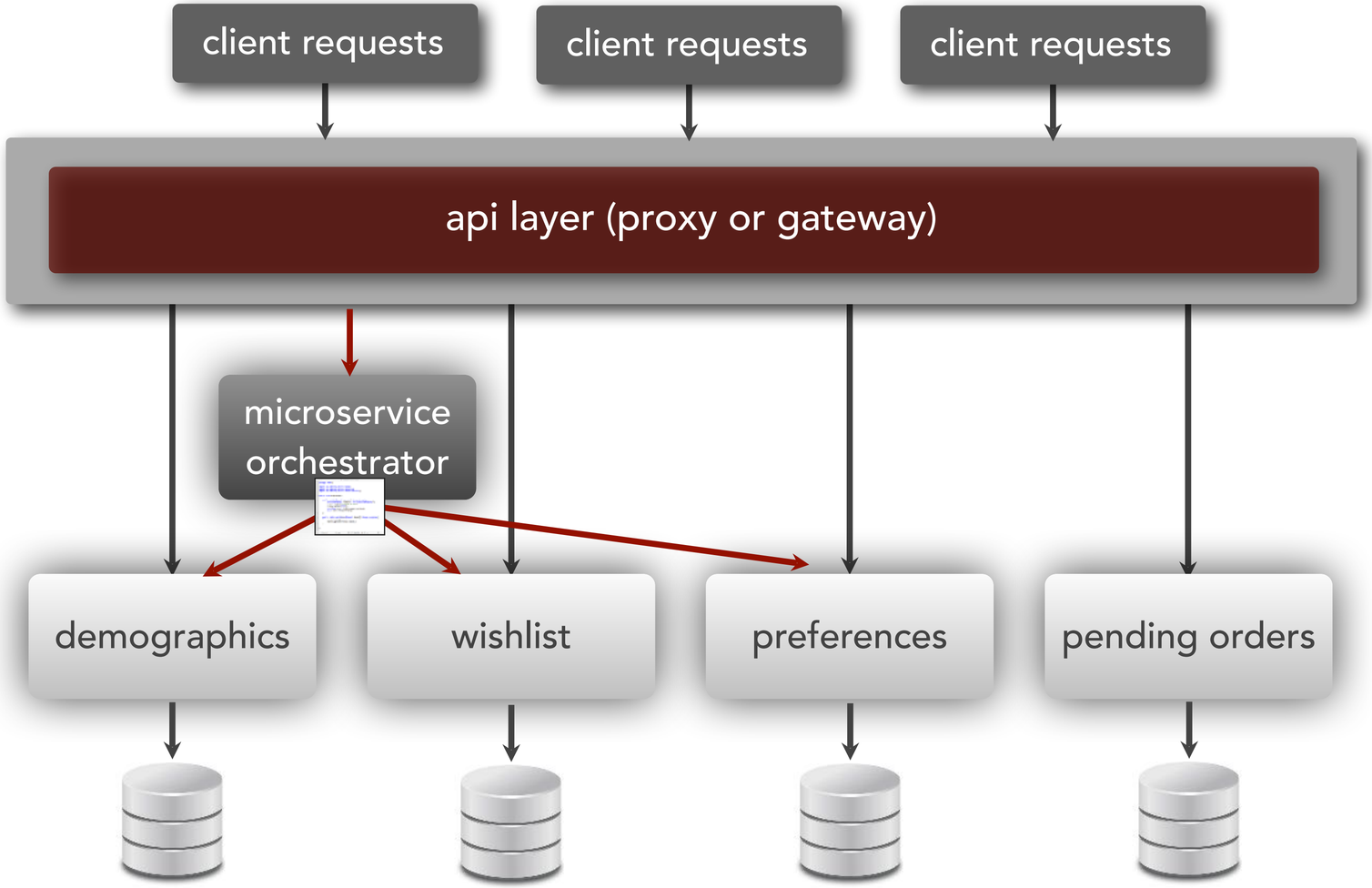
service types

orchestration services



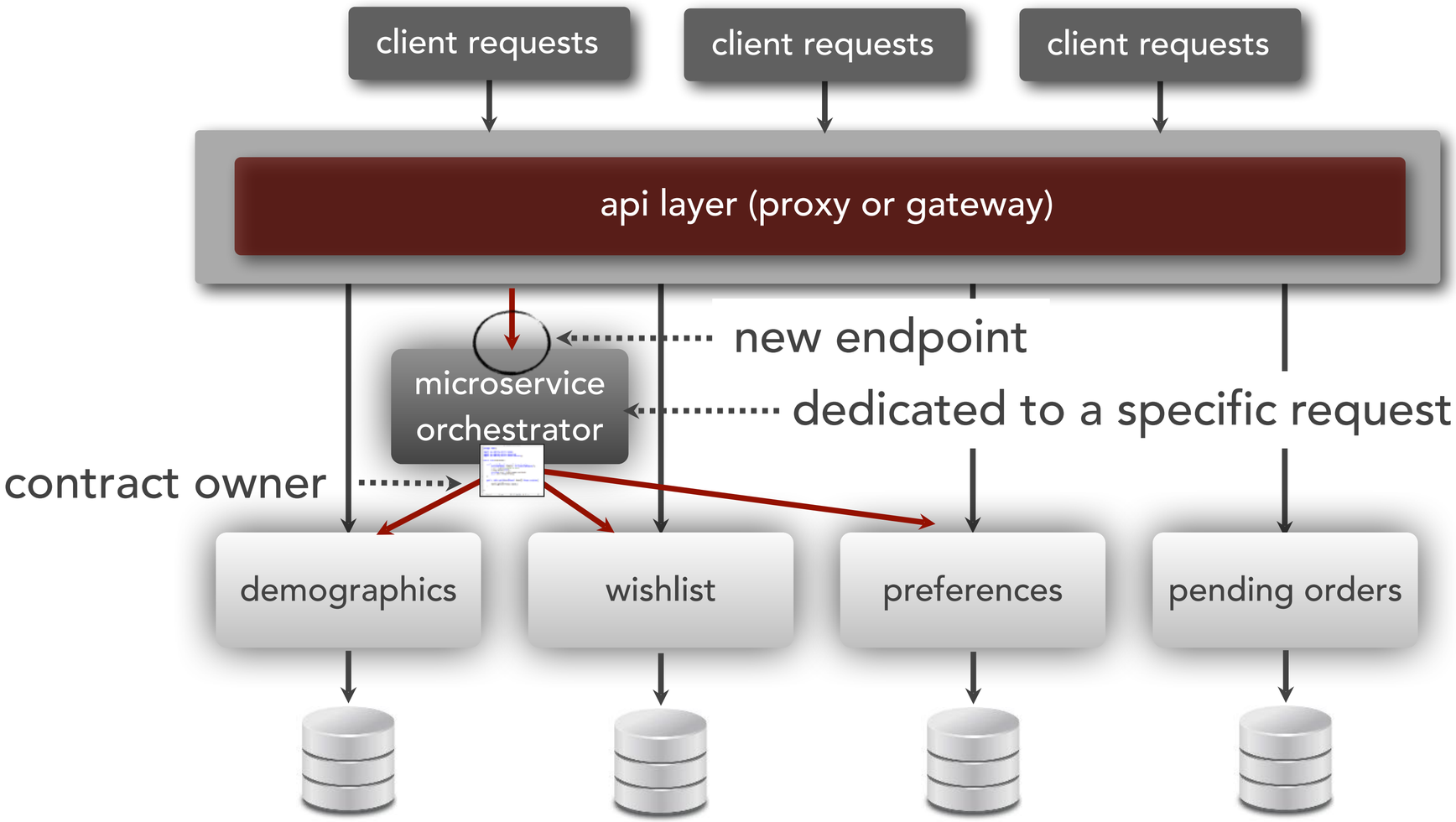
service types

orchestration services



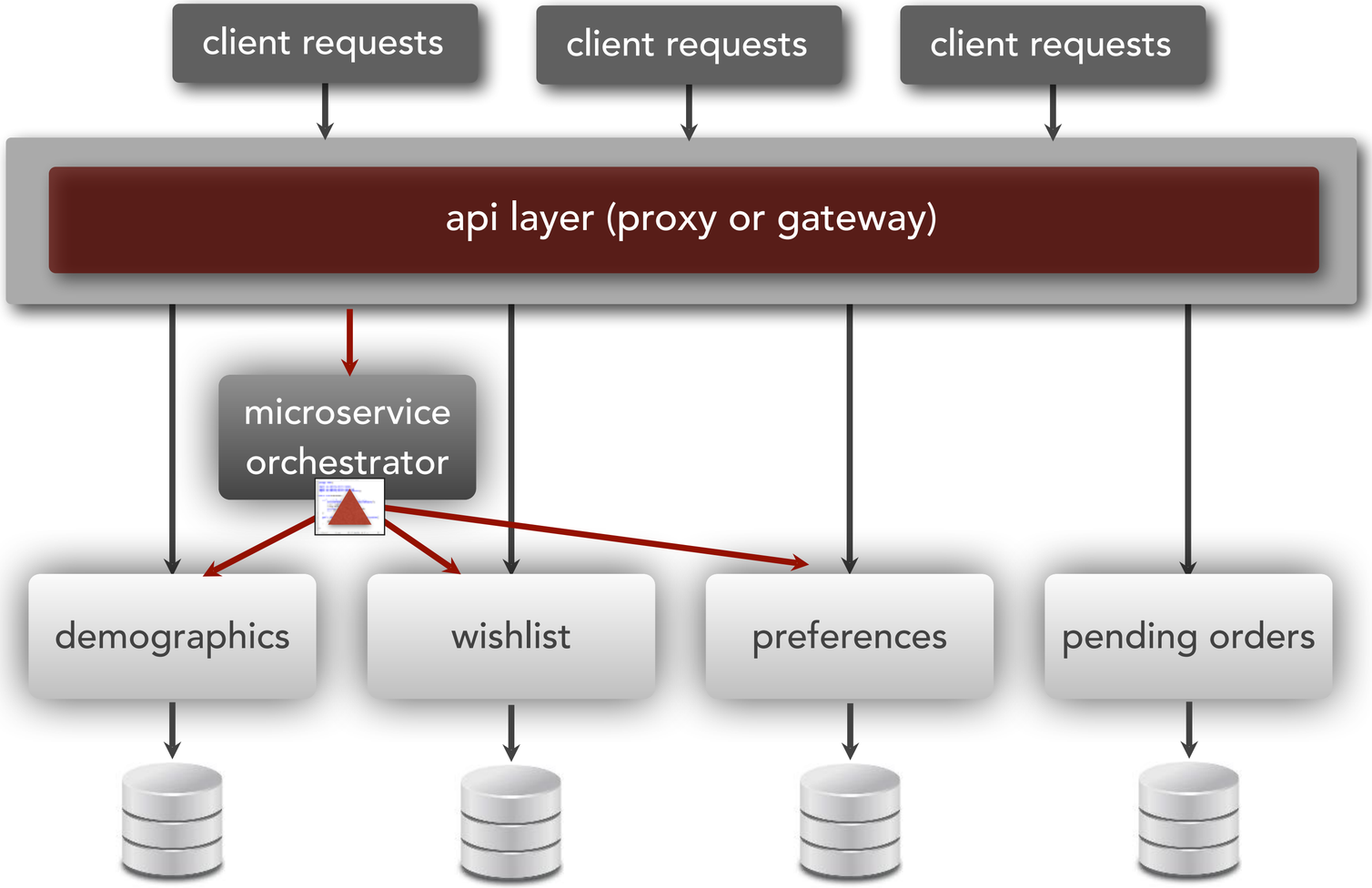
service types

orchestration services



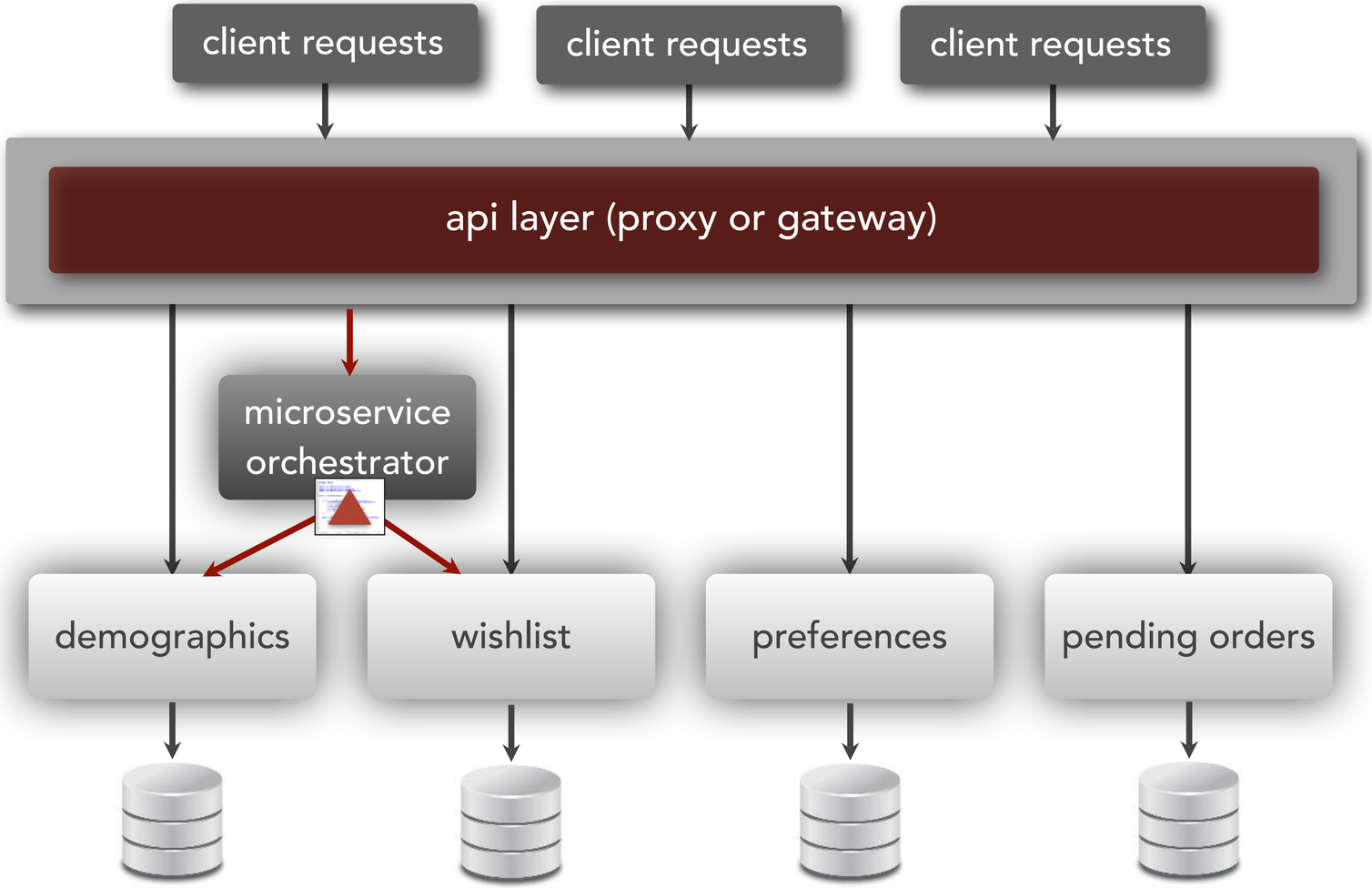
service types

orchestration services



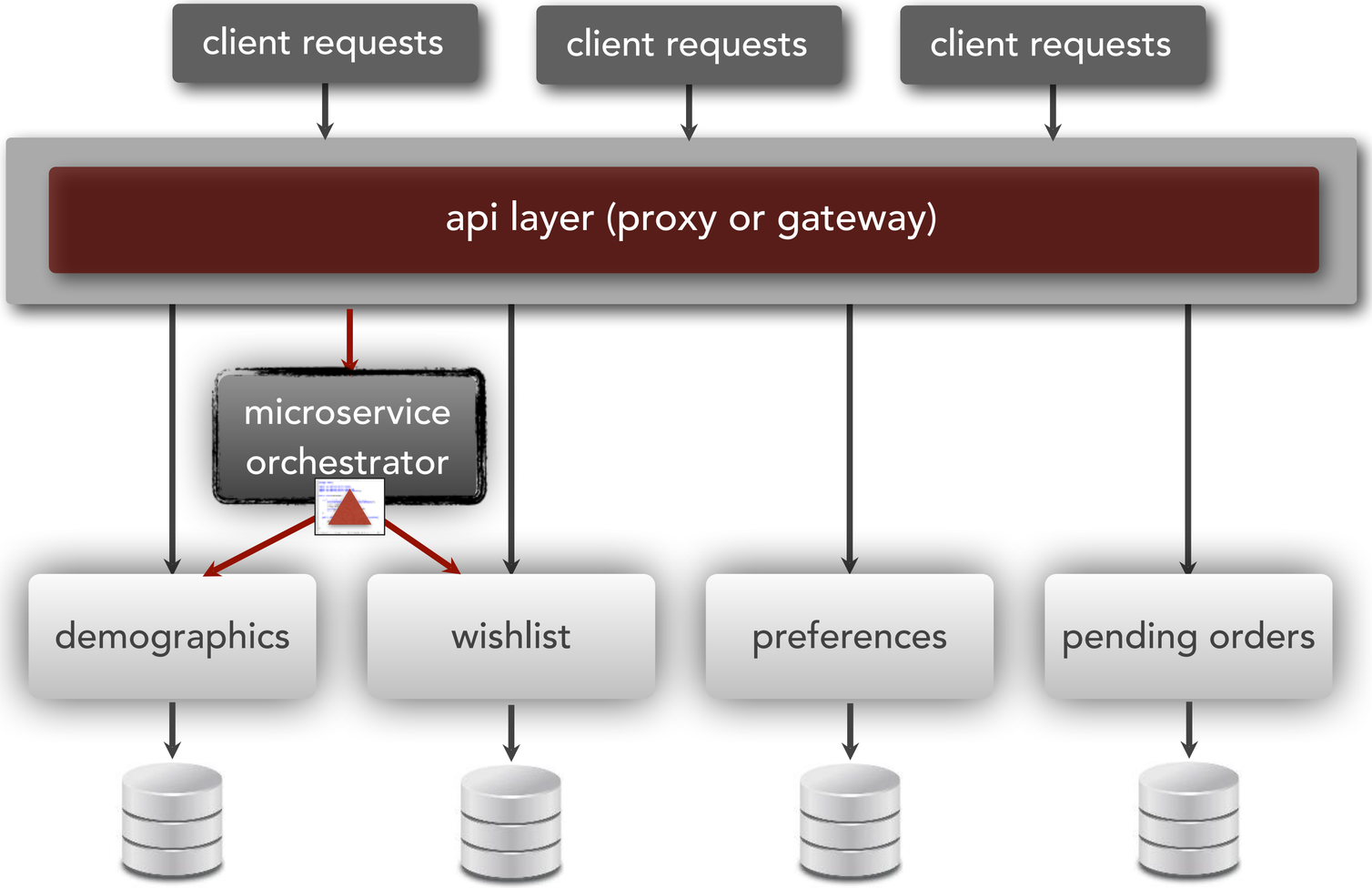
service types

orchestration services



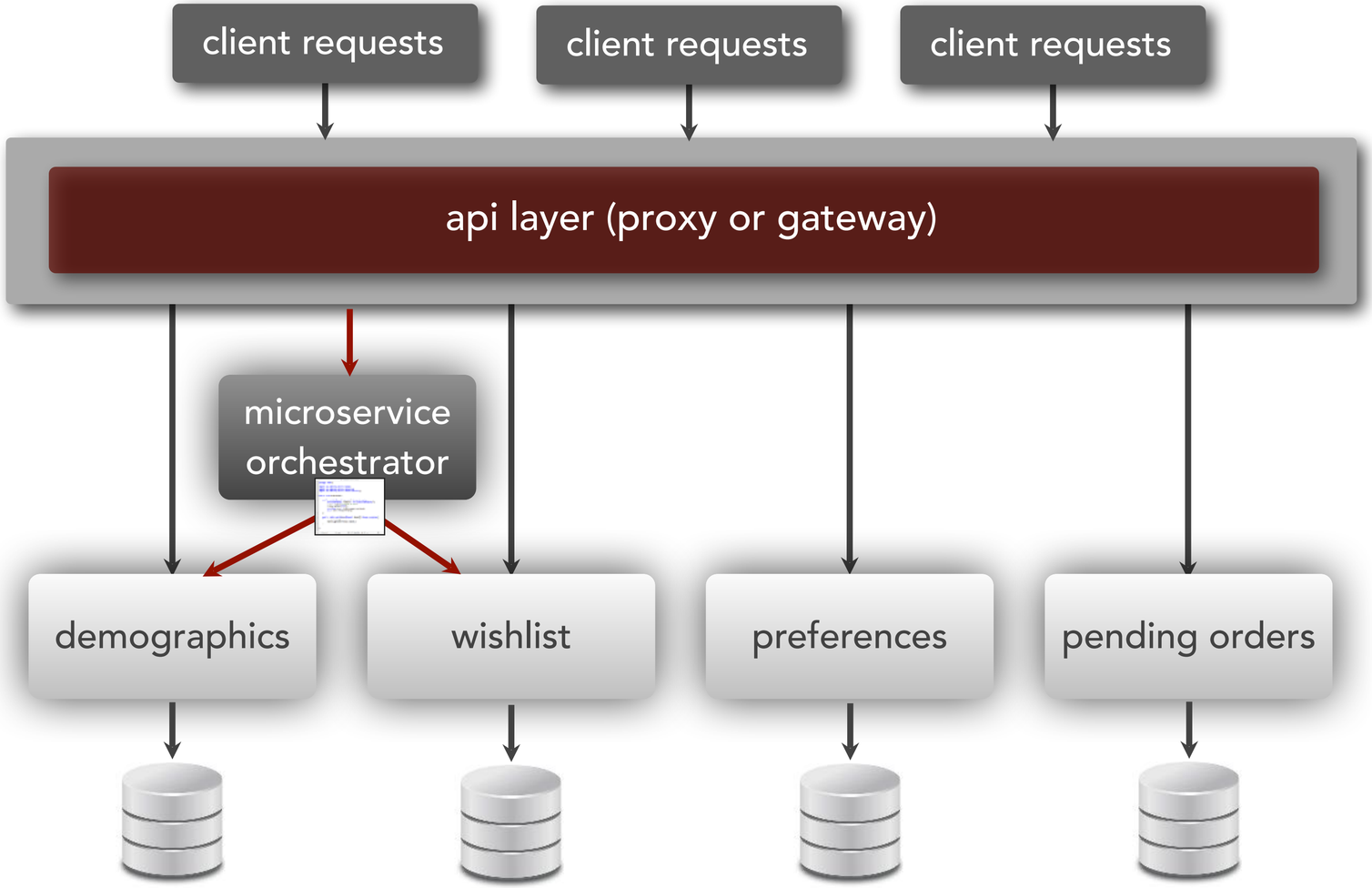
service types

orchestration services



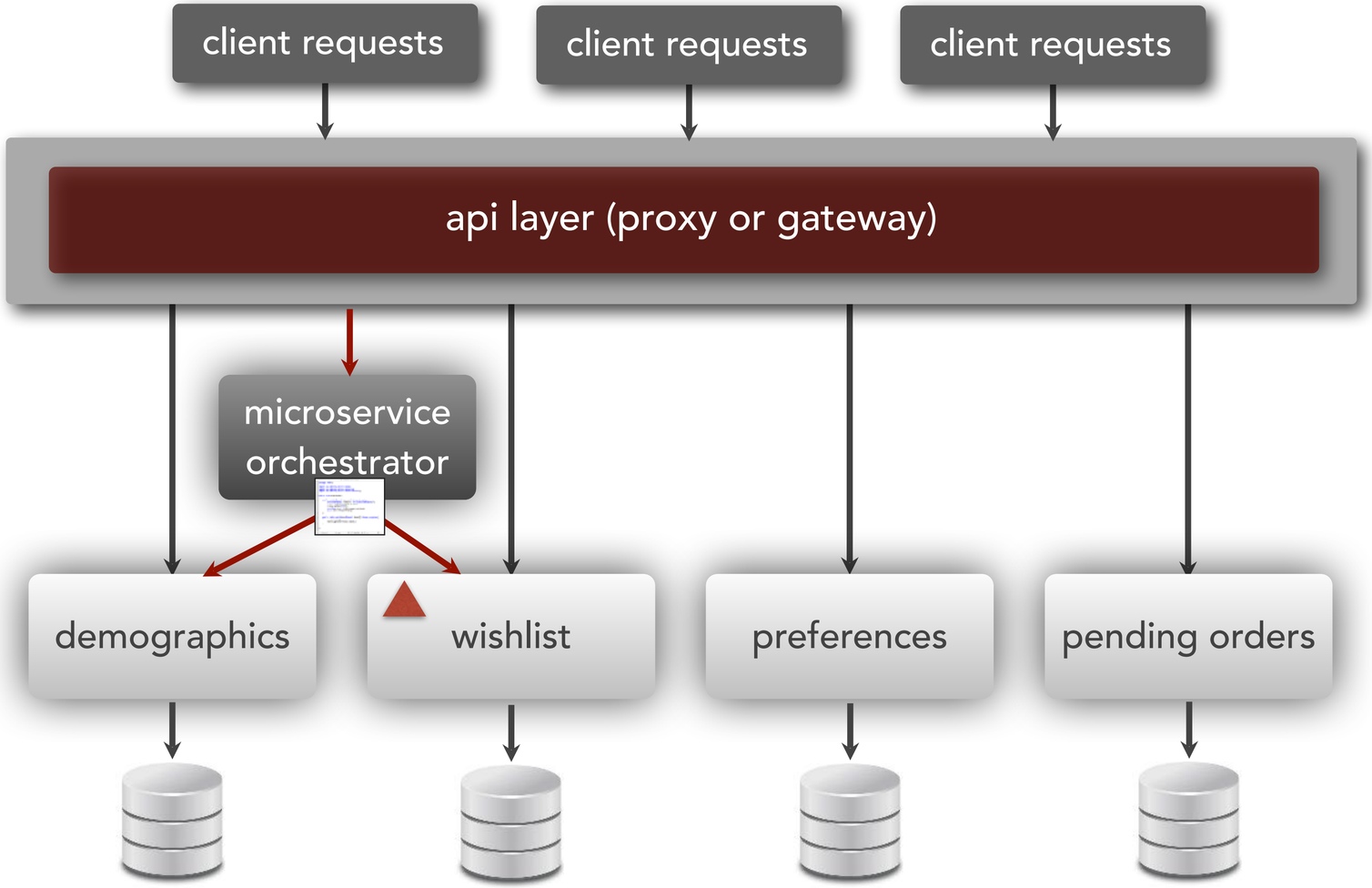
service types

orchestration services



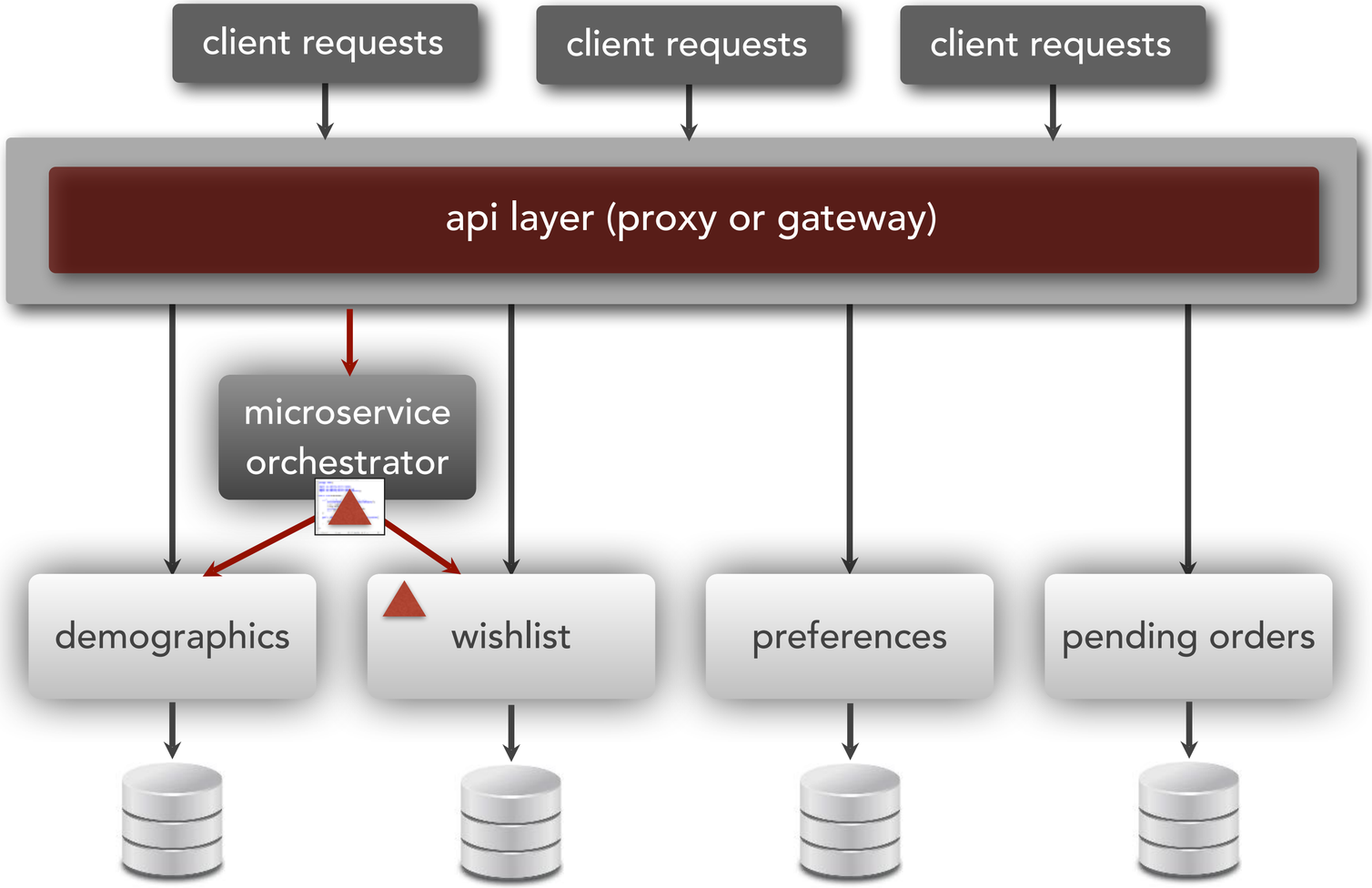
service types

orchestration services



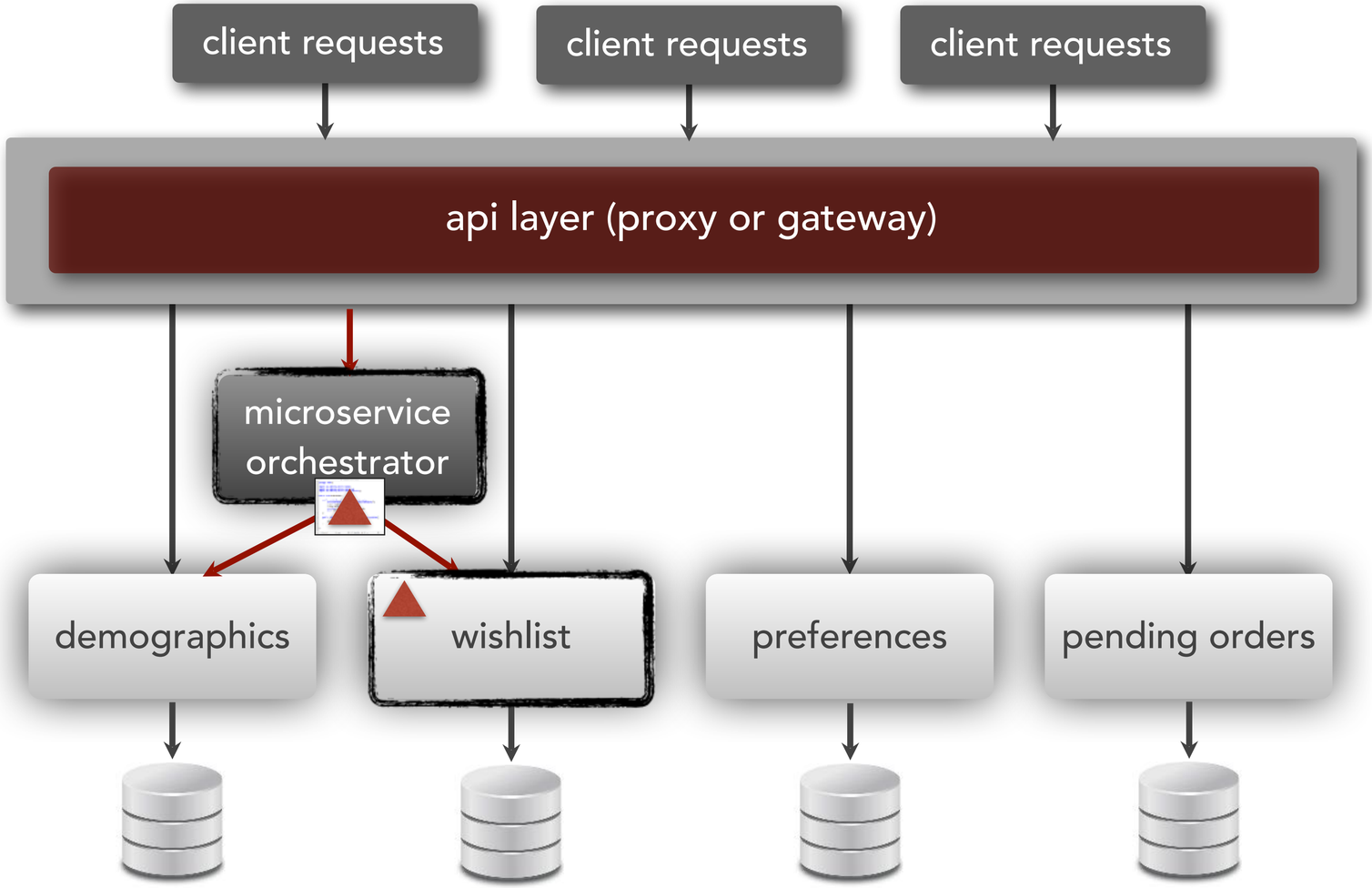
service types

orchestration services

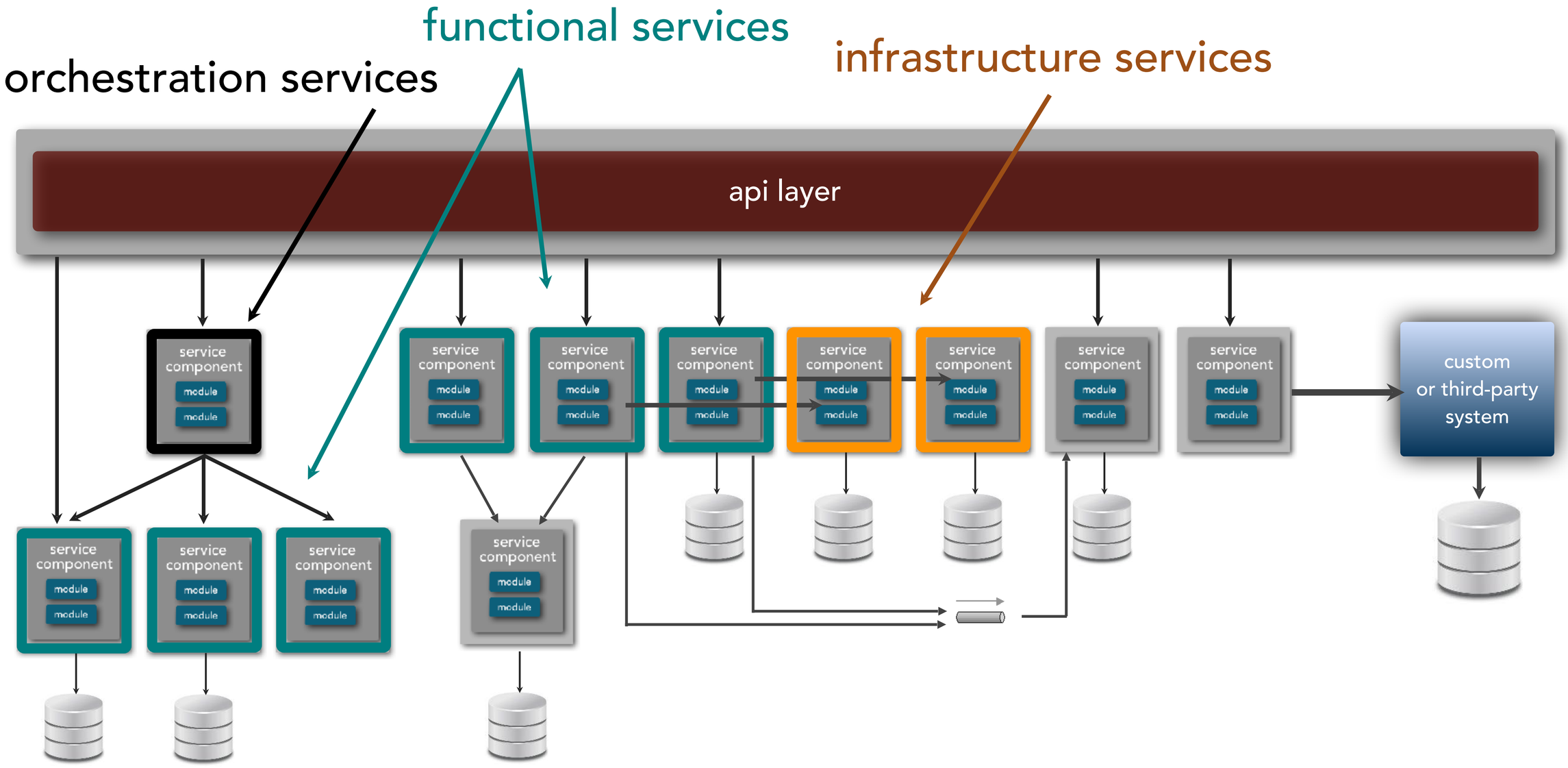


service types

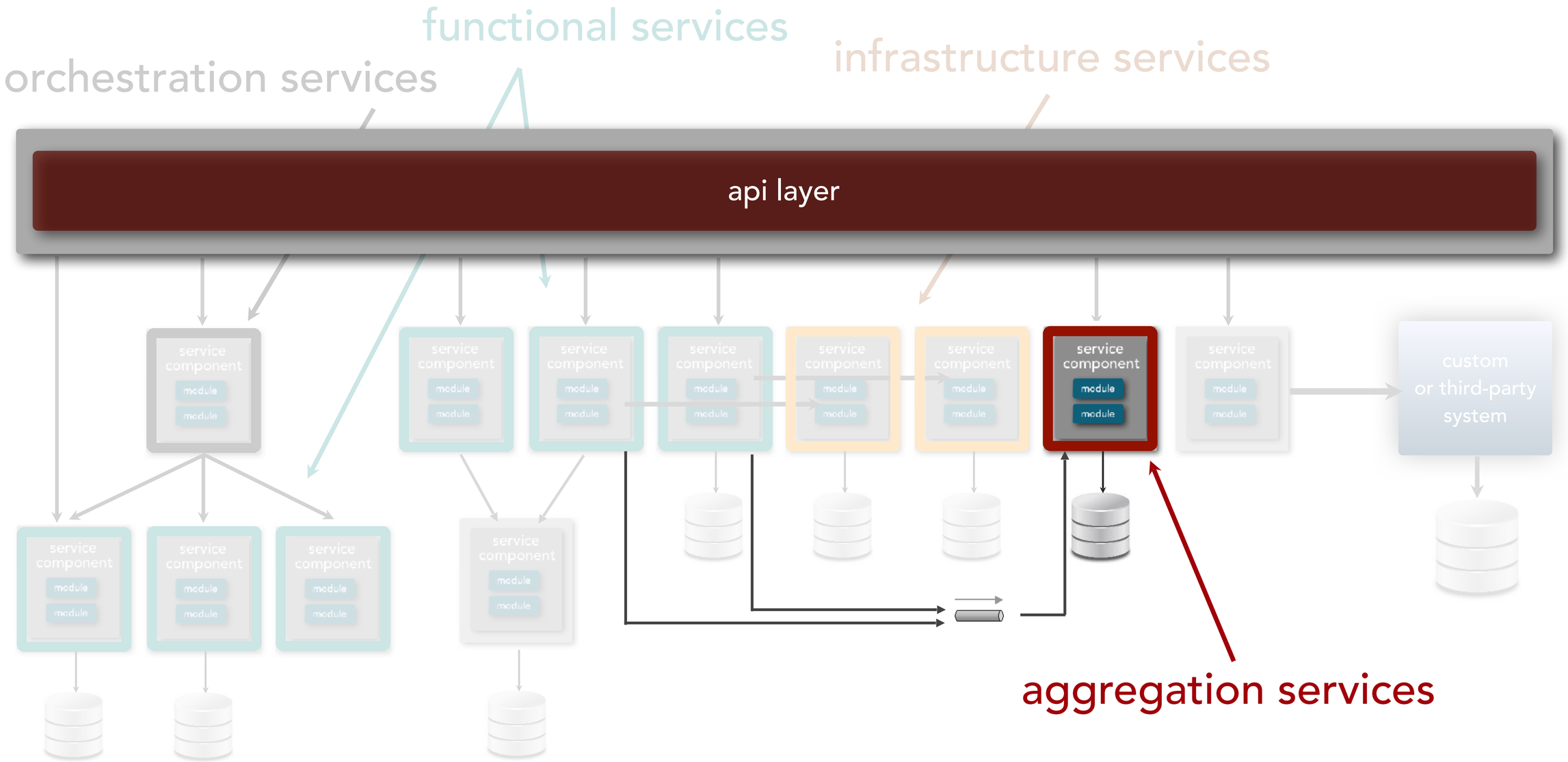
orchestration services



service types

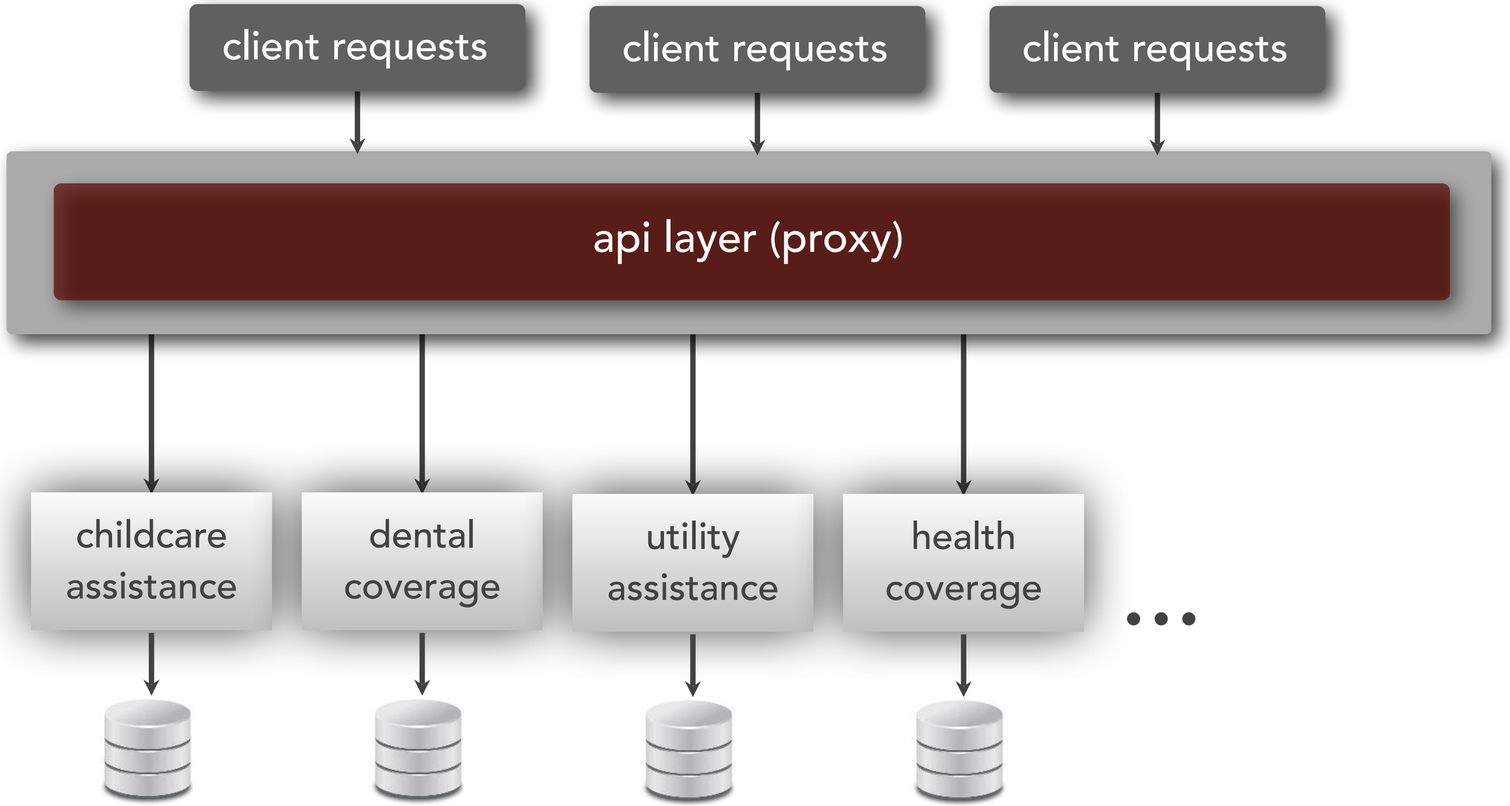


service types



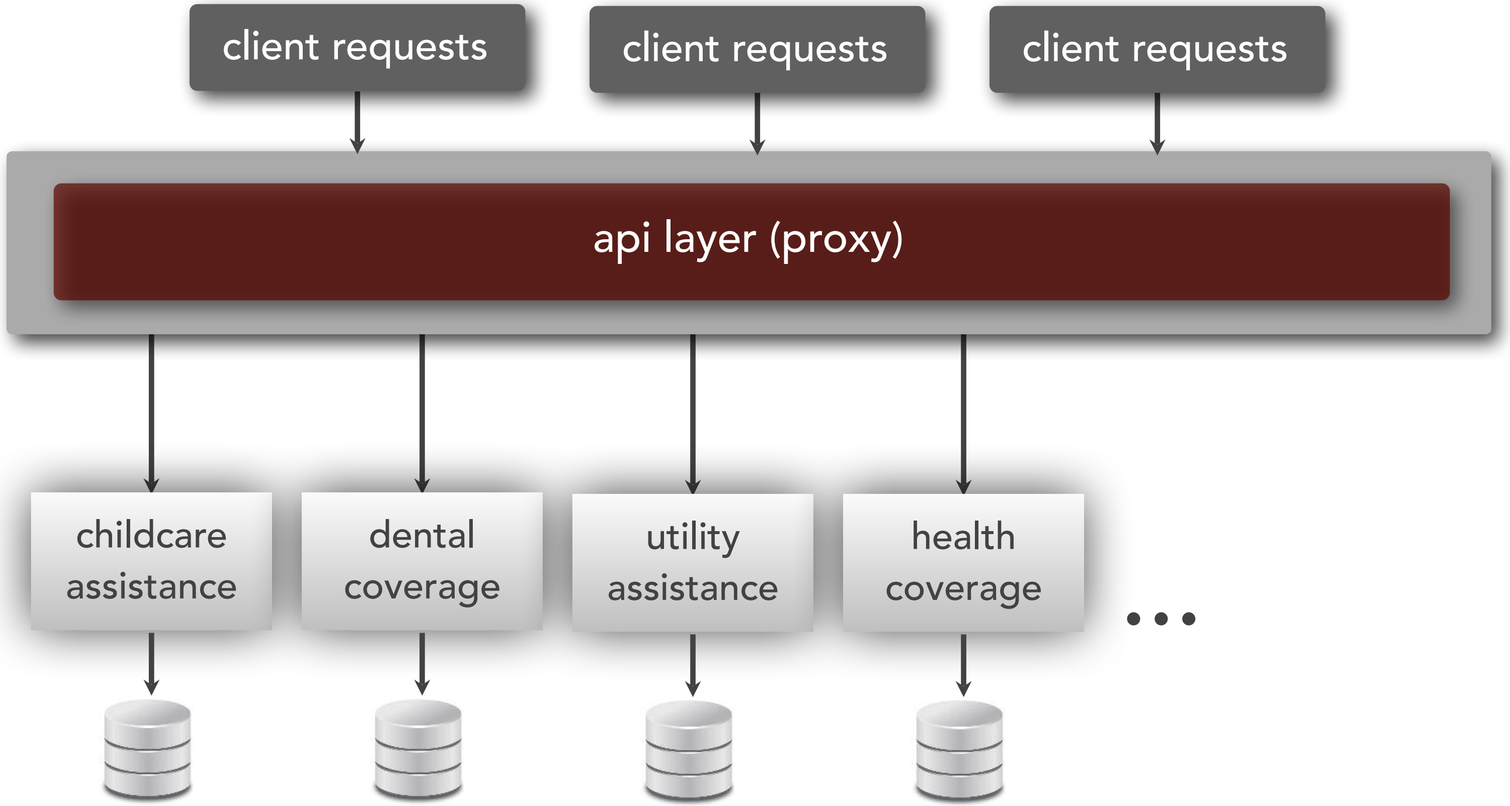
service types

aggregation services



service types

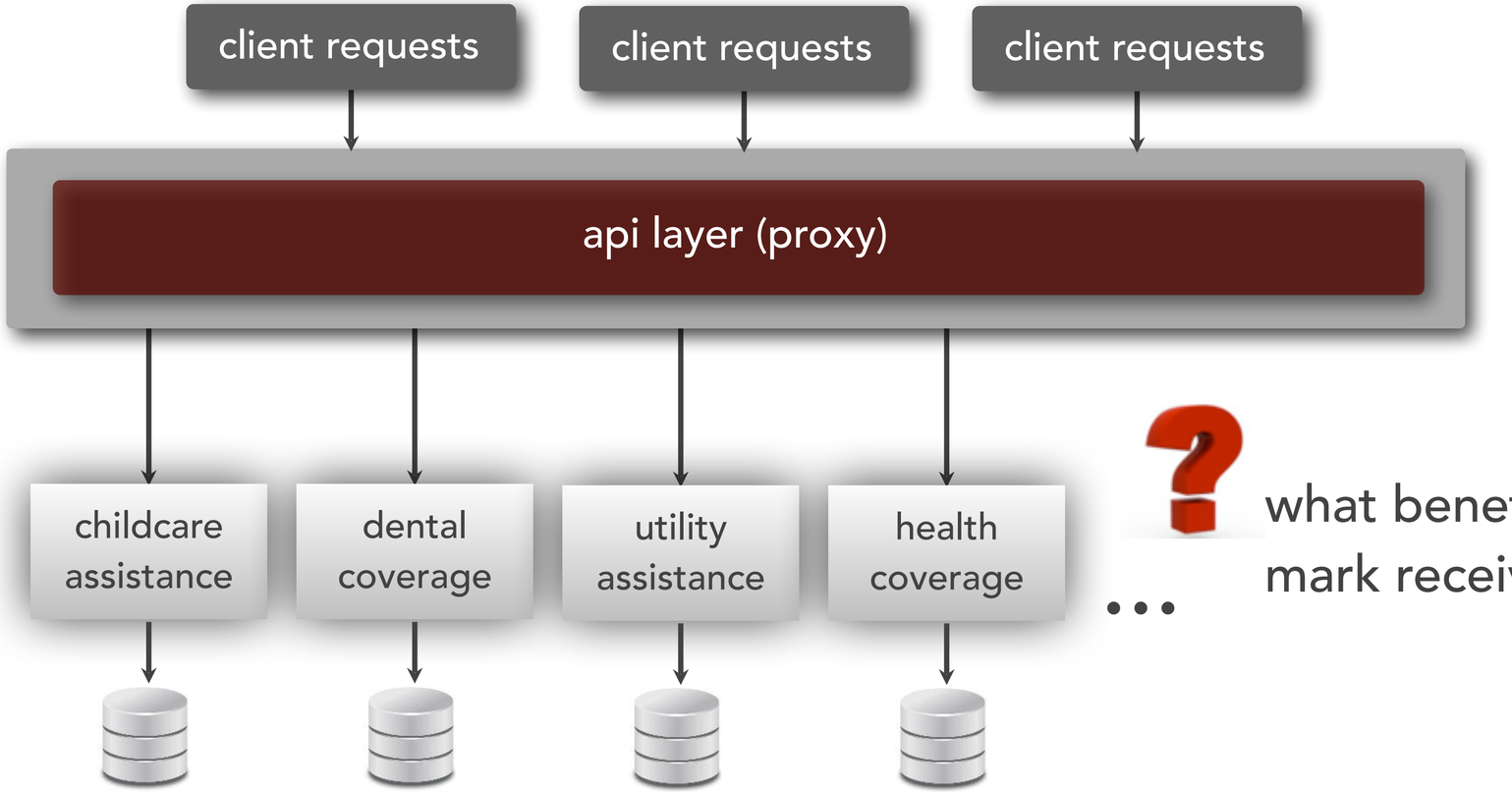
aggregation services



140 benefit services

service types

aggregation services

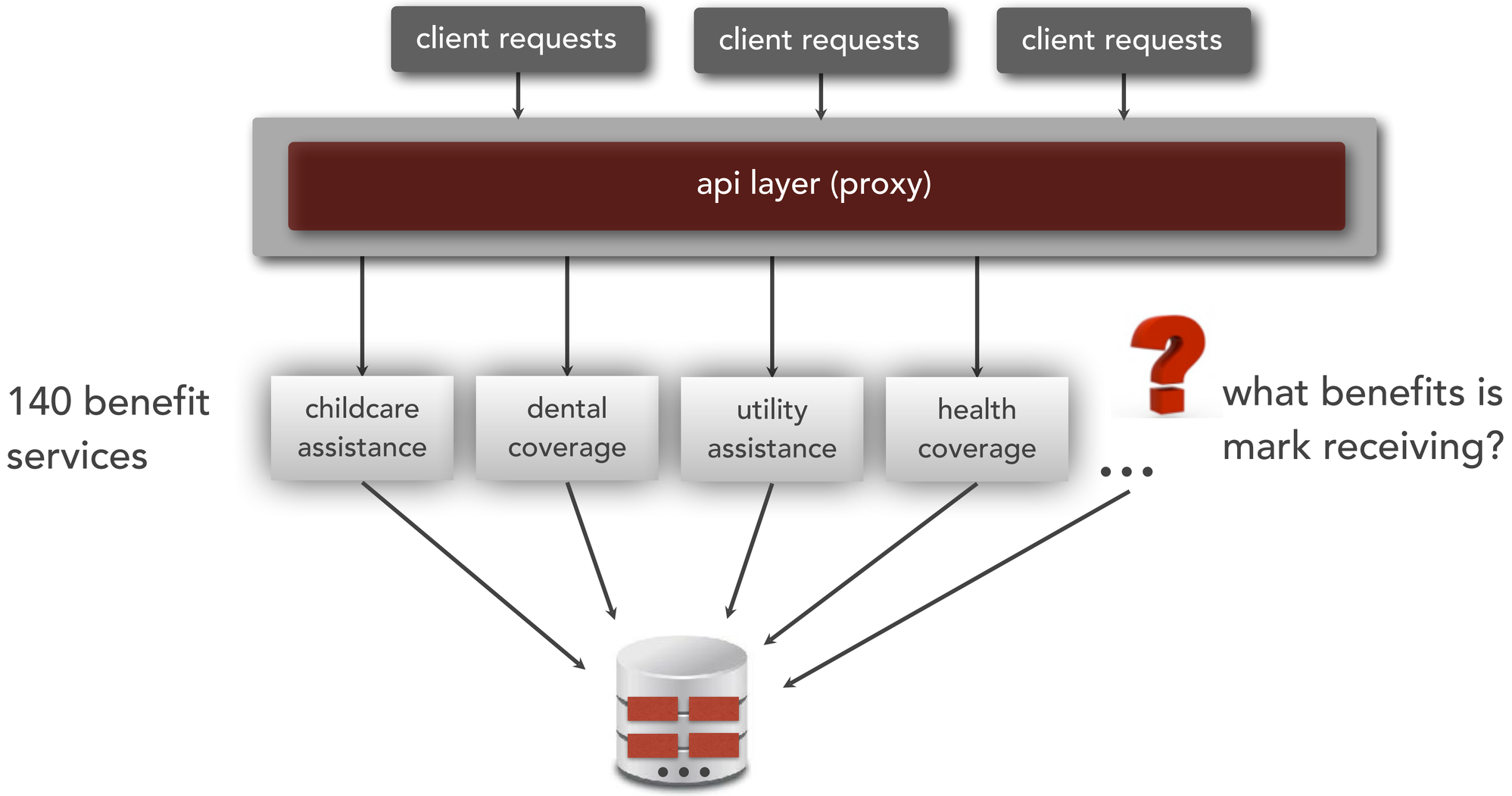


140 benefit services

what benefits is mark receiving?

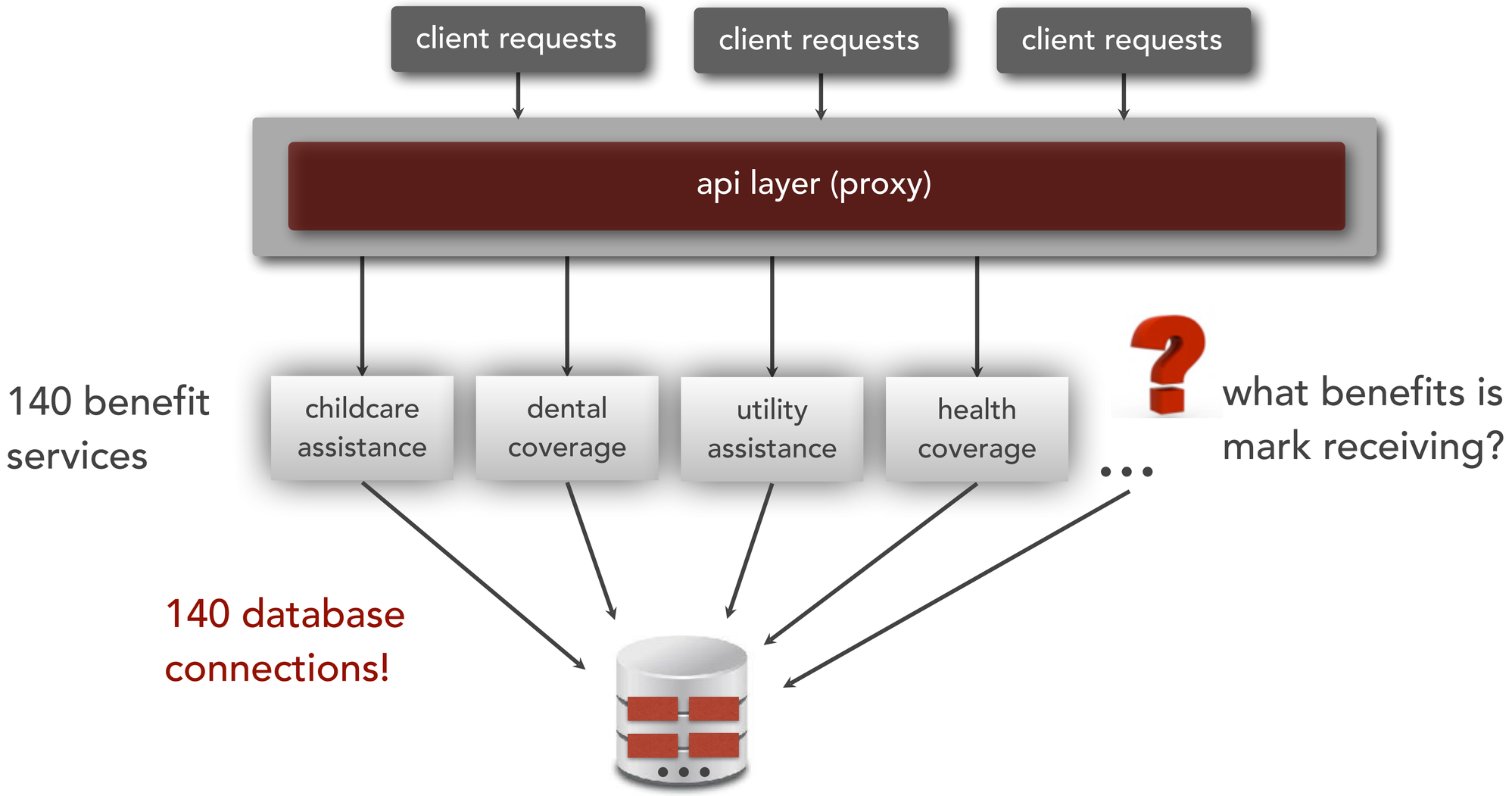
service types

aggregation services



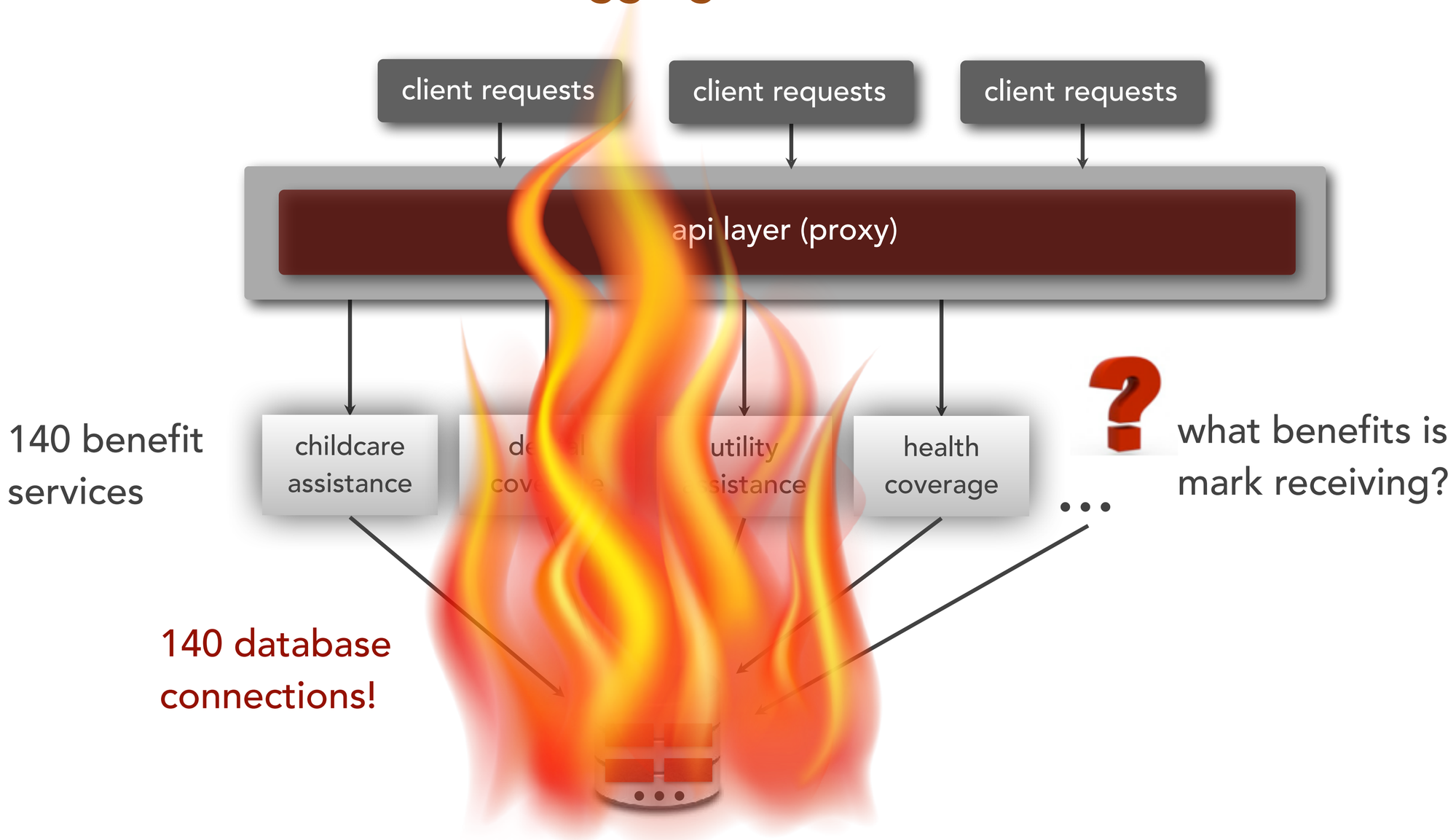
service types

aggregation services



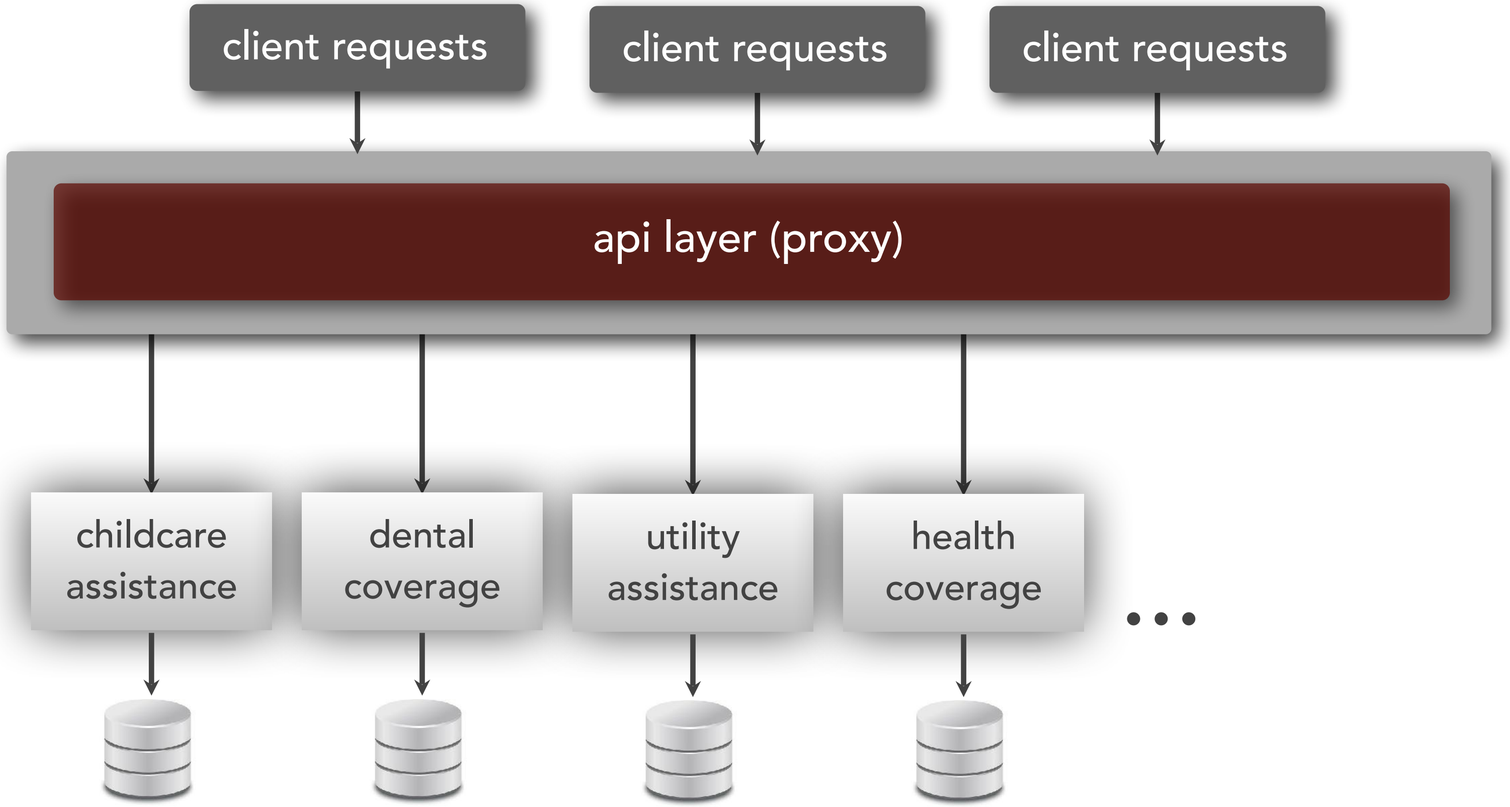
service types

aggregation services



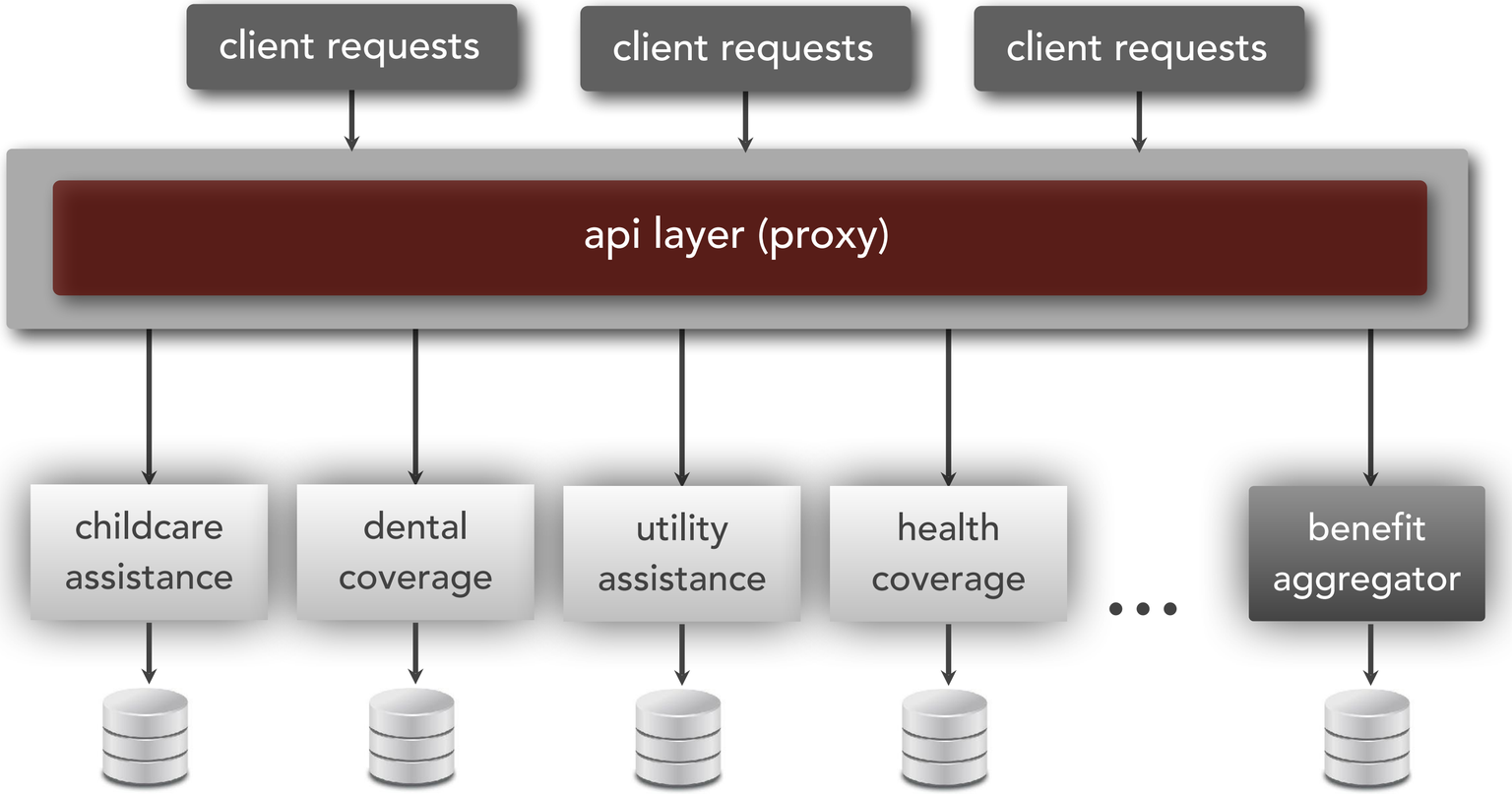
service types

aggregation services



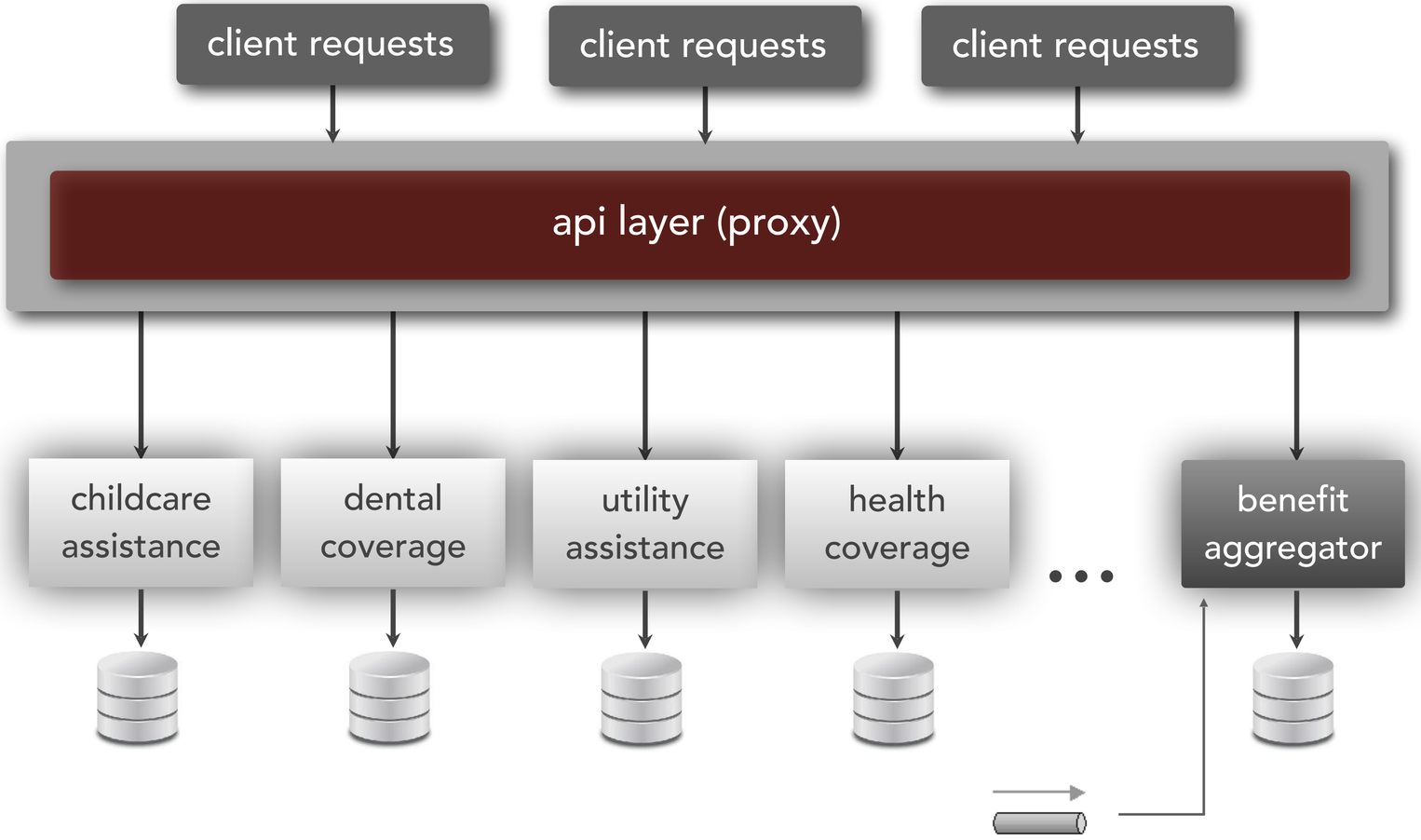
service types

aggregation services



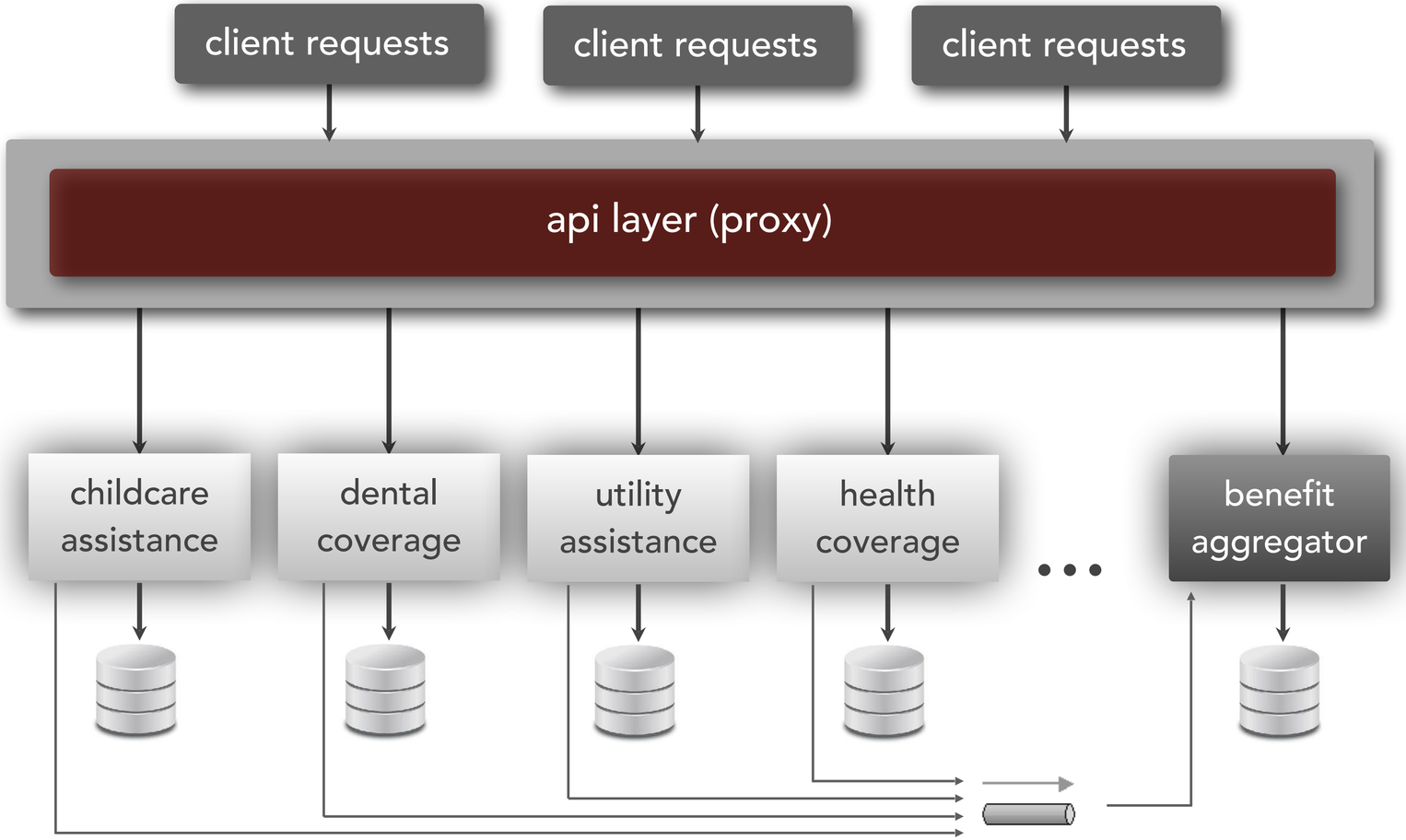
service types

aggregation services



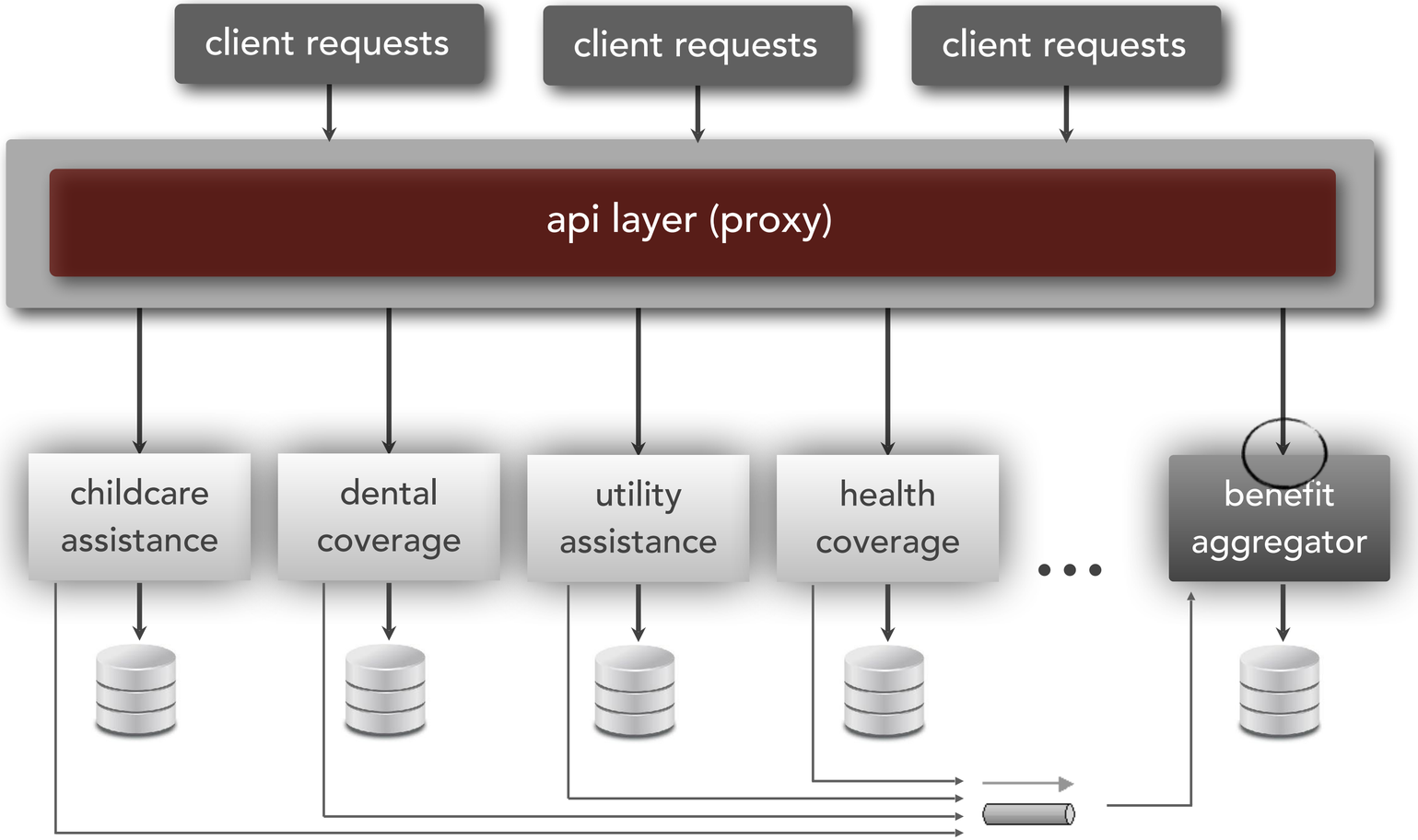
service types

aggregation services



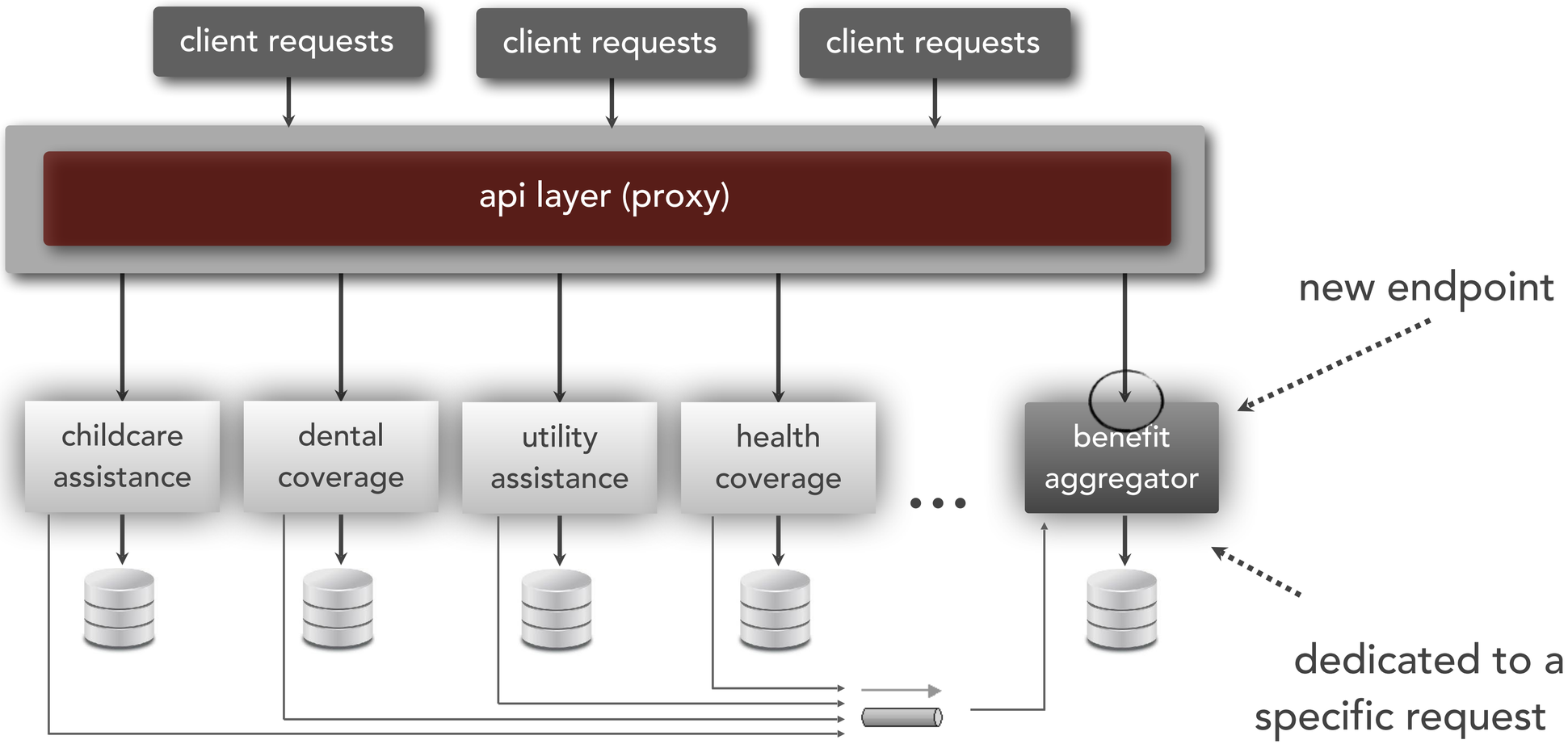
service types

aggregation services



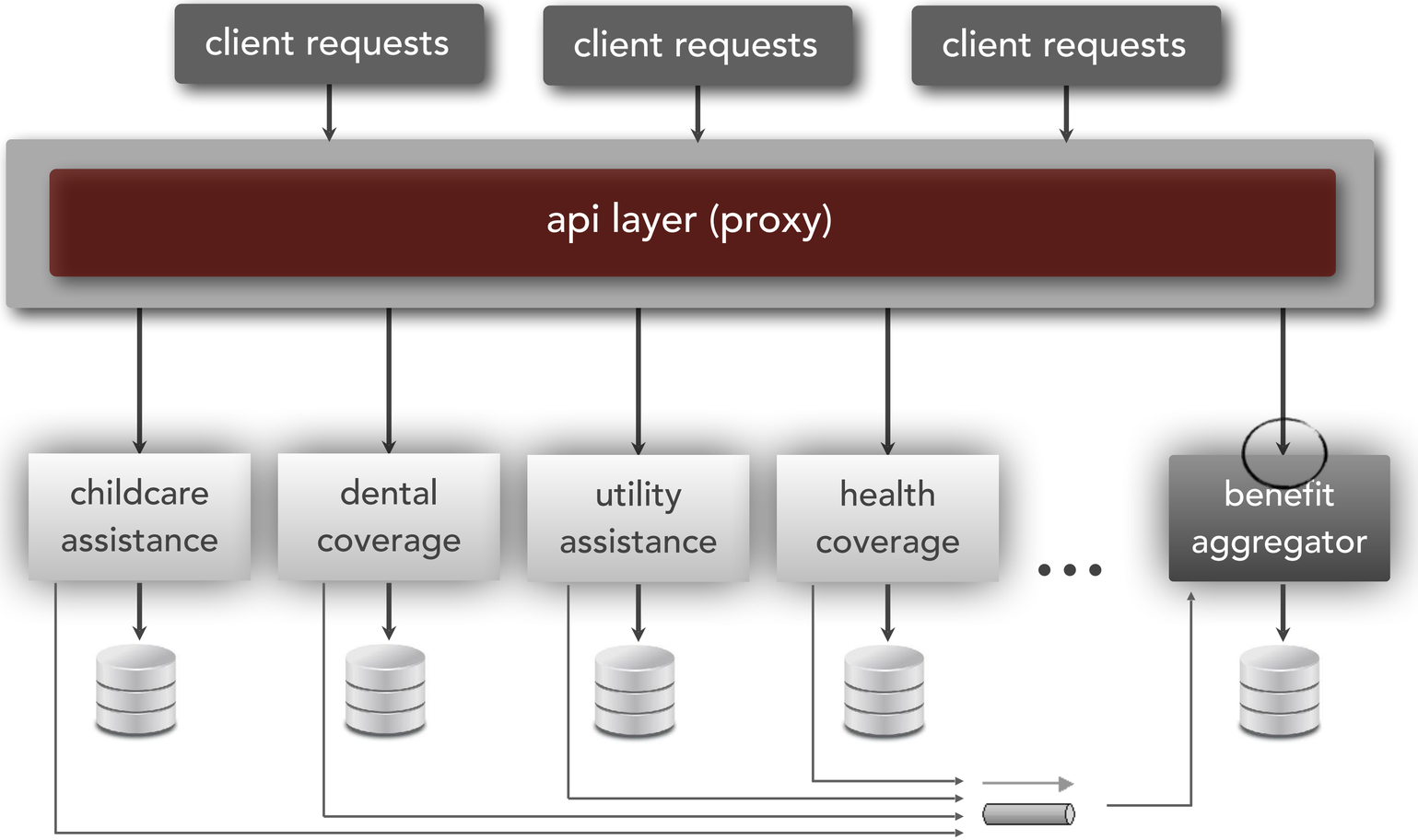
service types

aggregation services



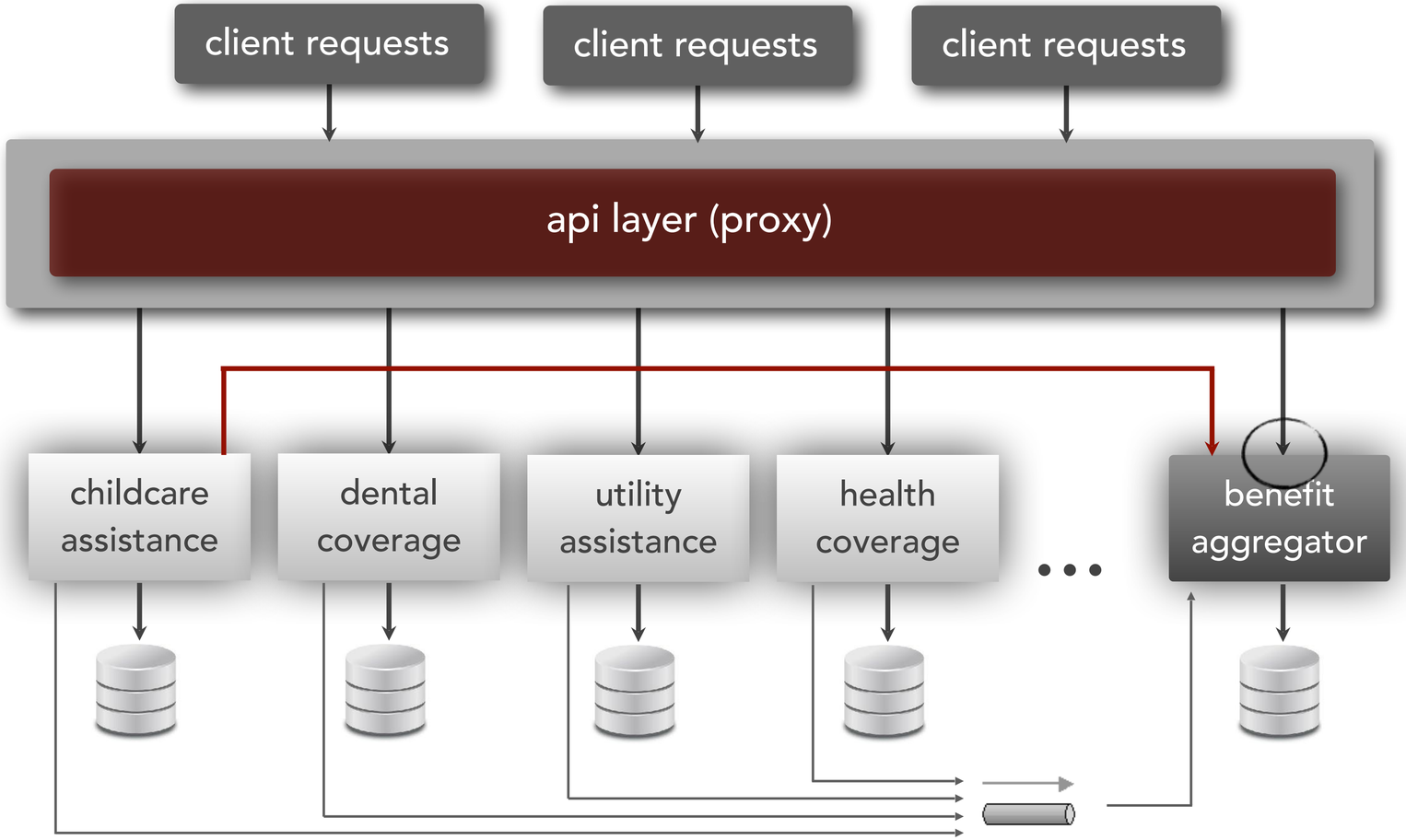
service types

aggregation services

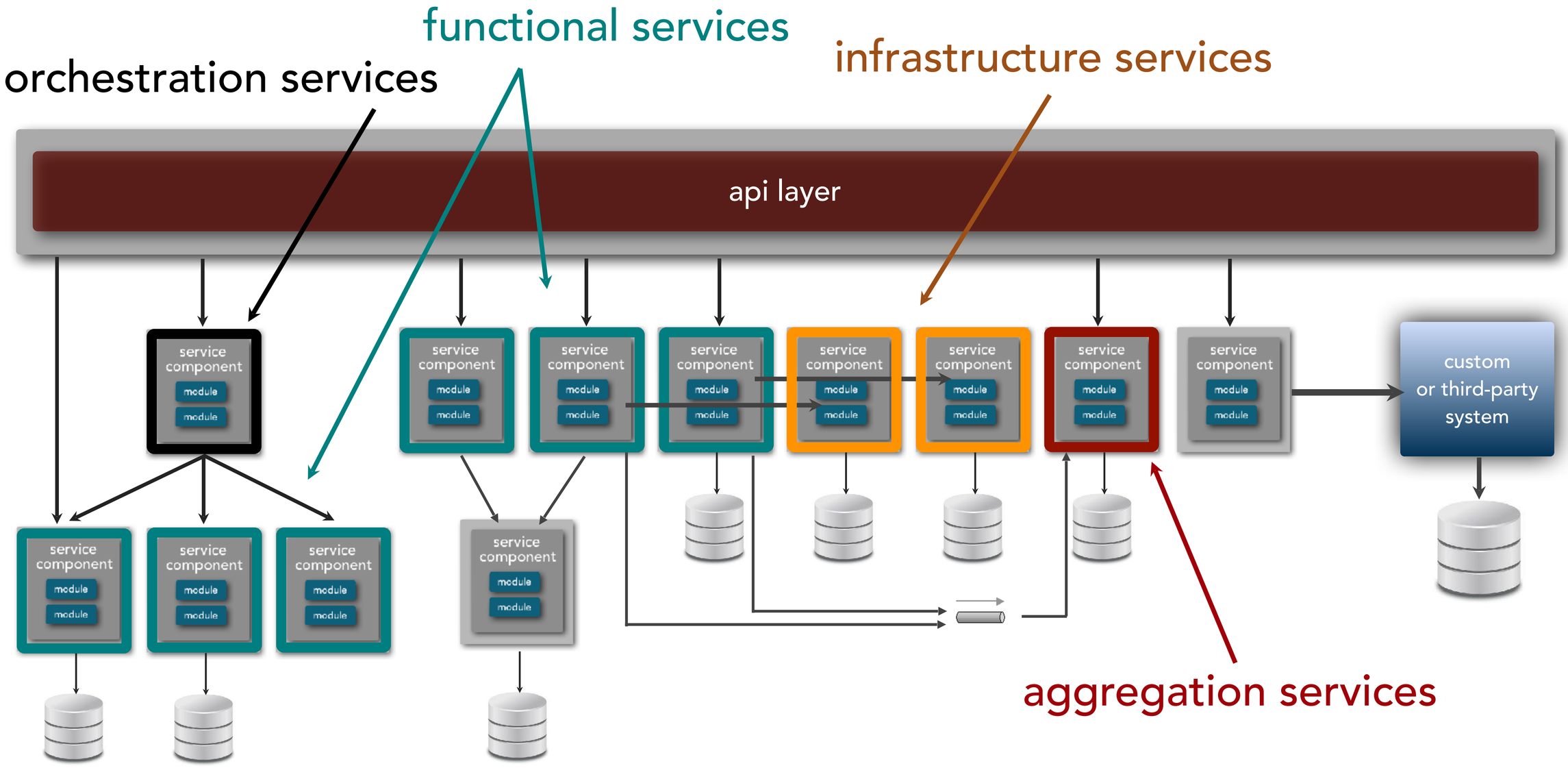


service types

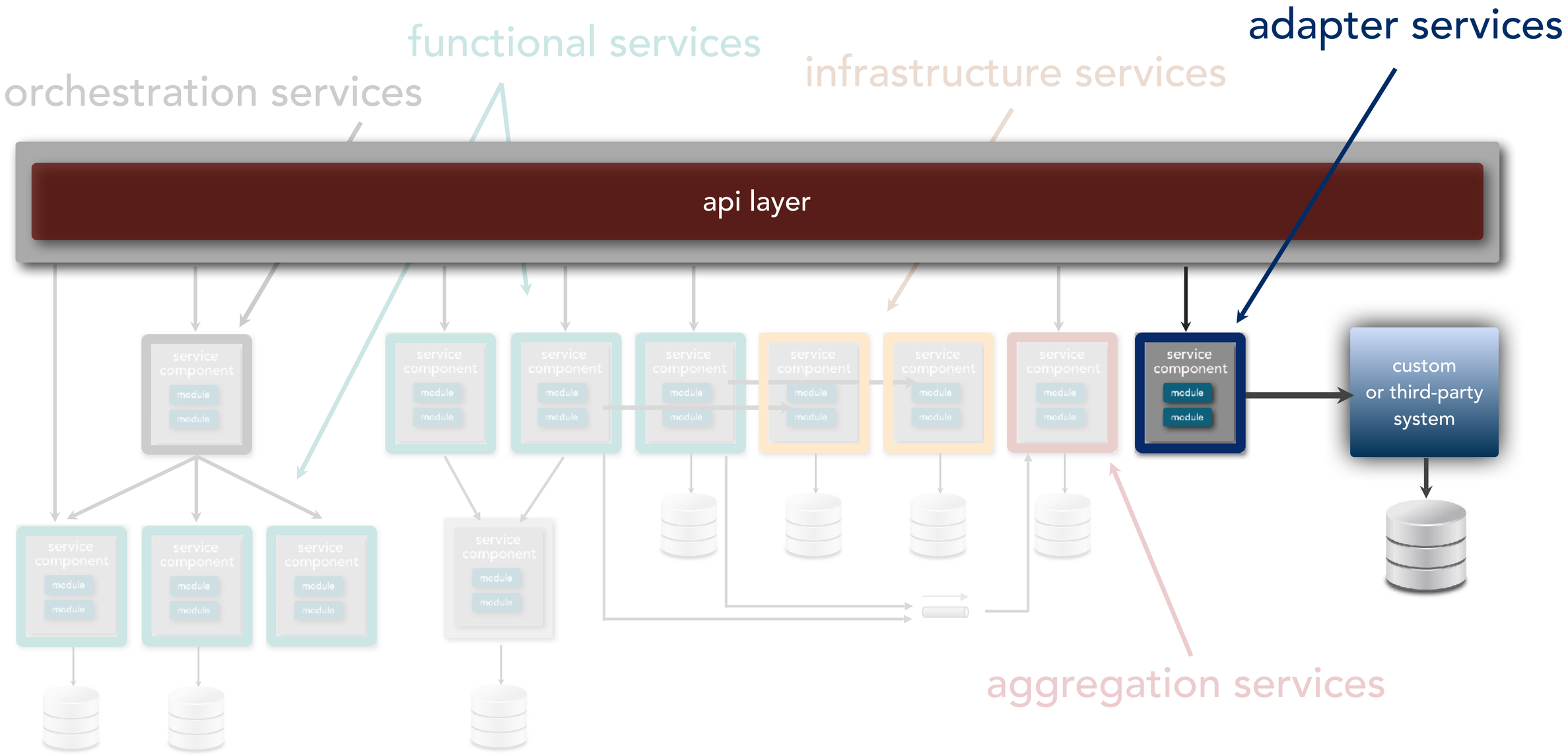
aggregation services



service types

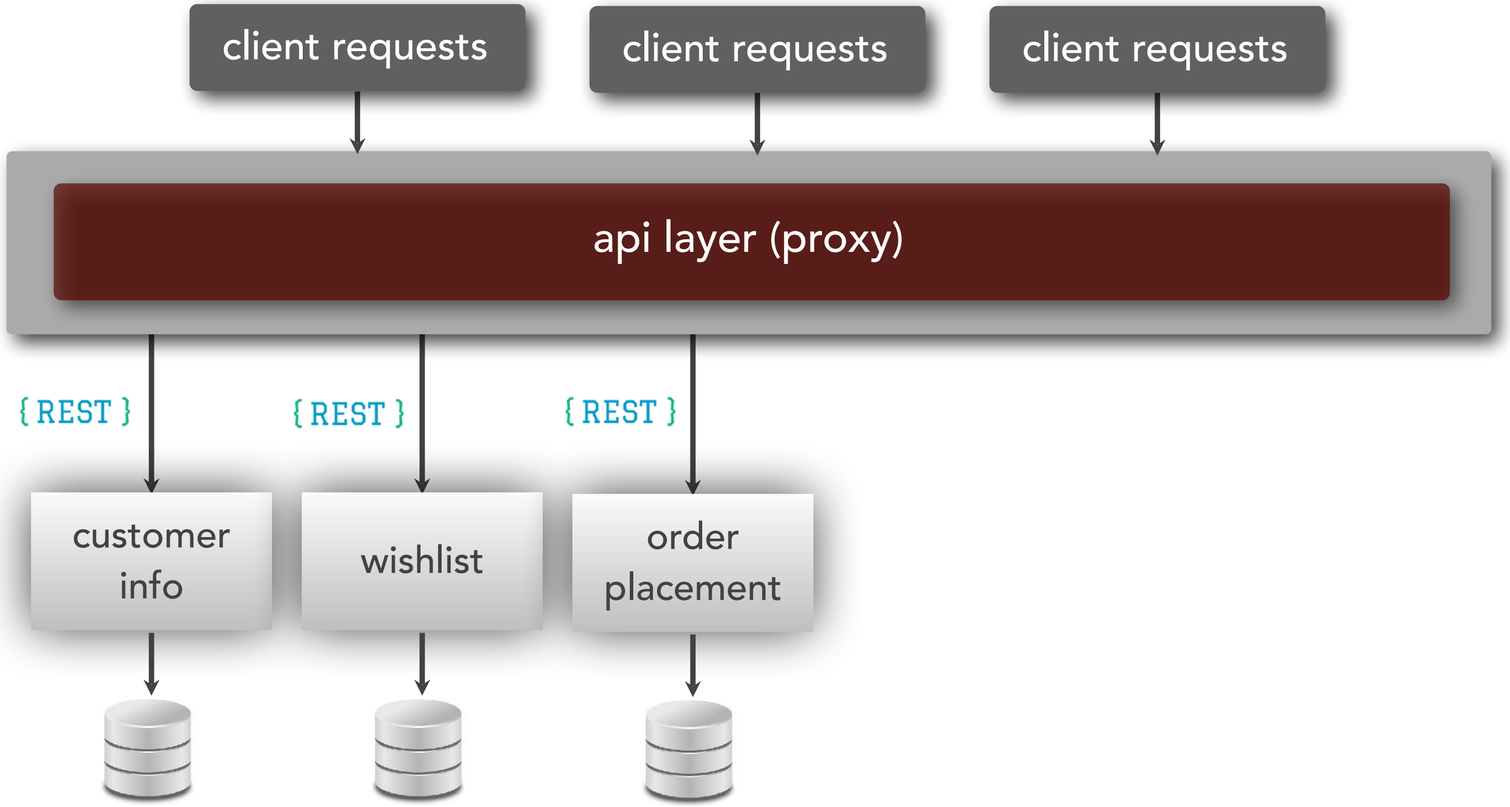


service types



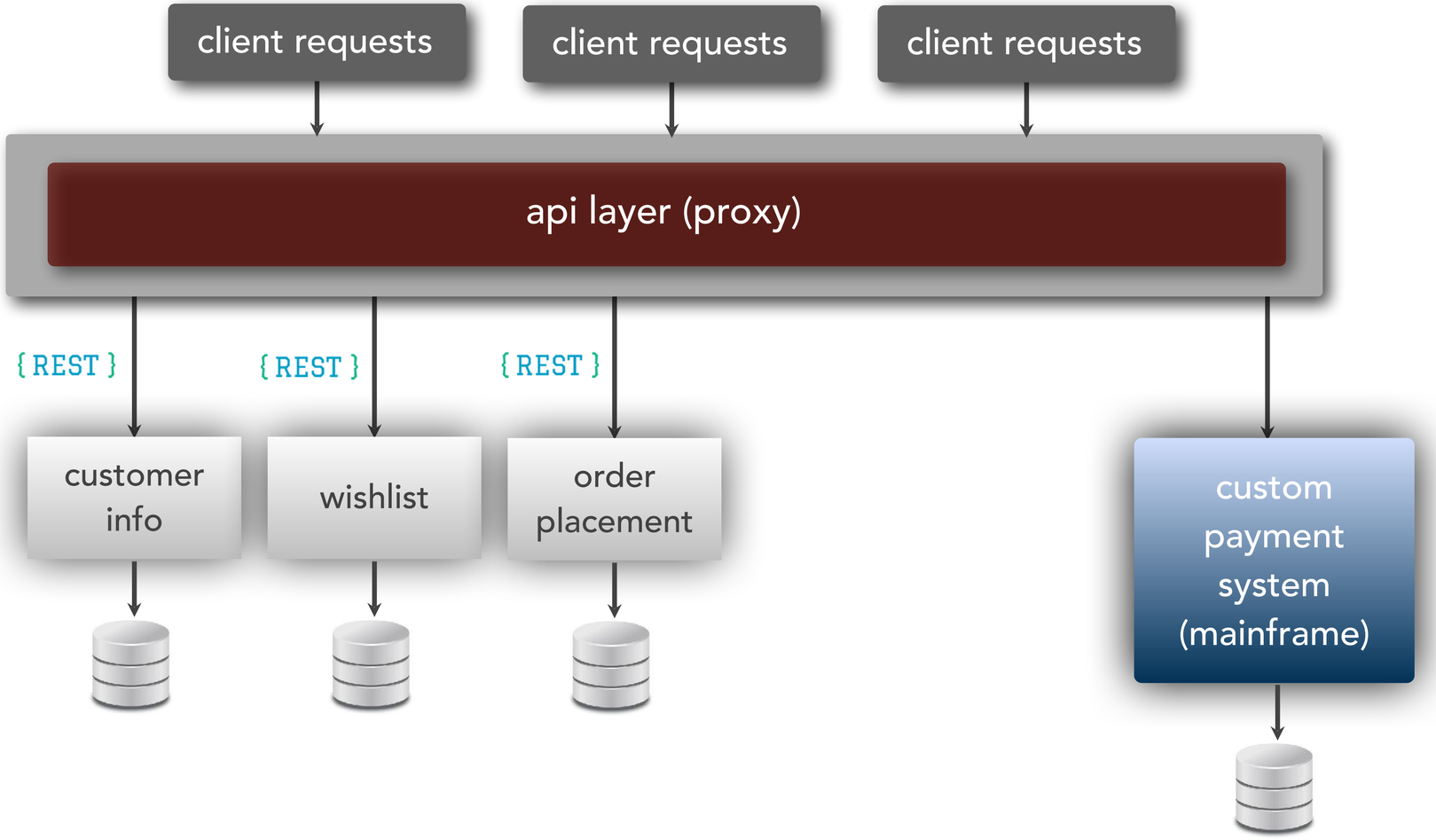
service types

adapter services



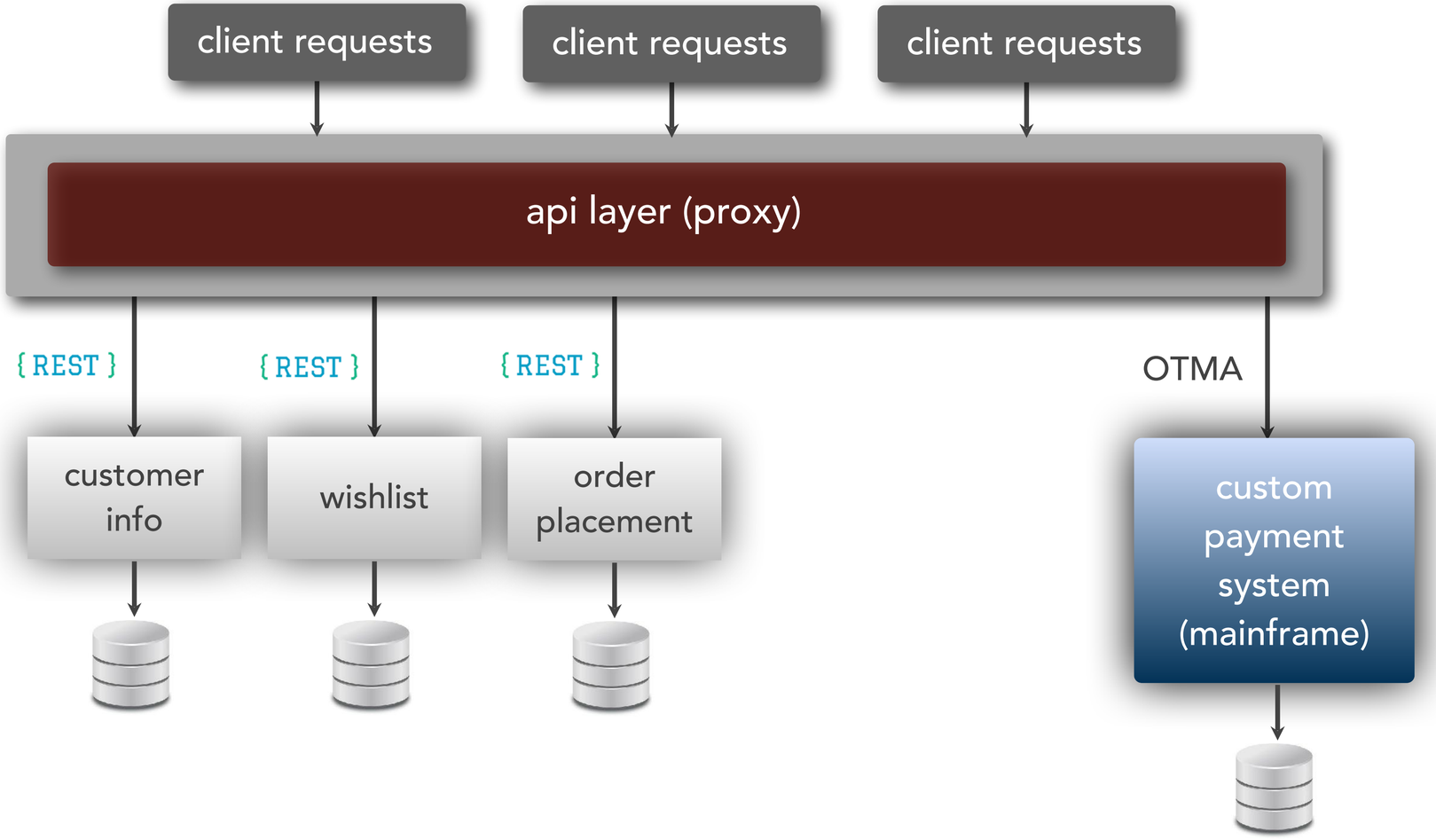
service types

adapter services



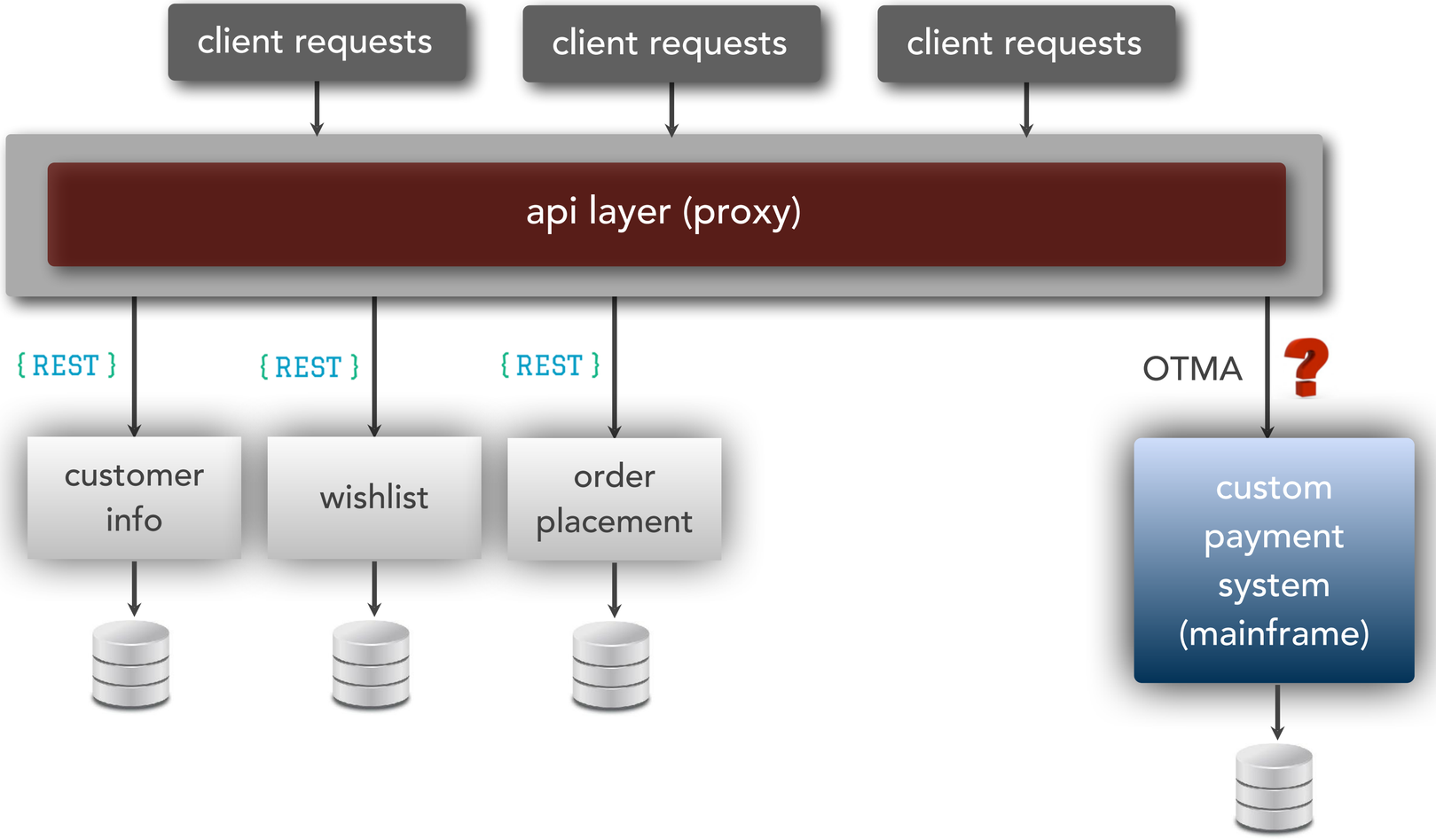
service types

adapter services



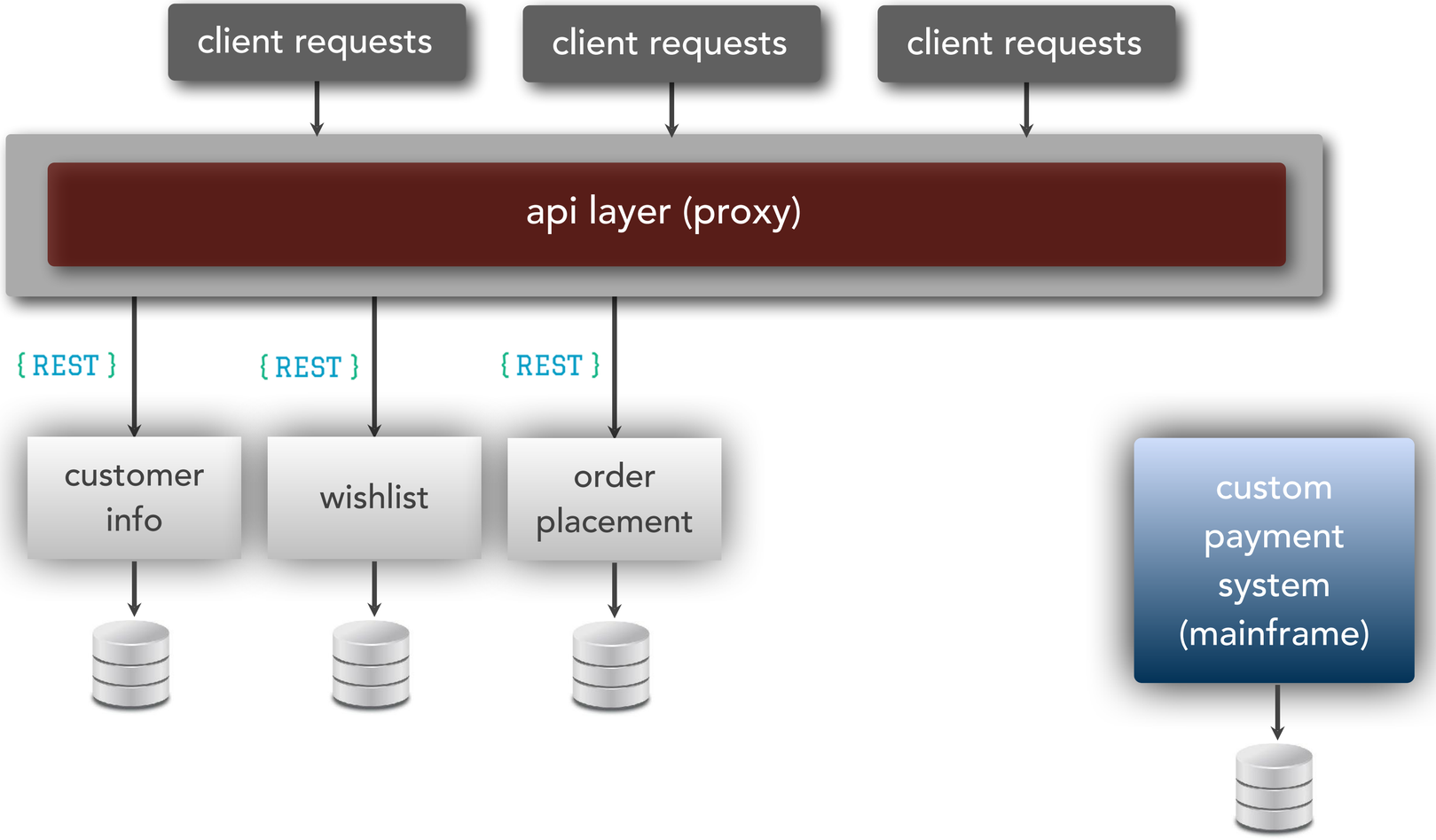
service types

adapter services



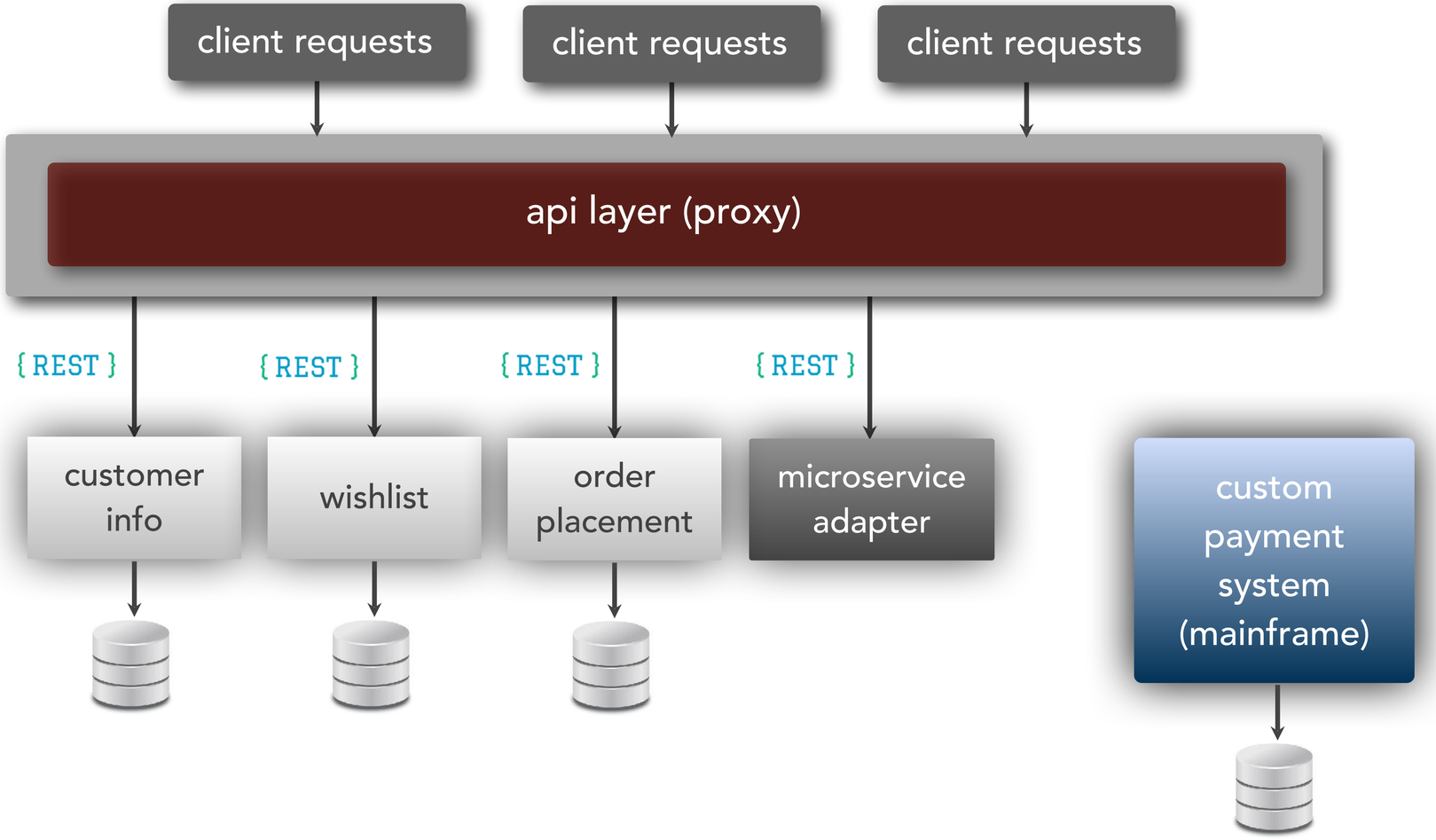
service types

adapter services



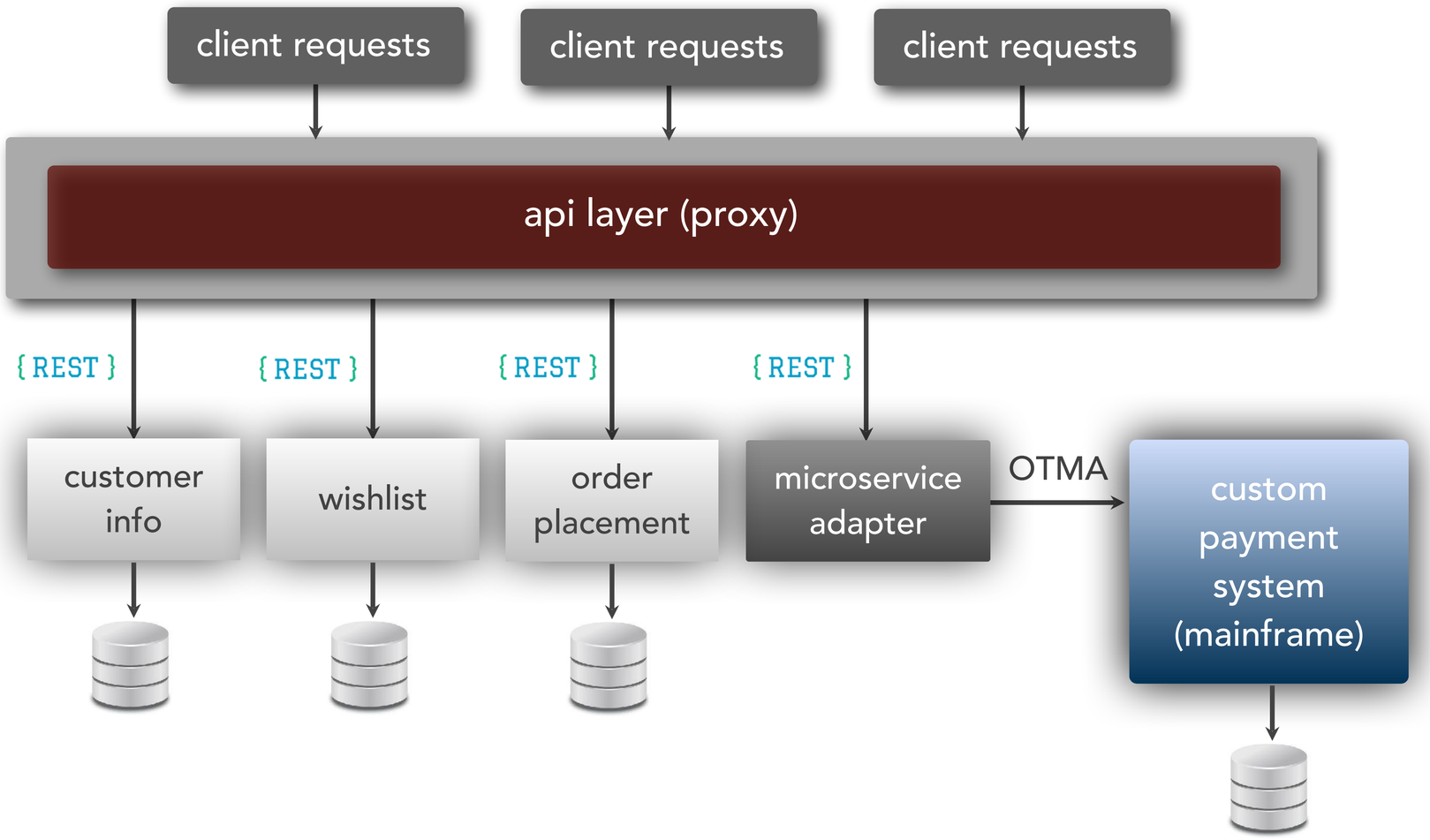
service types

adapter services



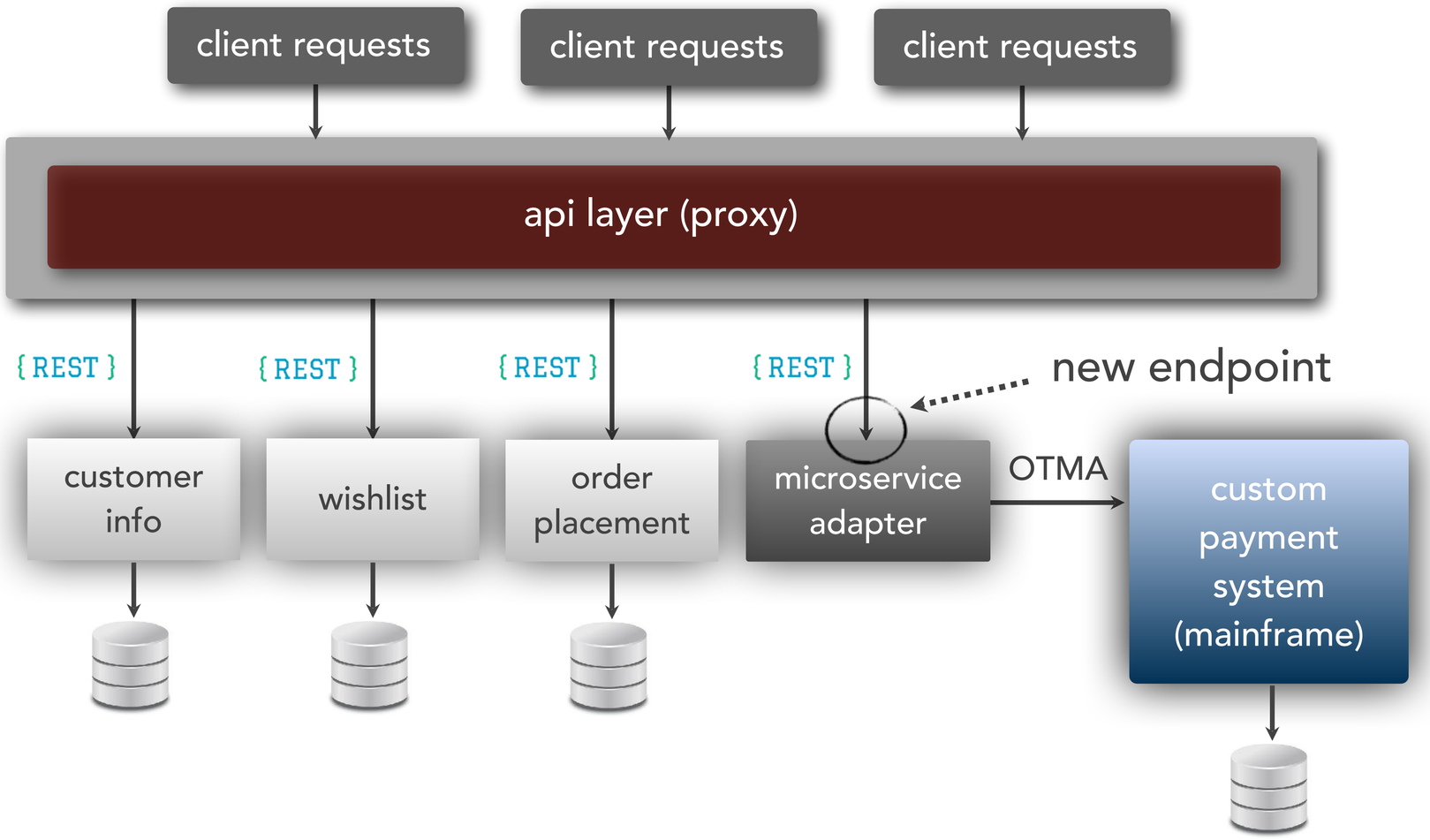
service types

adapter services



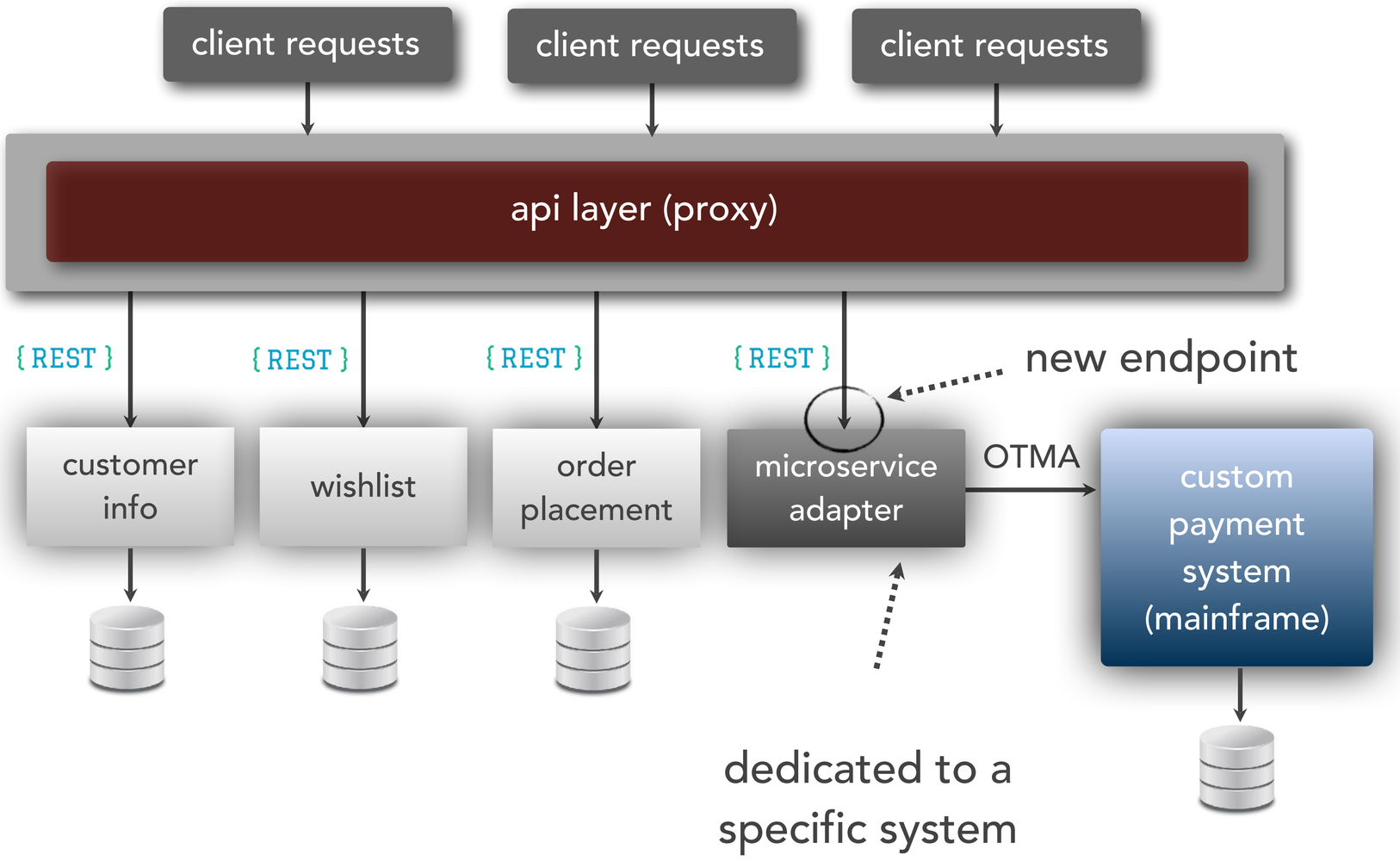
service types

adapter services



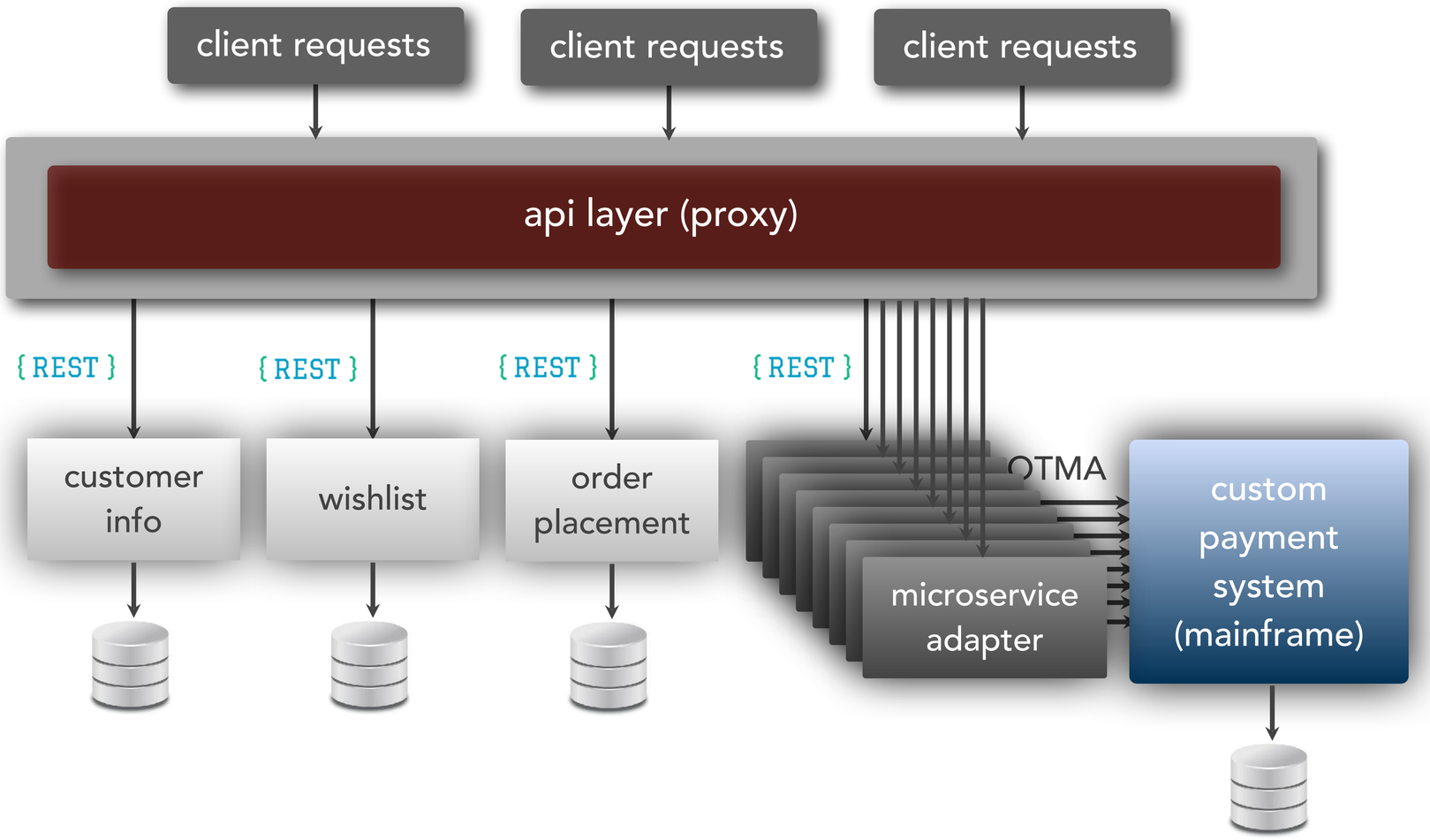
service types

adapter services



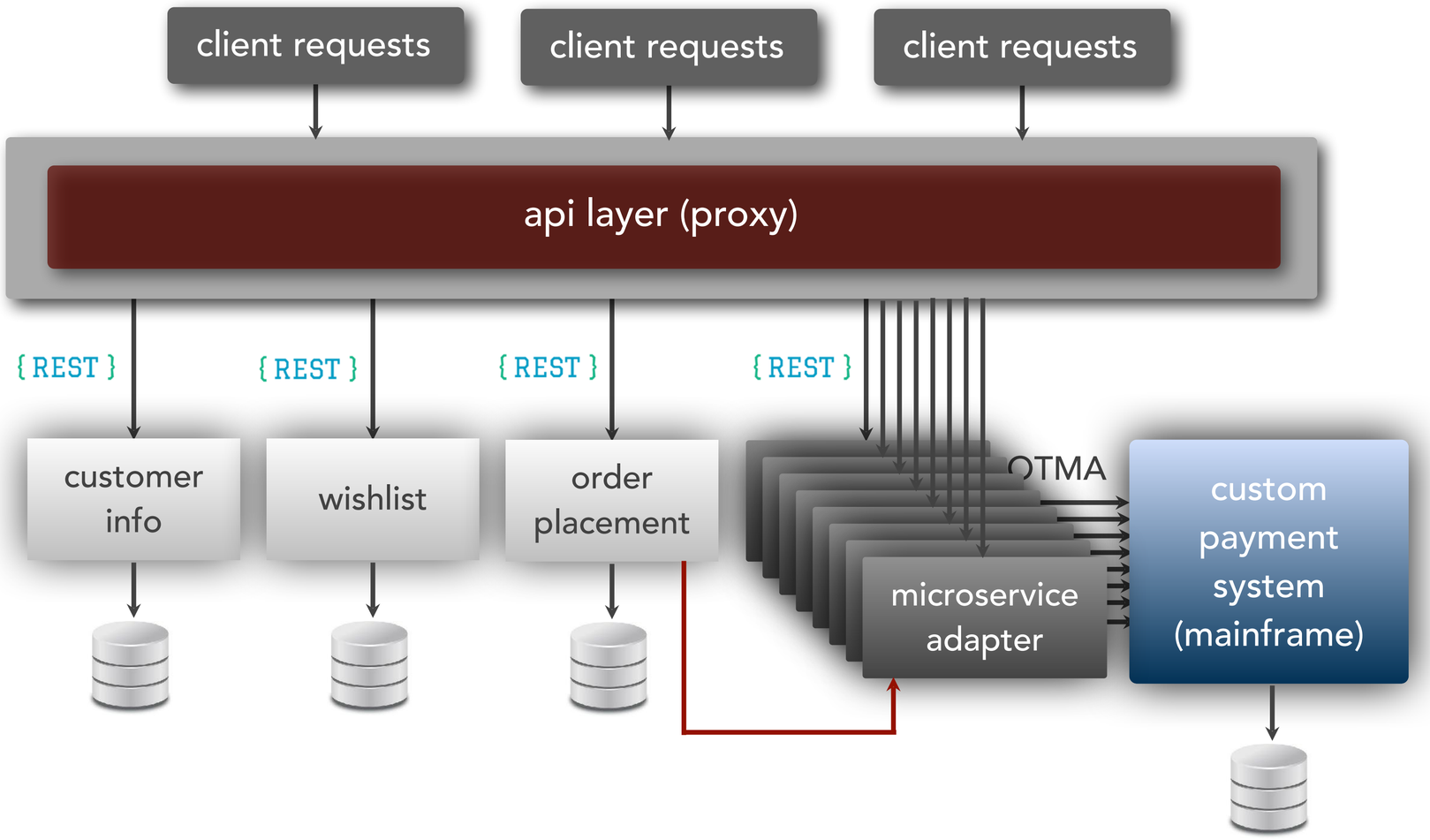
service types

adapter services



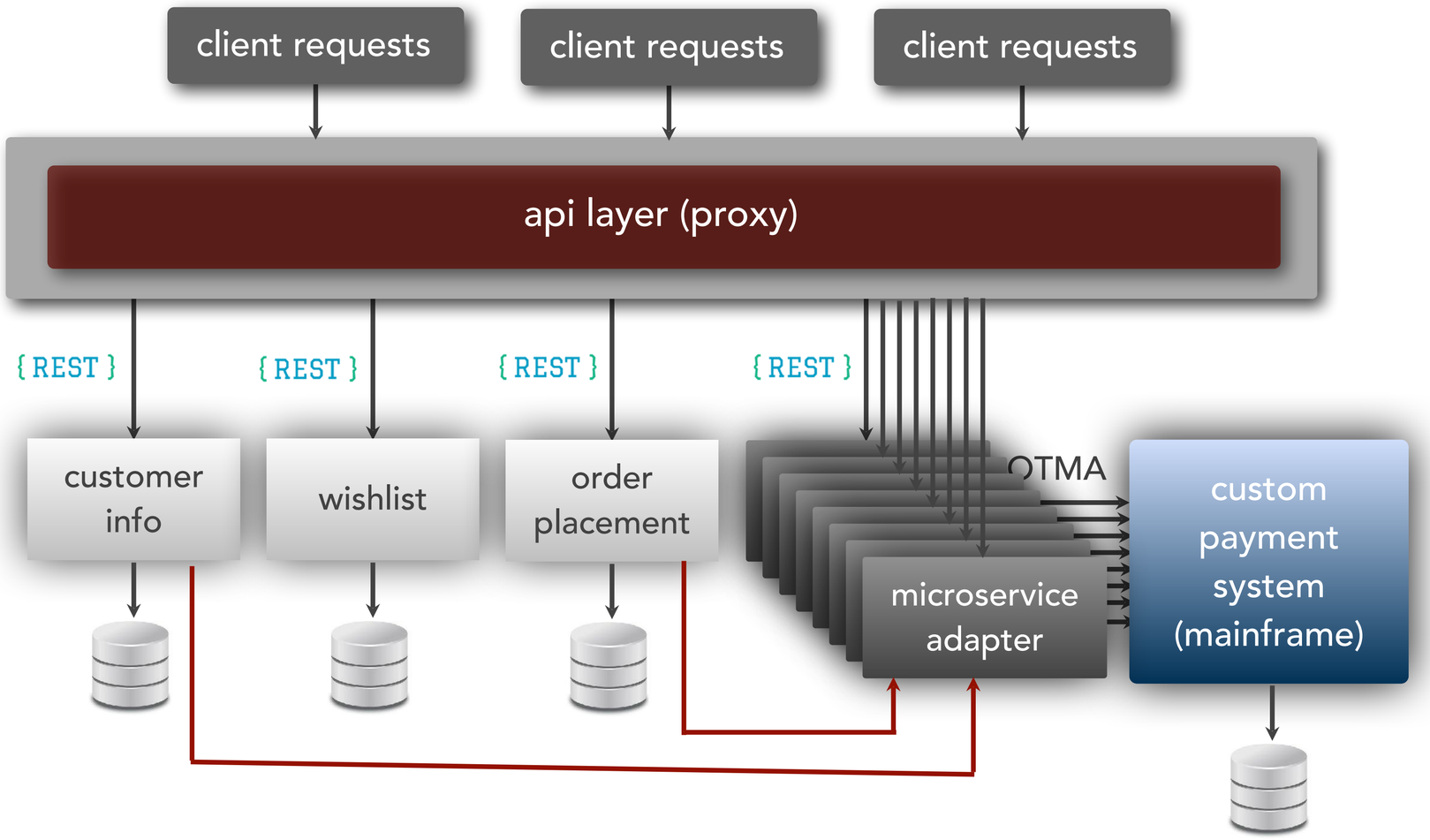
service types

adapter services



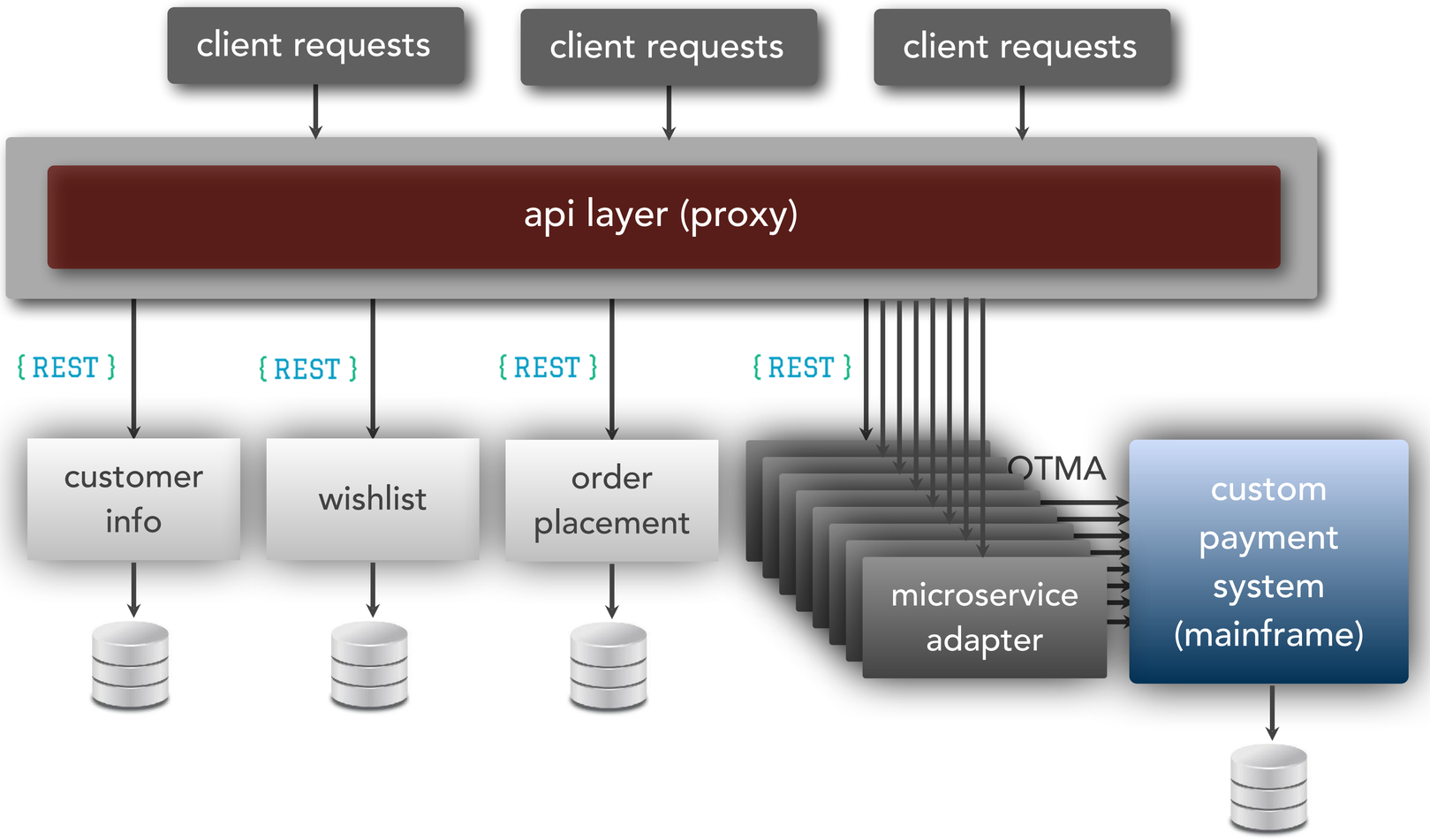
service types

adapter services



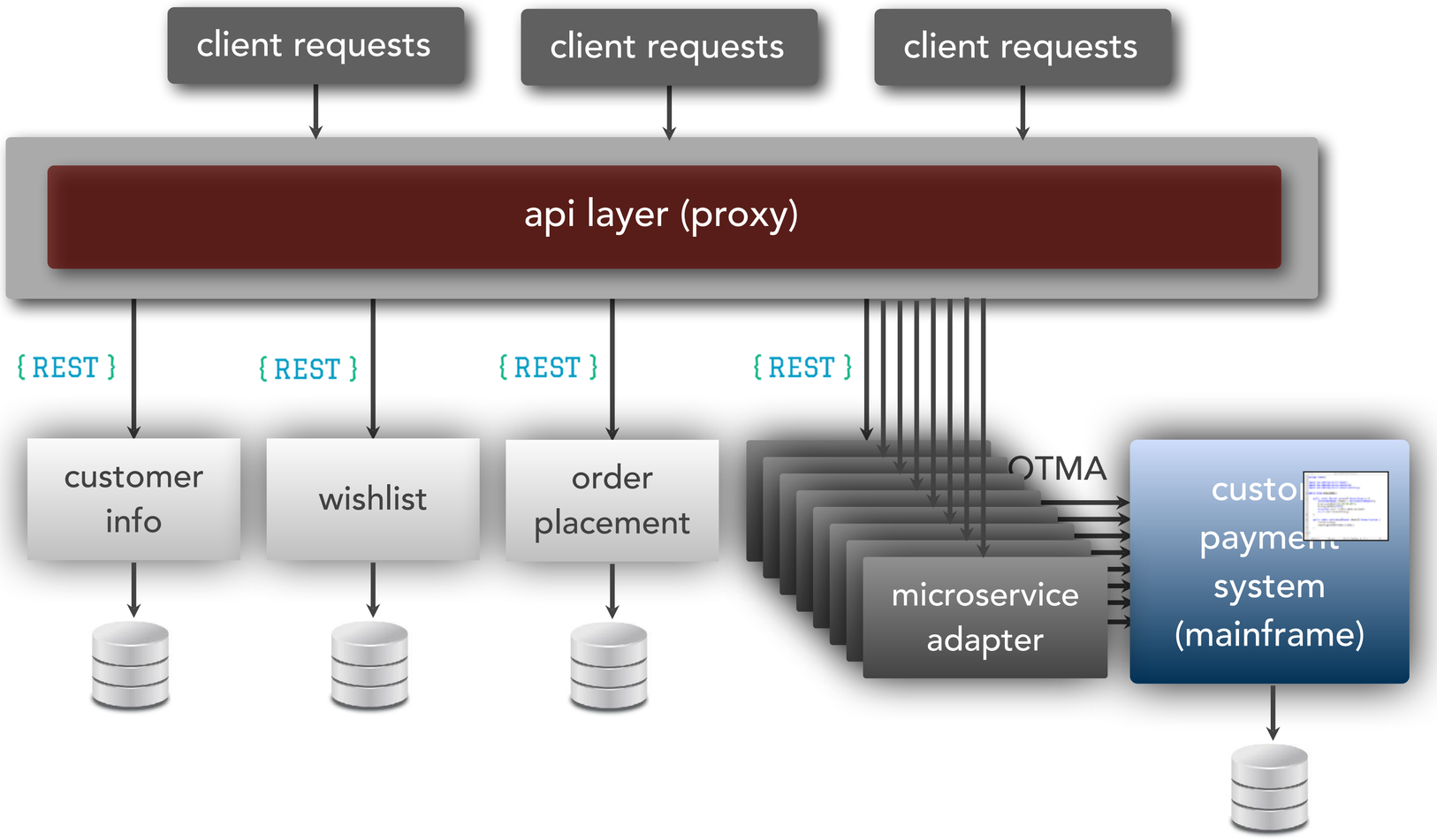
service types

adapter services



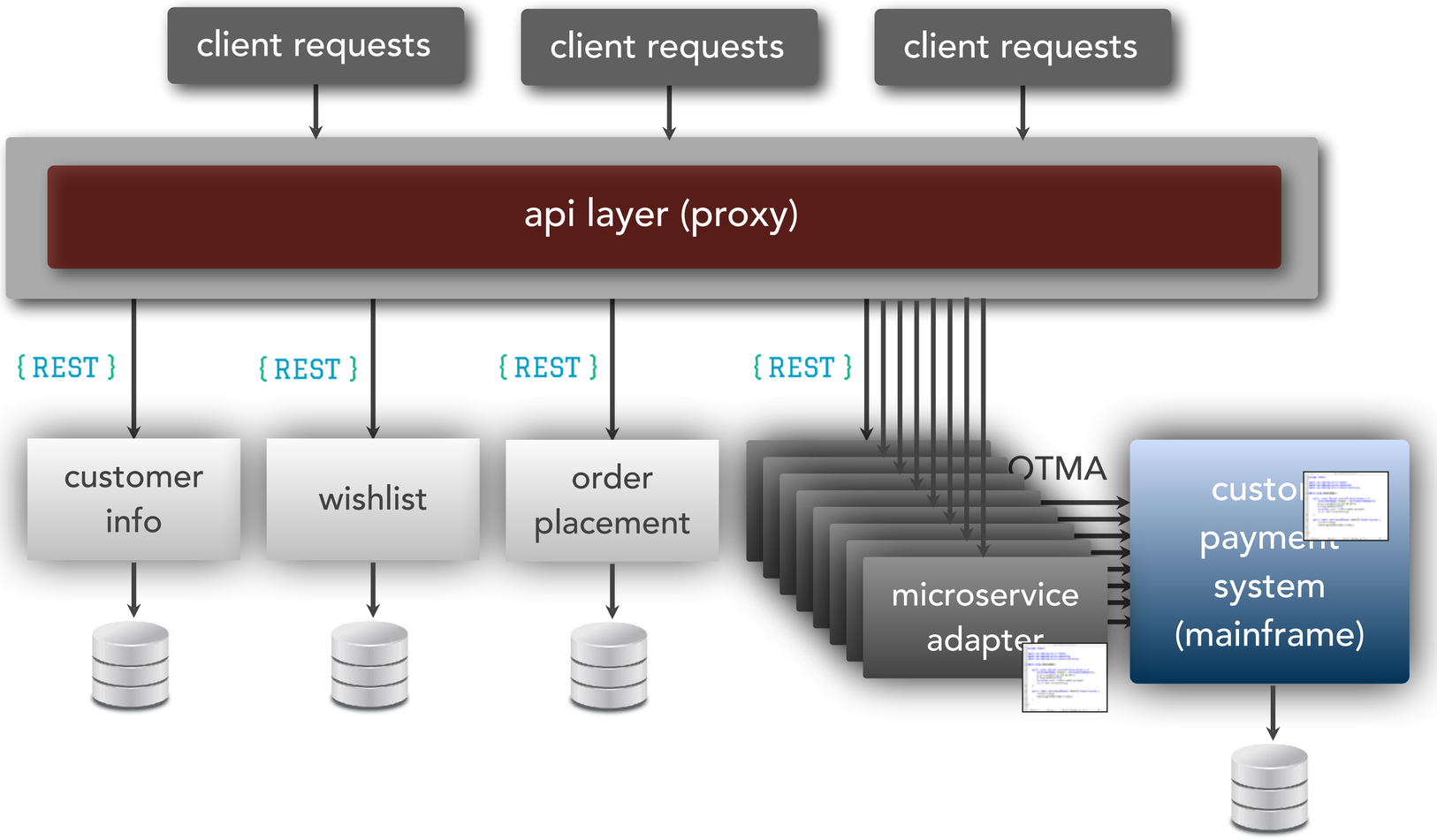
service types

adapter services



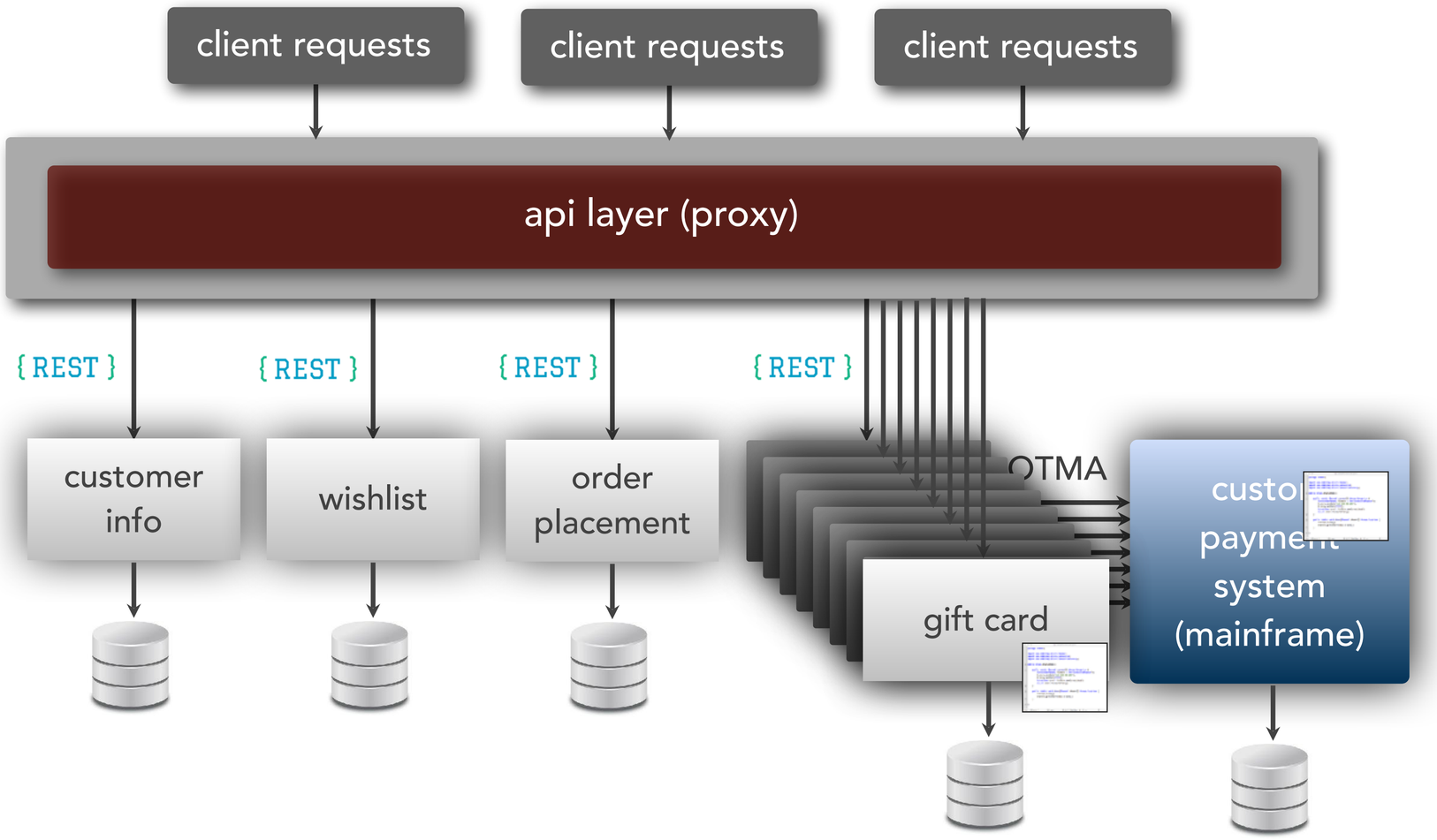
service types

adapter services

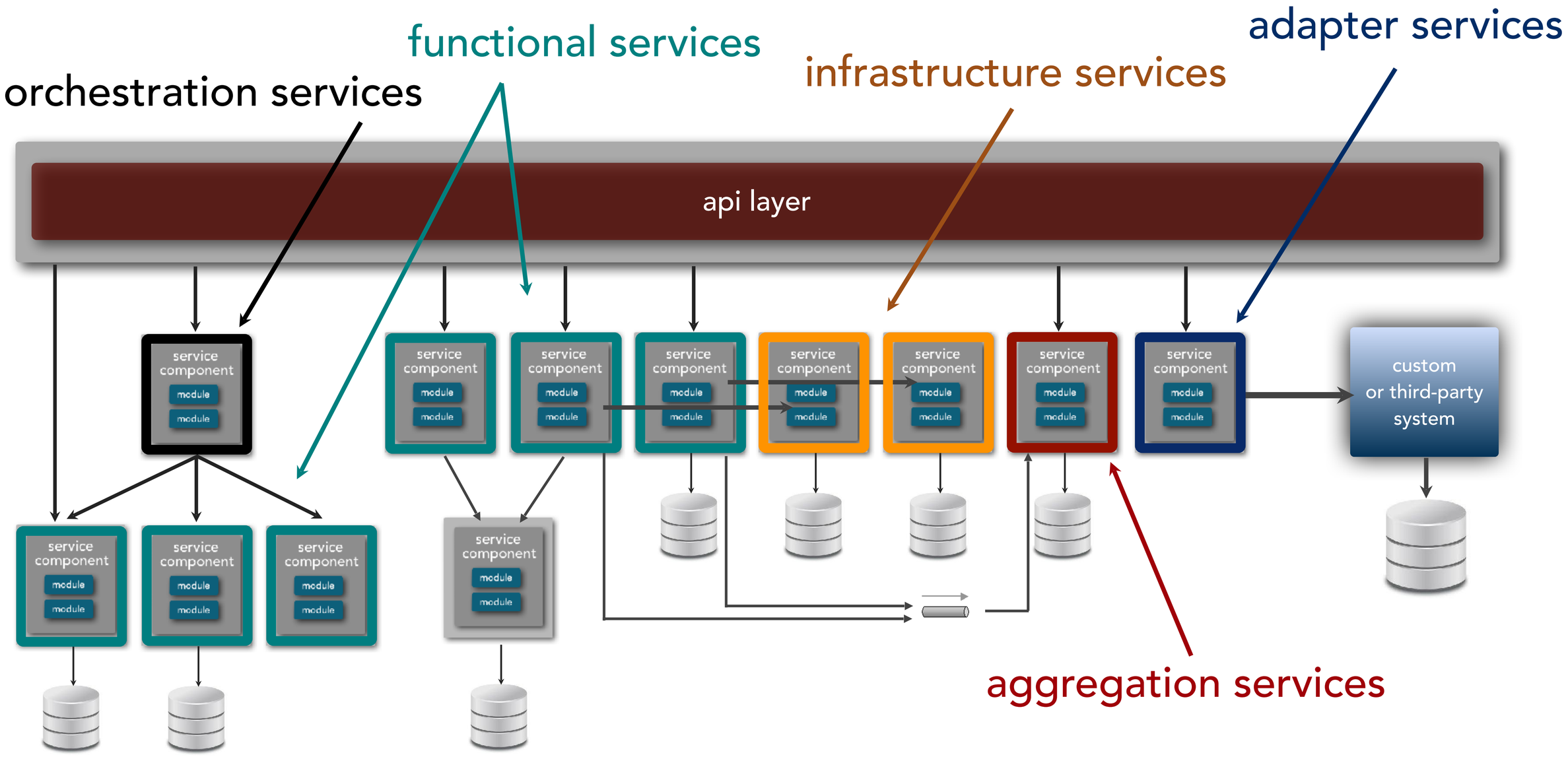


service types

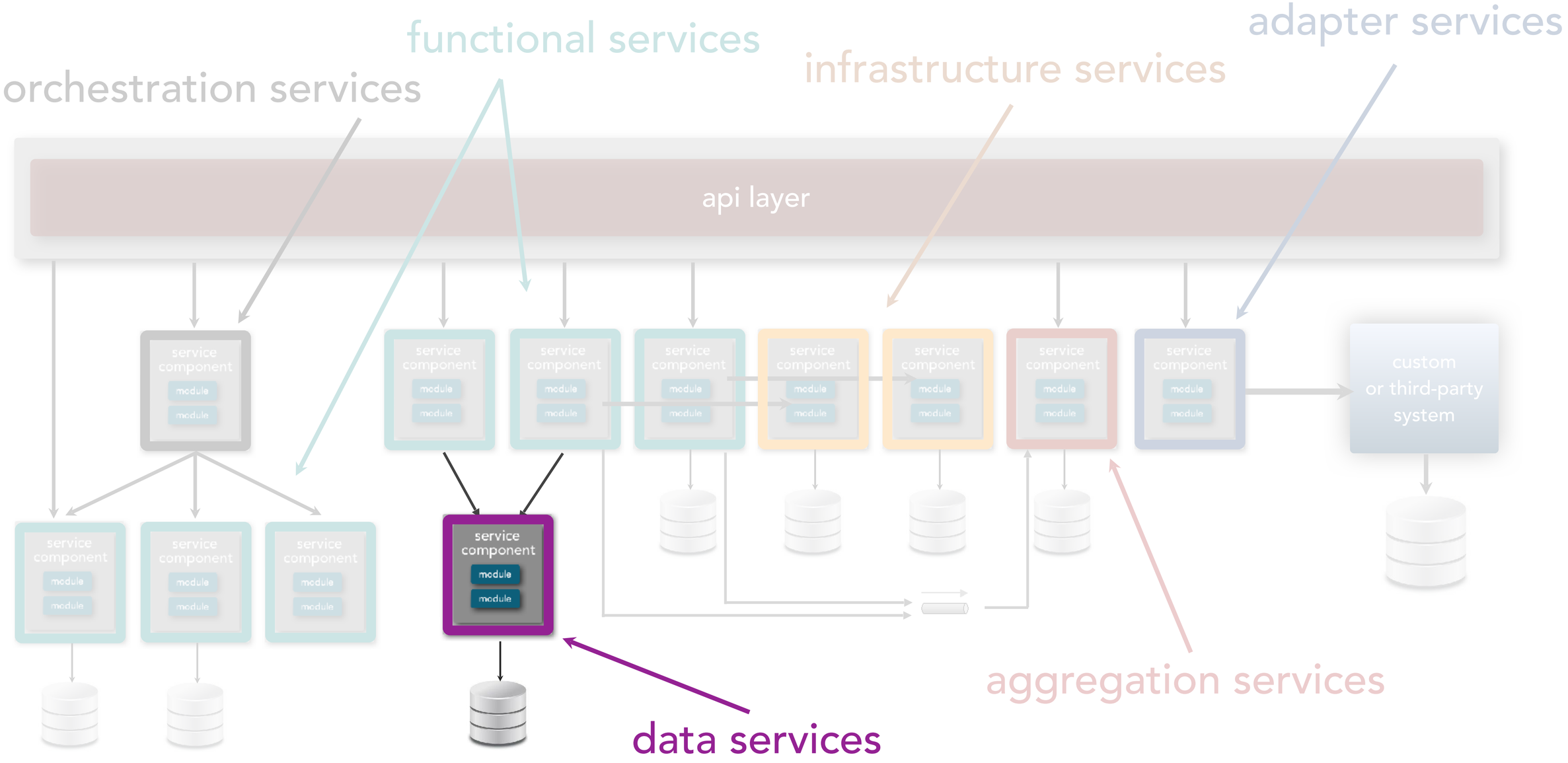
adapter services



service types

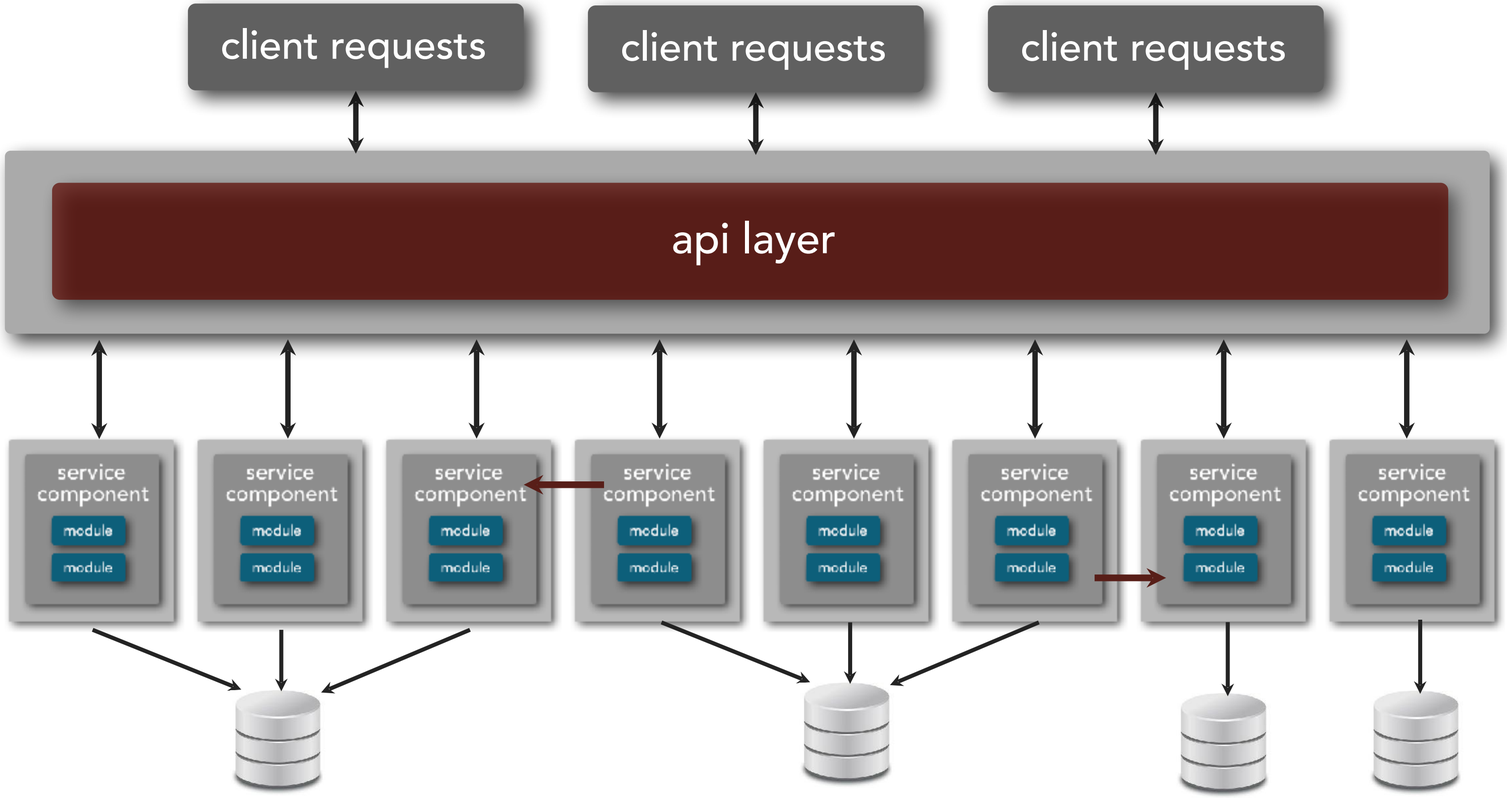


service types



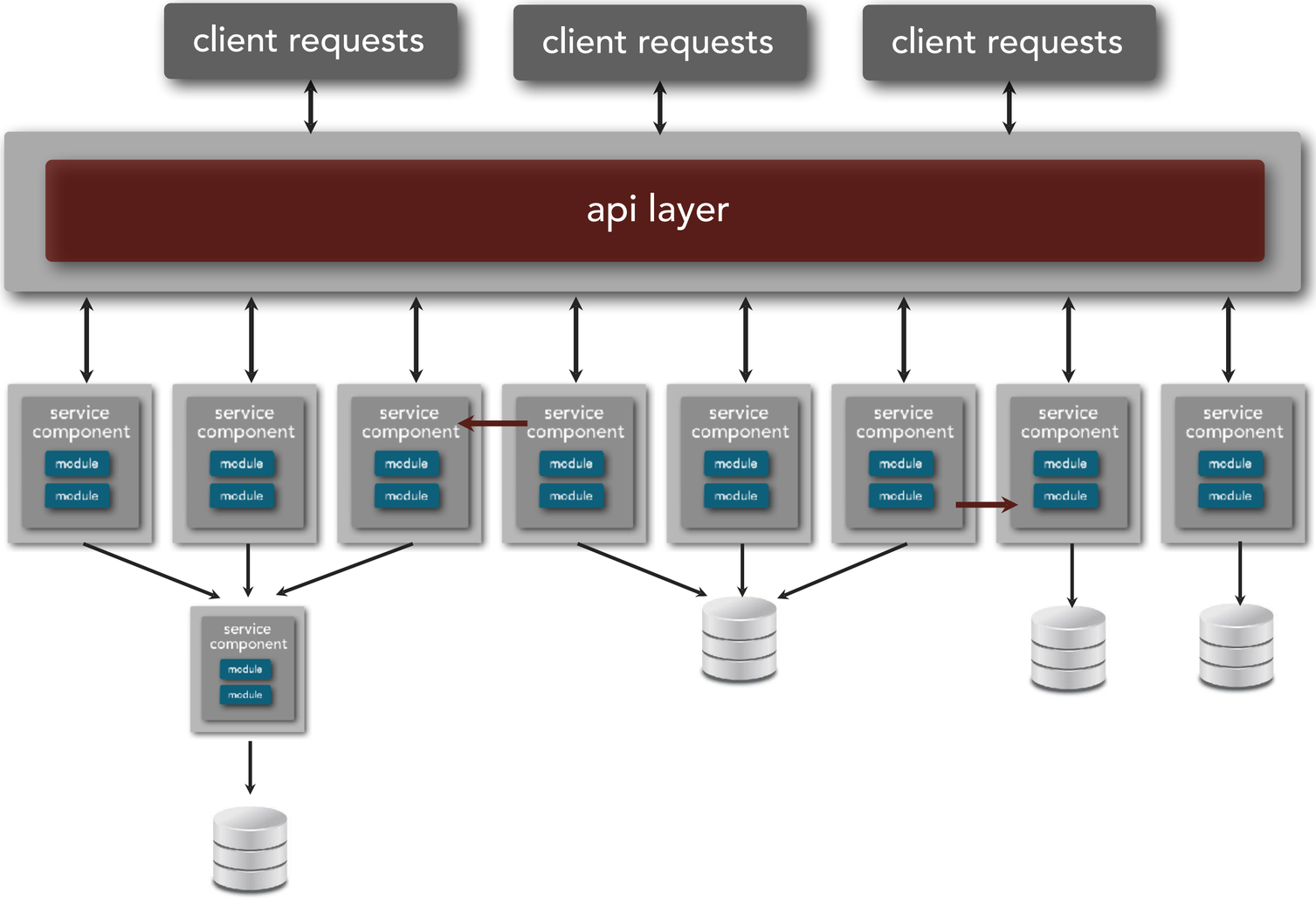
service types

data services



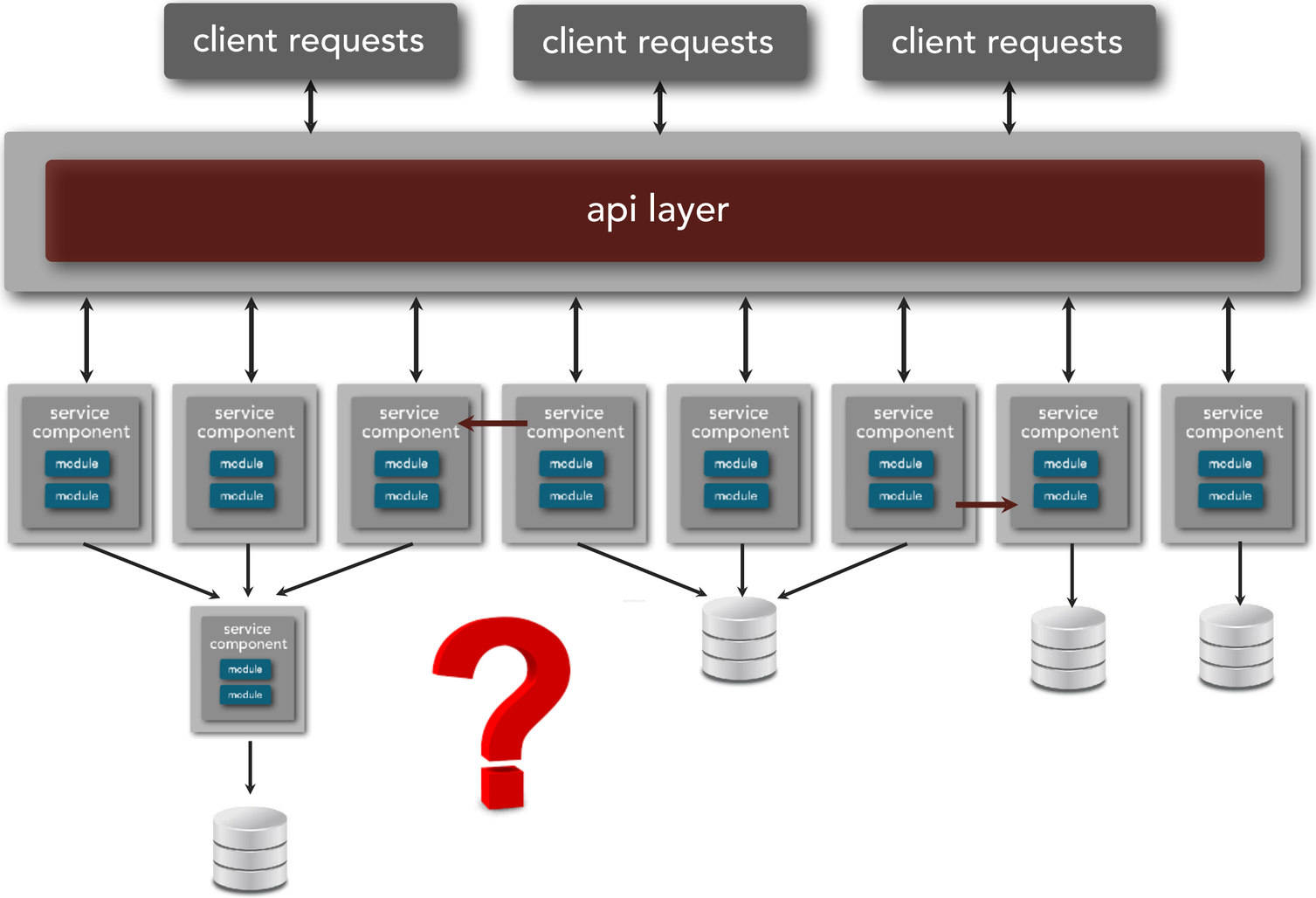
service types

data services



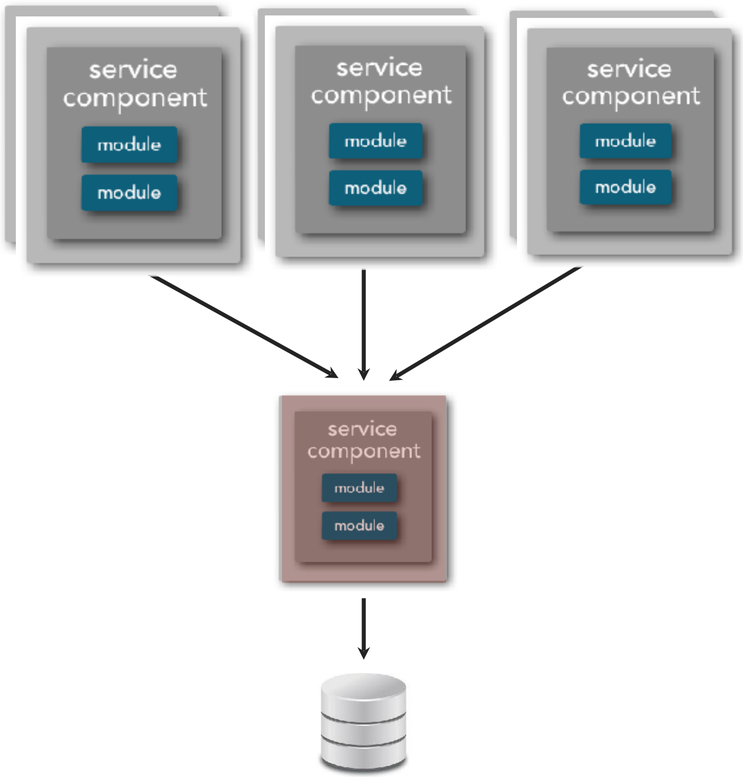
service types

data services



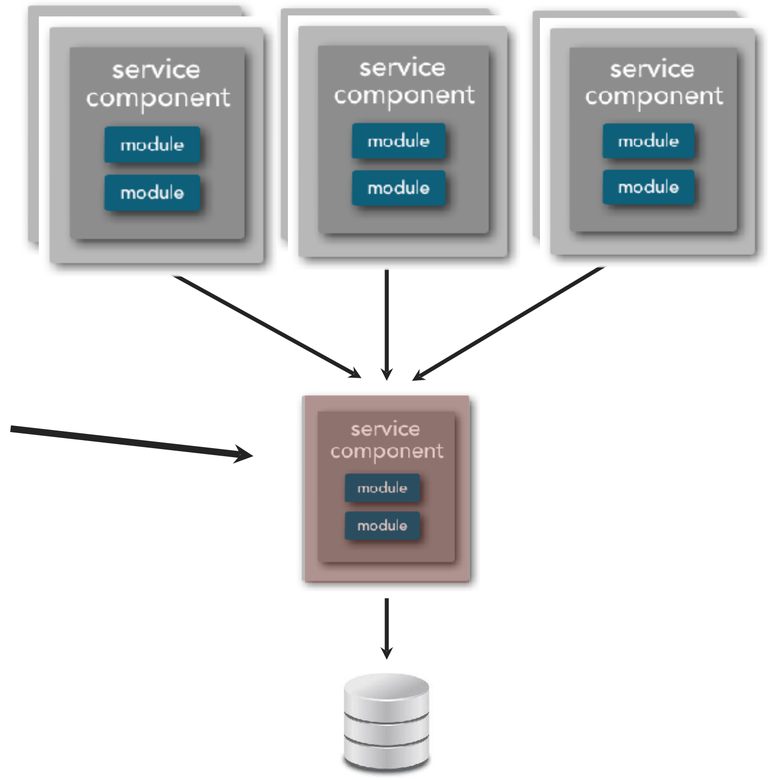
service types

data services



service types

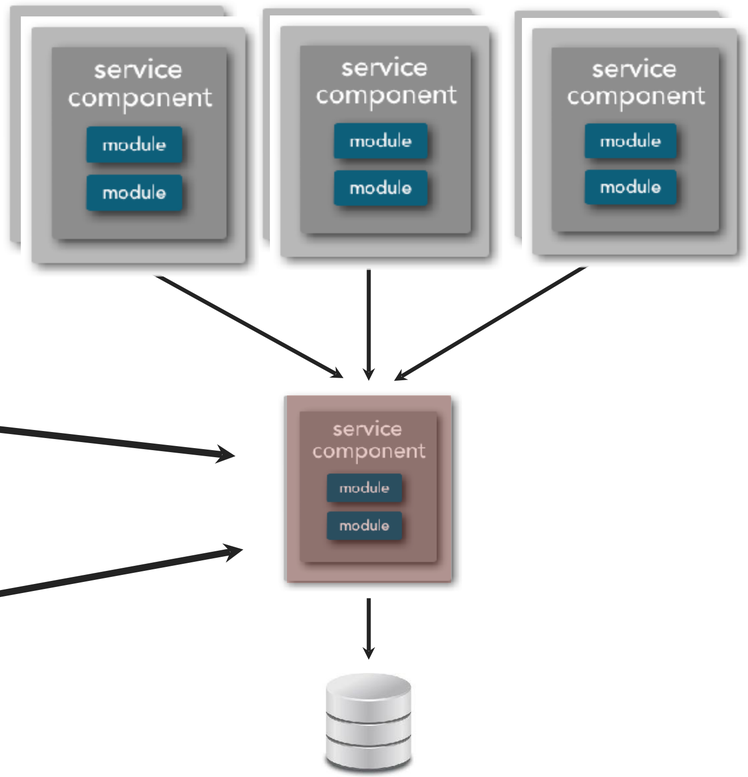
data services



single connection pool

service types

data services

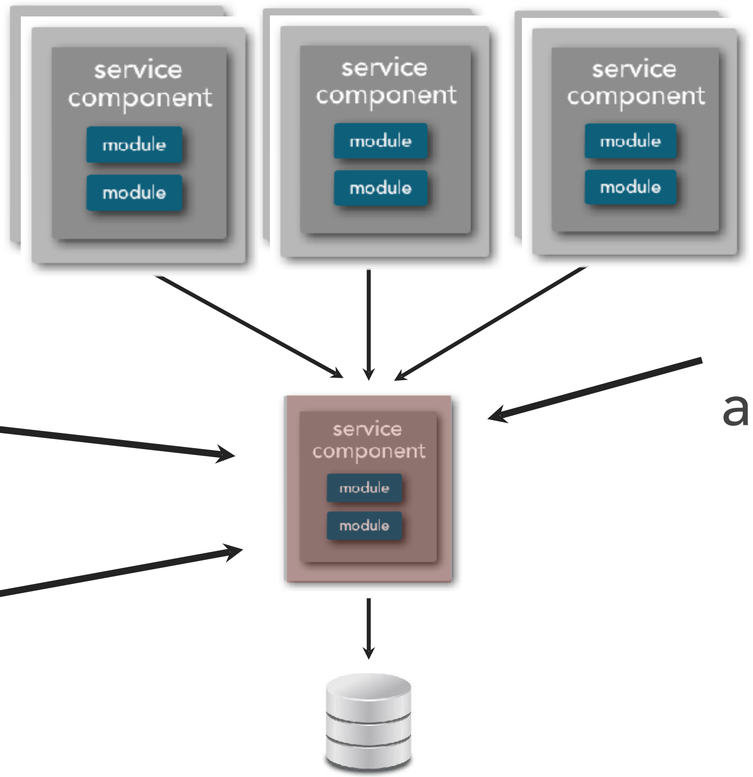


single connection pool

centrally manage data cache

service types

data services



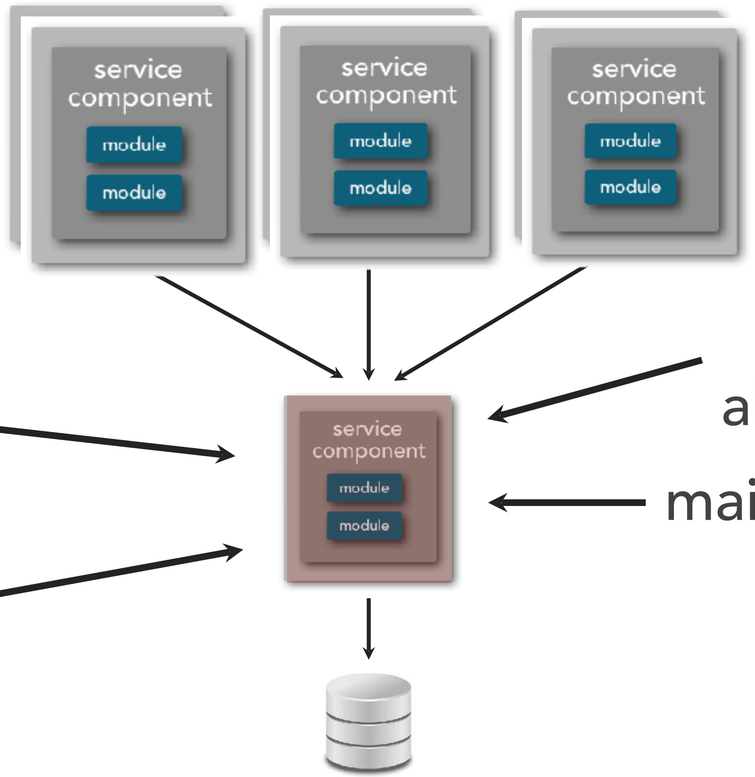
abstract database / sql queries

single connection pool

centrally manage data cache

service types

data services



abstract database / sql queries

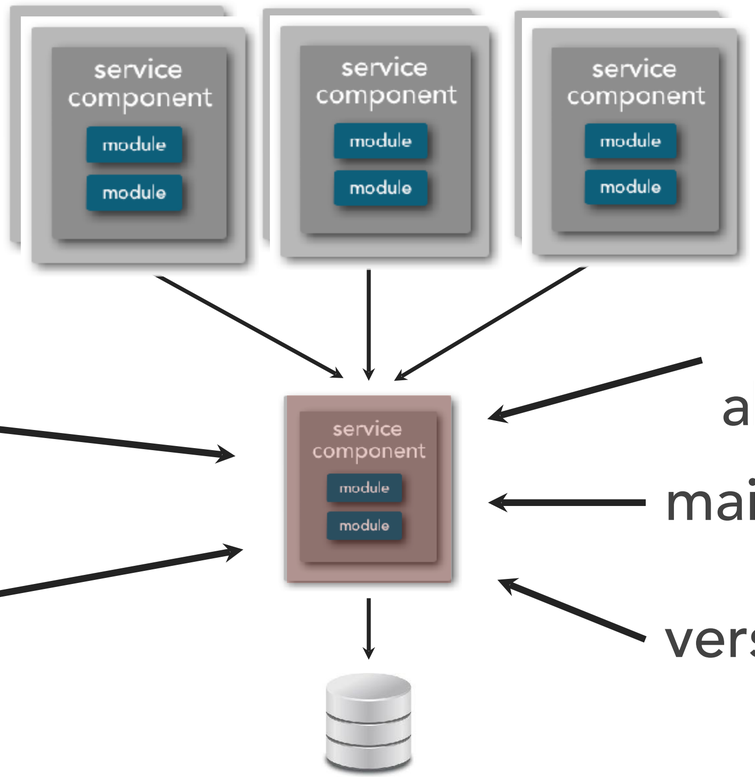
single connection pool

maintain write ownership

centrally manage data cache

service types

data services



abstract database / sql queries

single connection pool

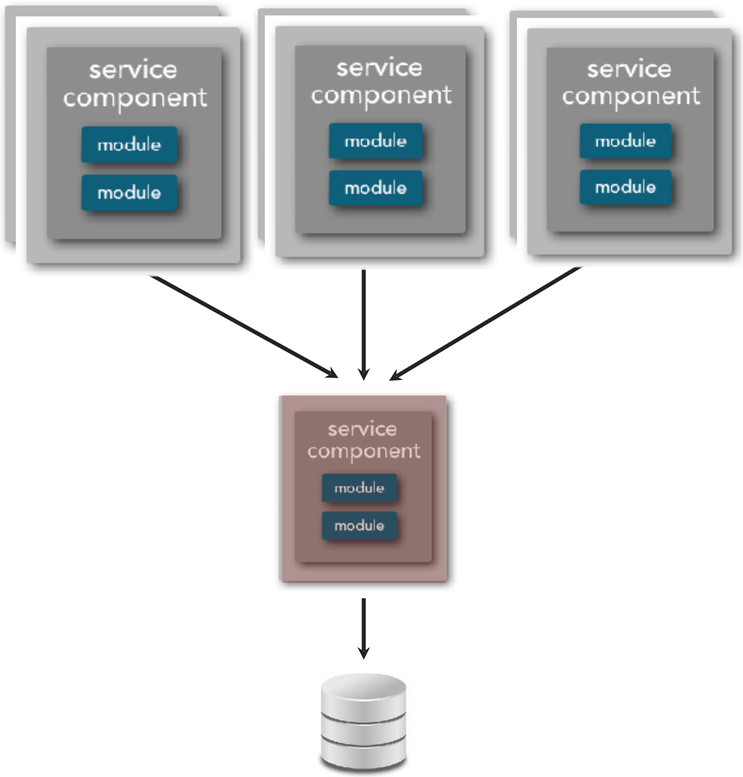
maintain write ownership

centrally manage data cache

version database changes

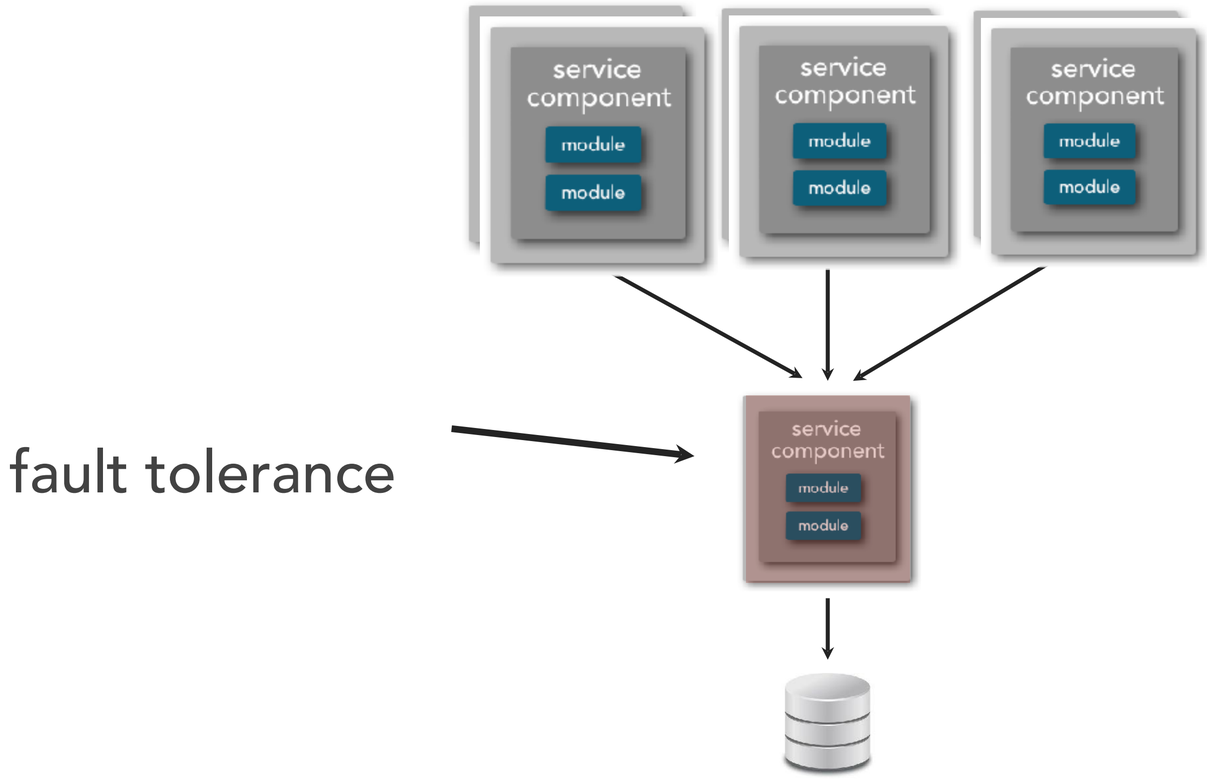
service types

data services



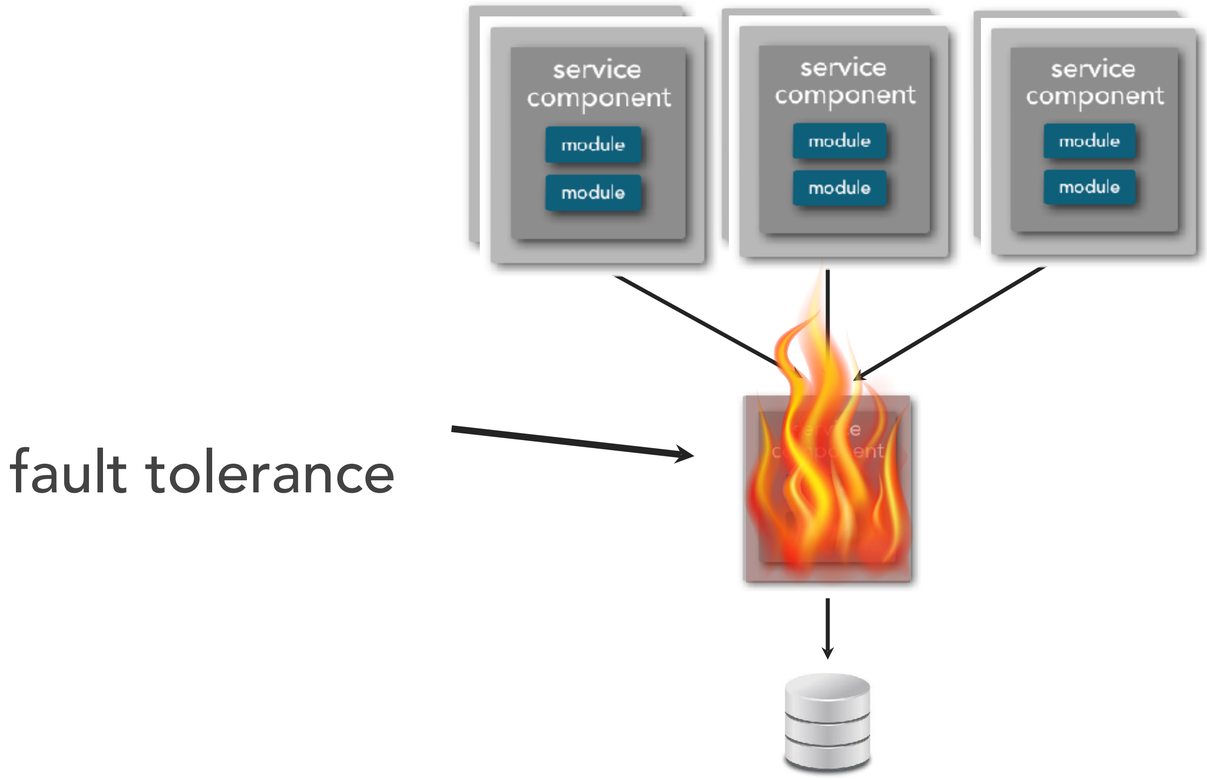
service types

data services

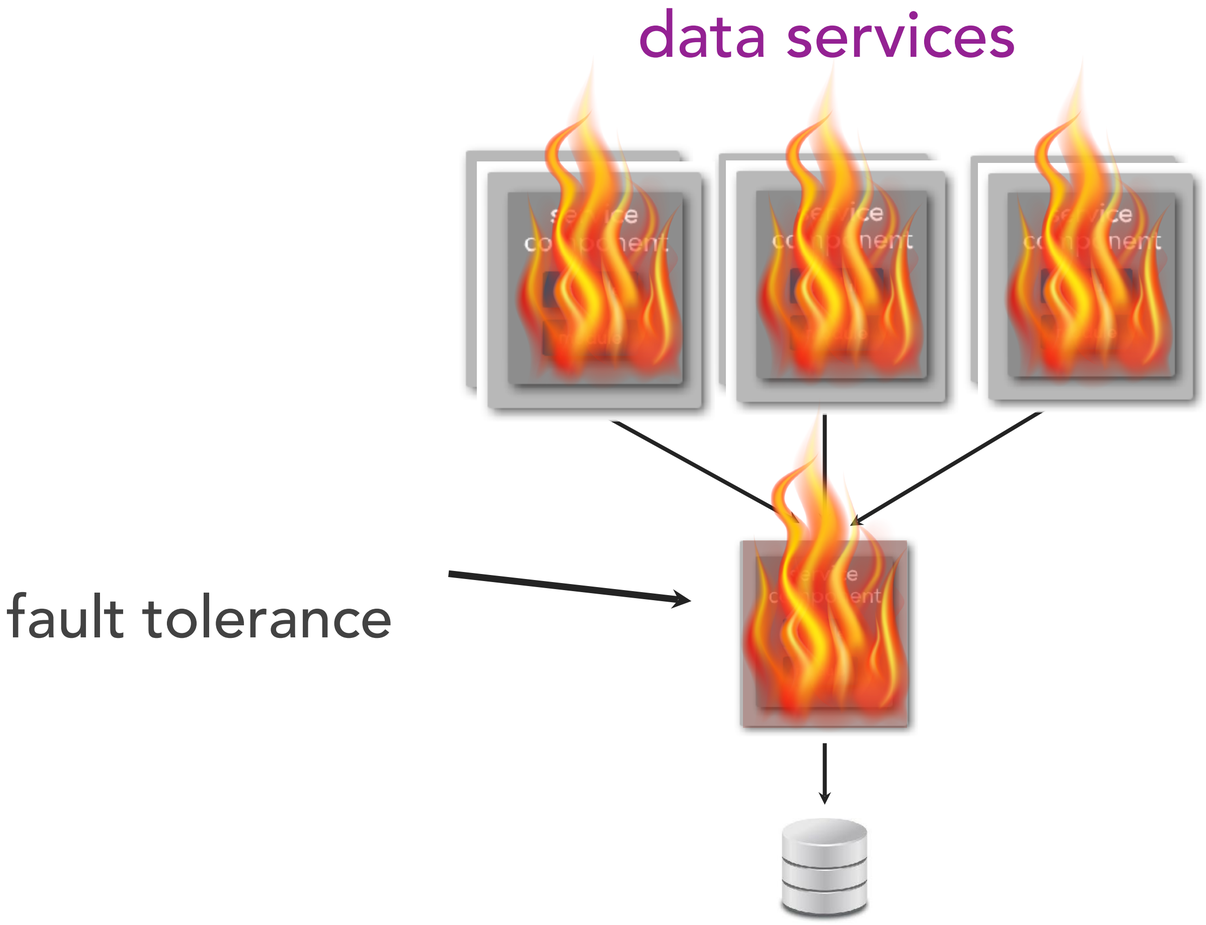


service types

data services

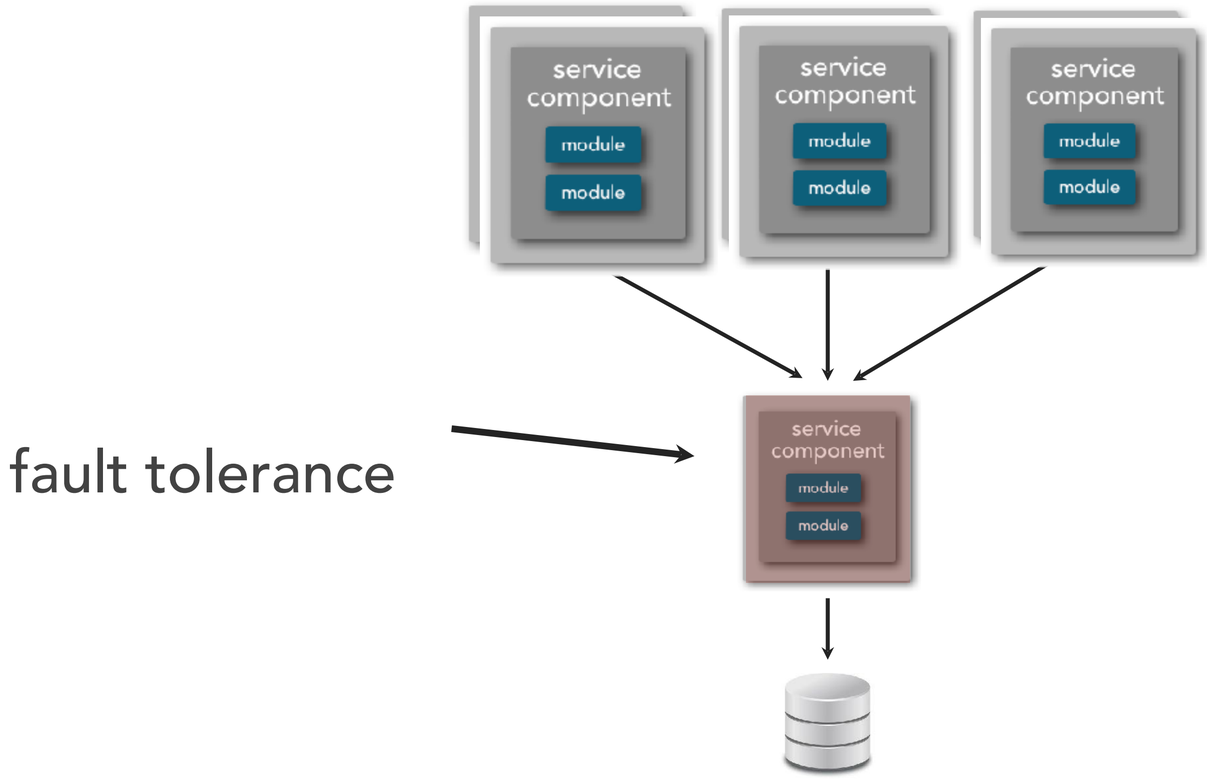


service types



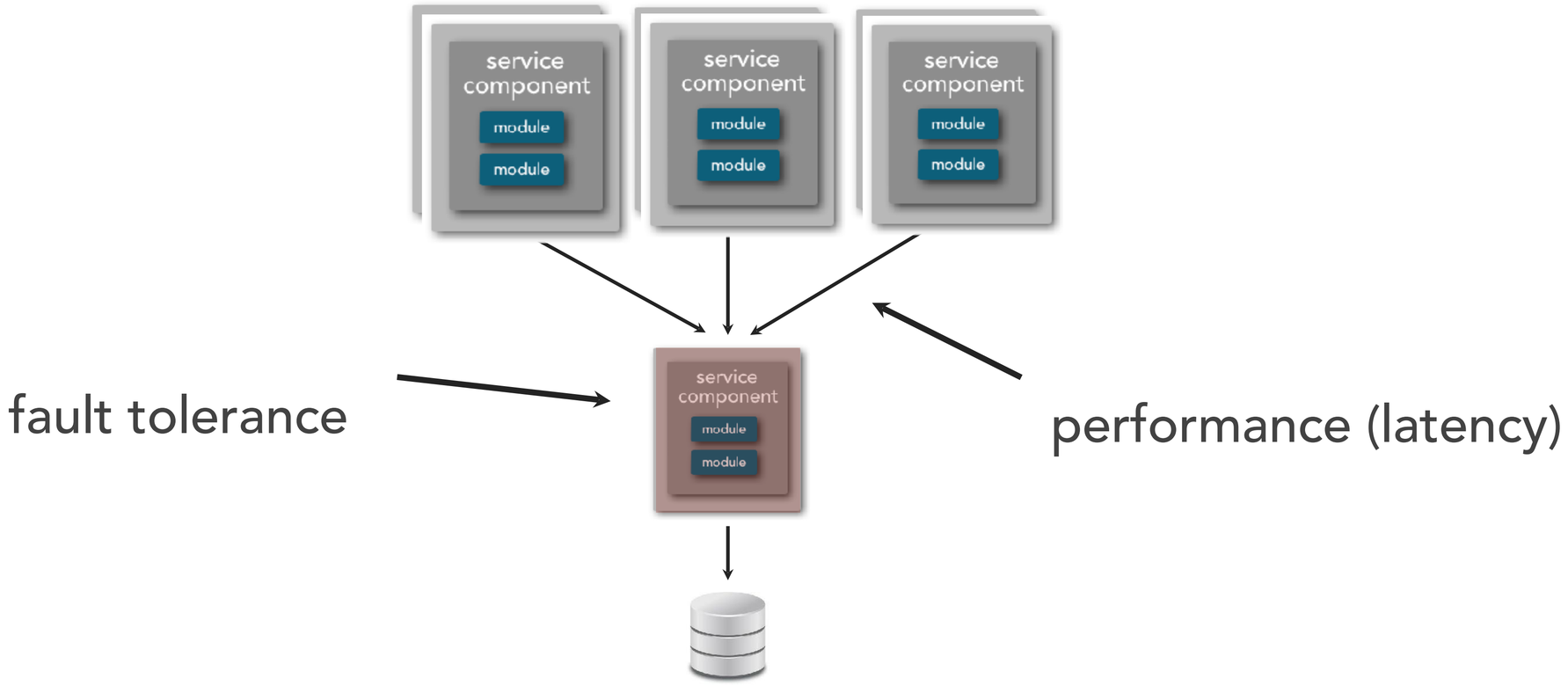
service types

data services



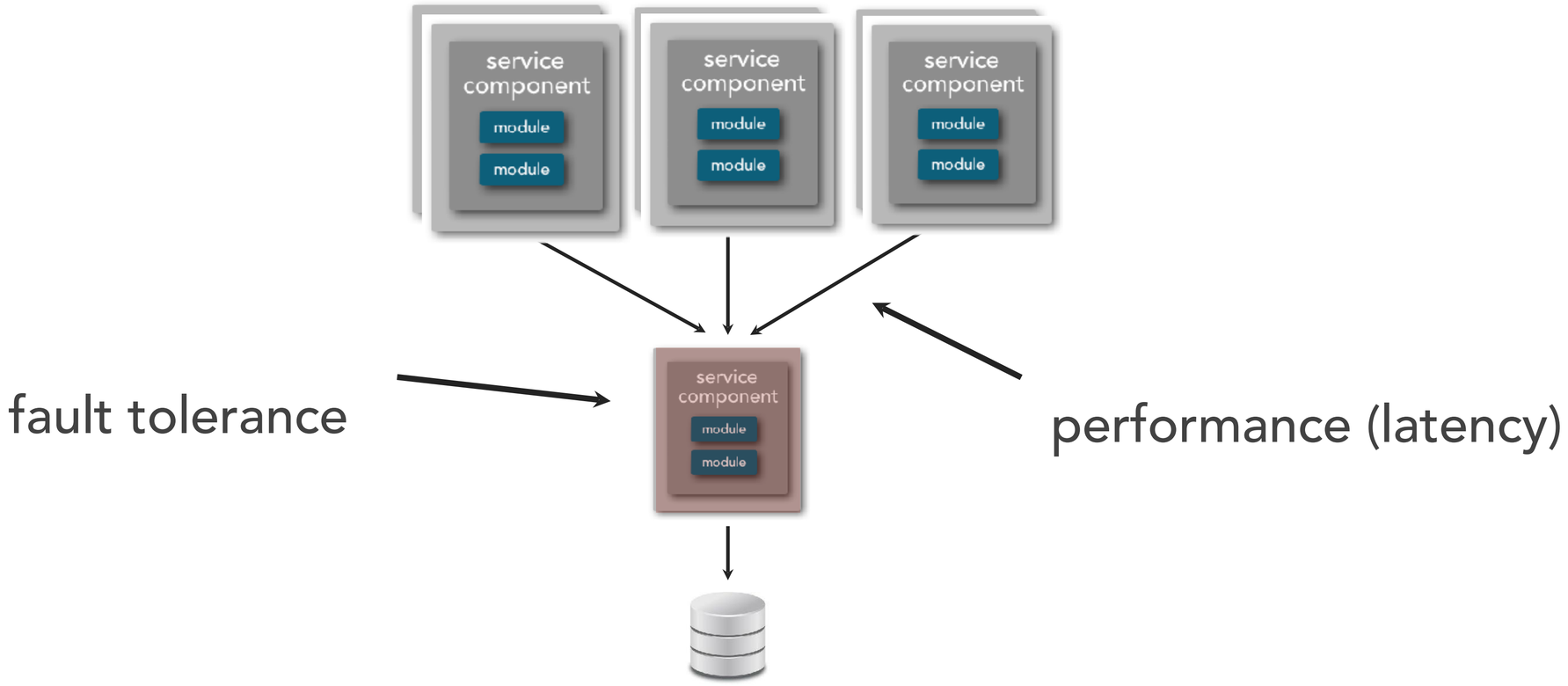
service types

data services



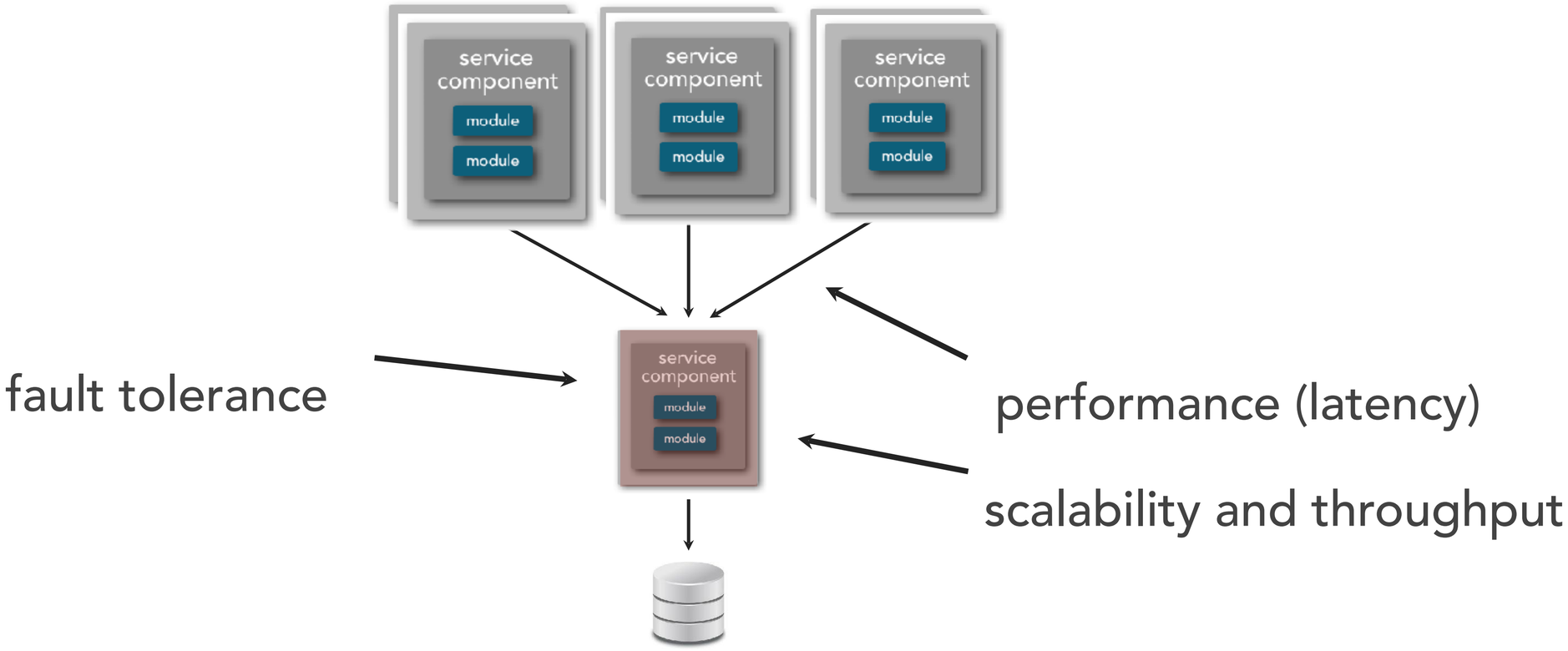
service types

data services



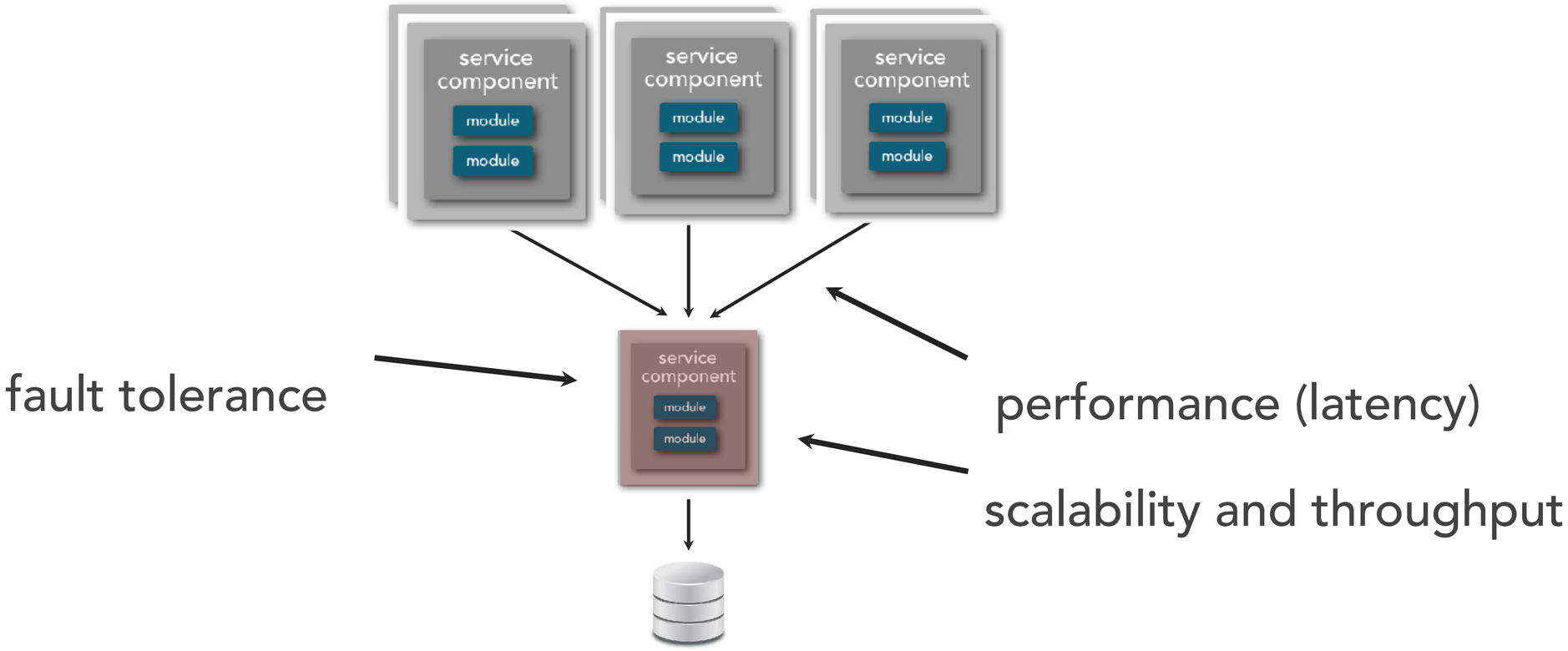
service types

data services



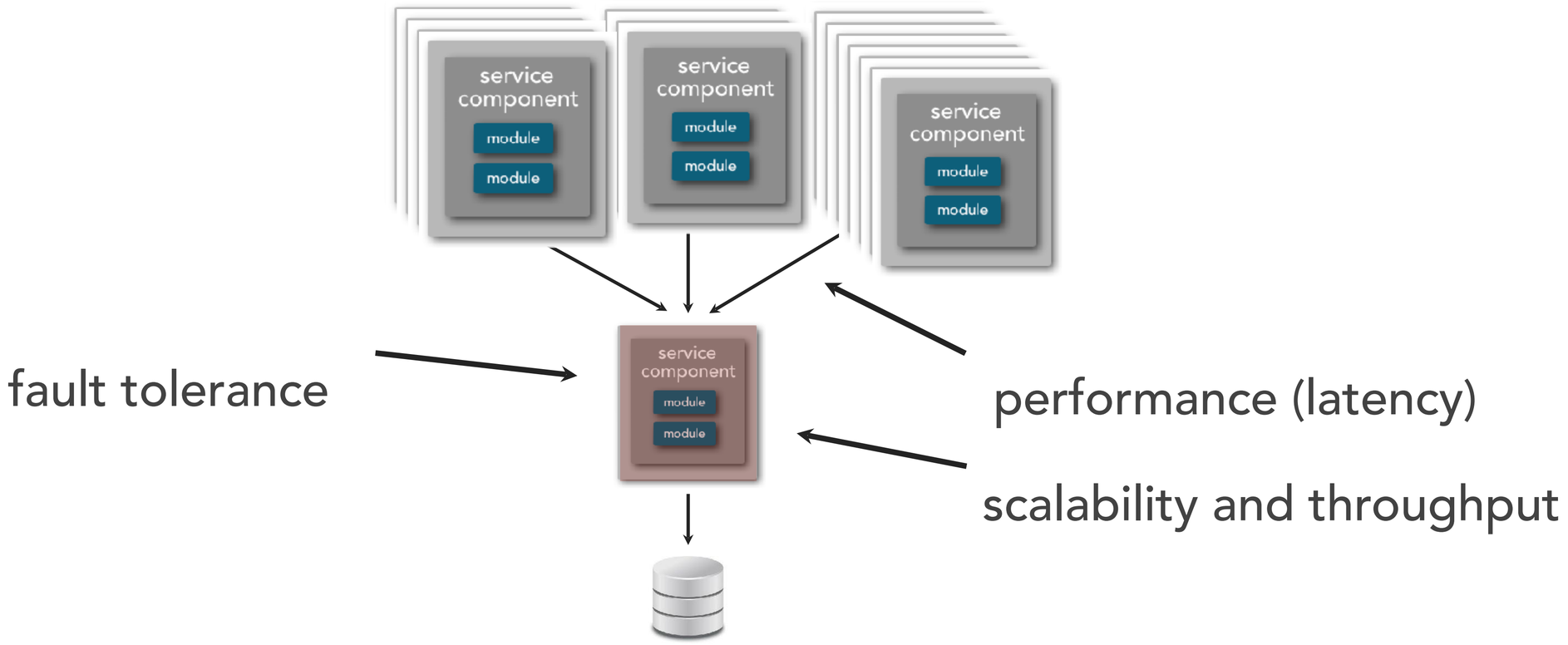
service types

data services



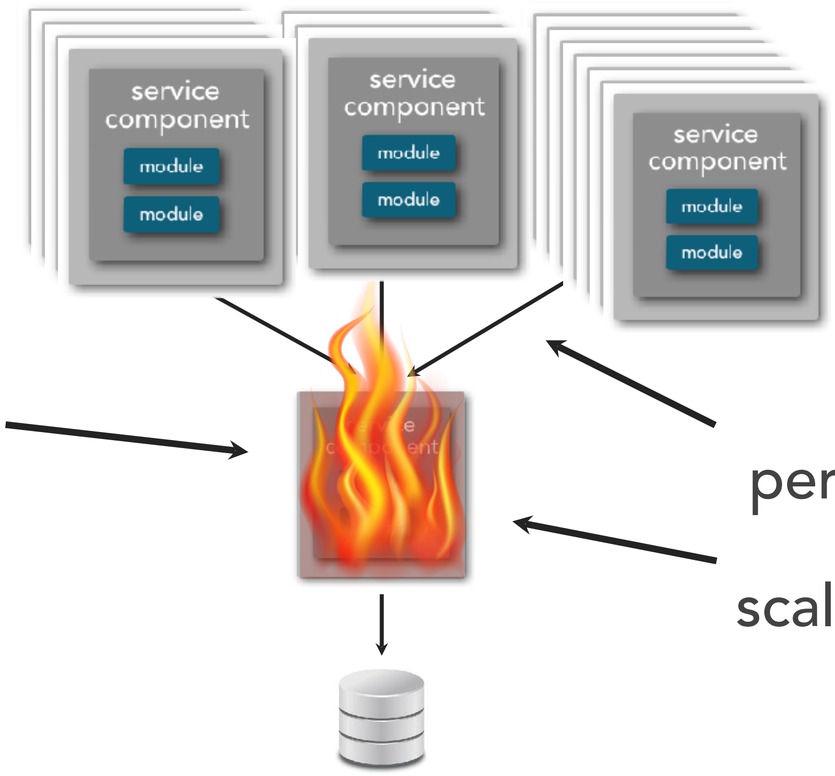
service types

data services



service types

data services



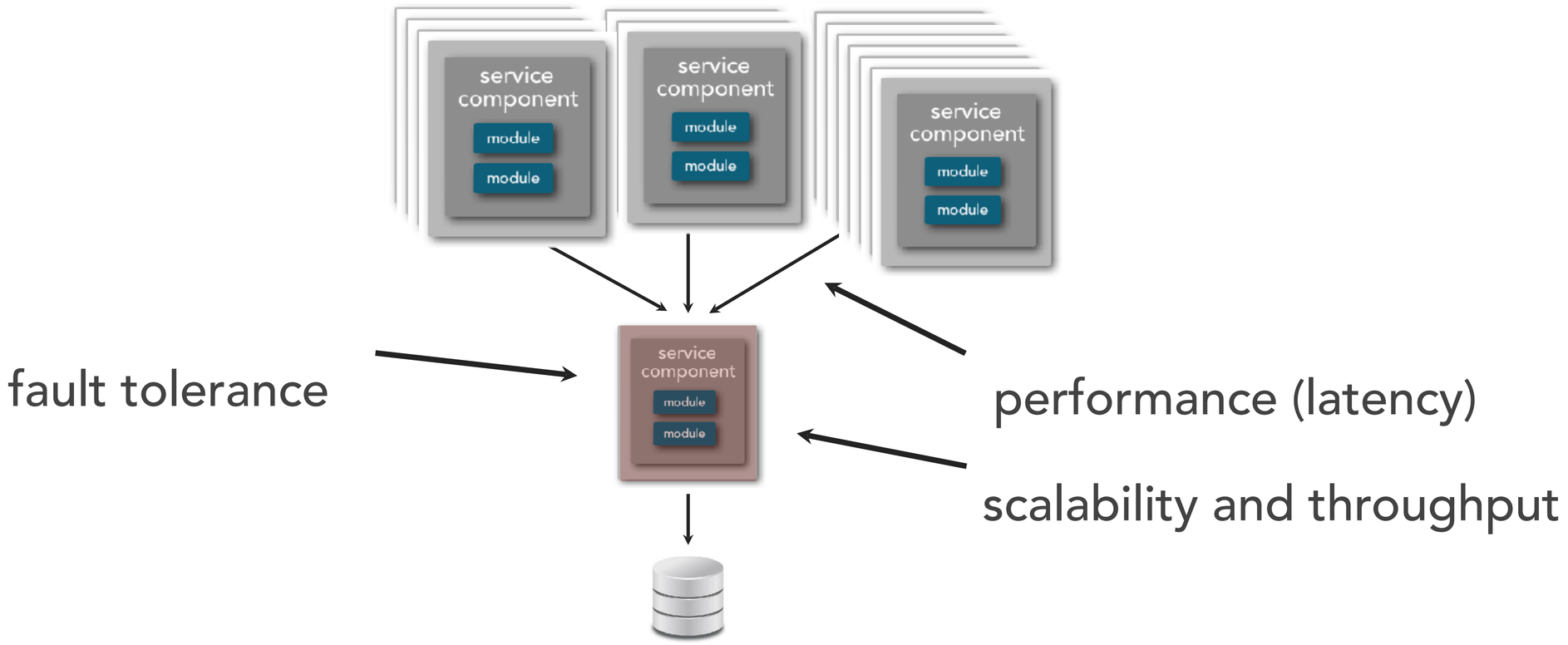
fault tolerance

performance (latency)

scalability and throughput

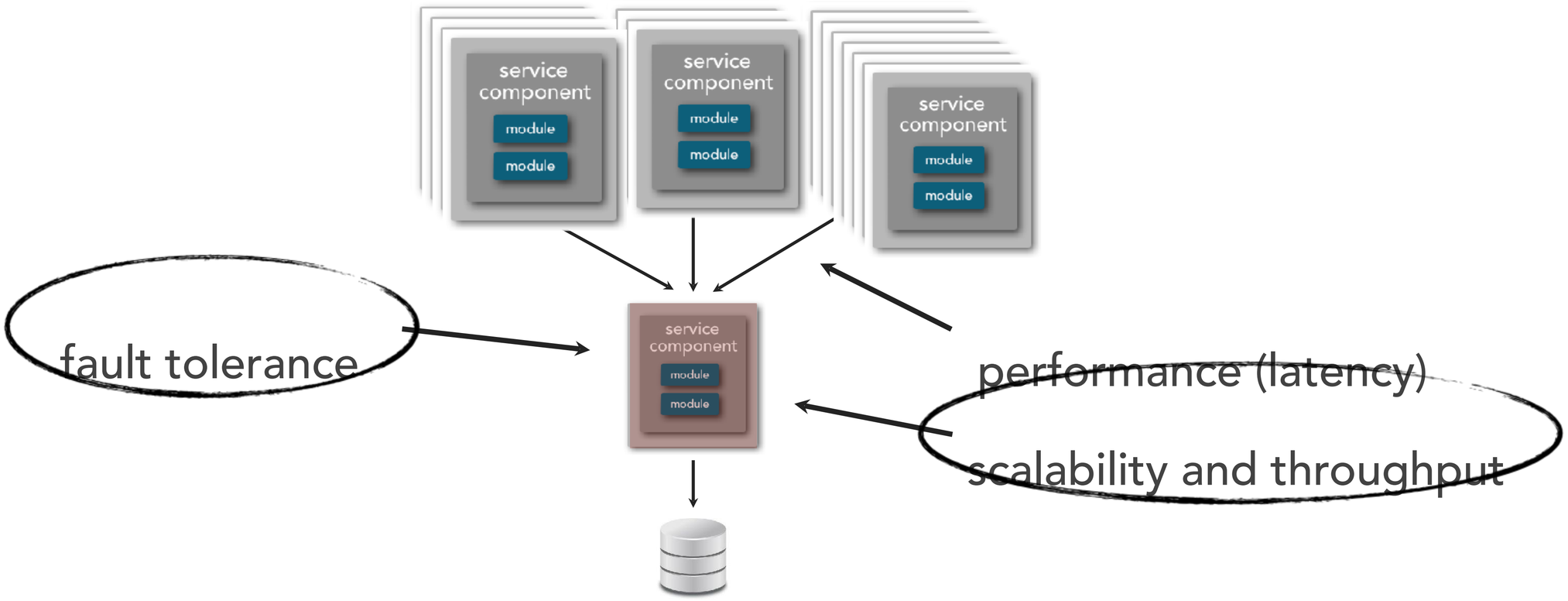
service types

data services



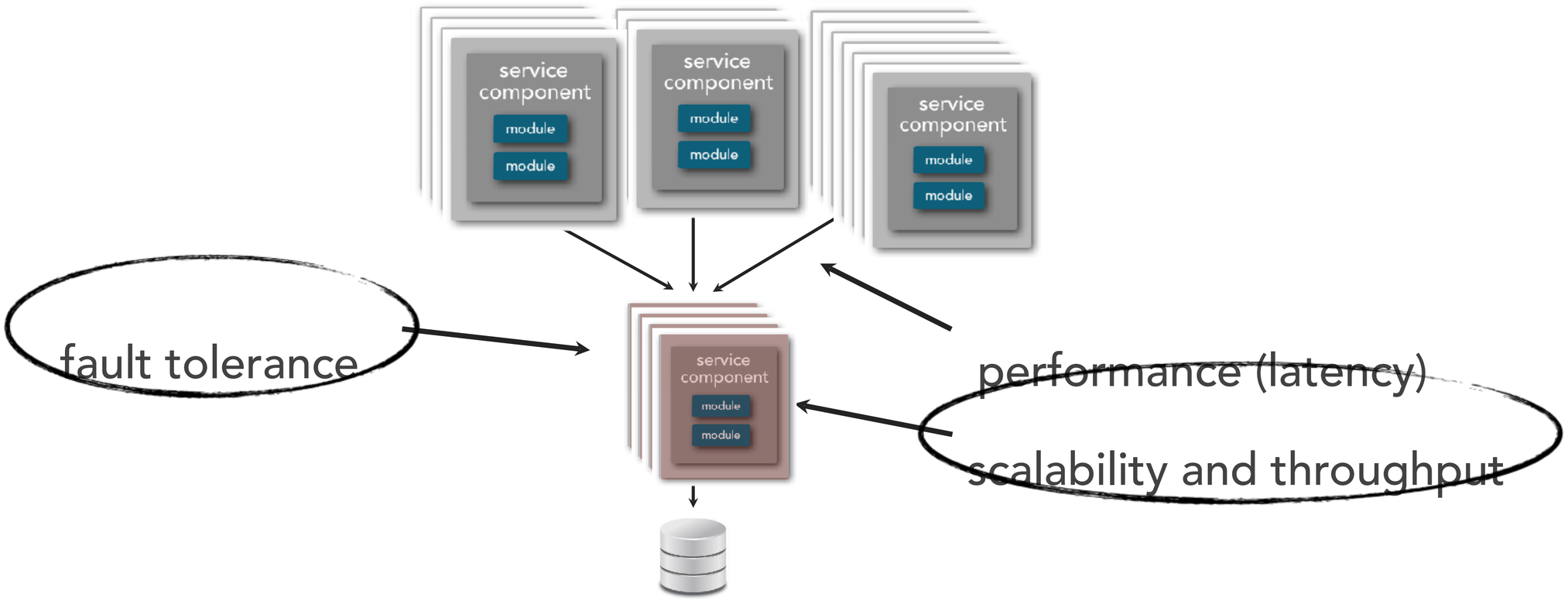
service types

data services



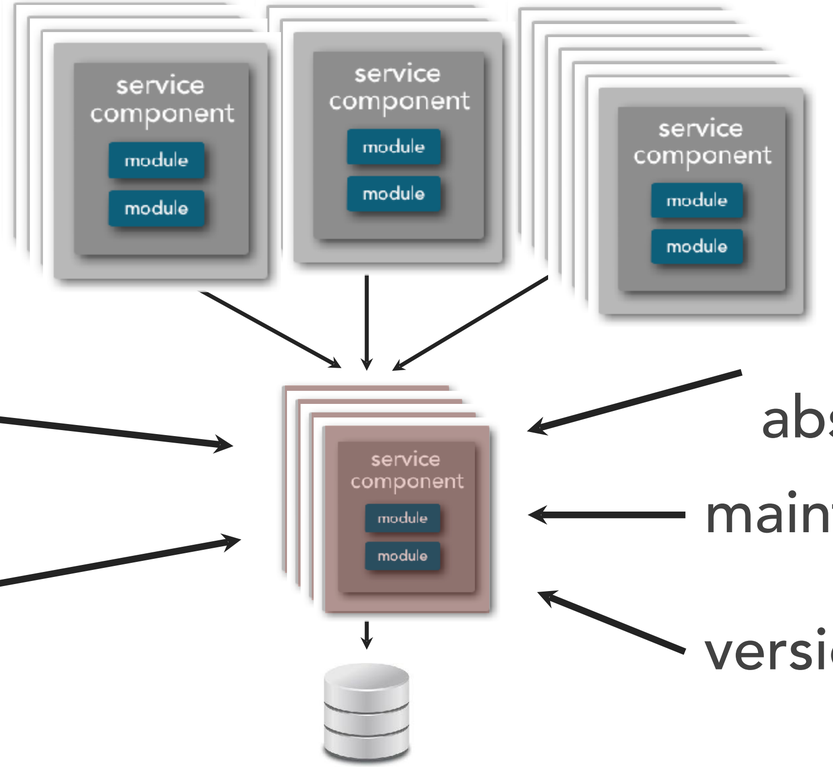
service types

data services



service types

data services



single connection pool

centrally manage data cache

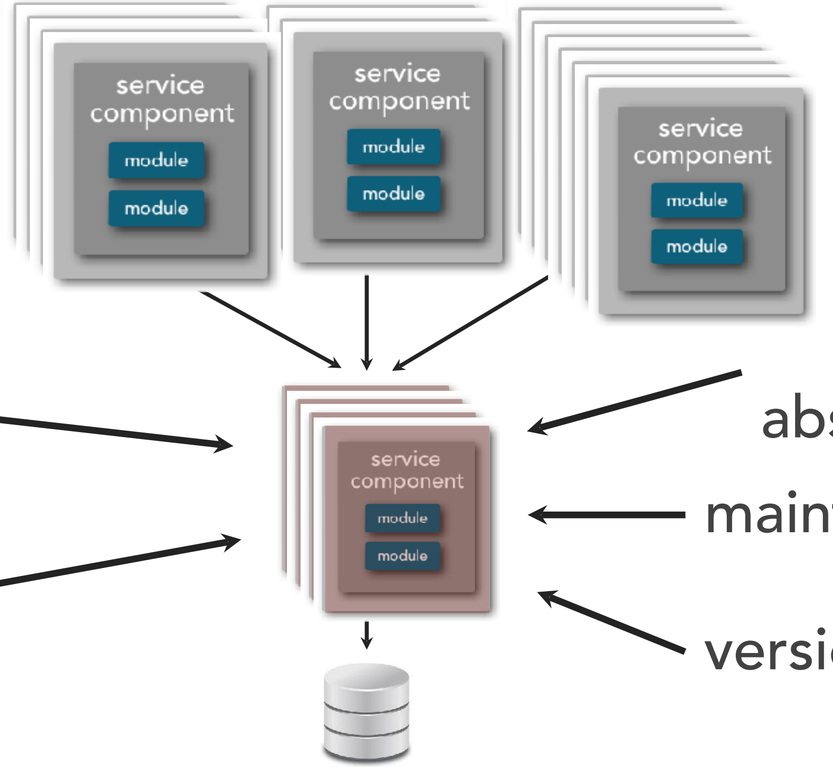
abstract database / sql queries

maintain write ownership

version database changes

service types

data services



single connection pool

centrally manage data cache

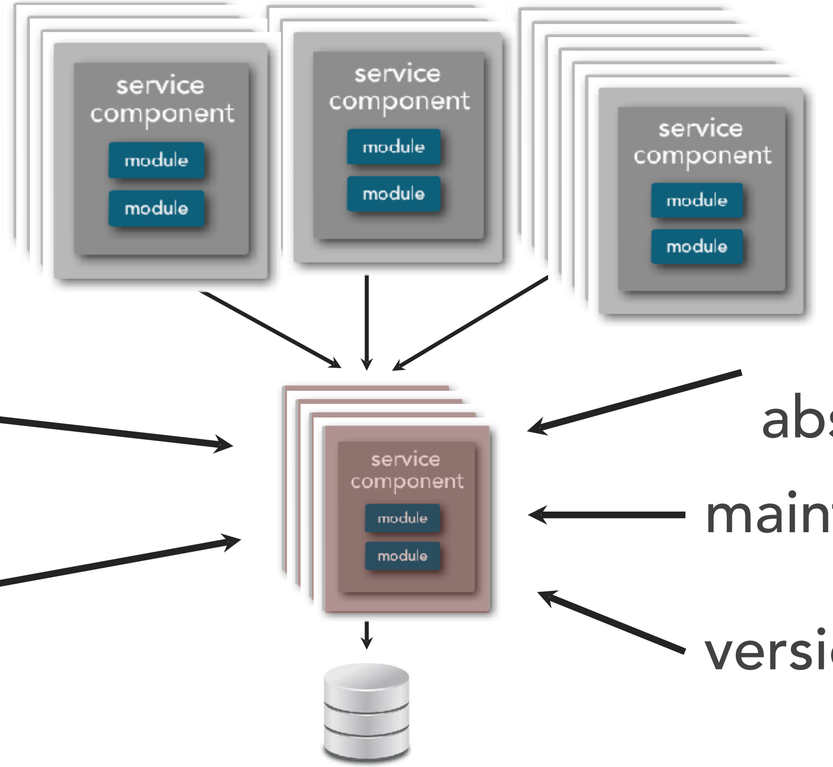
abstract database / sql queries

maintain write ownership

version database changes

service types

data services



single connection pool

centrally manage data cache

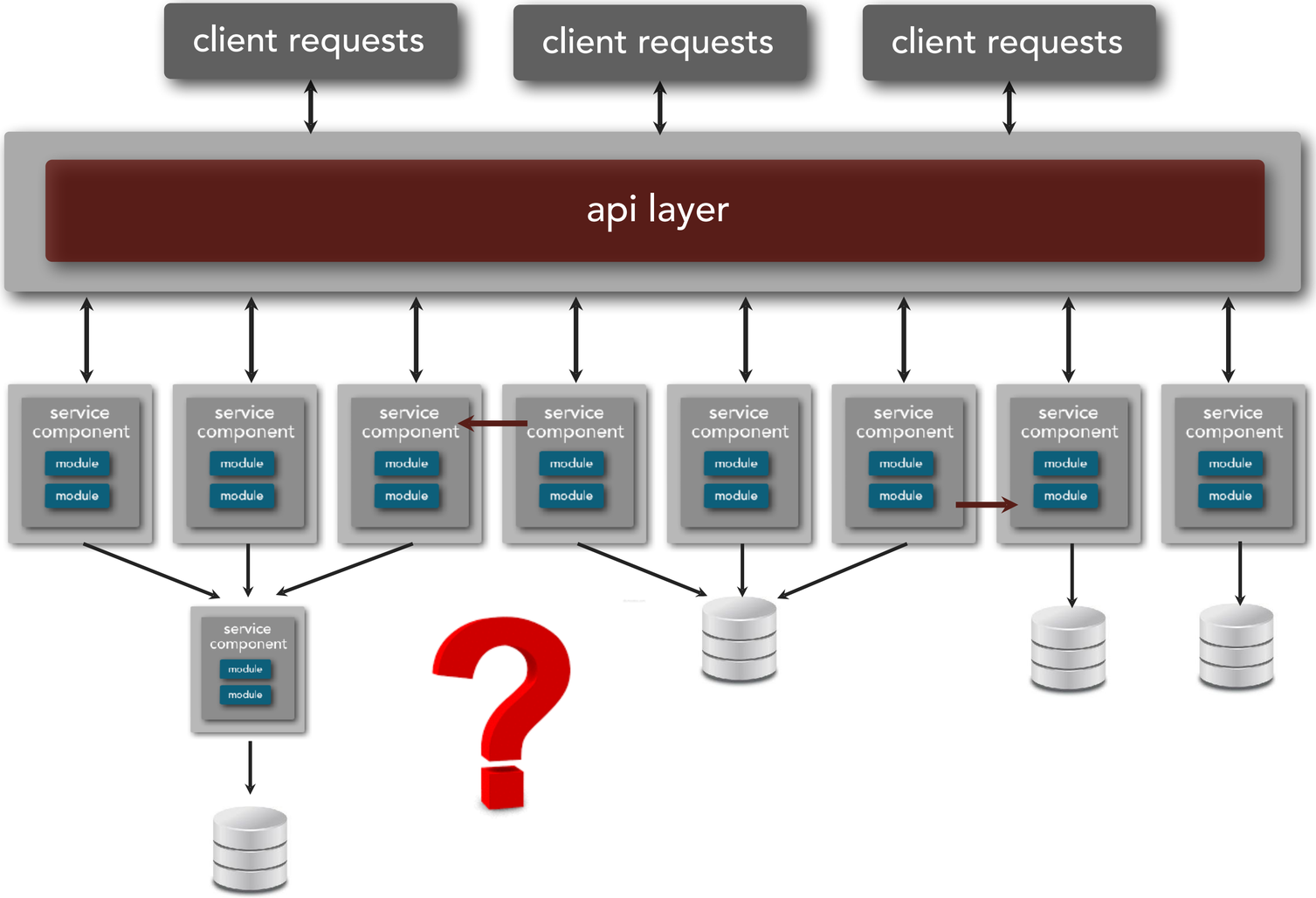
abstract database / sql queries

maintain write ownership

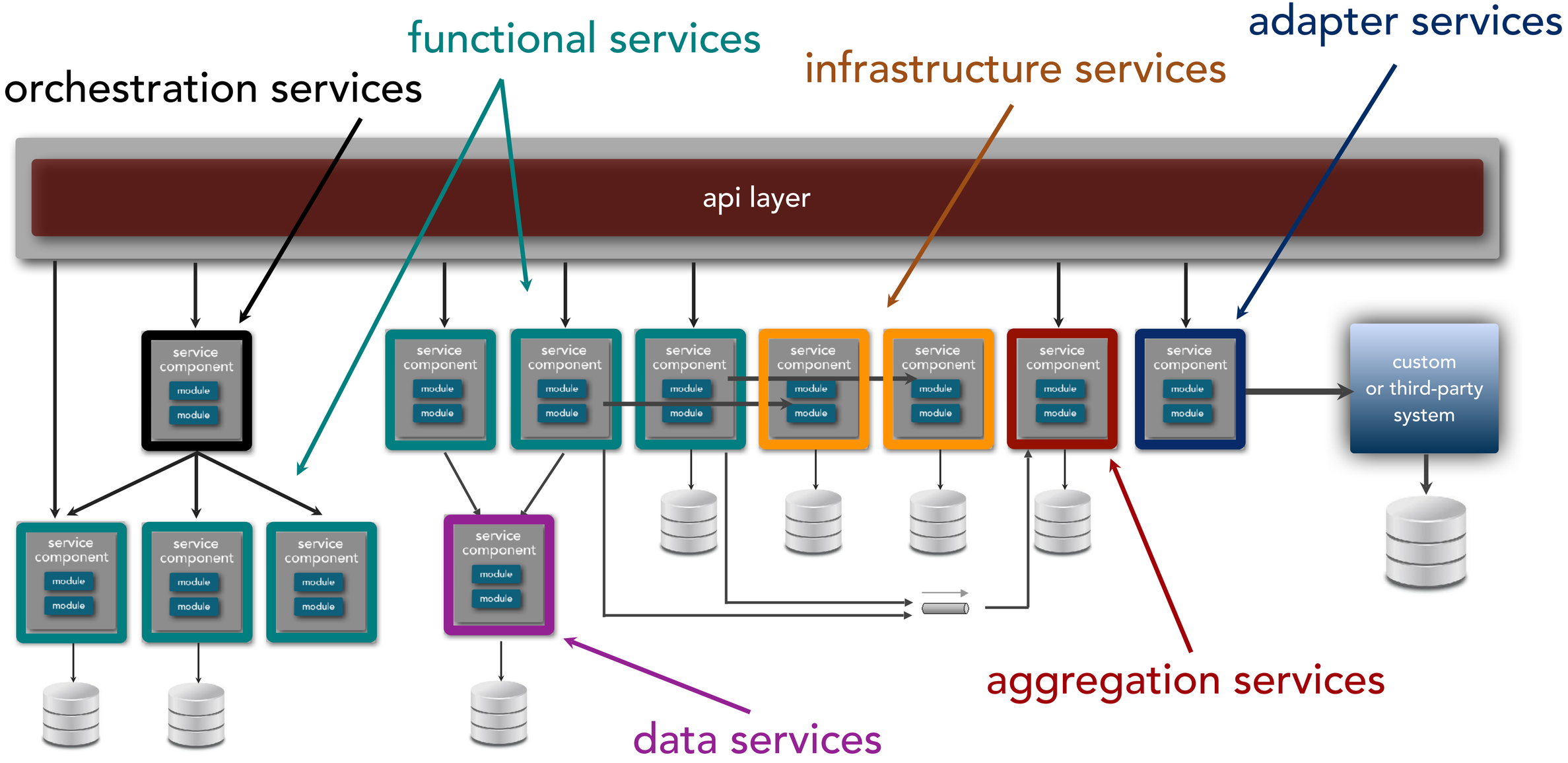
version database changes

service types

data services



service types



service types

service types

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface FunctionalService {}
```

service types

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface FunctionalService {}
```

```
@ServiceEntrypoint
@FunctionalService
public class PaymentServiceAPI {
    ...
}
```

service types

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface FunctionalService {}
```

```
@ServiceEntrypoint
@FunctionalService
public class PaymentServiceAPI {
    ...
}
```

service types

service types

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface OrchestrationService {}
```

service types

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface OrchestrationService {}
```

```
@ServiceEntrypoint
@OrchestrationService
public class CustomerRegistration {
    ...
}
```

service types

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface OrchestrationService {}
```

```
@ServiceEntrypoint
@OrchestrationService
public class CustomerRegistration {
    ...
}
```

service types

service types

```
[System.AttributeUsage(System.AttributeTargets.Class)]
class FunctionalService : System.Attribute {}
```

service types

```
[System.AttributeUsage(System.AttributeTargets.Class)]
class FunctionalService : System.Attribute {}
```

```
[ServiceEntrypoint]
[FunctionalService]
class PaymentServiceAPI {
    ...
}
```

service types

```
[System.AttributeUsage(System.AttributeTargets.Class)]
class FunctionalService : System.Attribute {}
```

```
[ServiceEntrypoint]
[FunctionalService]
class PaymentServiceAPI {
    ...
}
```

service types

service types

```
@FunctionalService
@OrchestrationService
@AggregationService
@AdapterService
@InfrastructureService
@DataService
```

service types

@FunctionalService	[FunctionalService]
@OrchestrationService	[OrchestrationService]
@AggregationService	[AggregationService]
@AdapterService	[AdapterService]
@InfrastructureService	[InfrastructureService]
@DataService	[DataService]

service types

`@FunctionalService`	`[FunctionalService]`
`@OrchestrationService`	`[OrchestrationService]`
`@AggregationService`	`[AggregationService]`
`@AdapterService`	`[AdapterService]`
`@InfrastructureService`	`[InfrastructureService]`
`@DataService`	`[DataService]`

✓ service identification and service context (developer info)

service types

`@FunctionalService`	`[FunctionalService]`
`@OrchestrationService`	`[OrchestrationService]`
`@AggregationService`	`[AggregationService]`
`@AdapterService`	`[AdapterService]`
`@InfrastructureService`	`[InfrastructureService]`
`@DataService`	`[DataService]`

✓ service identification and service context (developer info)

✓ metrics gathering to identify structural decay

e.g., no more than 10% orchestration services within an application context

e.g., no more than 5% aggregation services within an application context

service types

`@FunctionalService`	`[FunctionalService]`
`@OrchestrationService`	`[OrchestrationService]`
`@AggregationService`	`[AggregationService]`
`@AdapterService`	`[AdapterService]`
`@InfrastructureService`	`[InfrastructureService]`
`@DataService`	`[DataService]`

✓ service identification and service context (developer info)

✓ metrics gathering to identify structural decay

e.g., no more than 10% orchestration services within an application context

e.g., no more than 5% aggregation services within an application context

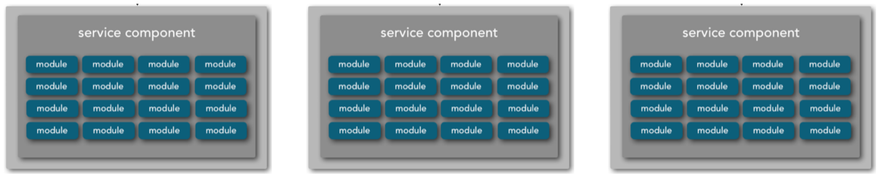
✓ architectural governance

e.g., adapter services cannot own or access data

Identifying Services

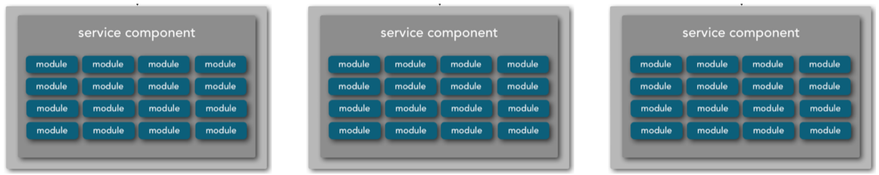
identifying microservices

identifying microservices

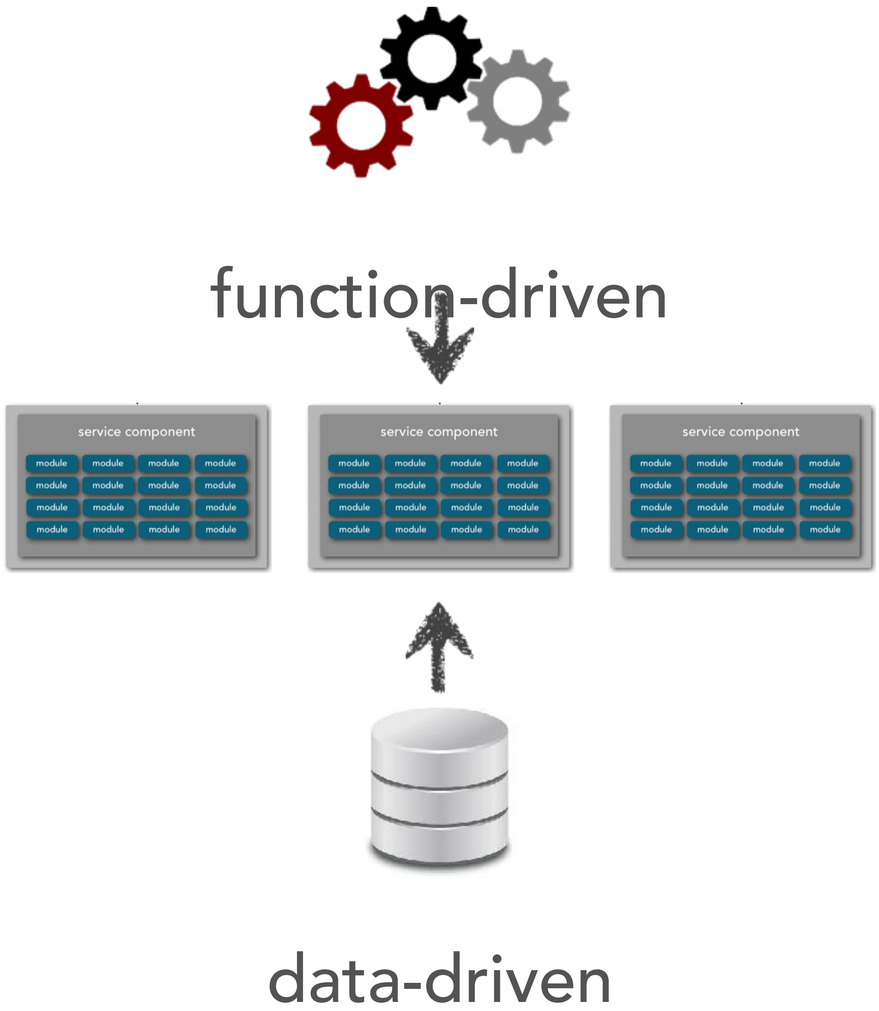


identifying microservices

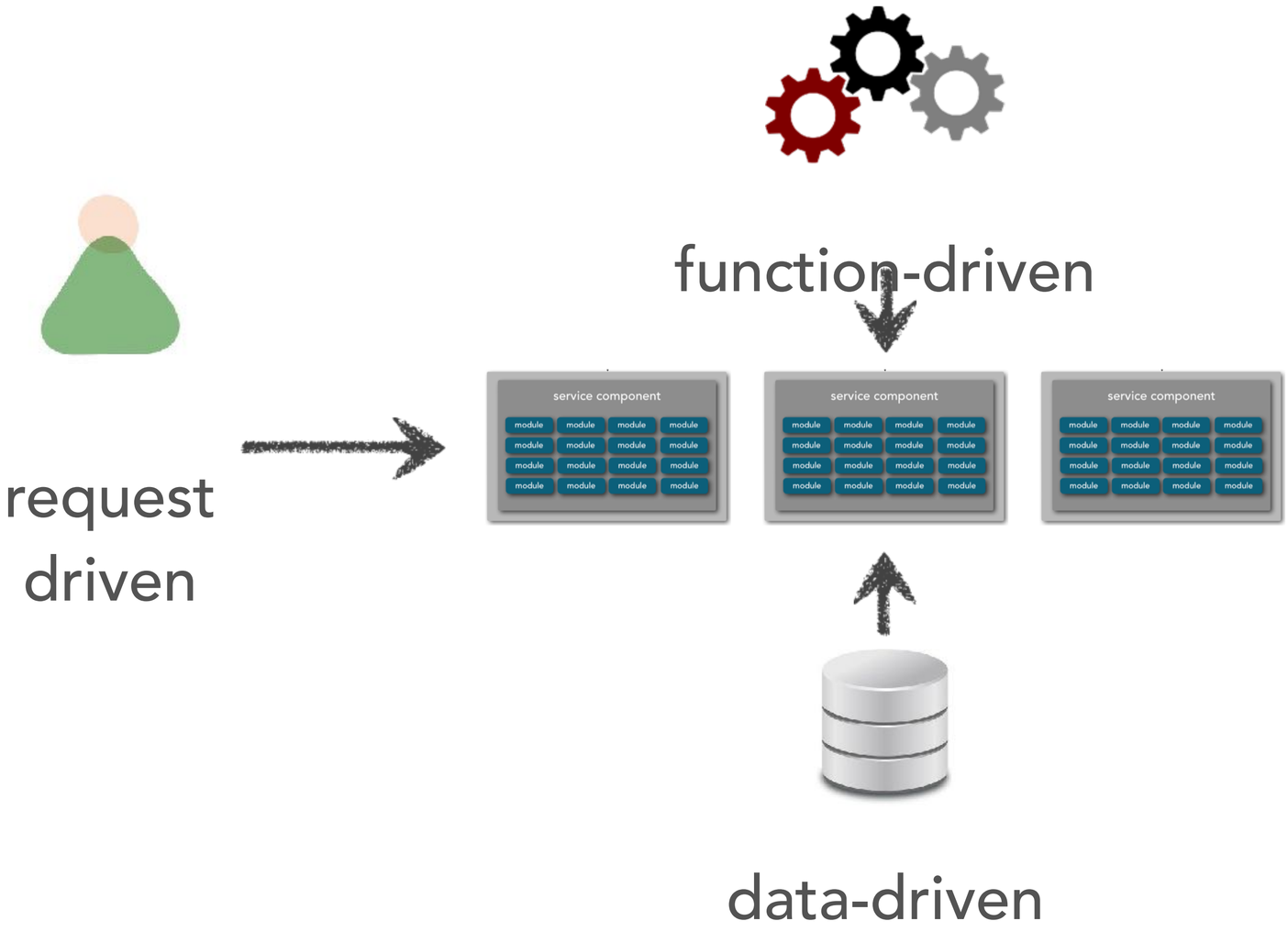
function-driven



identifying microservices



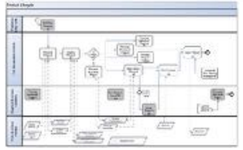
identifying microservices



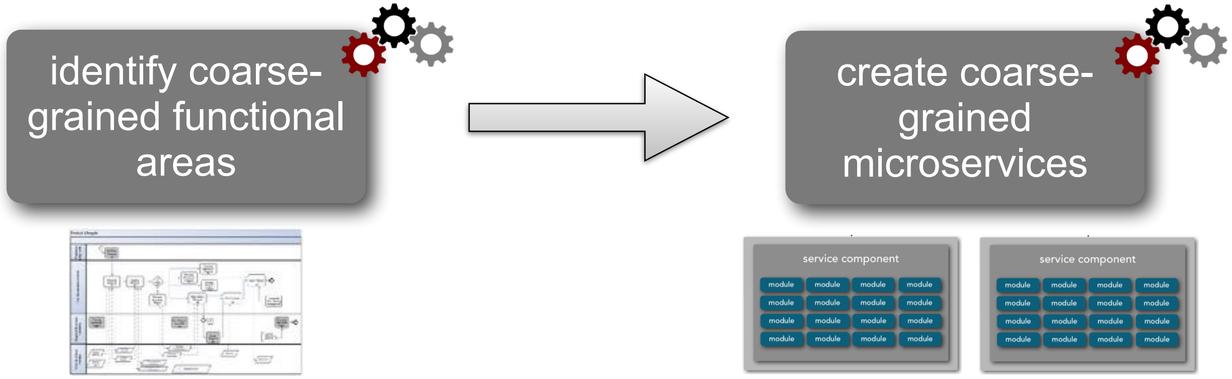
identifying microservices

identifying microservices

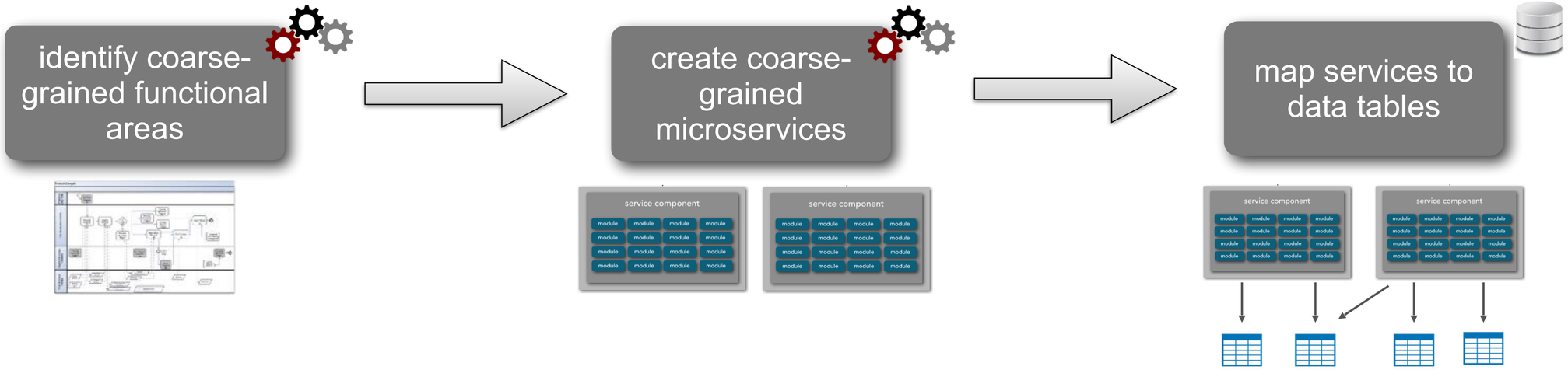
identify coarse-grained functional areas



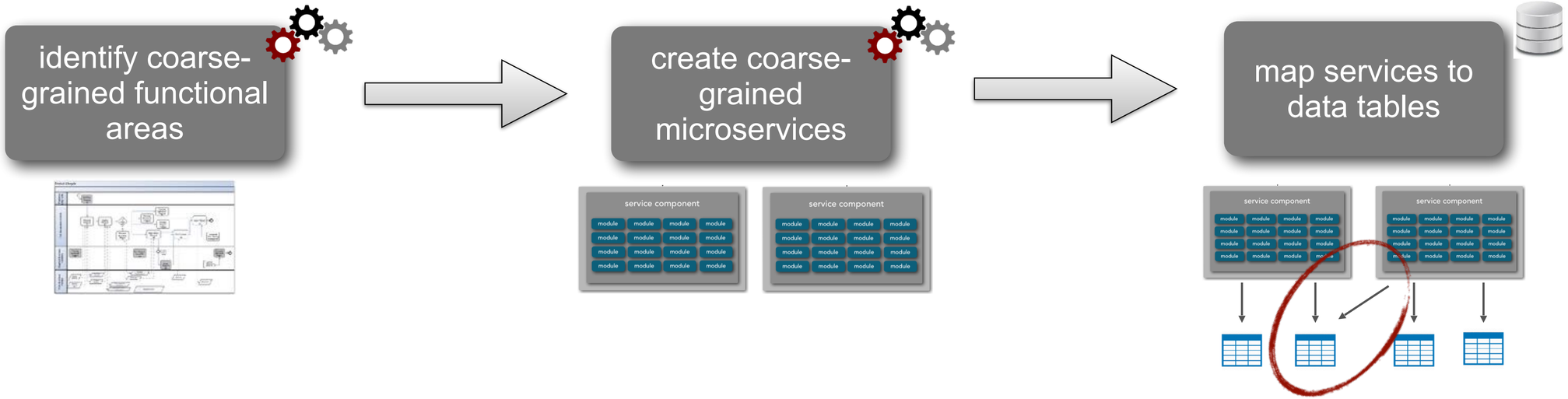
identifying microservices



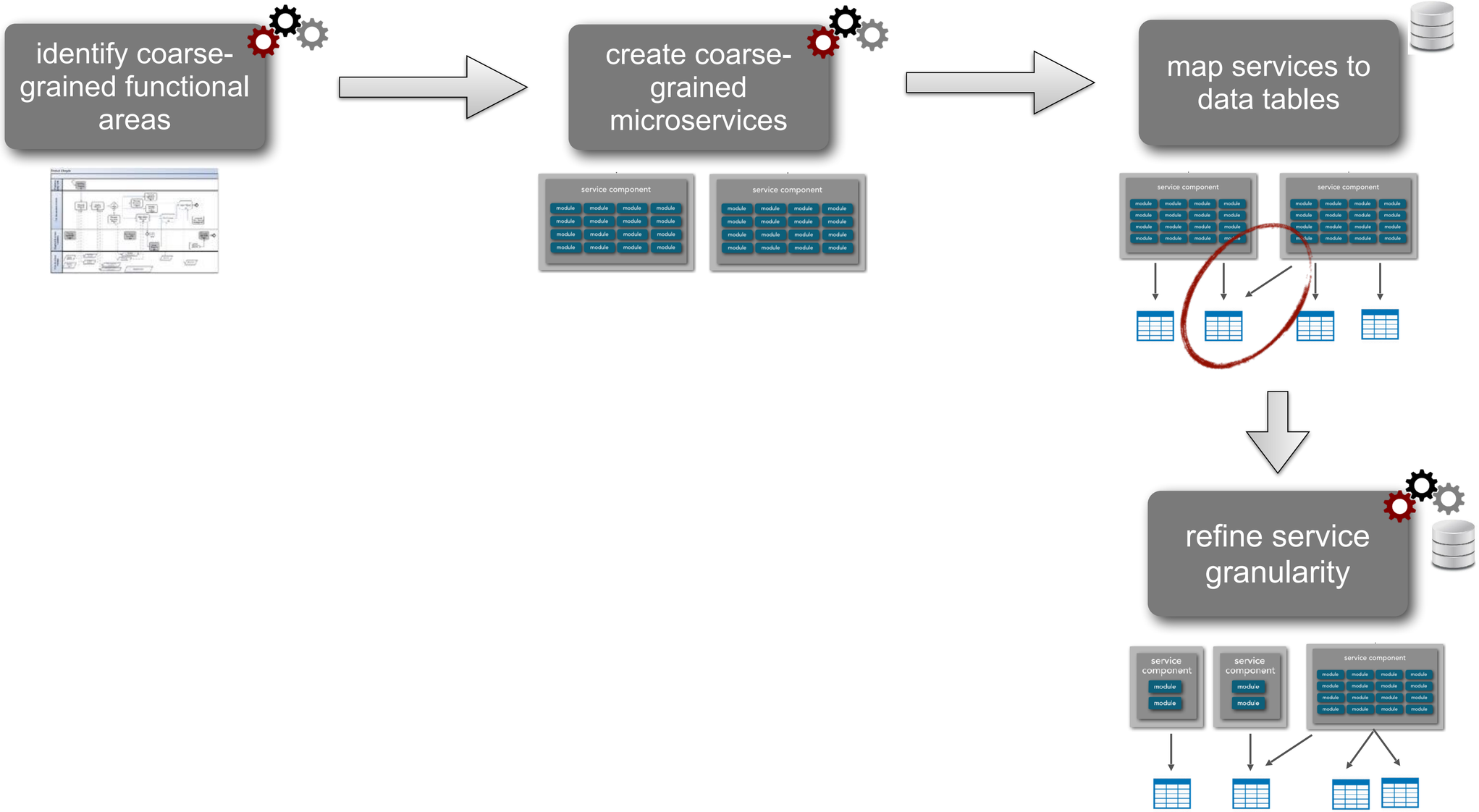
identifying microservices



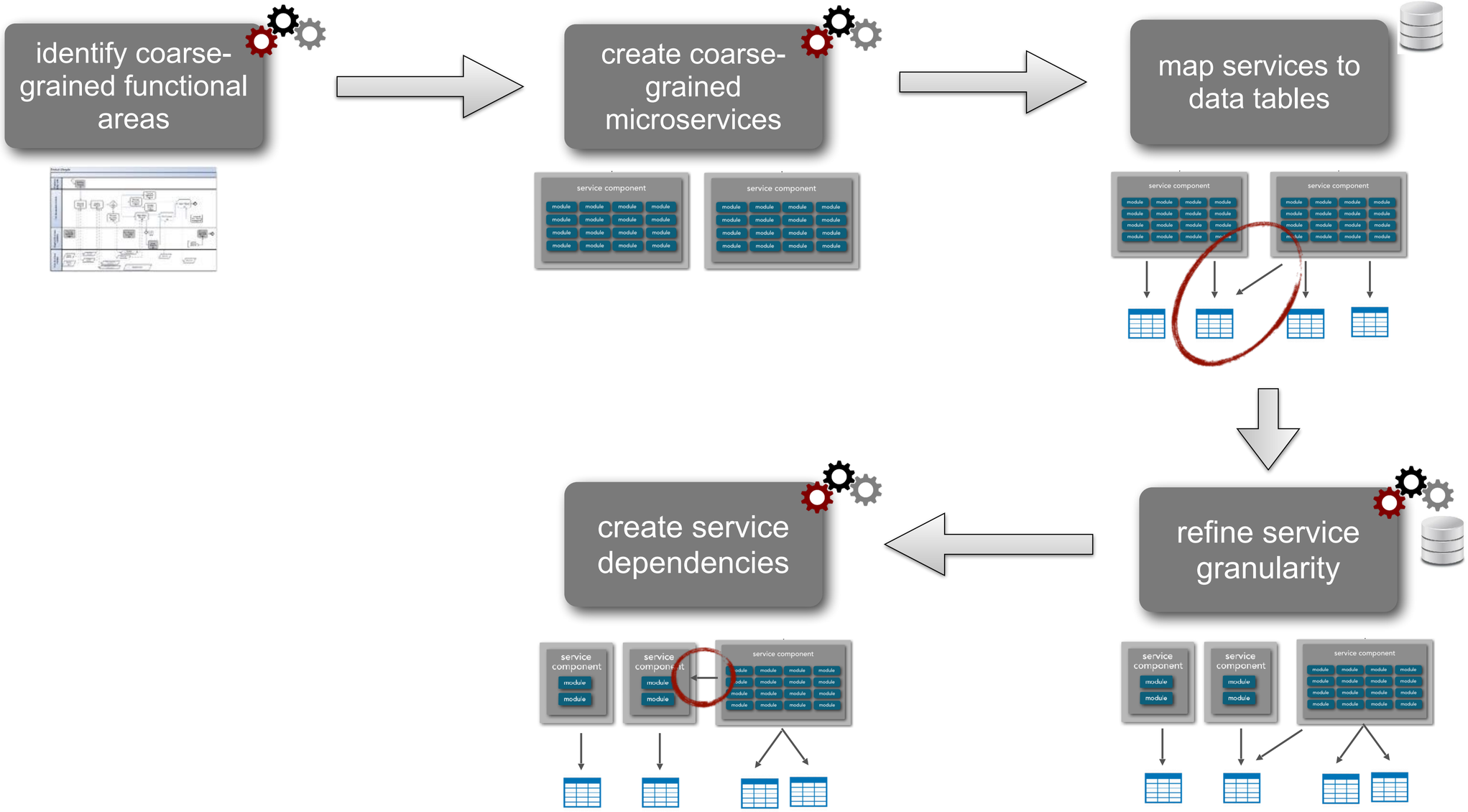
identifying microservices



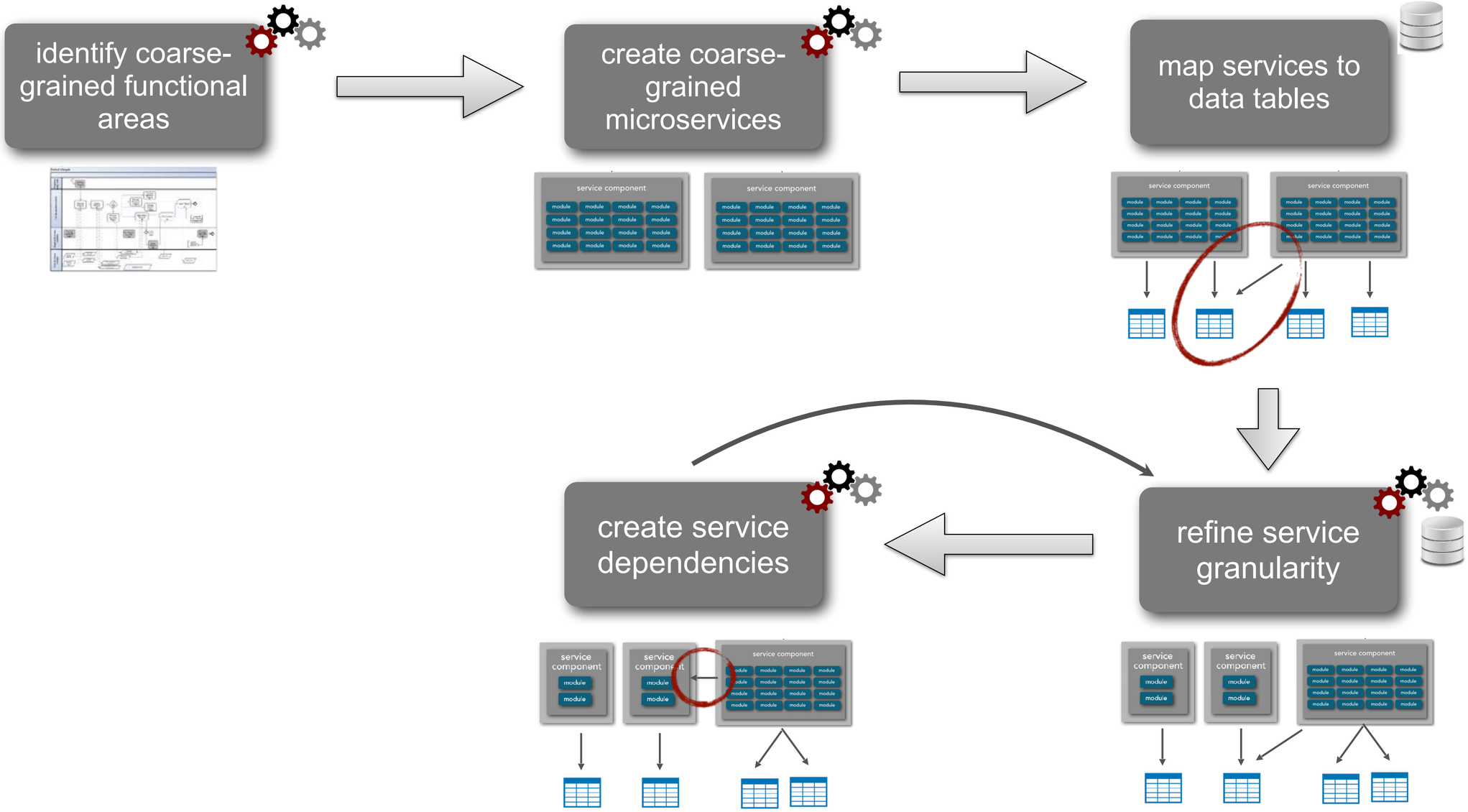
identifying microservices



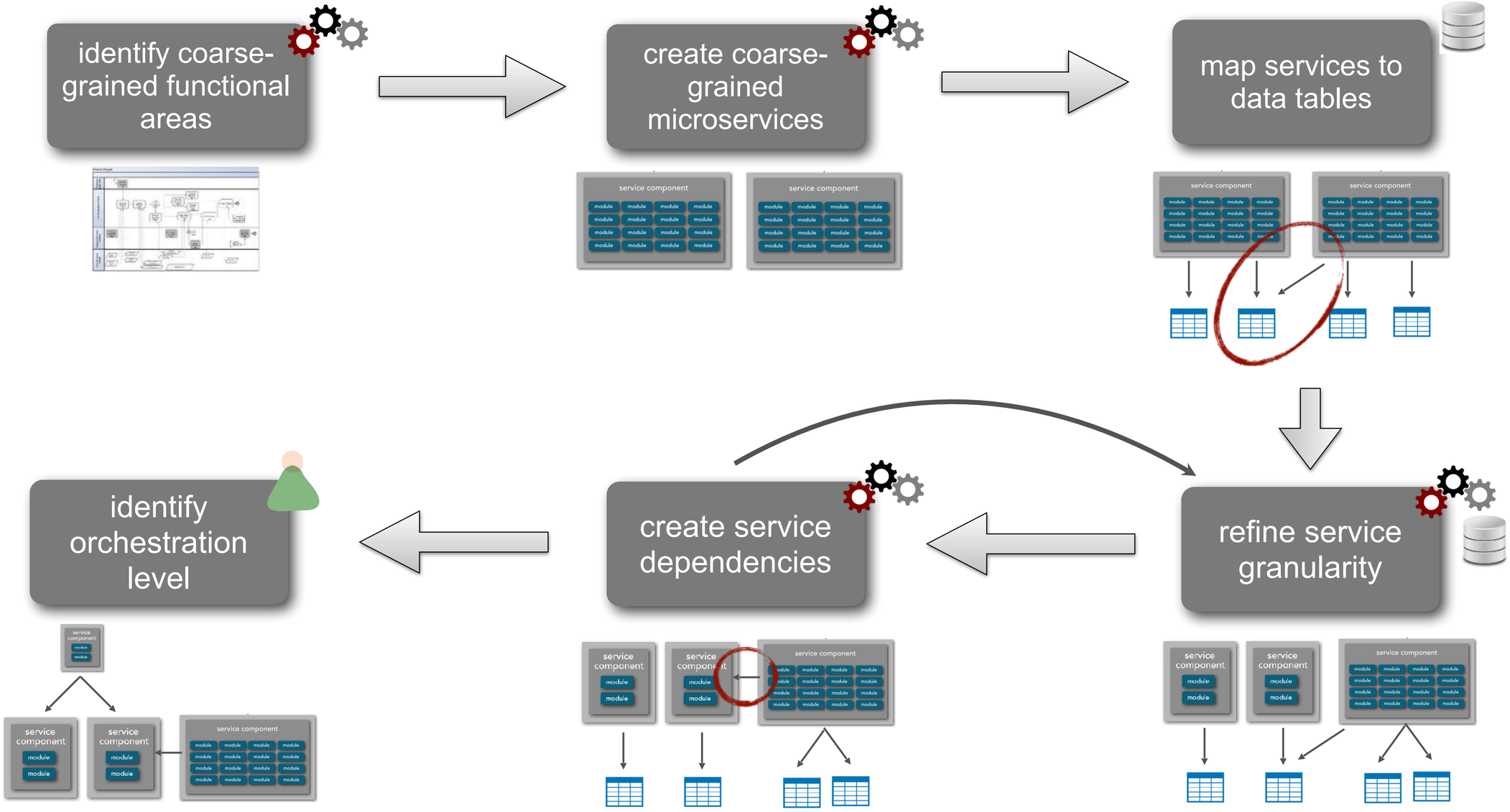
identifying microservices



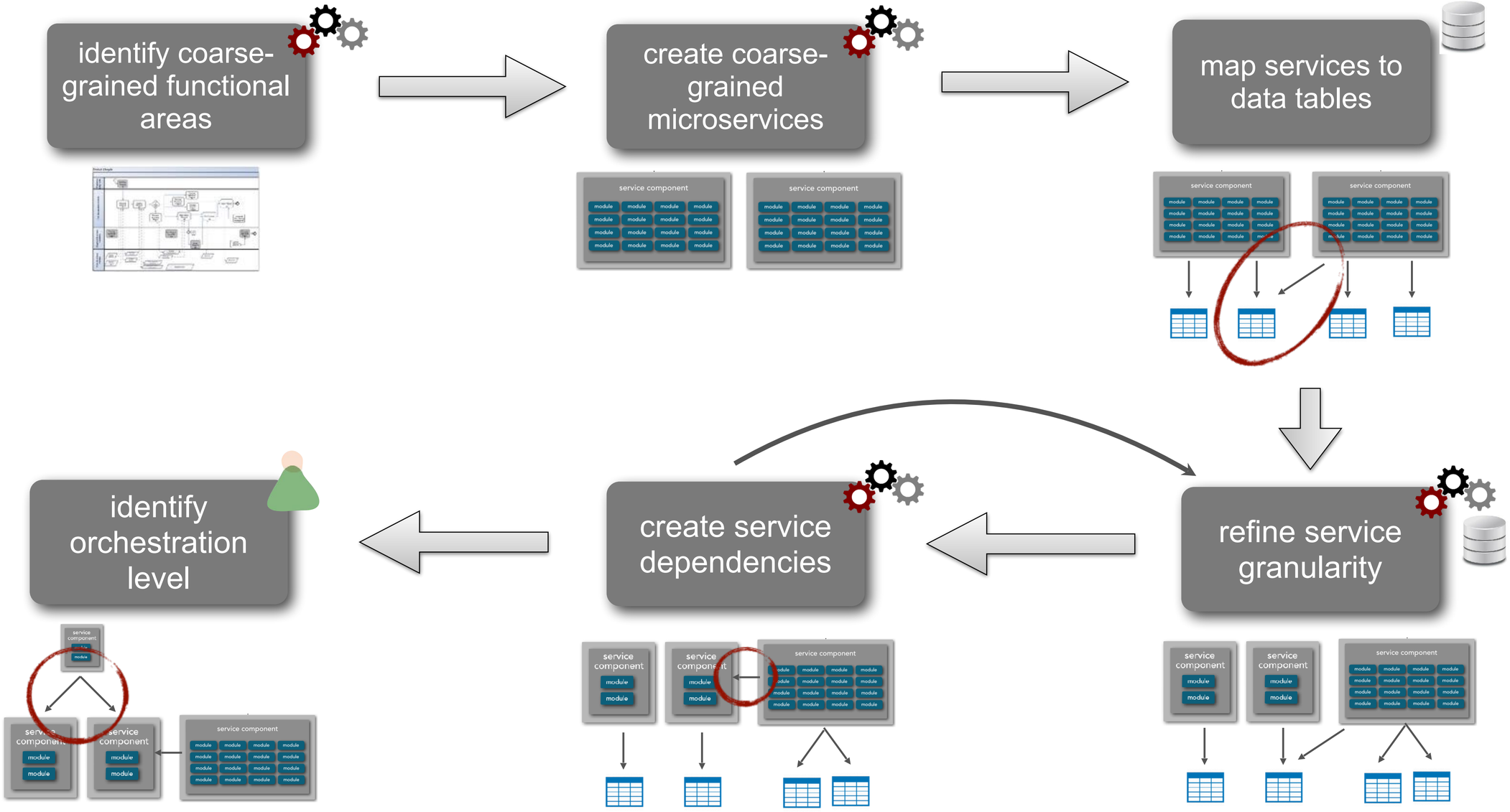
identifying microservices



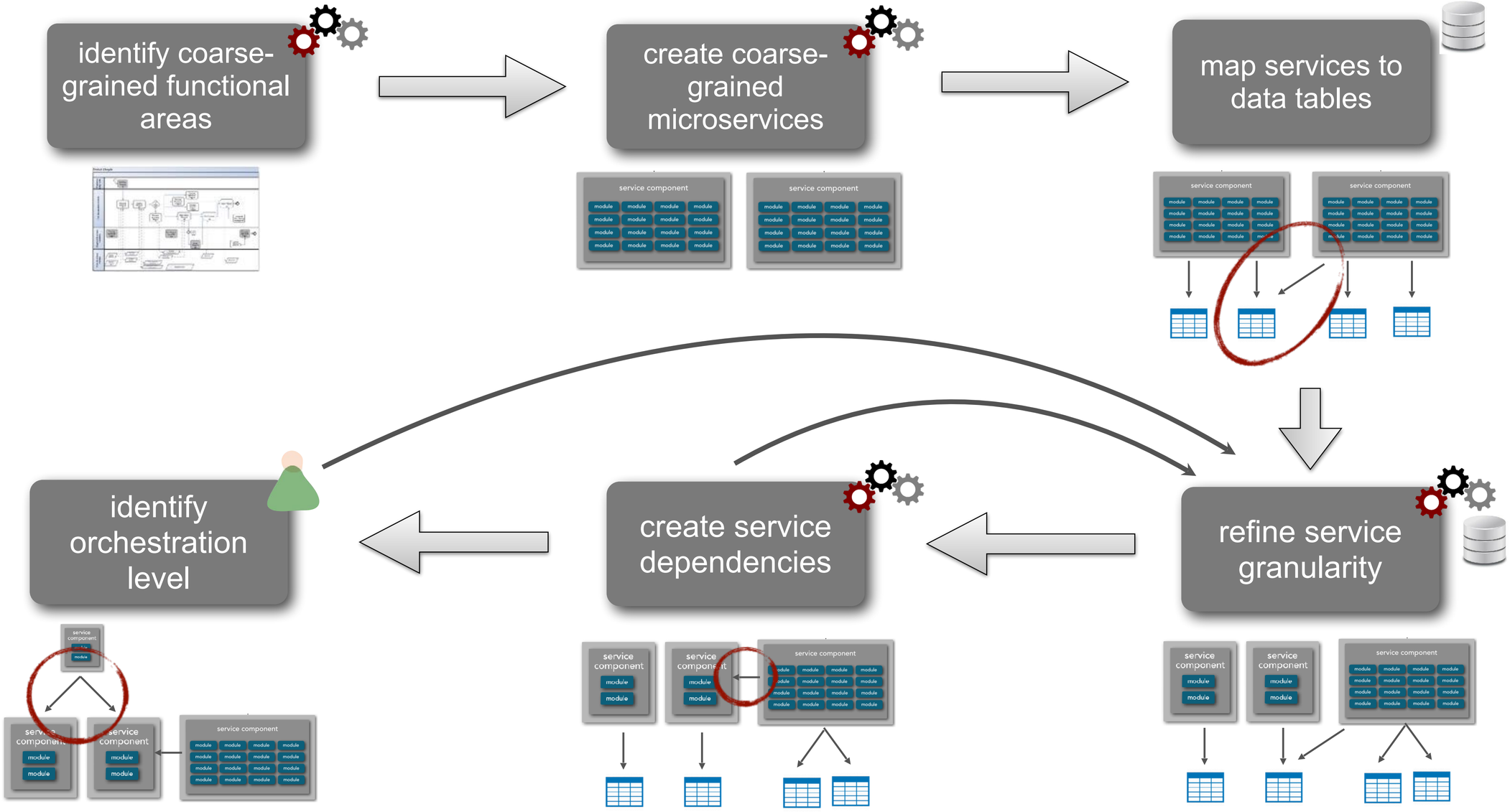
identifying microservices



identifying microservices



identifying microservices



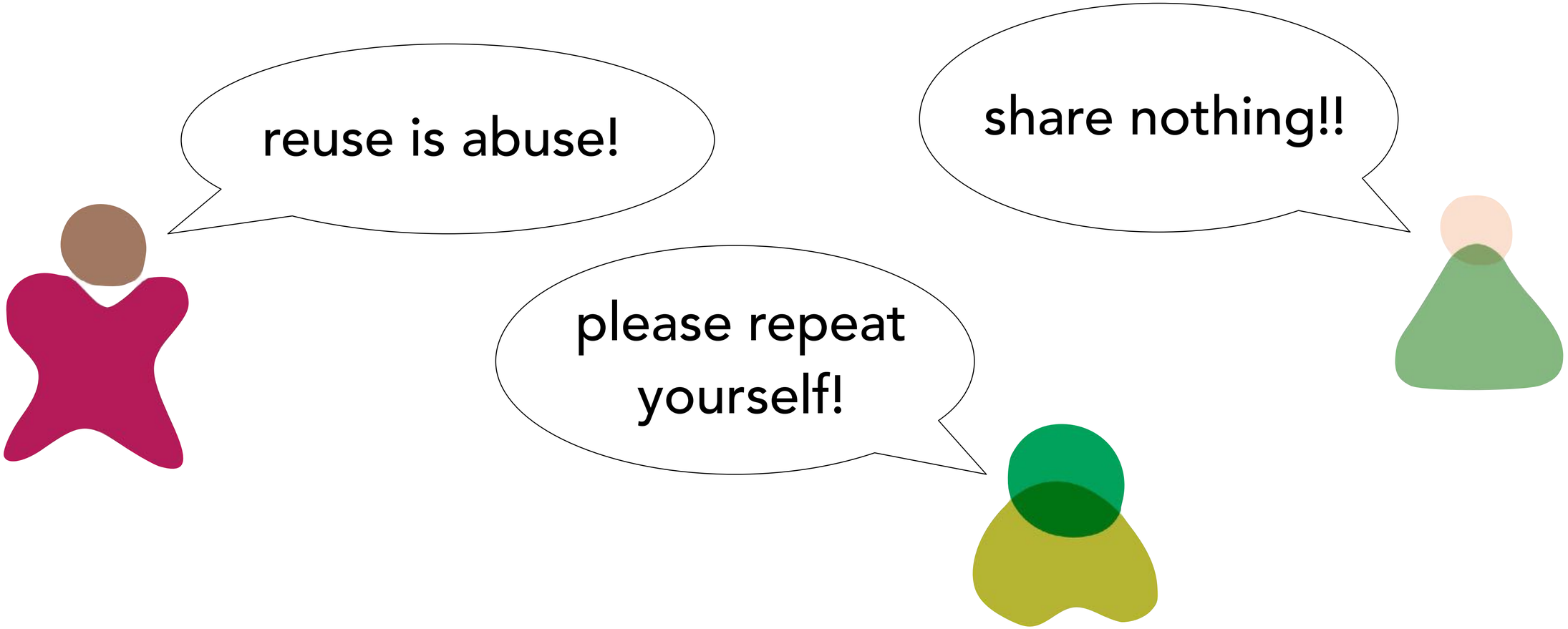
Code Sharing Techniques

code sharing techniques

code sharing techniques

code sharing techniques

code sharing techniques



code sharing techniques

code sharing techniques

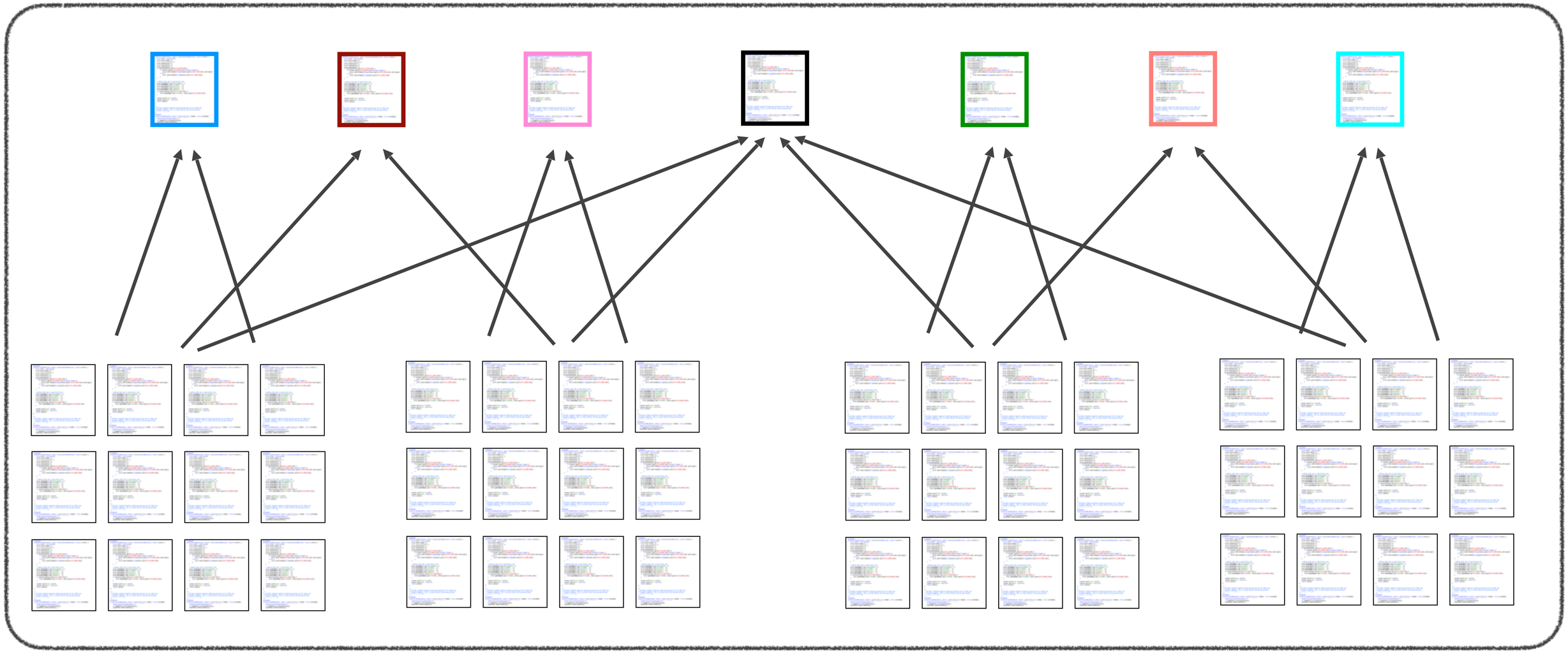
code sharing techniques

code sharing techniques

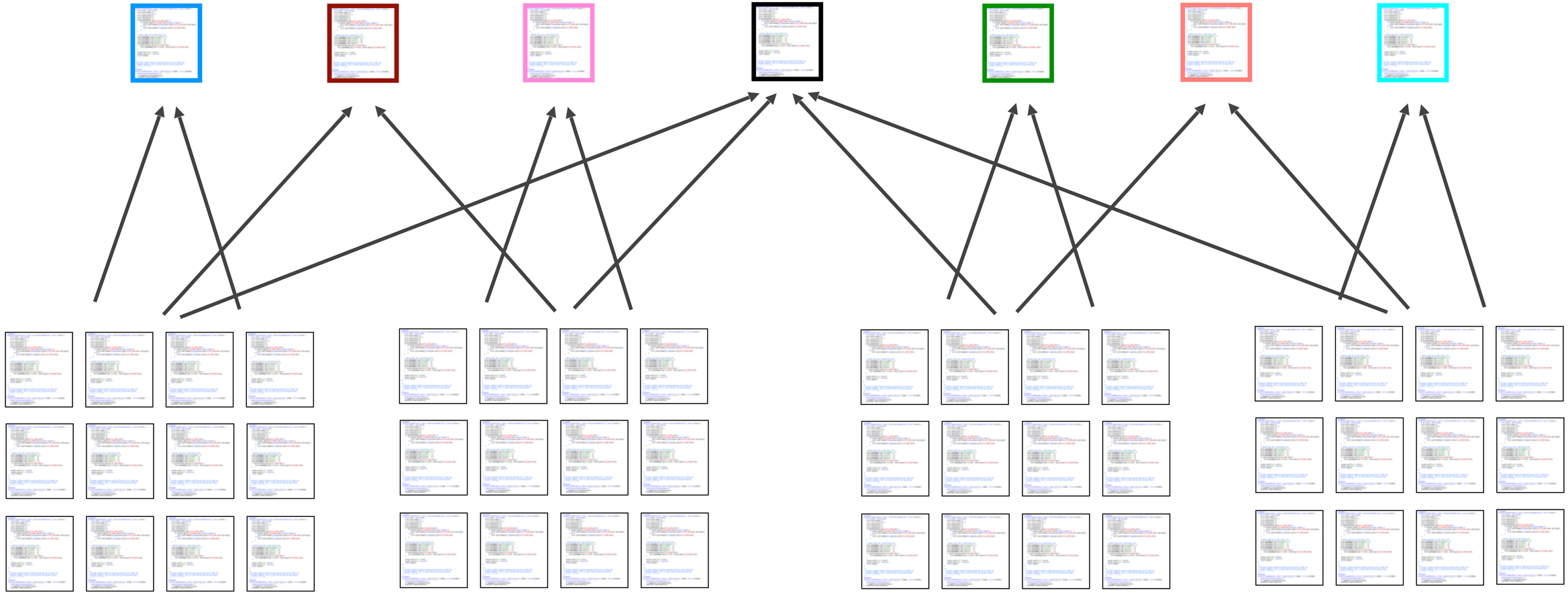
code sharing techniques

code sharing techniques

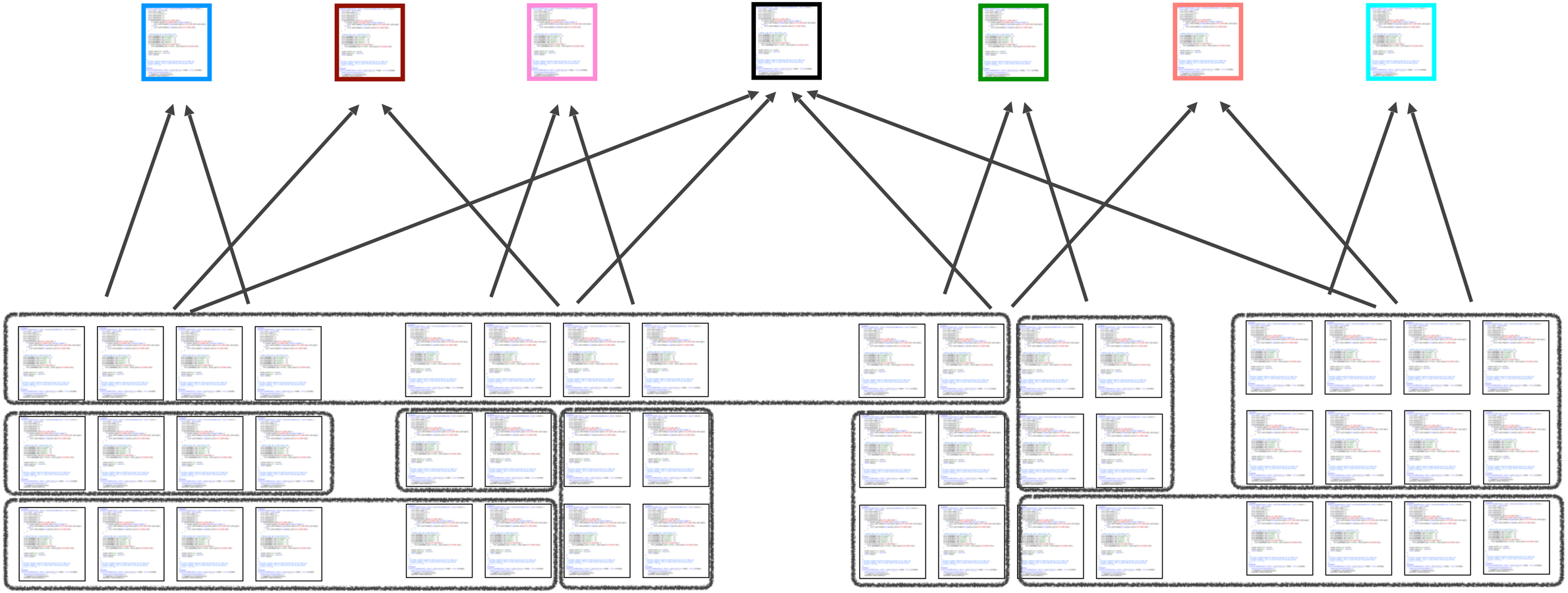
code sharing techniques



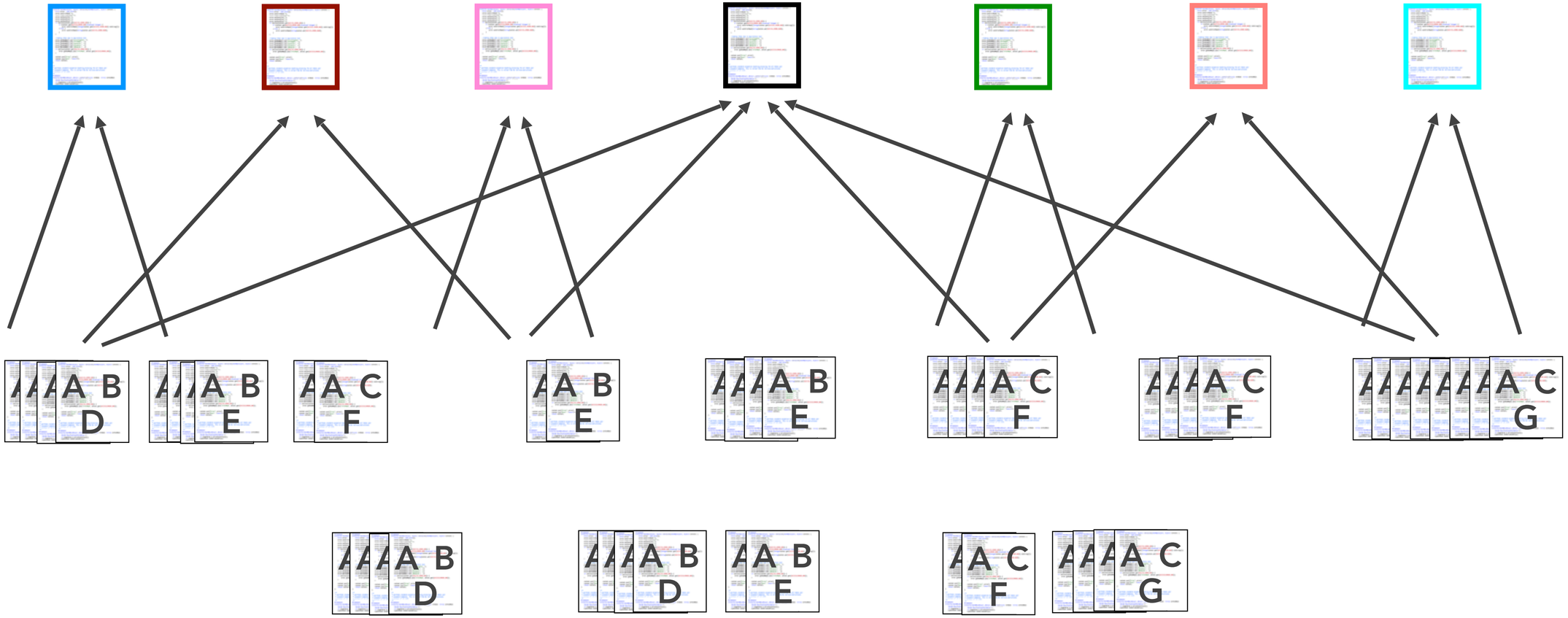
code sharing techniques



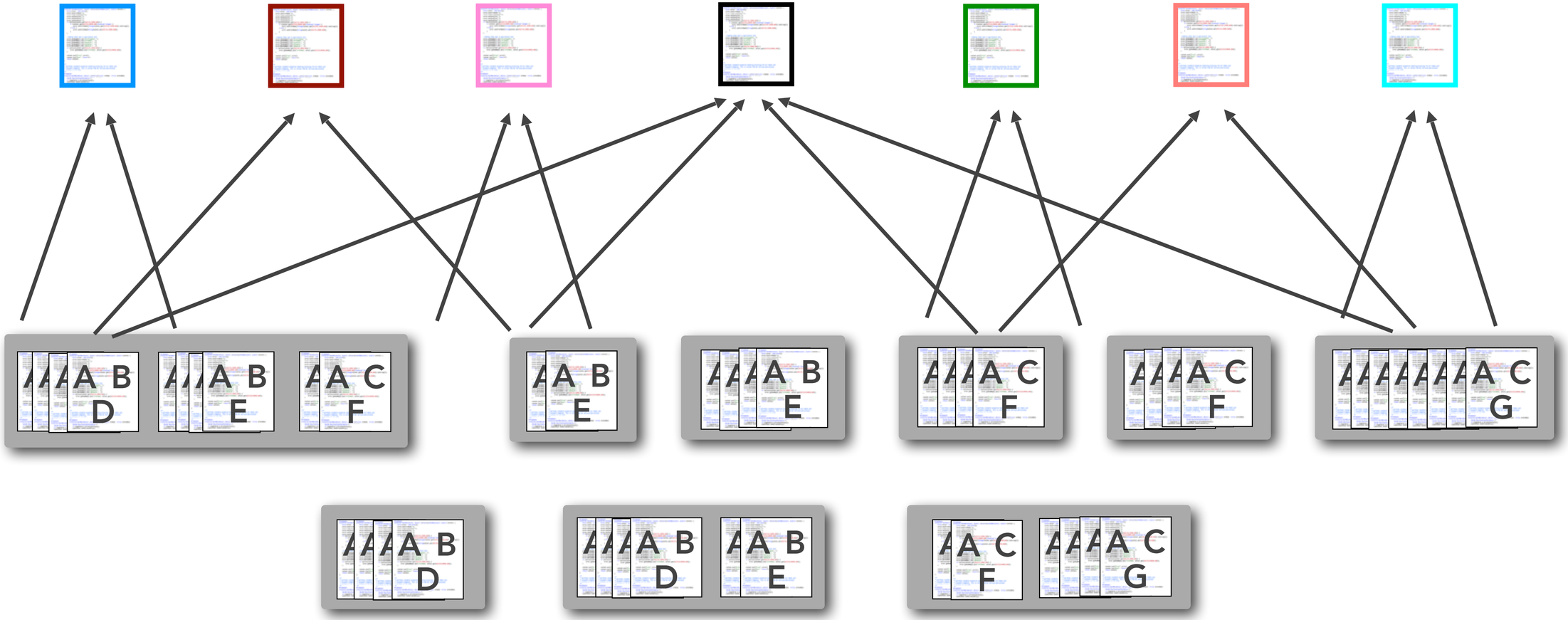
code sharing techniques



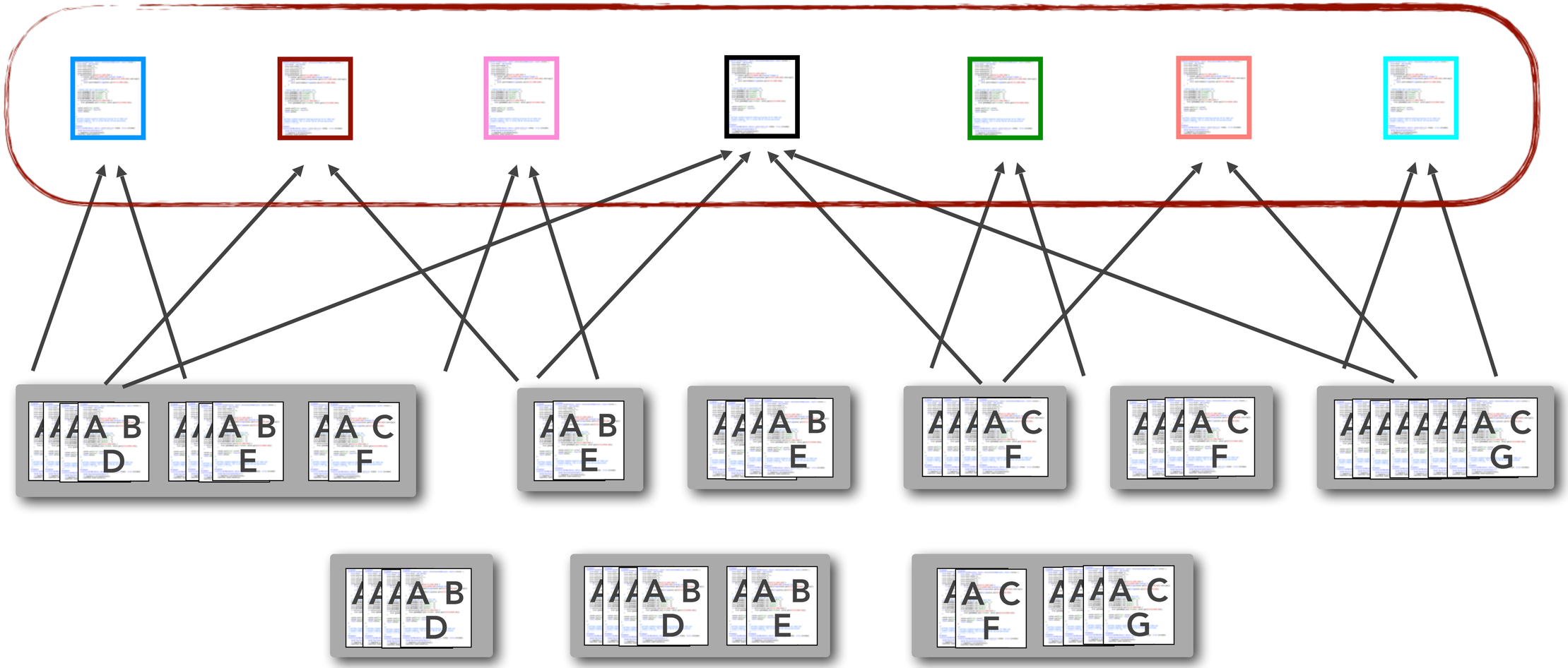
code sharing techniques



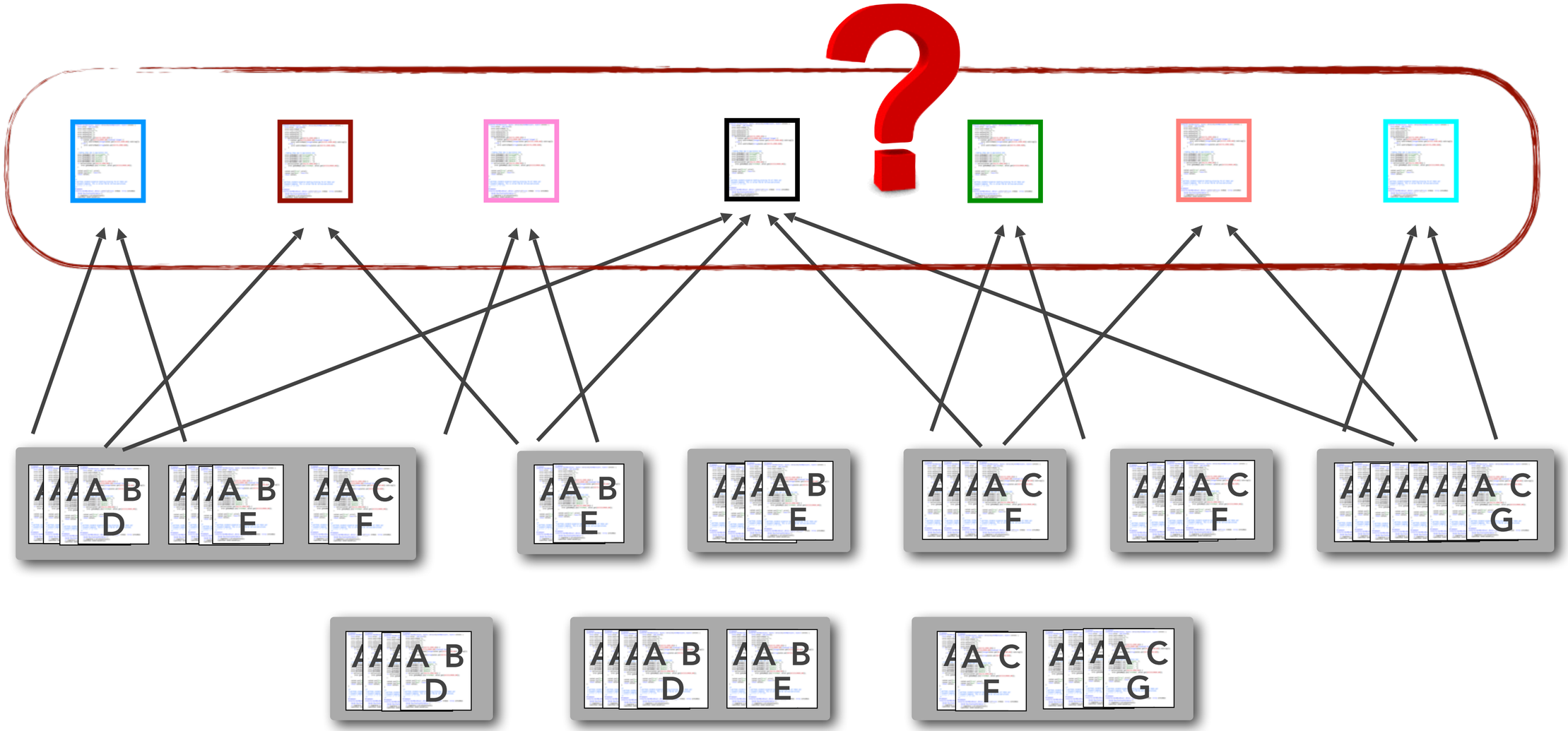
code sharing techniques



code sharing techniques



code sharing techniques



code sharing techniques

code replication

code sharing techniques

code replication

ServiceEntrypoint.java

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface ServiceEntrypoint {}
```

code sharing techniques

code replication

ServiceEntrypoint.java

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface ServiceEntrypoint {}
```

```
@ServiceEntrypoint
public class PaymentServiceAPI {
    ...
}
```

code sharing techniques

code replication

code sharing techniques

code replication

ServiceEntrypoint.cs

```
[AttributeUsage(AttributeTargets.Class)]
class ServiceEntrypoint : Attribute {}
```

code sharing techniques

code replication

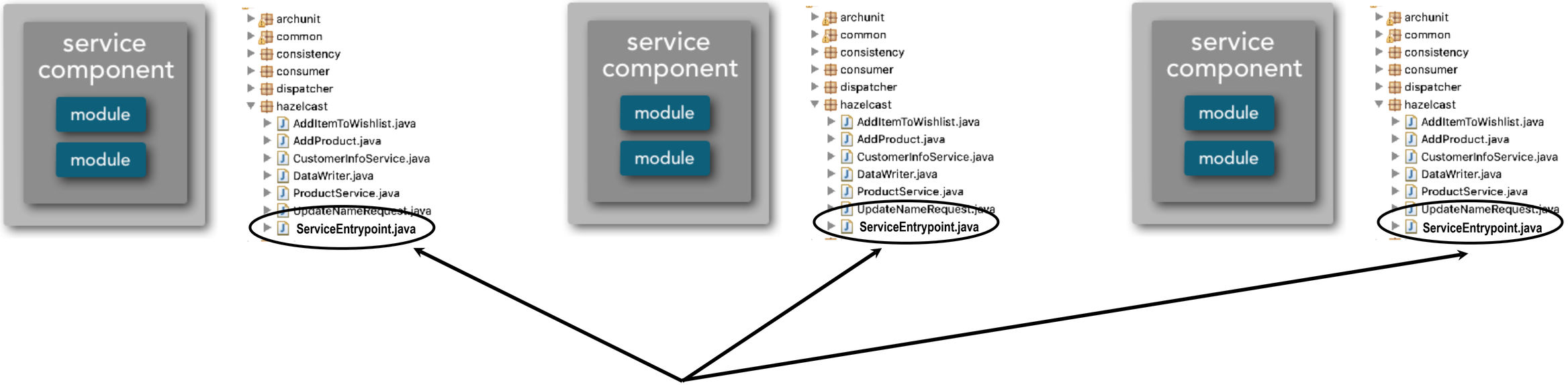
ServiceEntrypoint.cs

```
[AttributeUsage(AttributeTargets.Class)]
class ServiceEntrypoint : Attribute {}
```

```
[ServiceEntrypoint]
class PaymentServiceAPI {
    ...
}
```

code sharing techniques

code replication

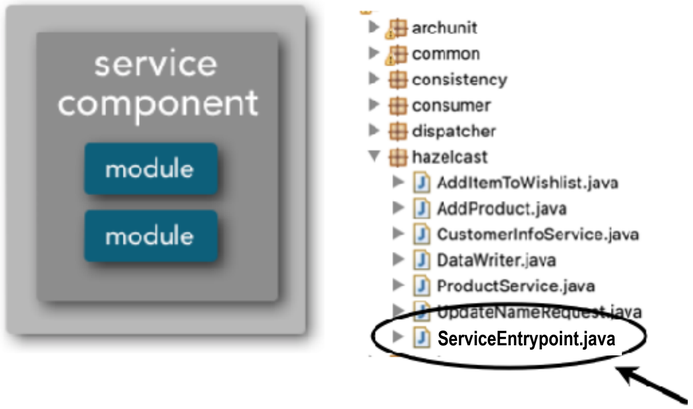


ServiceEntrypoint.java

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface ServiceEntrypoint {}
```

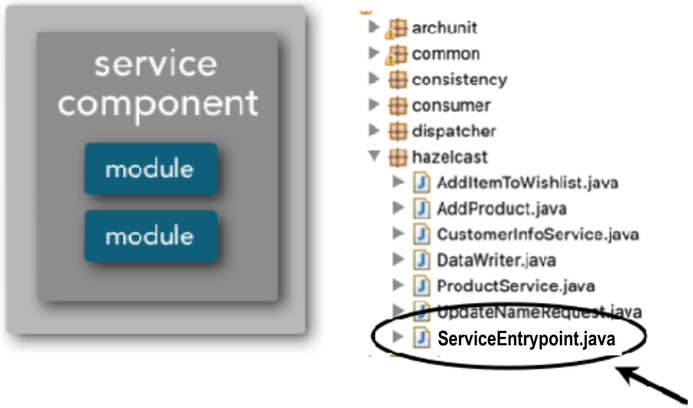
code sharing techniques

code replication



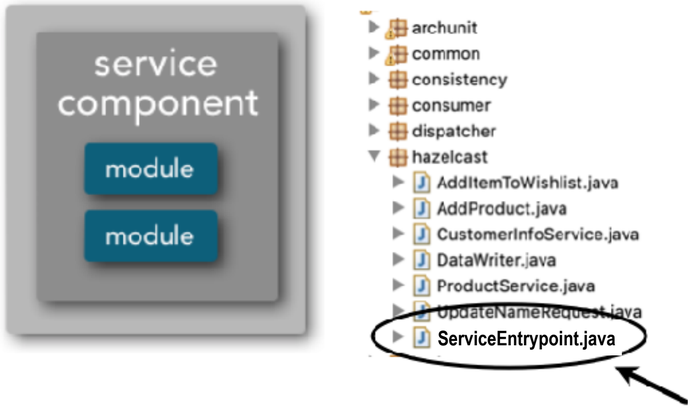
code sharing techniques

code replication



code sharing techniques

code replication

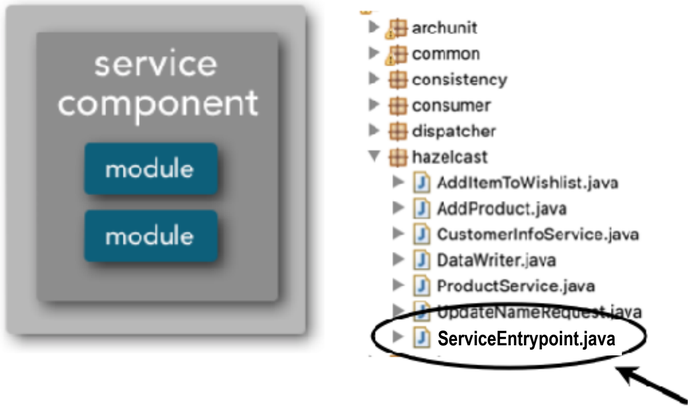


✓ no code sharing

✓ good for static code

code sharing techniques

code replication



✓ no code sharing

✓ good for static code

✗ code changes

✗ difficult to expand context

code sharing techniques

shared library

ServiceEntrypoint.java

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface ServiceEntrypoint {}
```

code sharing techniques

shared library

ServiceEntrypoint.java

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface ServiceEntrypoint {}
```

annotations.jar v1.0

ServiceEntrypoint.java

code sharing techniques

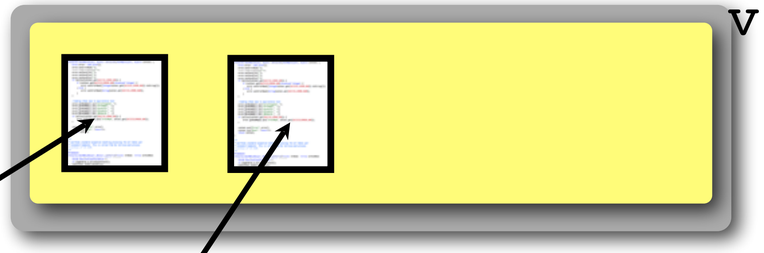
shared library

ServiceEntrypoint.java

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface ServiceEntrypoint {}
```

annotations.jar v1.0

v1.1



ServiceEntrypoint.java

OrchestrationService.java

code sharing techniques

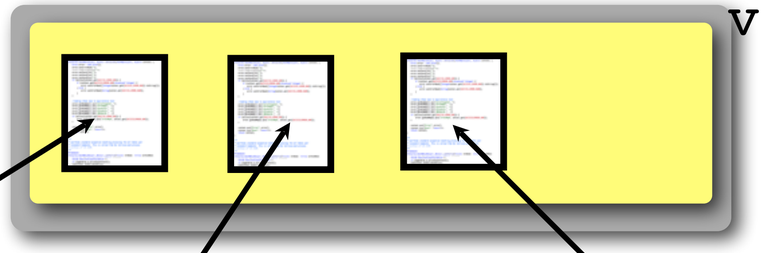
shared library

ServiceEntrypoint.java

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface ServiceEntrypoint {}
```

annotations.jar v1.0

v1.1 v1.2



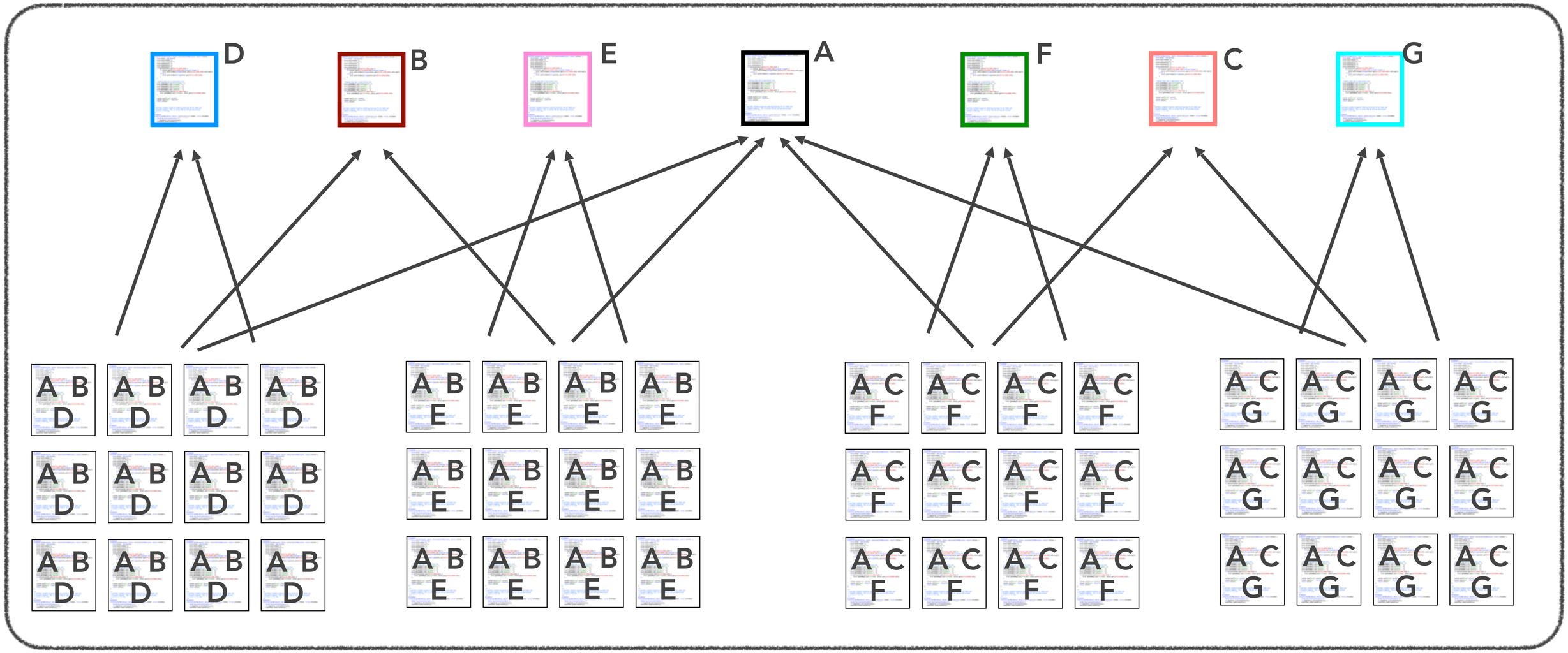
ServiceEntrypoint.java

OrchestrationService.java

DataService.java

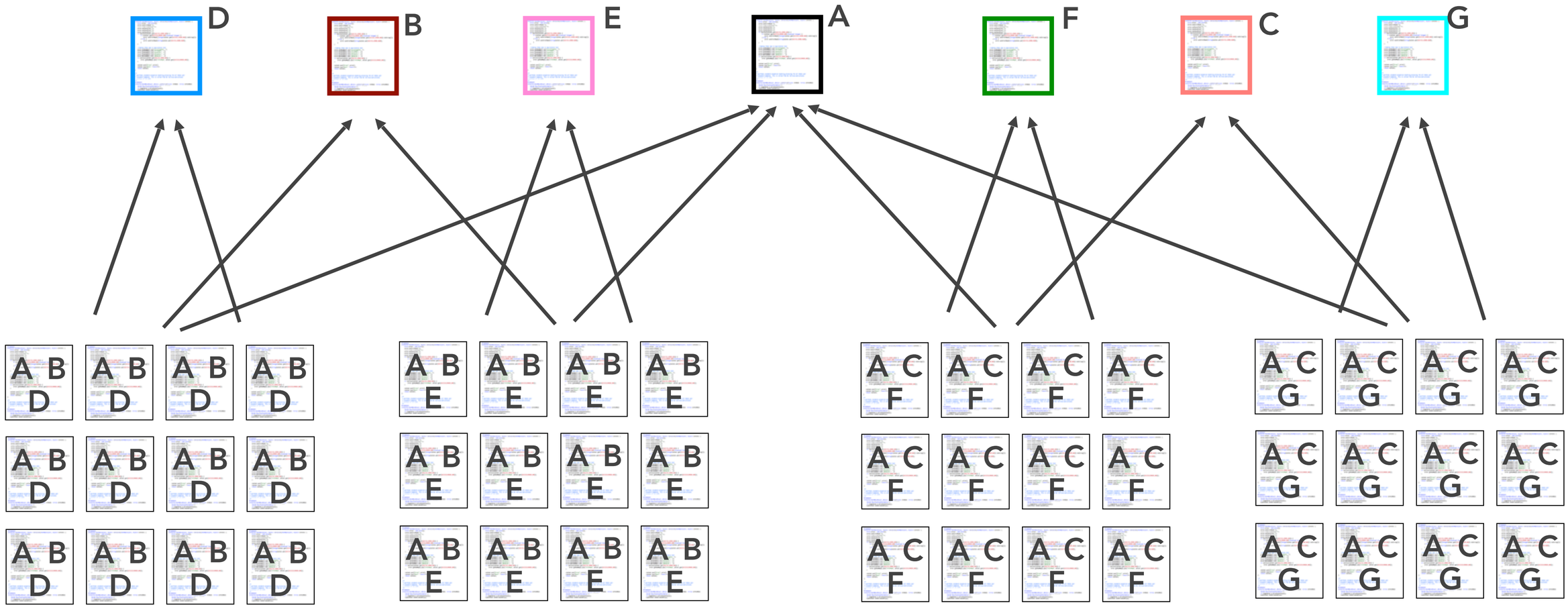
code sharing techniques

shared library



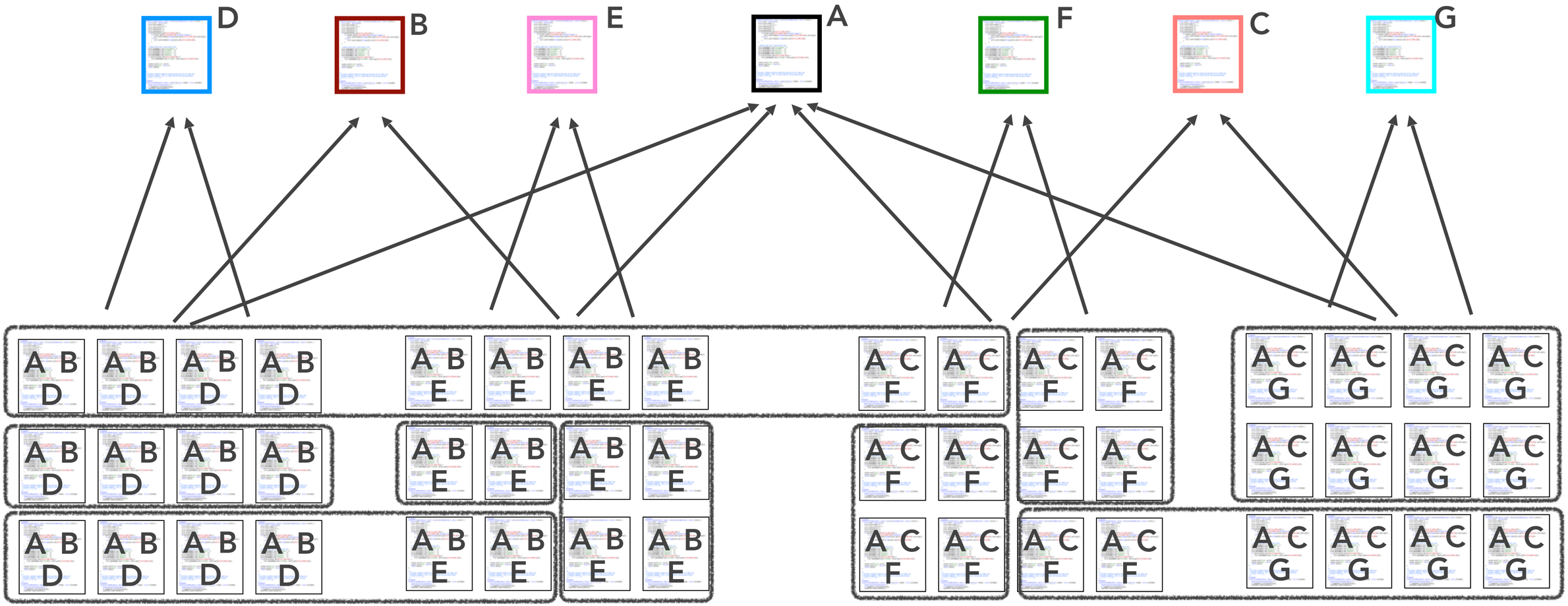
code sharing techniques

shared library



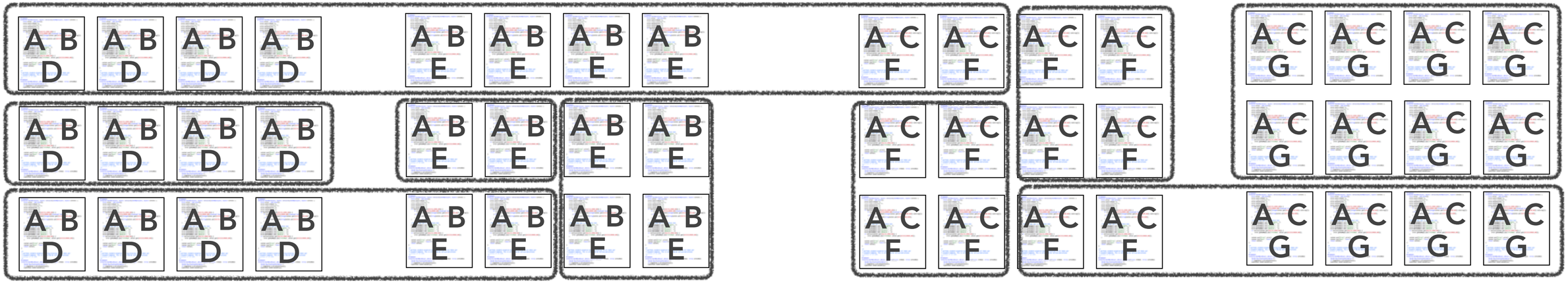
code sharing techniques

shared library



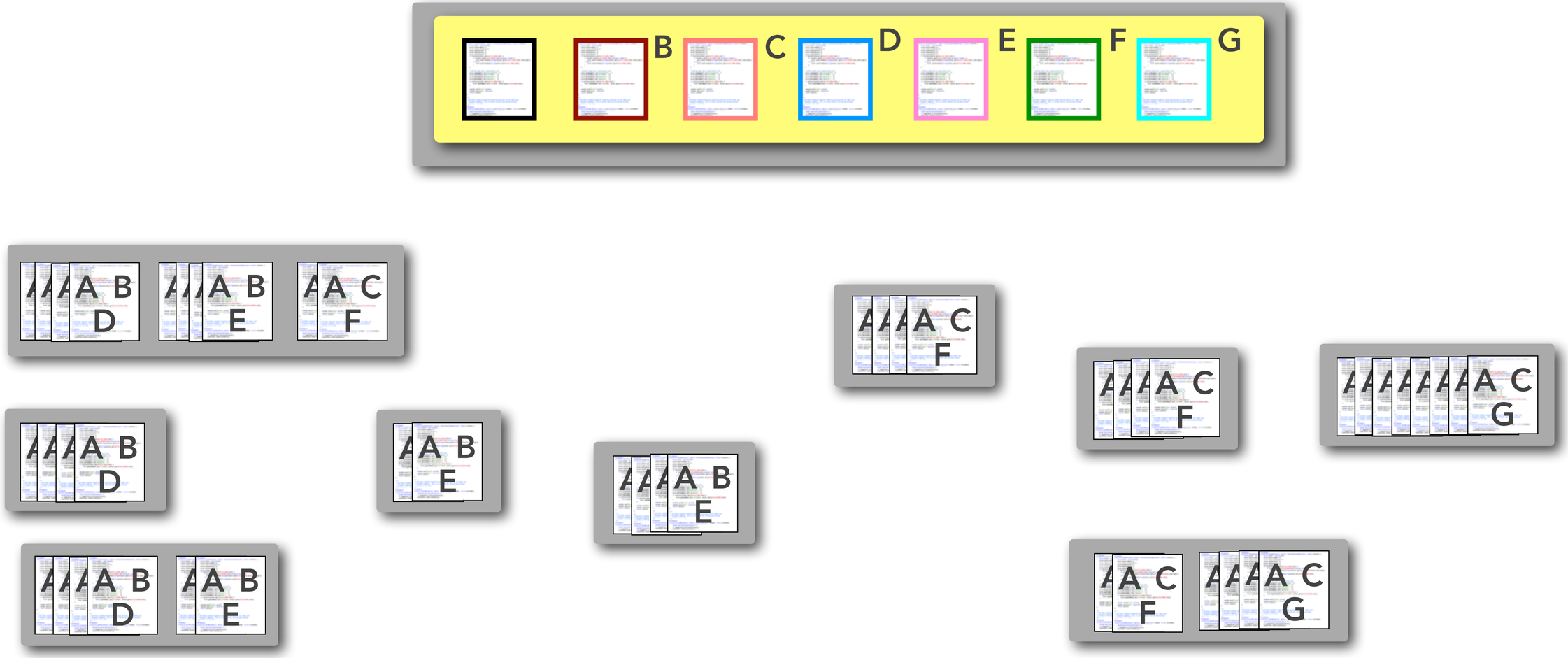
code sharing techniques

shared library



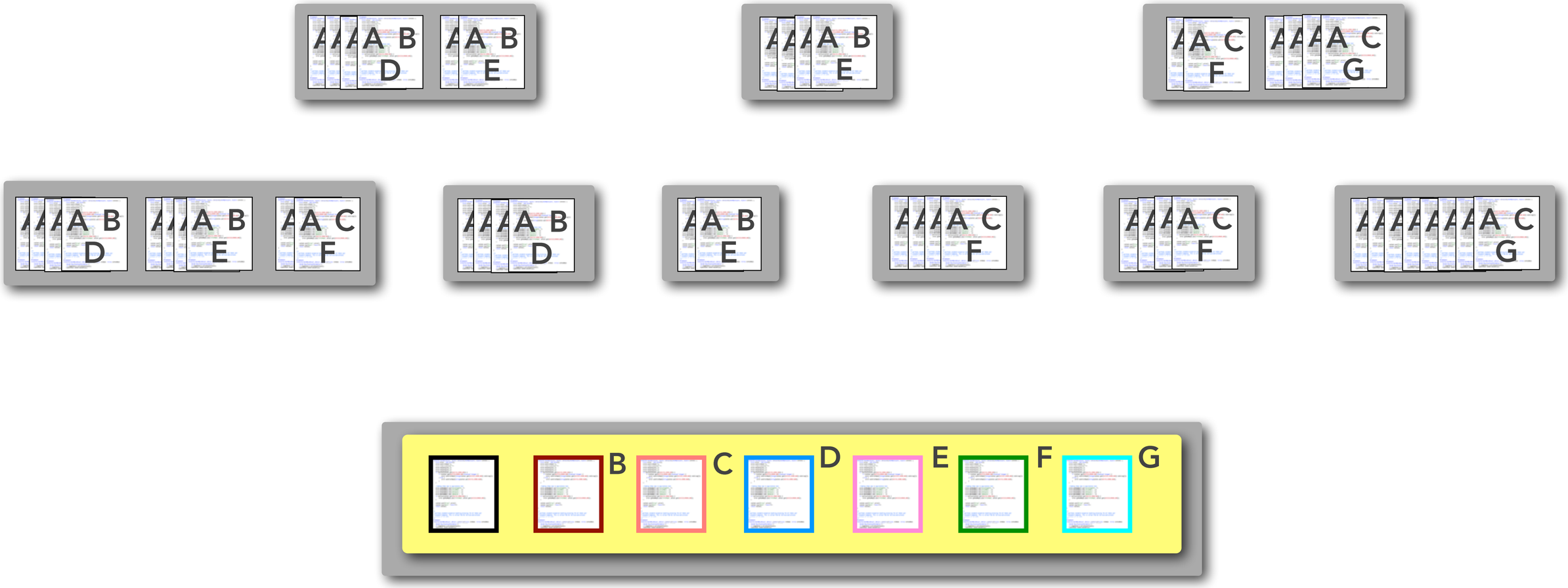
code sharing techniques

shared library



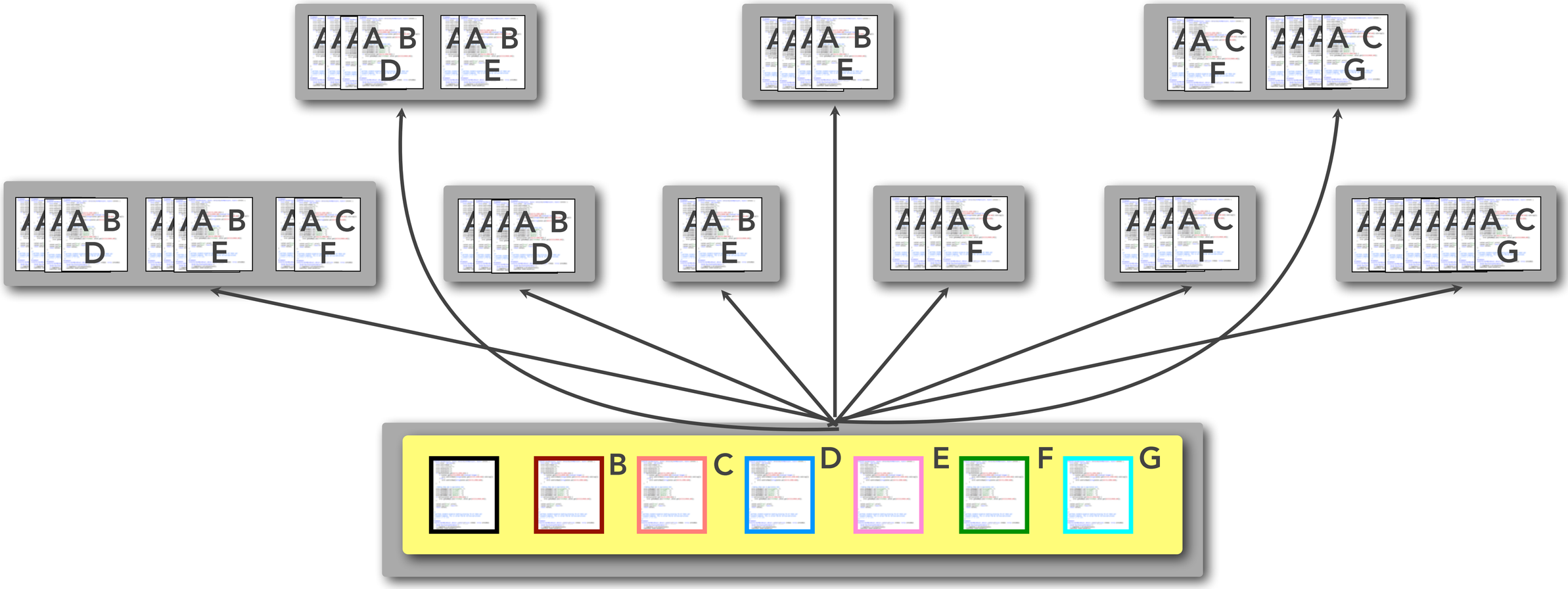
code sharing techniques

shared library



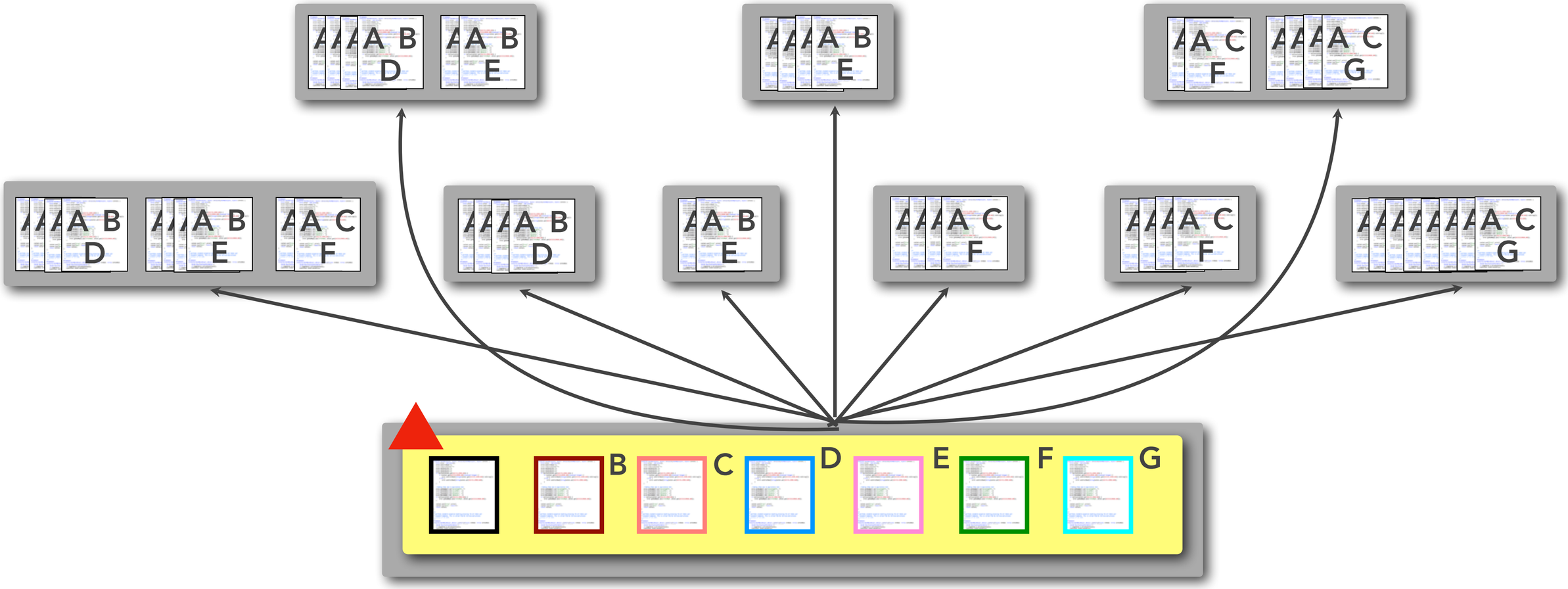
code sharing techniques

shared library



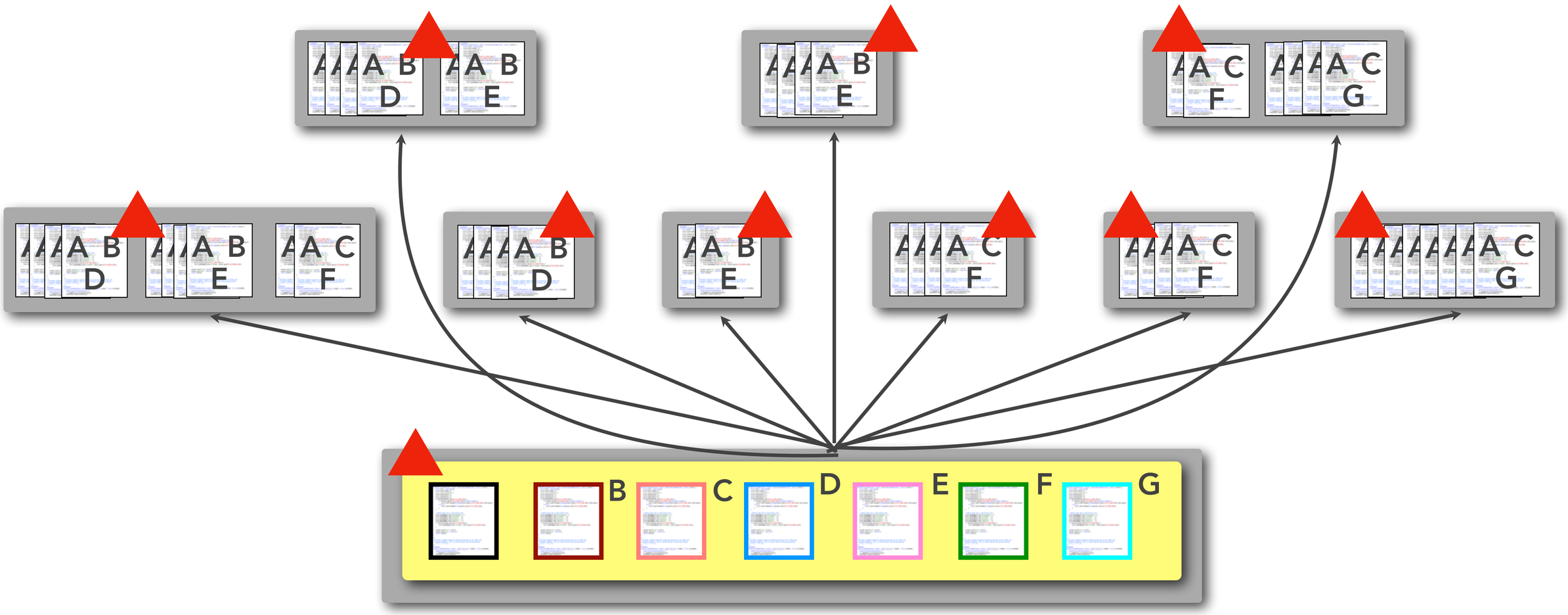
code sharing techniques

shared library



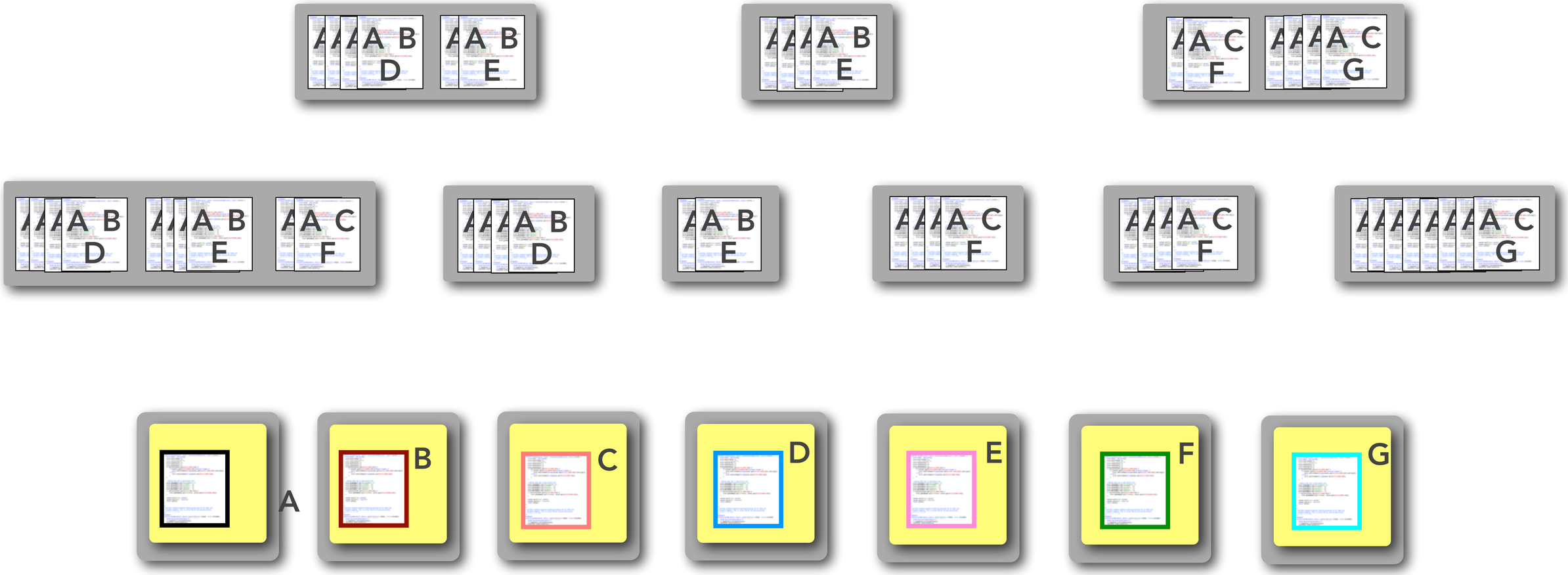
code sharing techniques

shared library



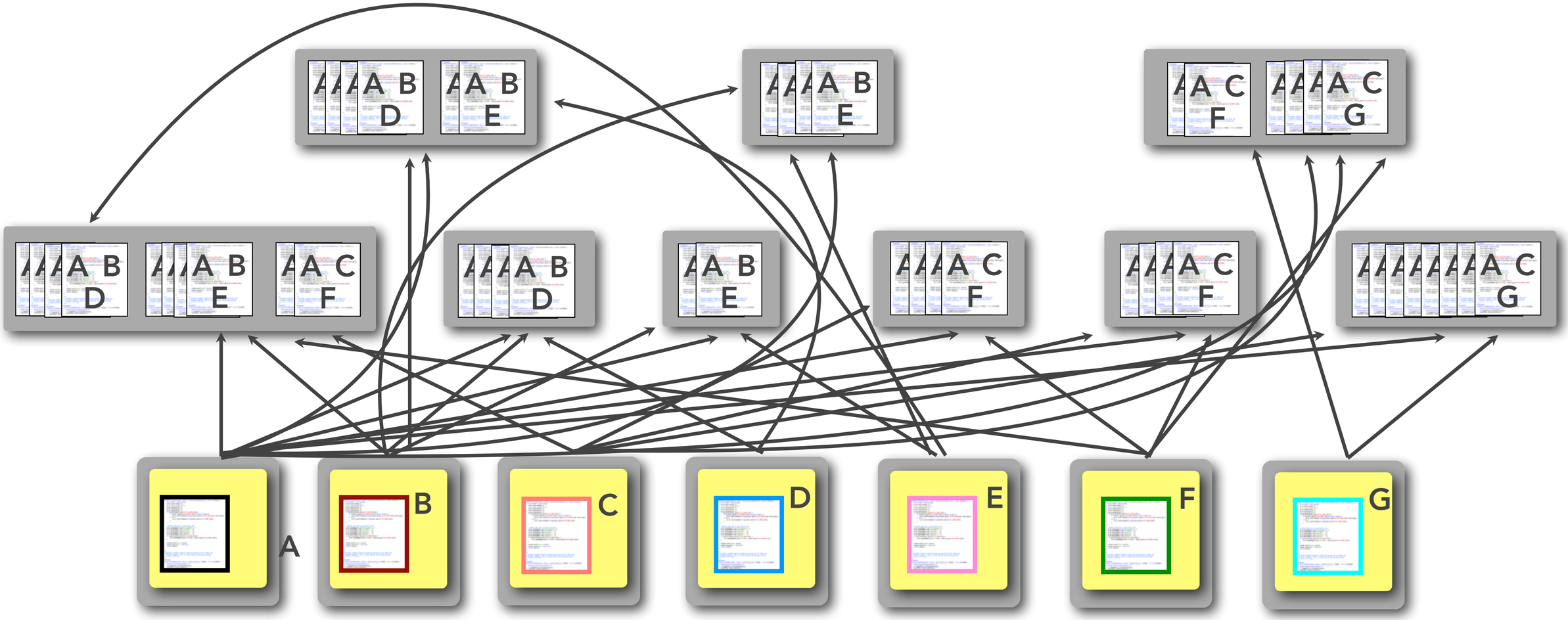
code sharing techniques

shared library



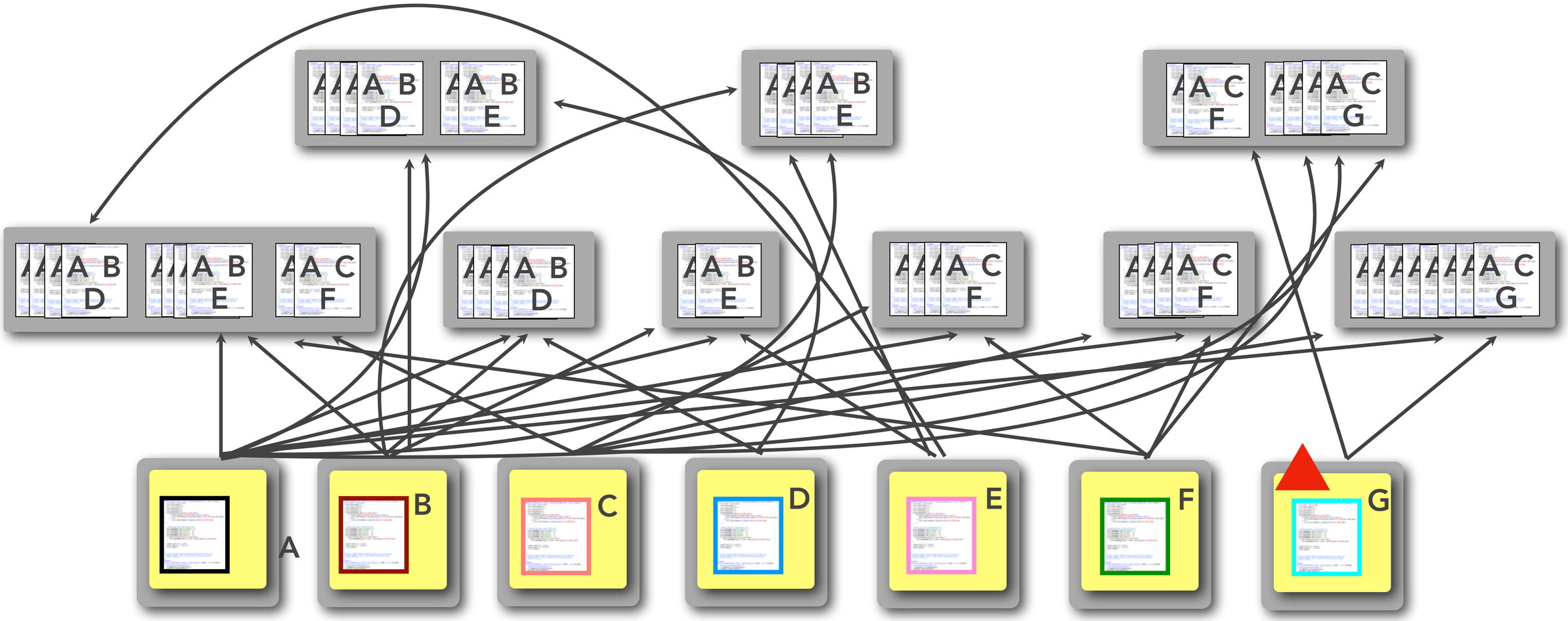
code sharing techniques

shared library



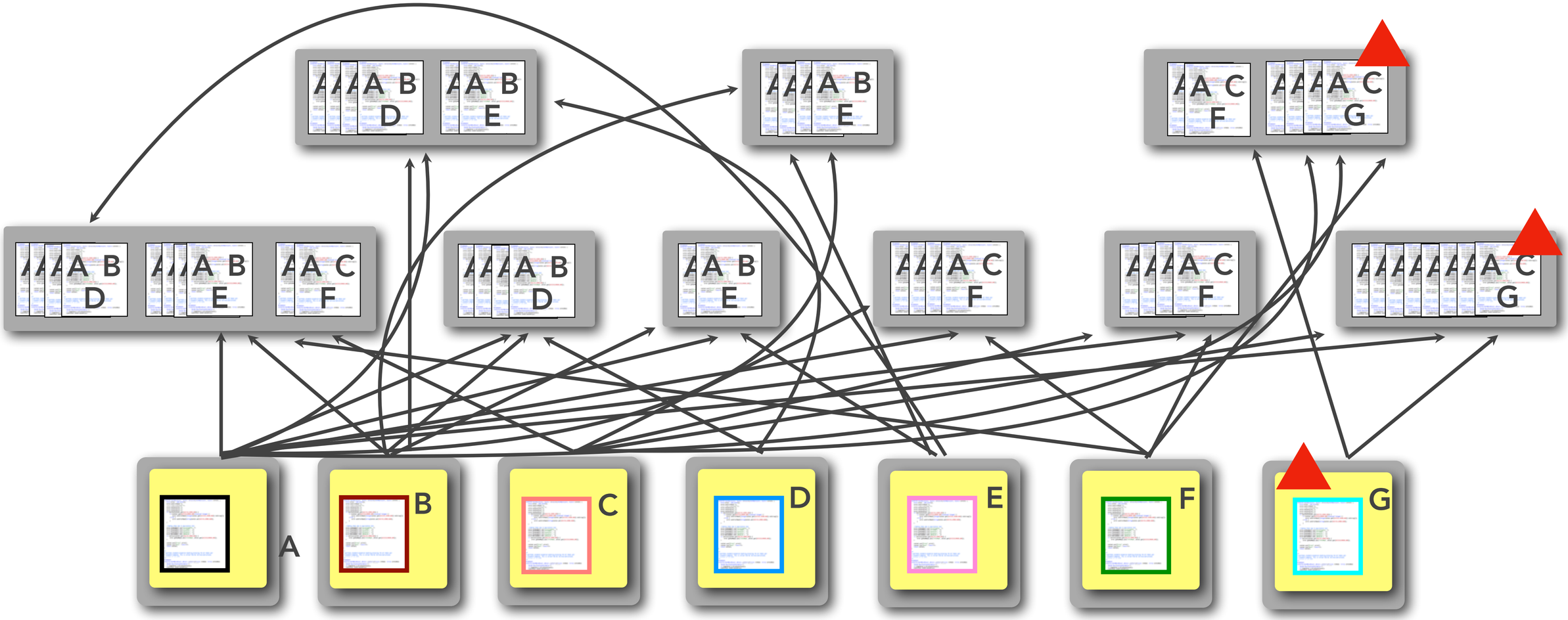
code sharing techniques

shared library



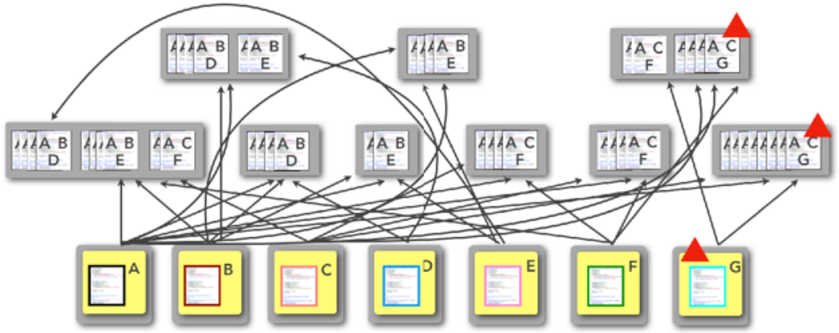
code sharing techniques

shared library



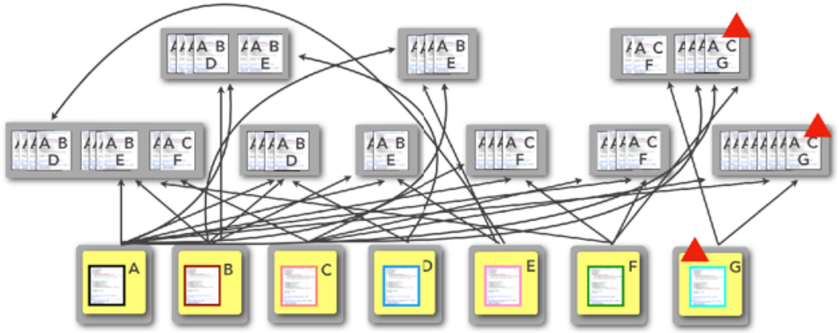
code sharing techniques

shared library



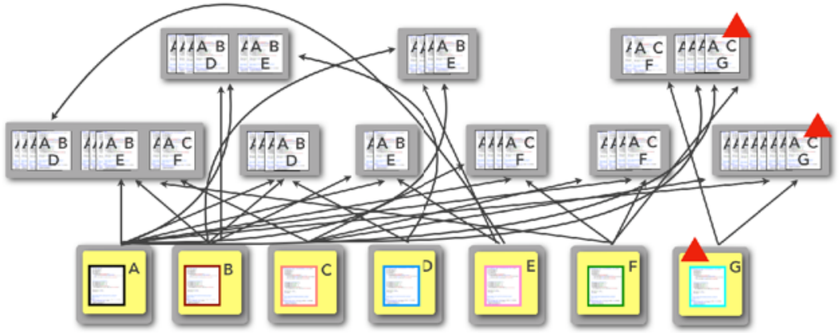
code sharing techniques

shared library



code sharing techniques

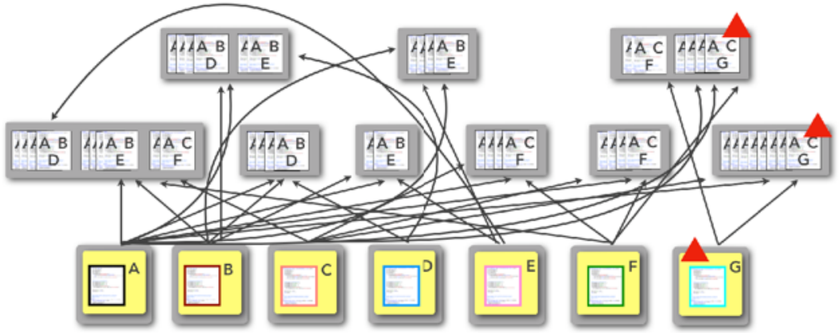
shared library



- ✓ ability to version changes
- ✓ easy to expand

code sharing techniques

shared library



✓ ability to version changes

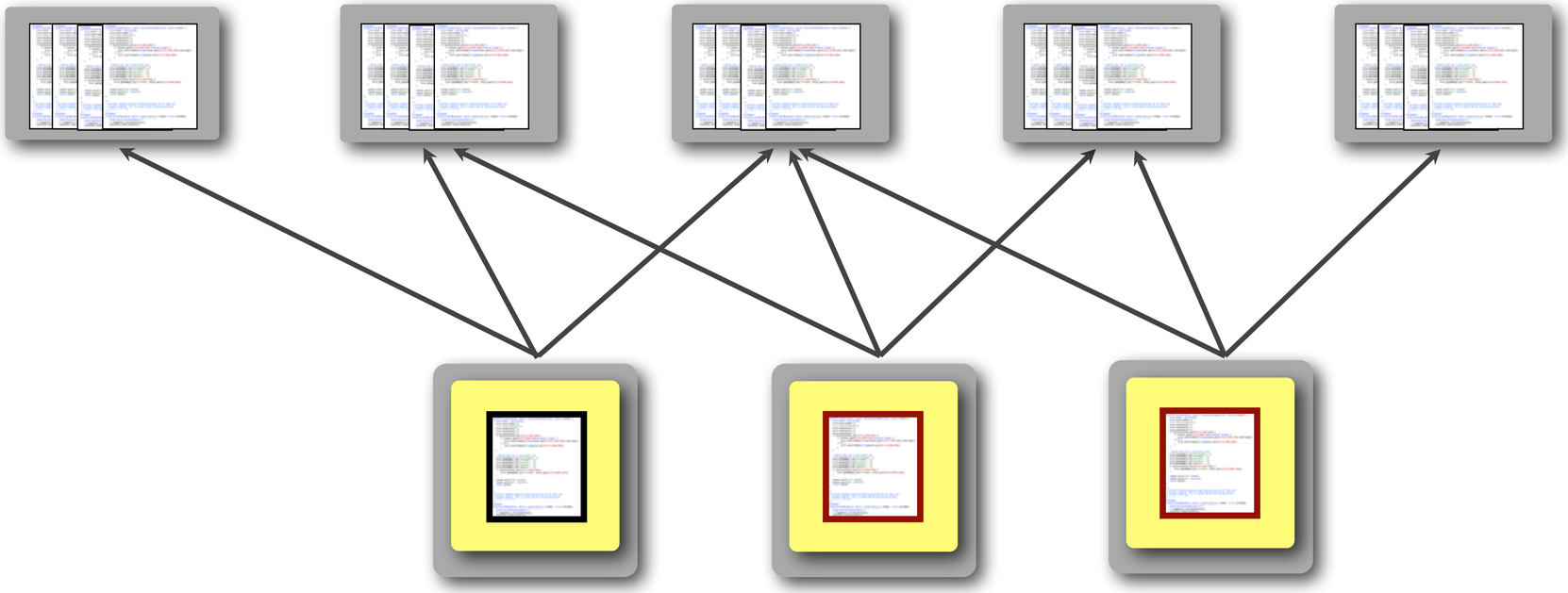
✓ easy to expand

✗ dependency management

✗ heterogeneous code bases

code sharing techniques

shared service

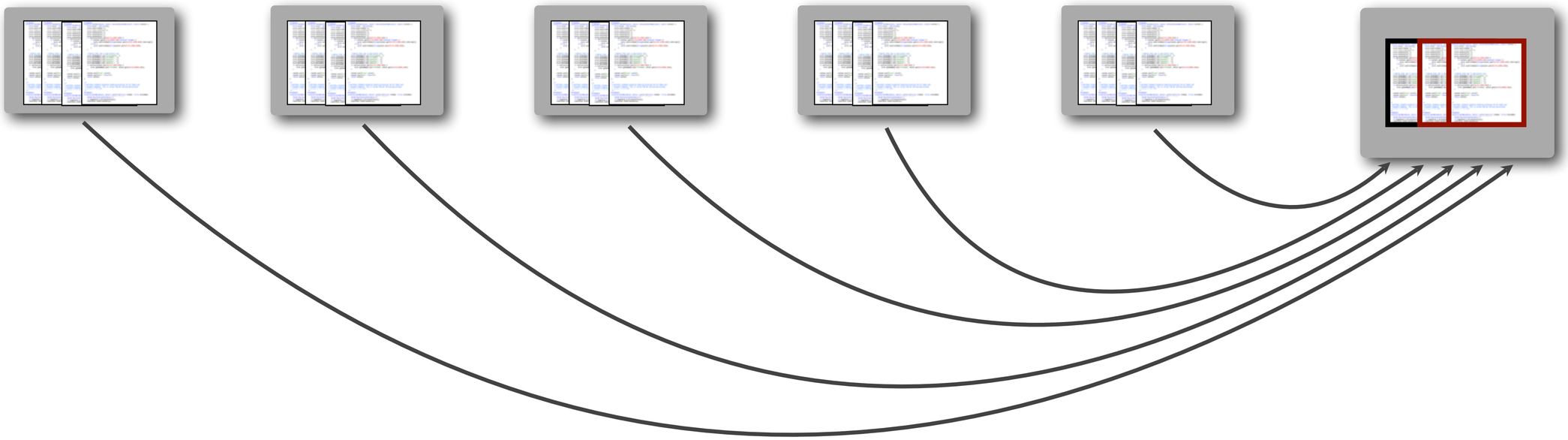


code sharing techniques

shared service

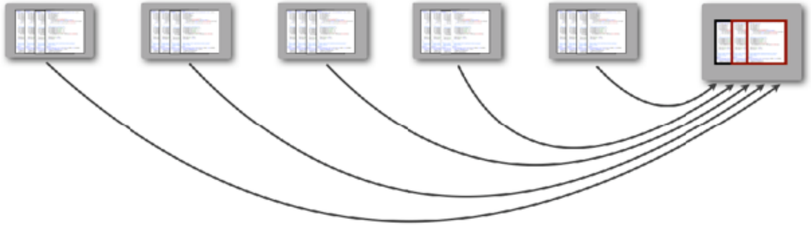
code sharing techniques

shared service



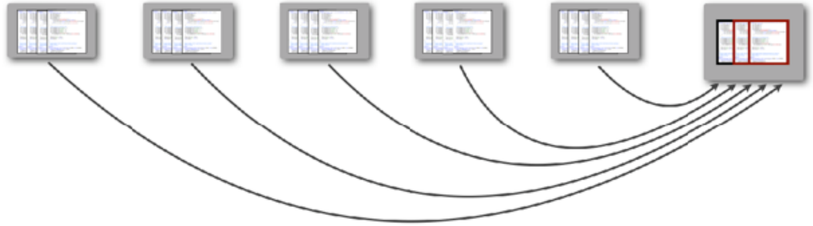
code sharing techniques

shared service



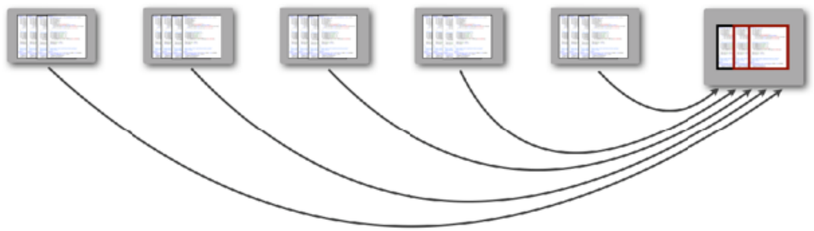
code sharing techniques

shared service



code sharing techniques

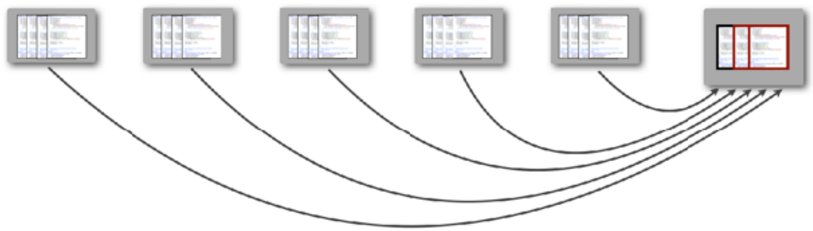
shared service



- ✓ high code volatility
- ✓ heterogeneous code

code sharing techniques

shared service



✓ high code volatility

✓ heterogeneous code

✗ versioning is difficult

✗ performance issues due to latency

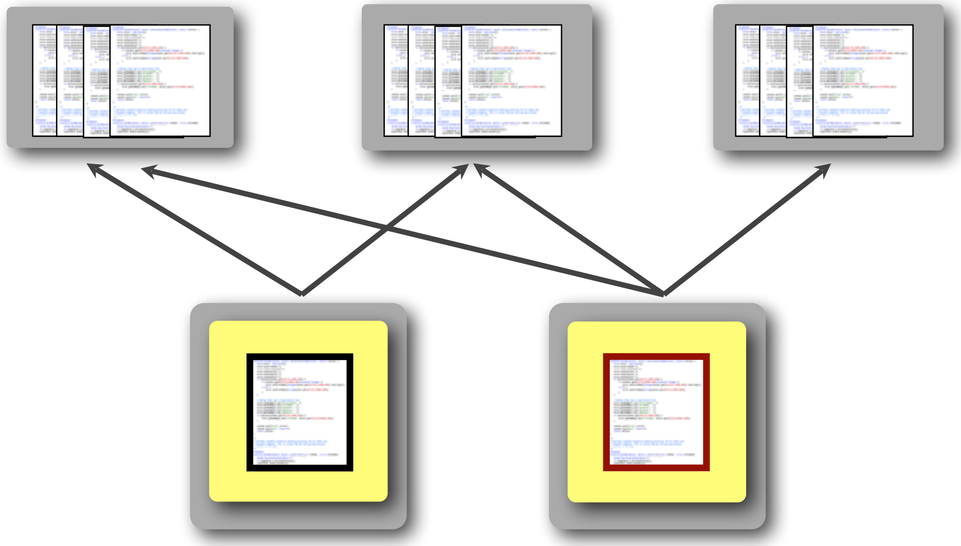
✗ availability / fault tolerance

✗ scalability / throughput

✗ runtime changes (things break)

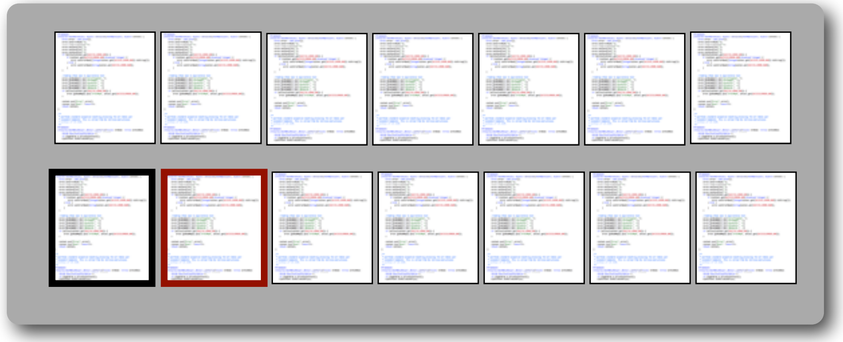
code sharing techniques

service consolidation



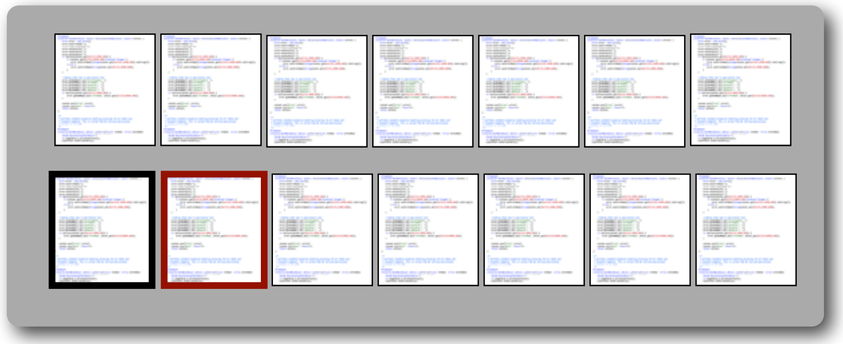
code sharing techniques

service consolidation



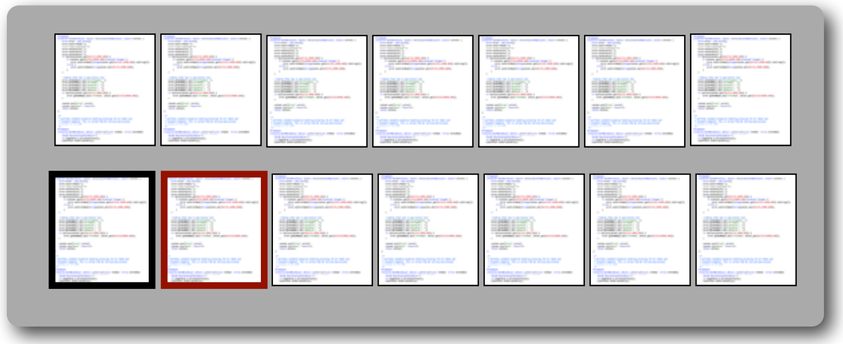
code sharing techniques

service consolidation



code sharing techniques

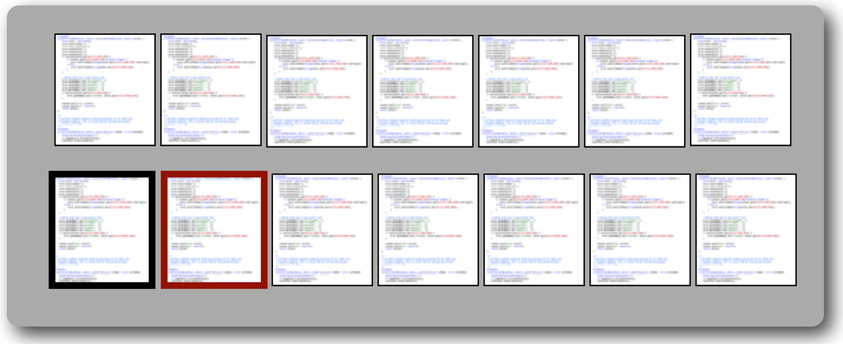
service consolidation



✓ no code sharing

code sharing techniques

service consolidation



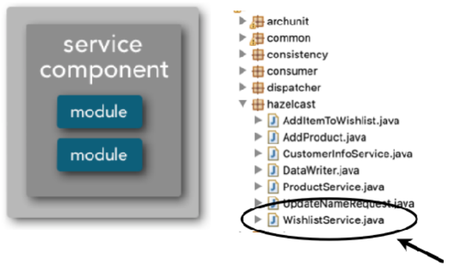
✓ no code sharing

✗ larger testing scope

✗ greater risk of deployment

✗ less agility

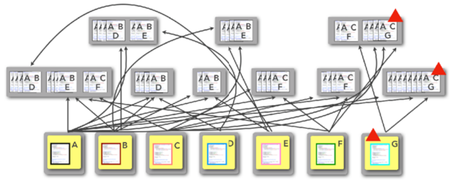
code sharing techniques



replicated code

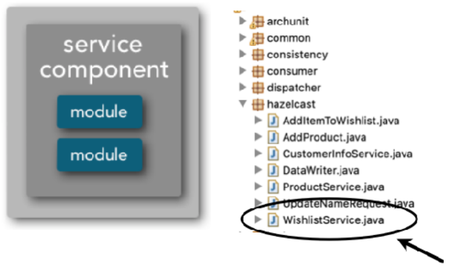
consolidate services

shared service



shared library

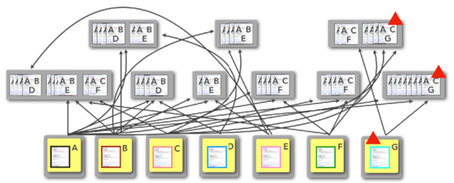
code sharing techniques



replicated code

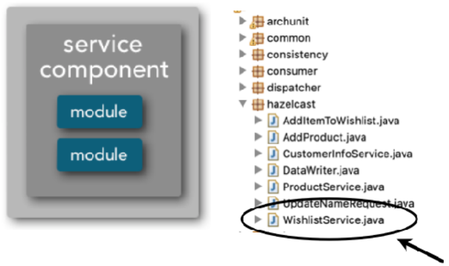
consolidate services

shared service

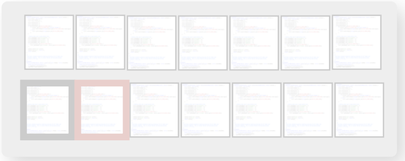


shared library

code sharing techniques



replicated code



consolidate services

shared service

shared library

✓ shared code that doesn't change (static code)

code sharing techniques

replicated code

consolidate services

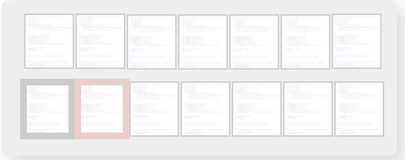
shared service

shared library

✓ minimal services (2-3) dependent on shared code that changes frequently

code sharing techniques

replicated code



consolidate services

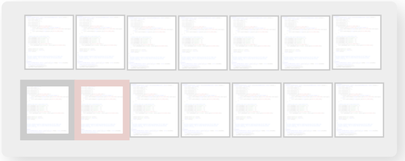
shared service

shared library

✓ shared code in a heterogeneous environment, or shared code in homogeneous environment that frequently changes

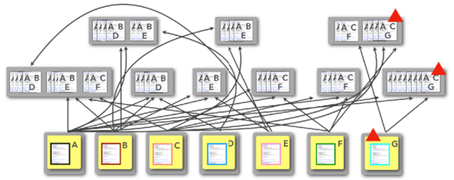
code sharing techniques

replicated code



consolidate services

shared service

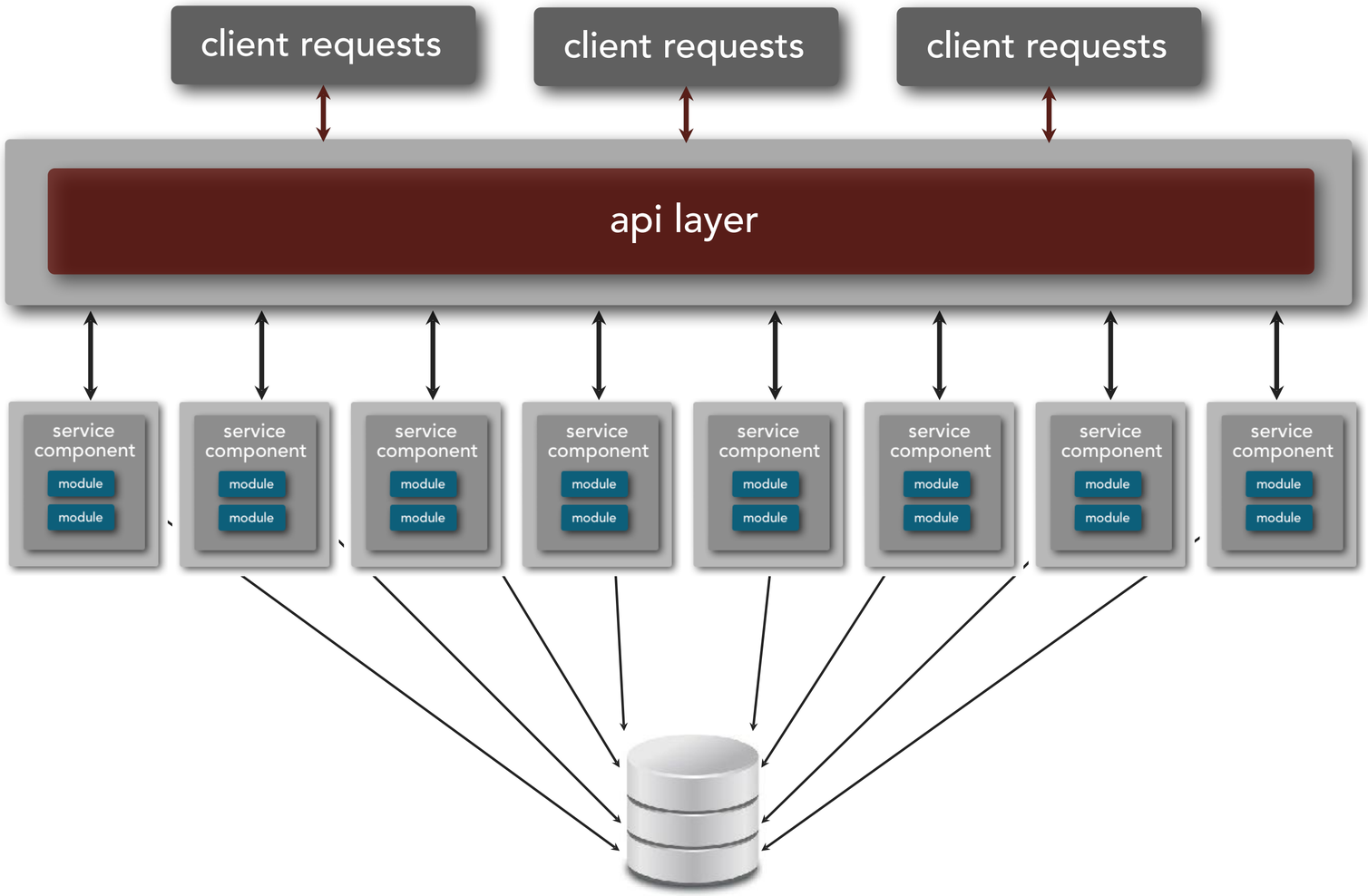


shared library

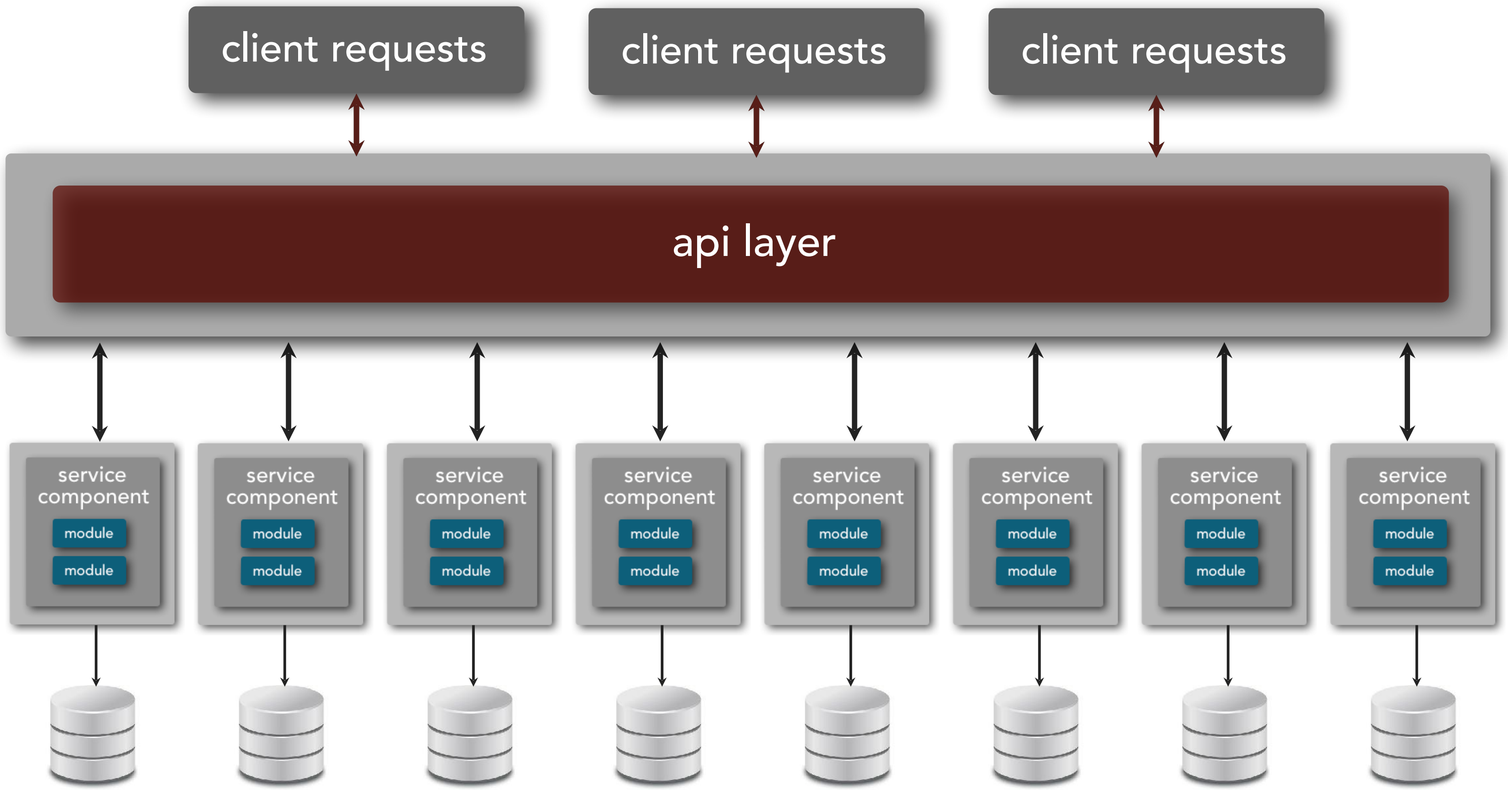
✓ shared code in a homogeneous environment that sometimes changes

Decomposing Data

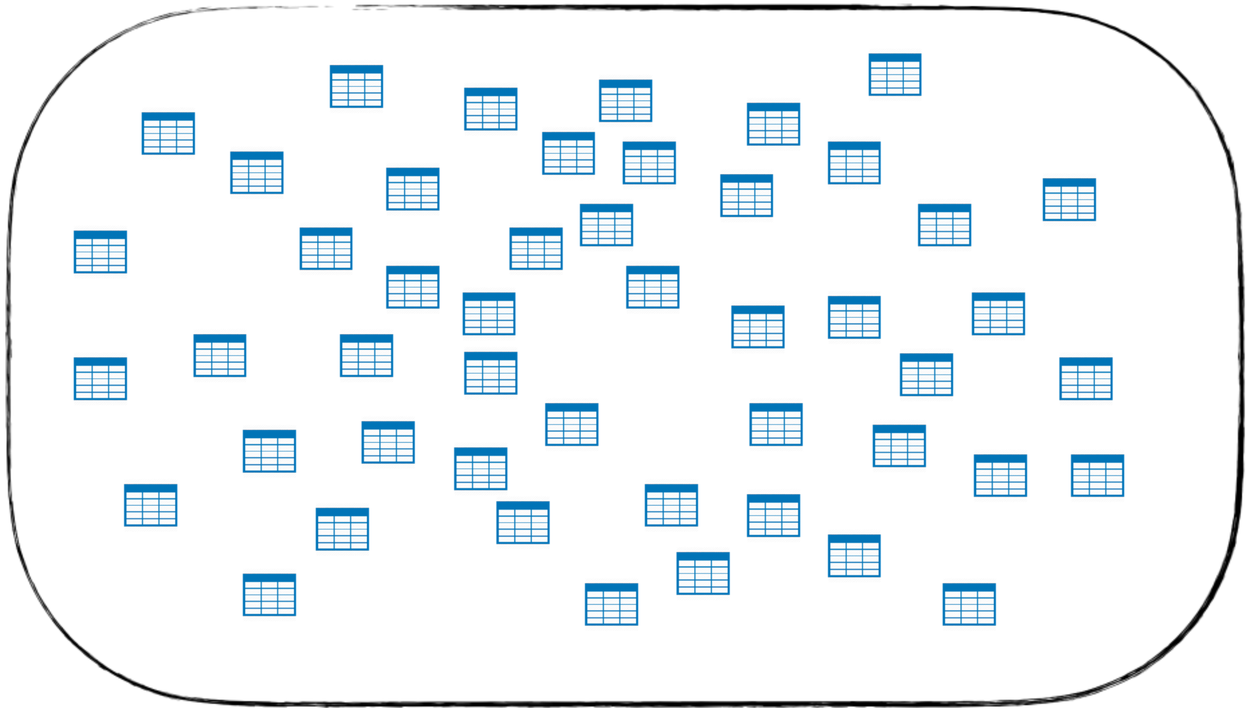
decomposing data



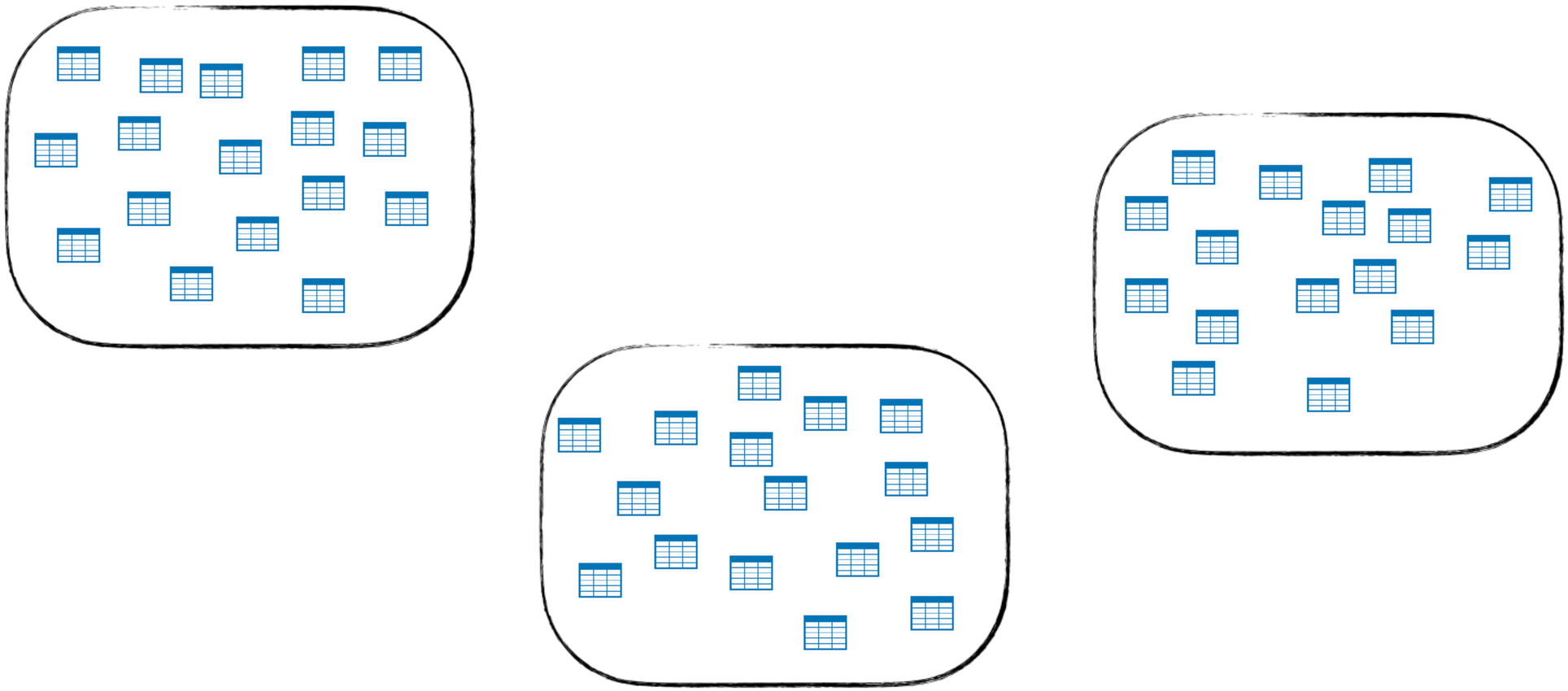
decomposing data



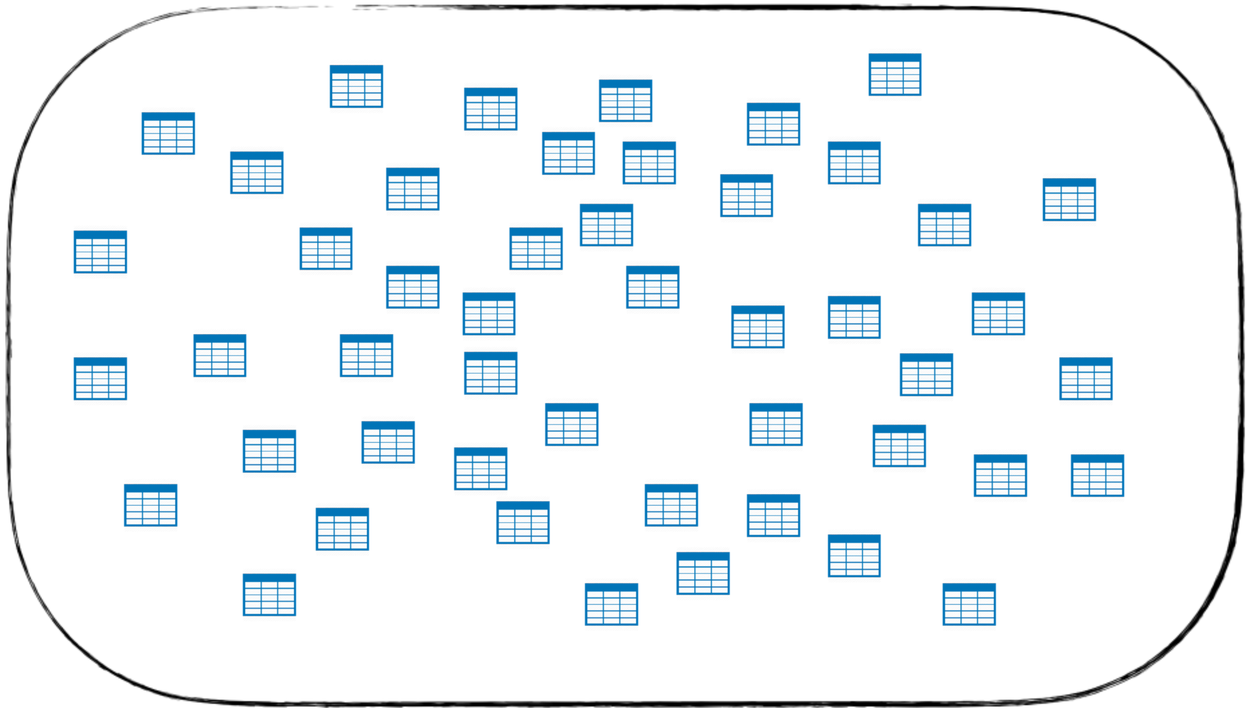
decomposing data



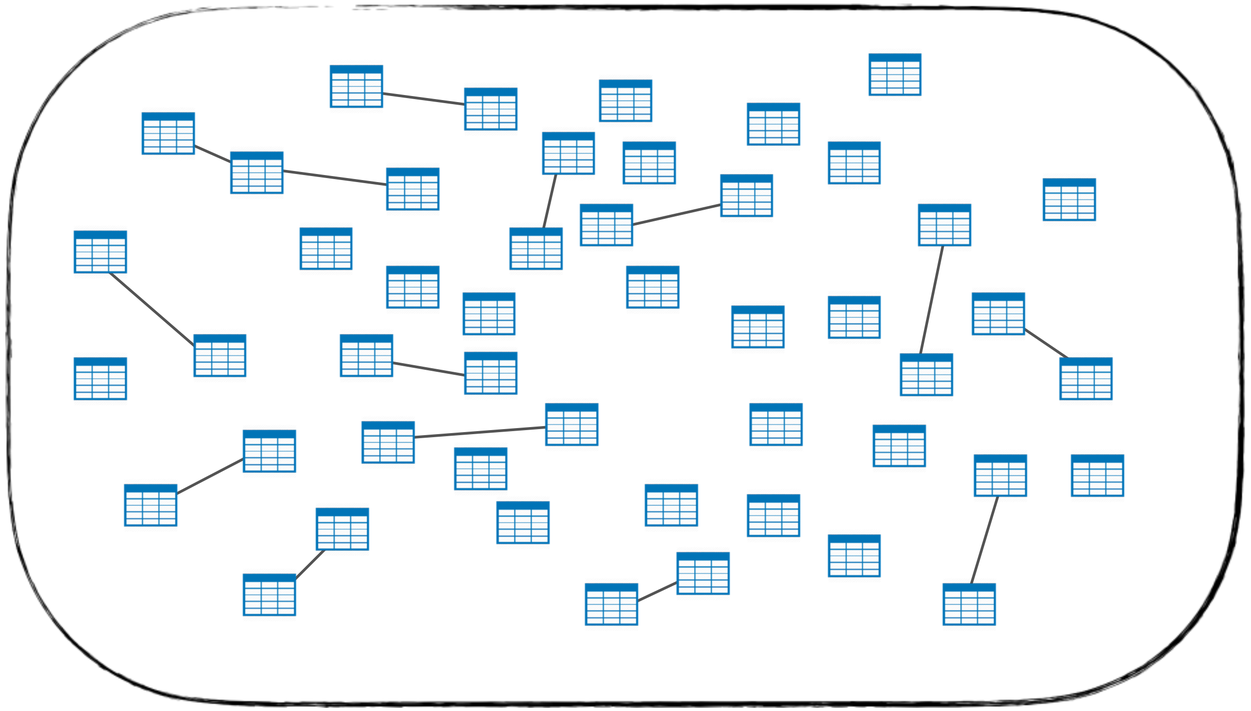
decomposing data



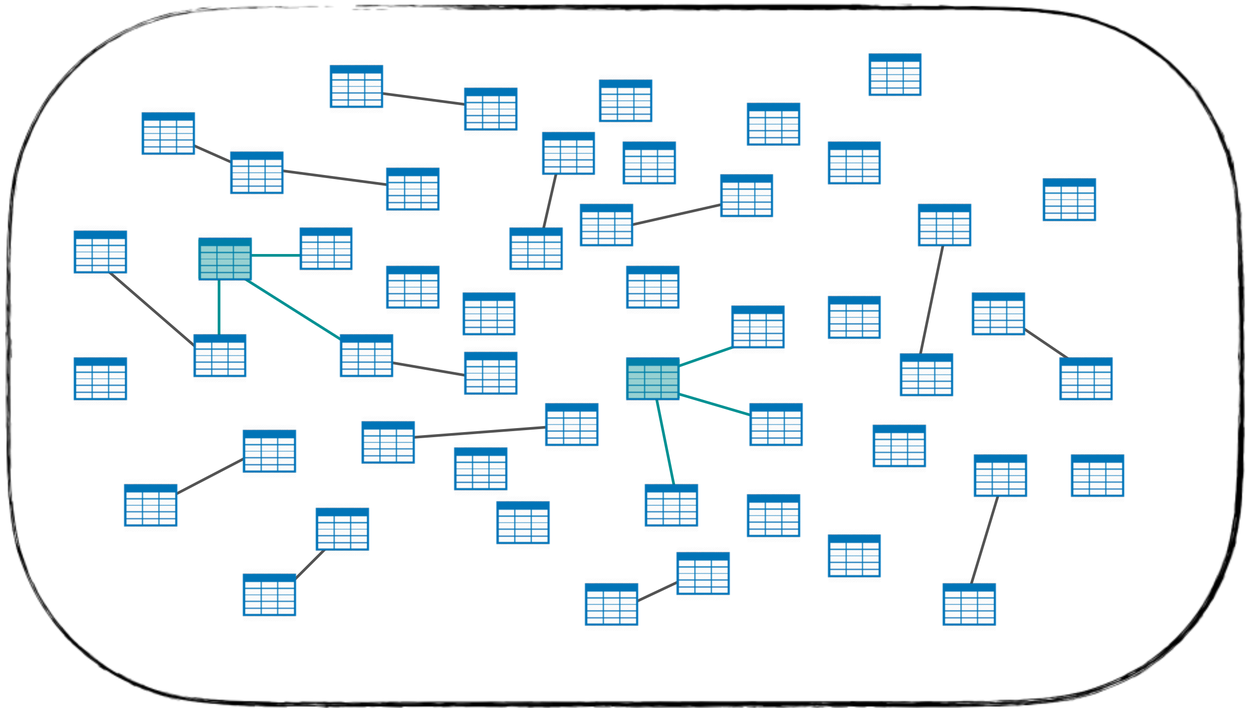
decomposing data



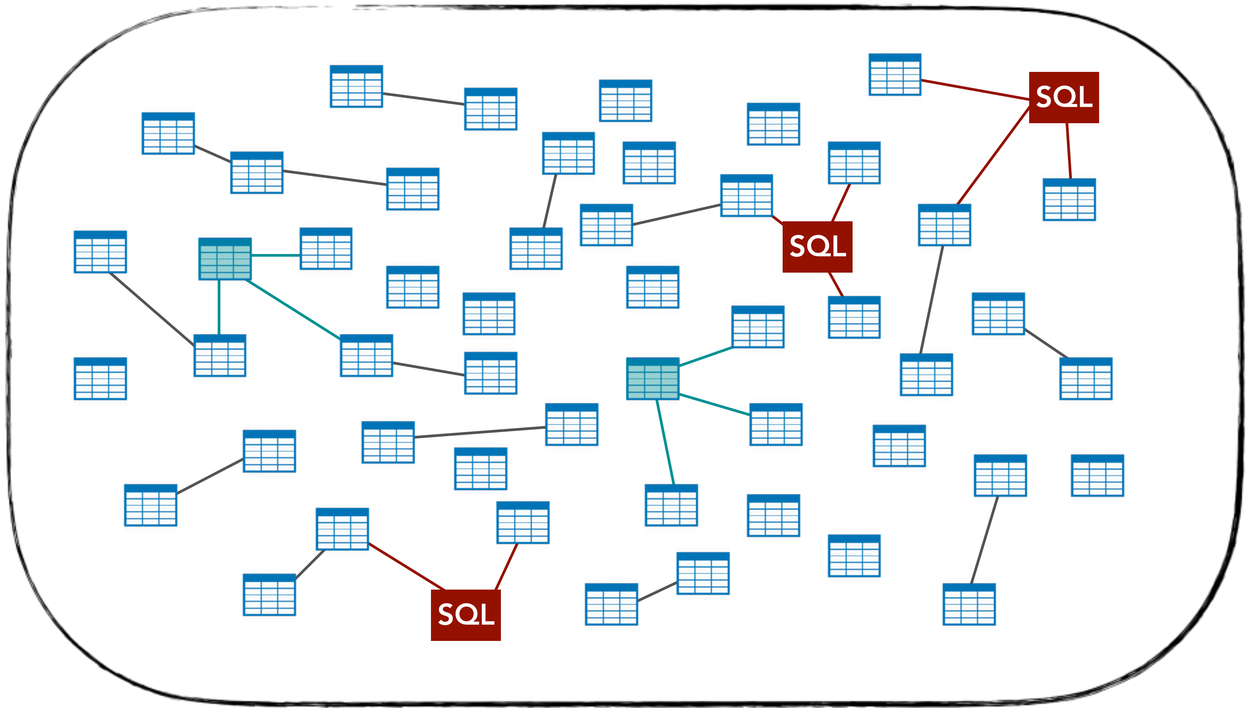
decomposing data



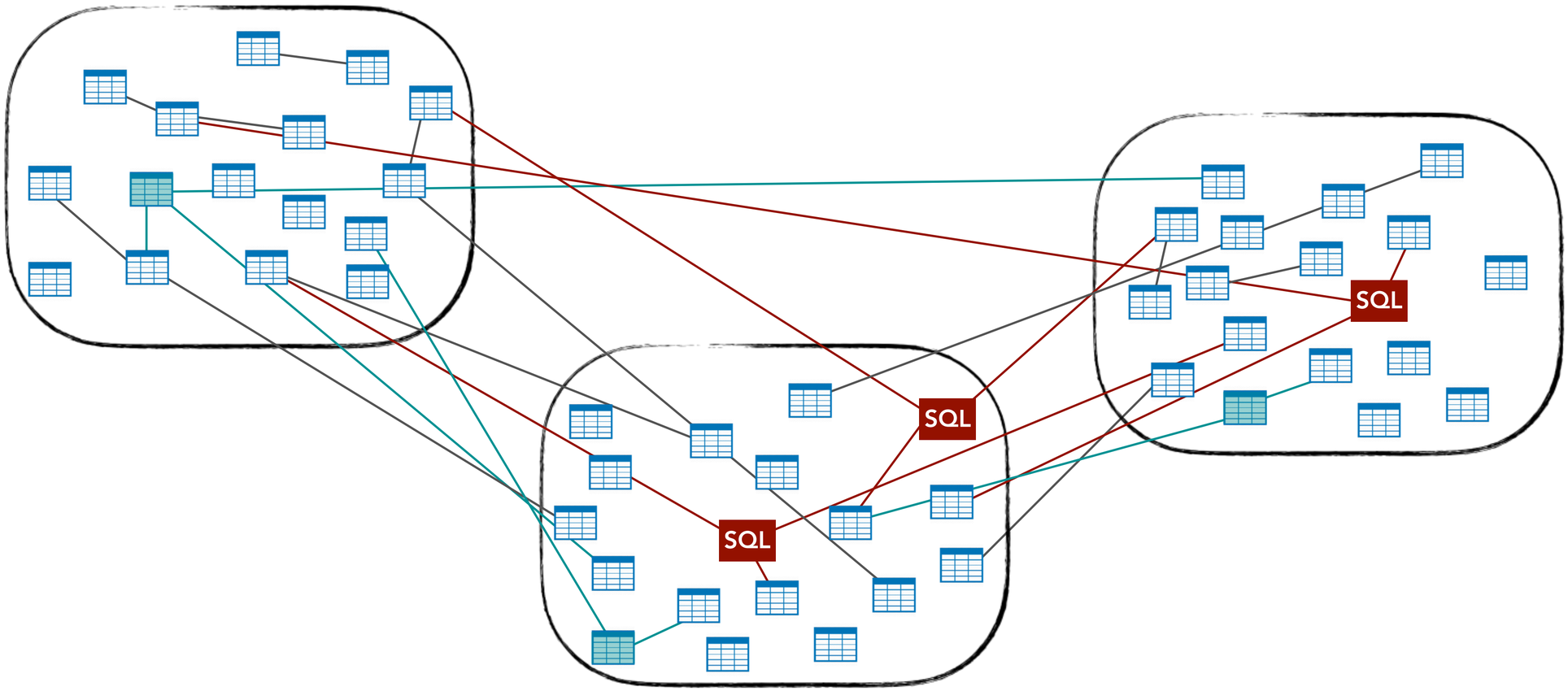
decomposing data



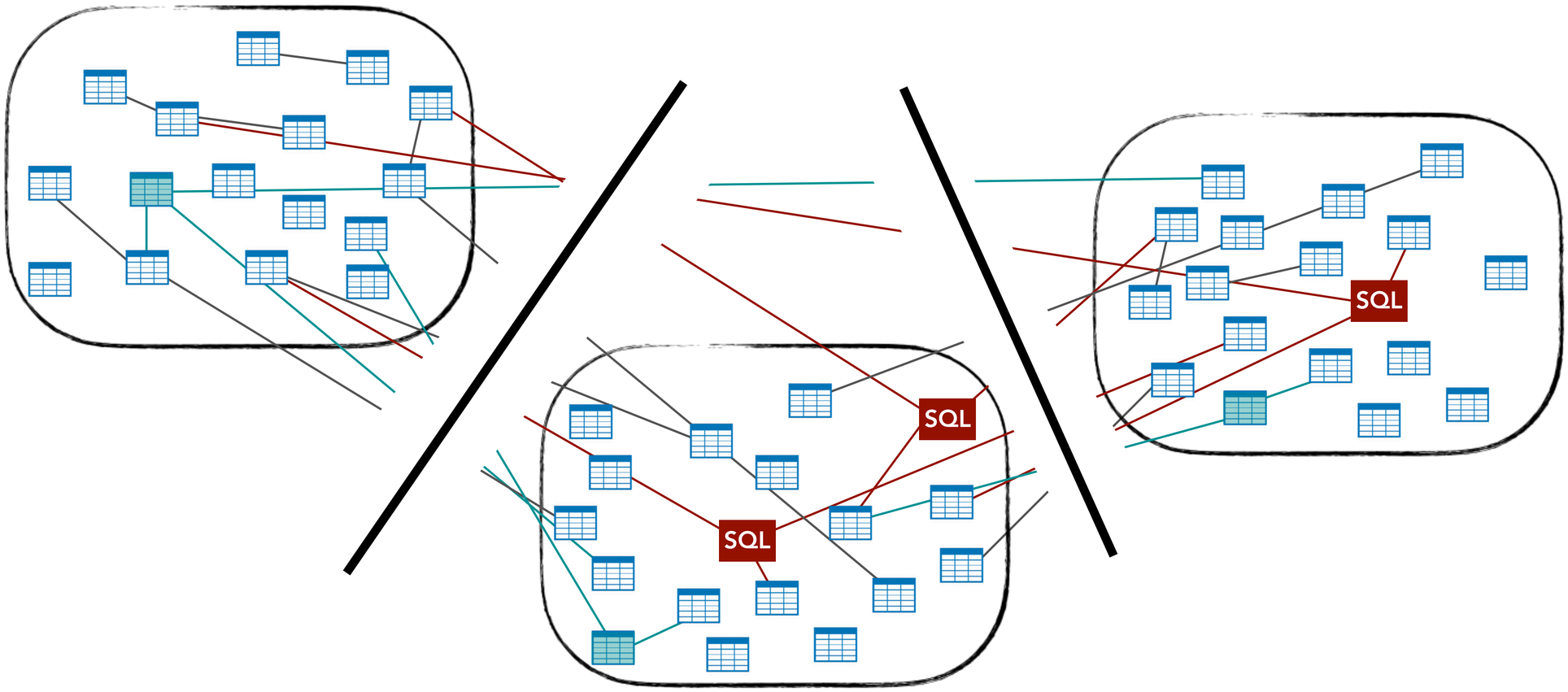
decomposing data



decomposing data

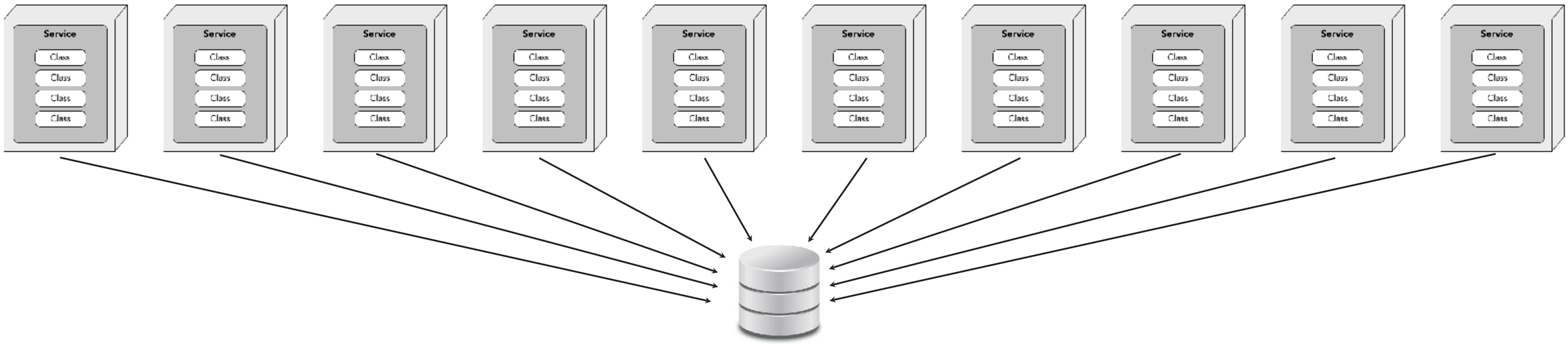


decomposing data

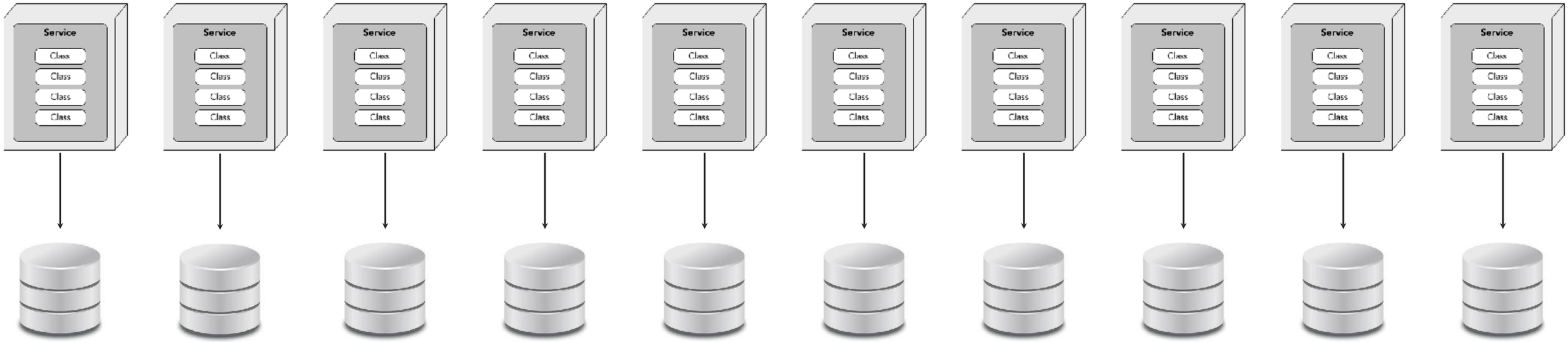


Data Domains

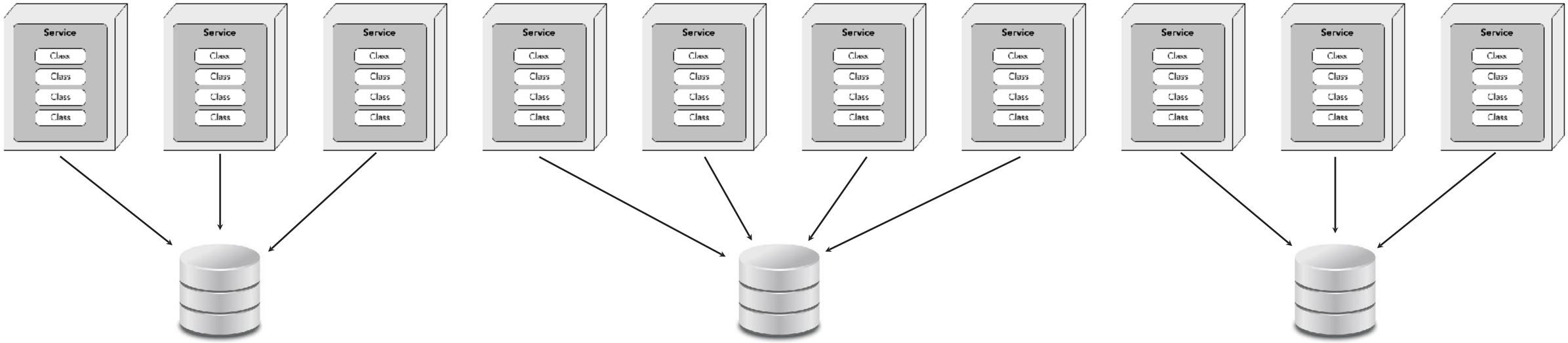
single monolithic database



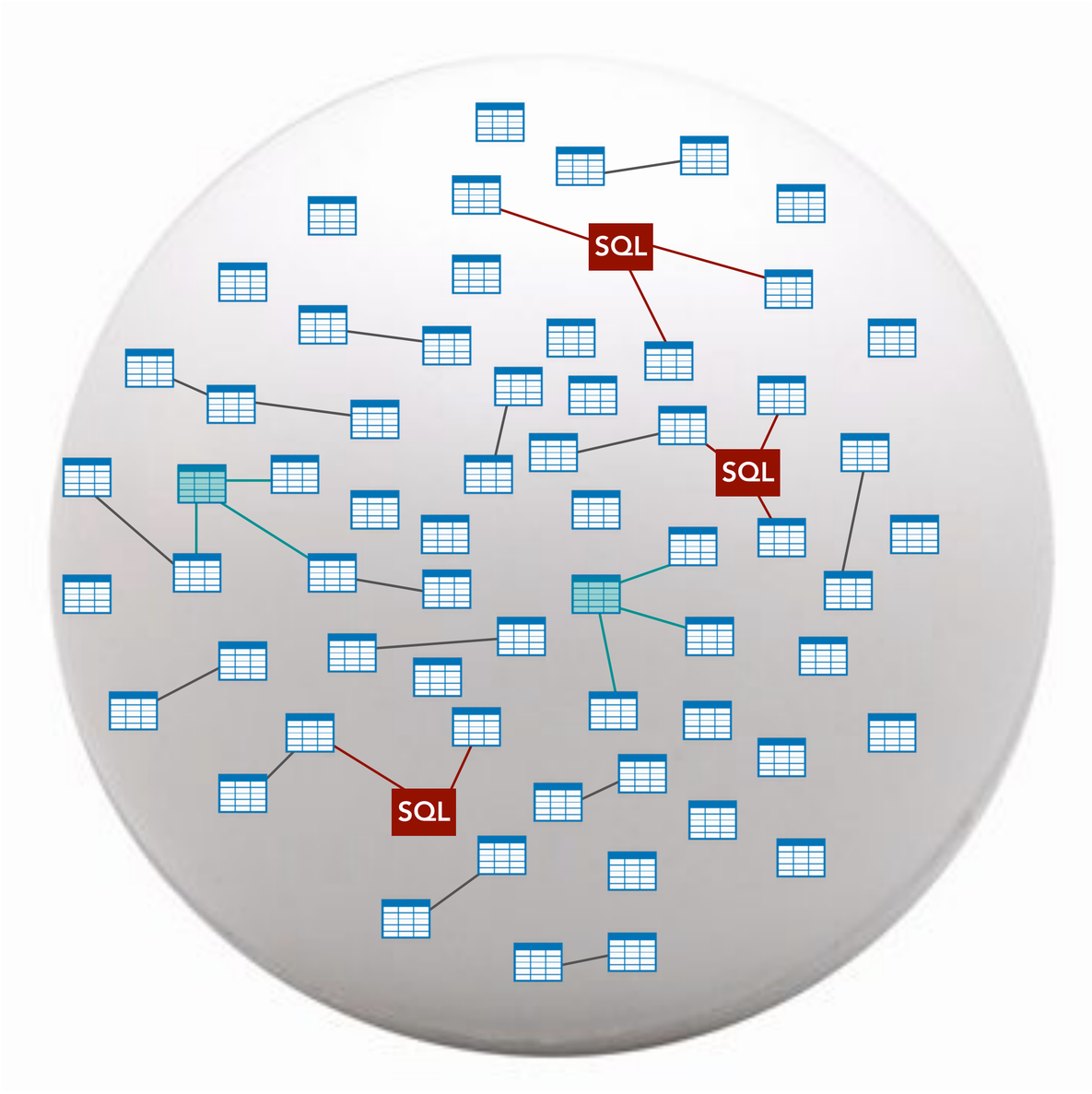
separate databases for each service



data domains

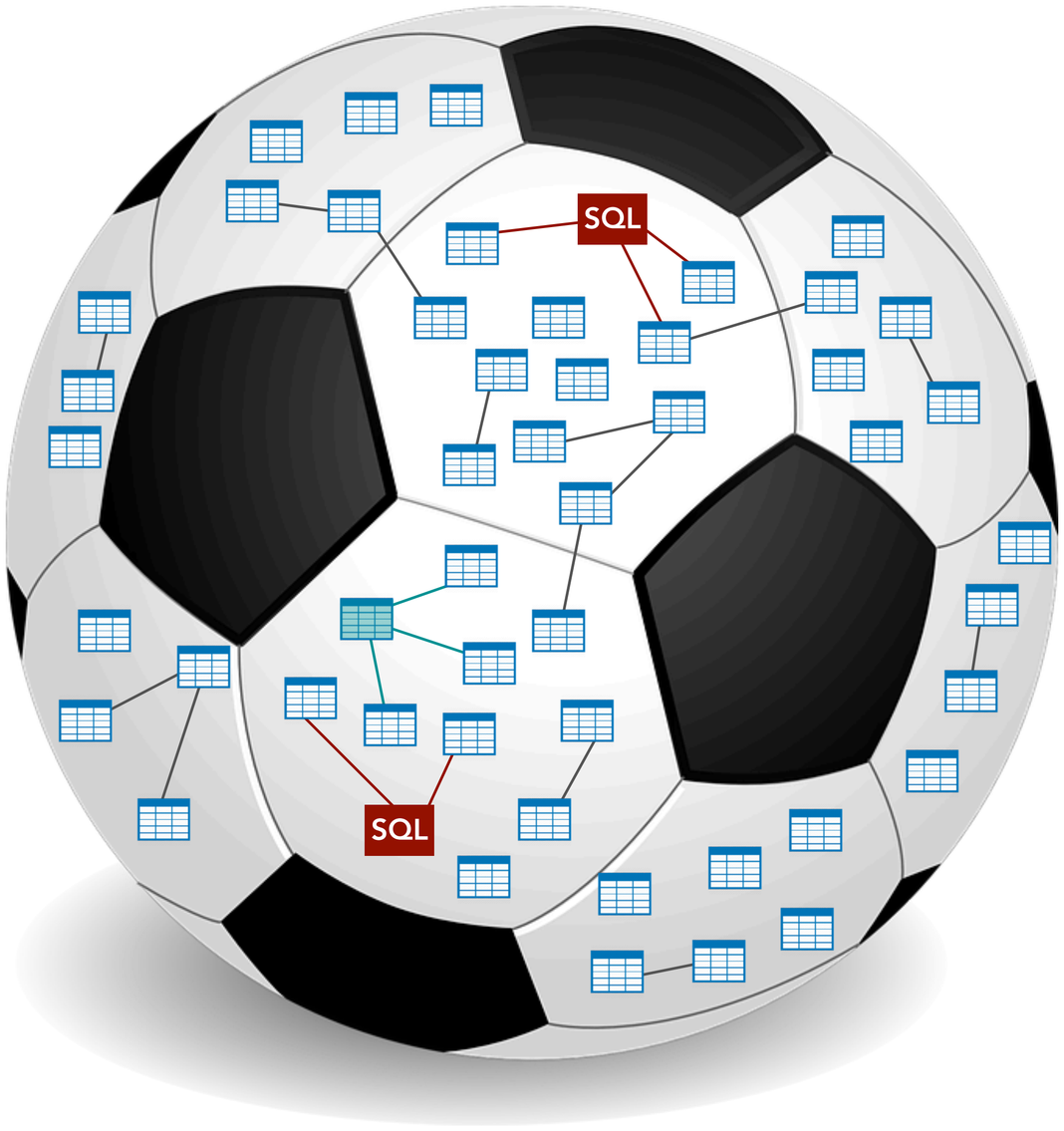


data domains



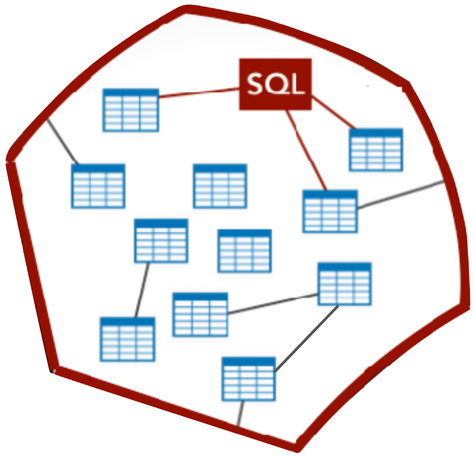
data domains

data domains

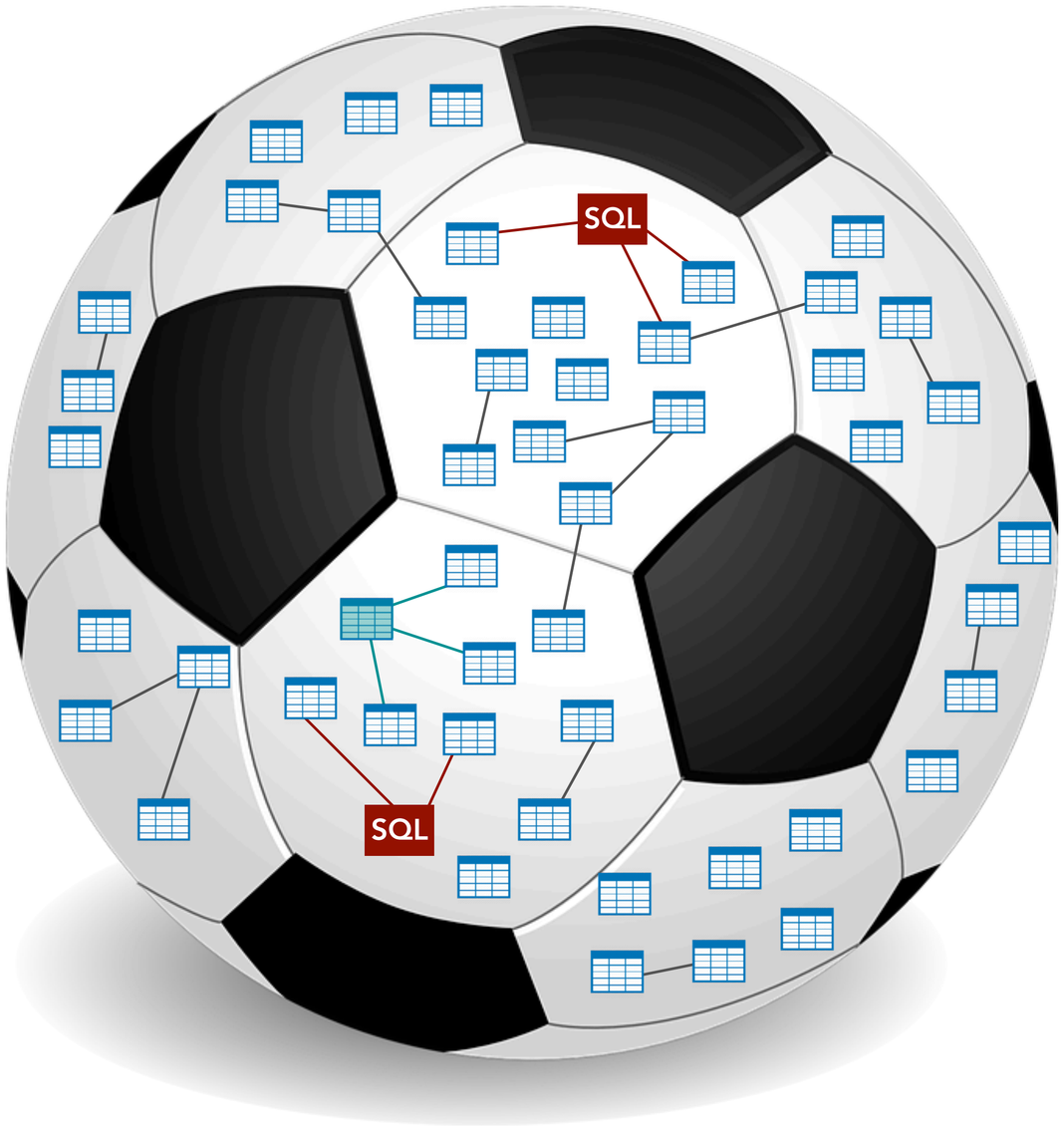


data domains

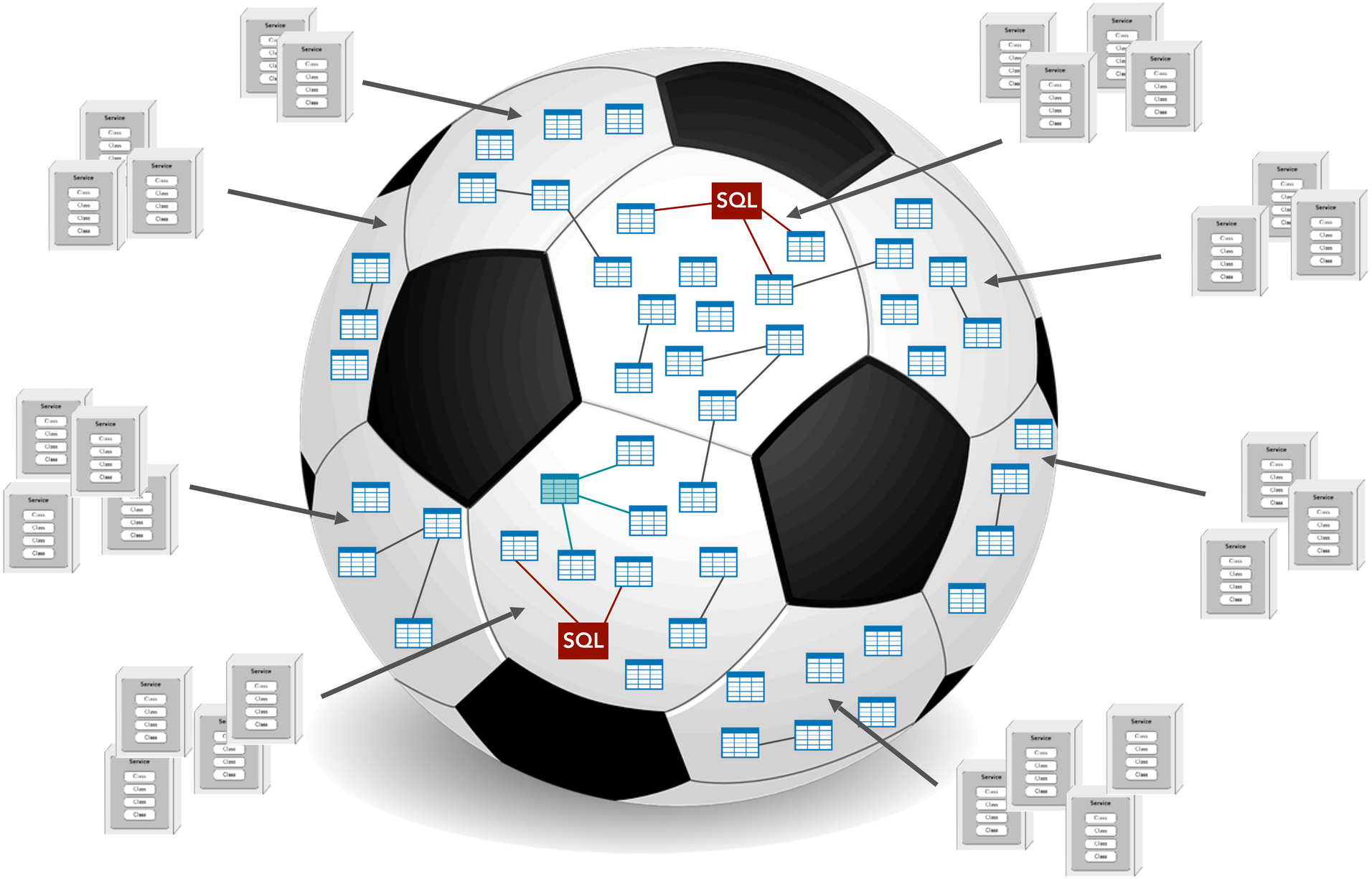
data domains



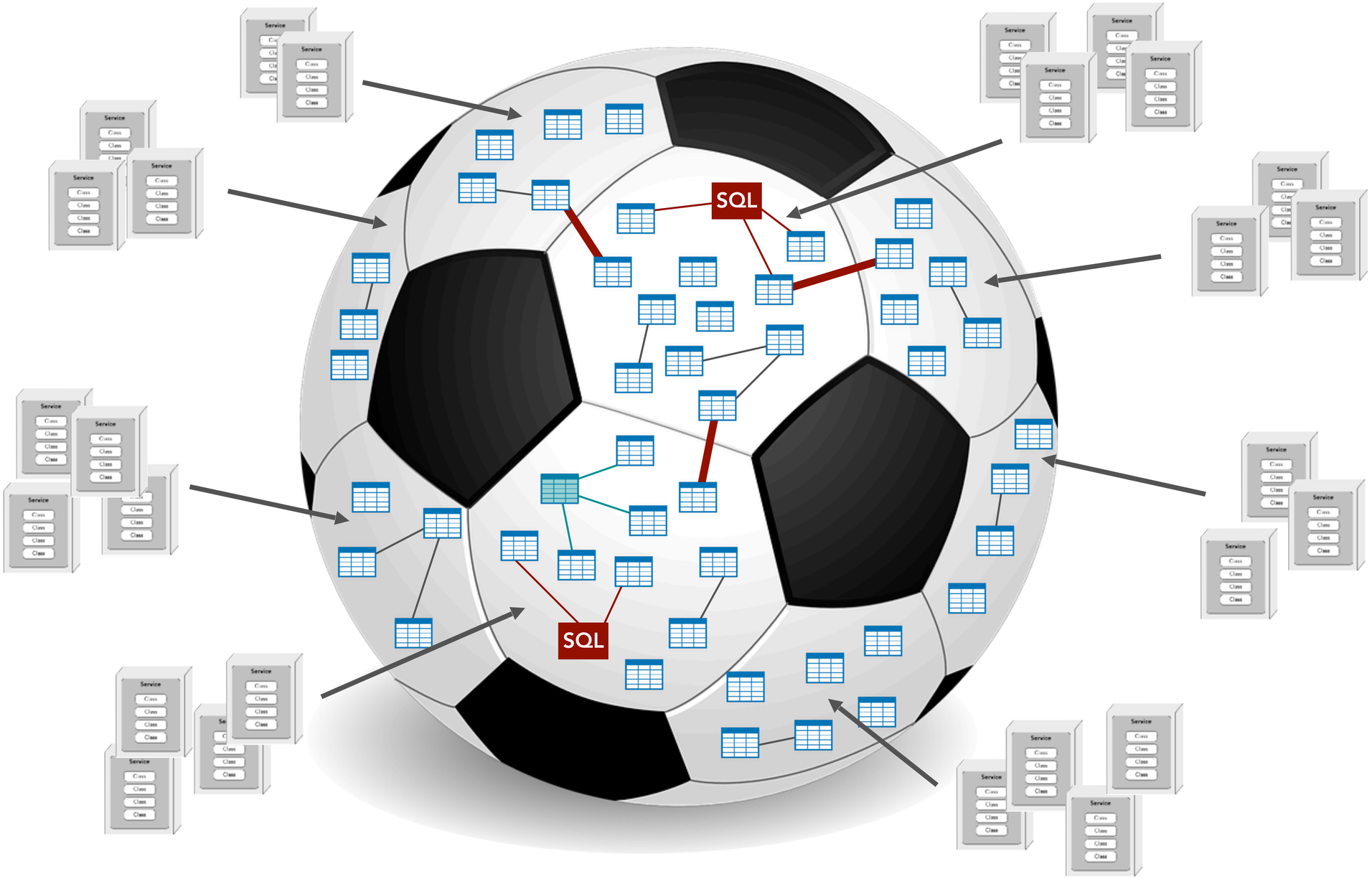
data domains



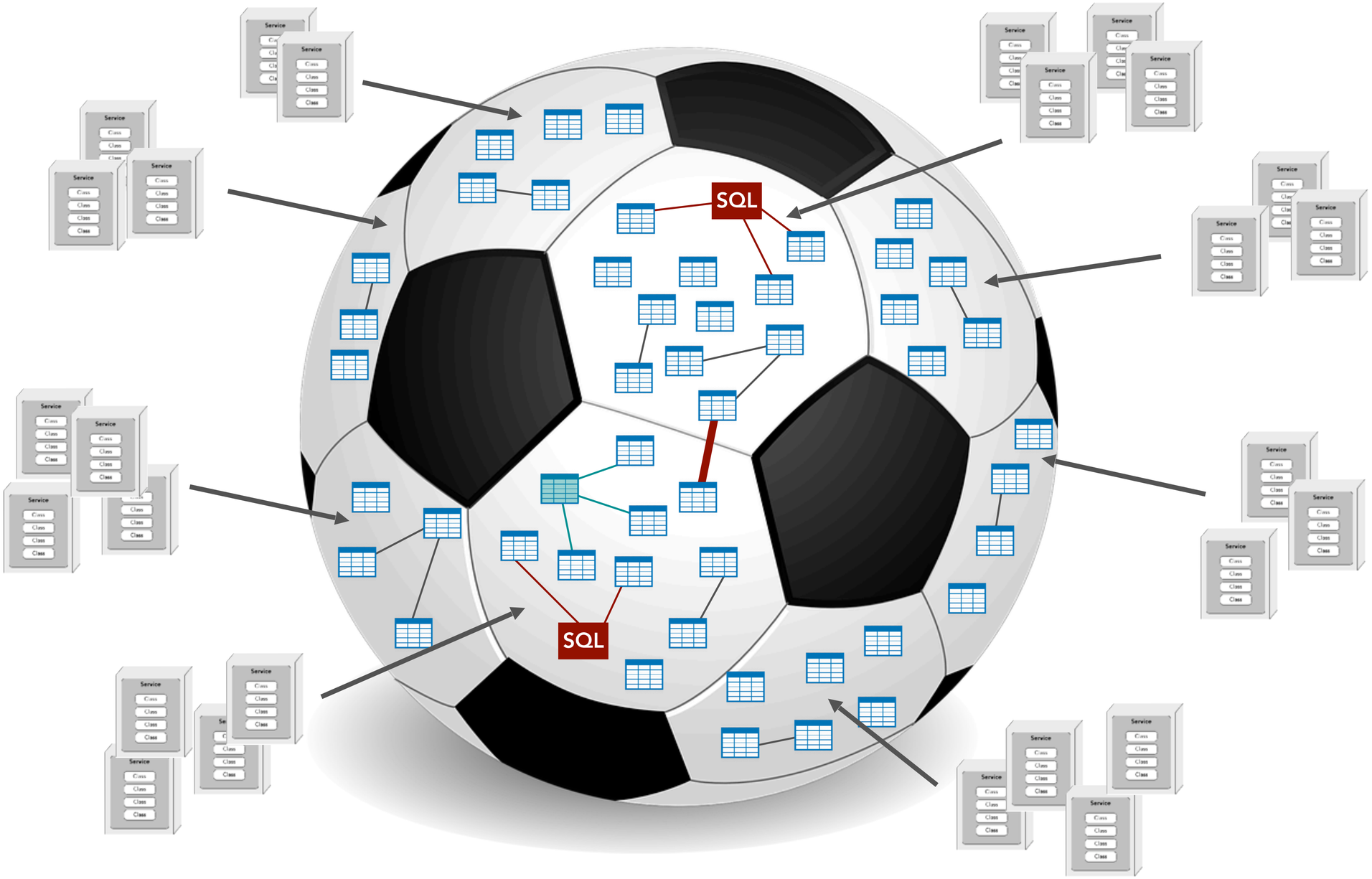
data domains



data domains

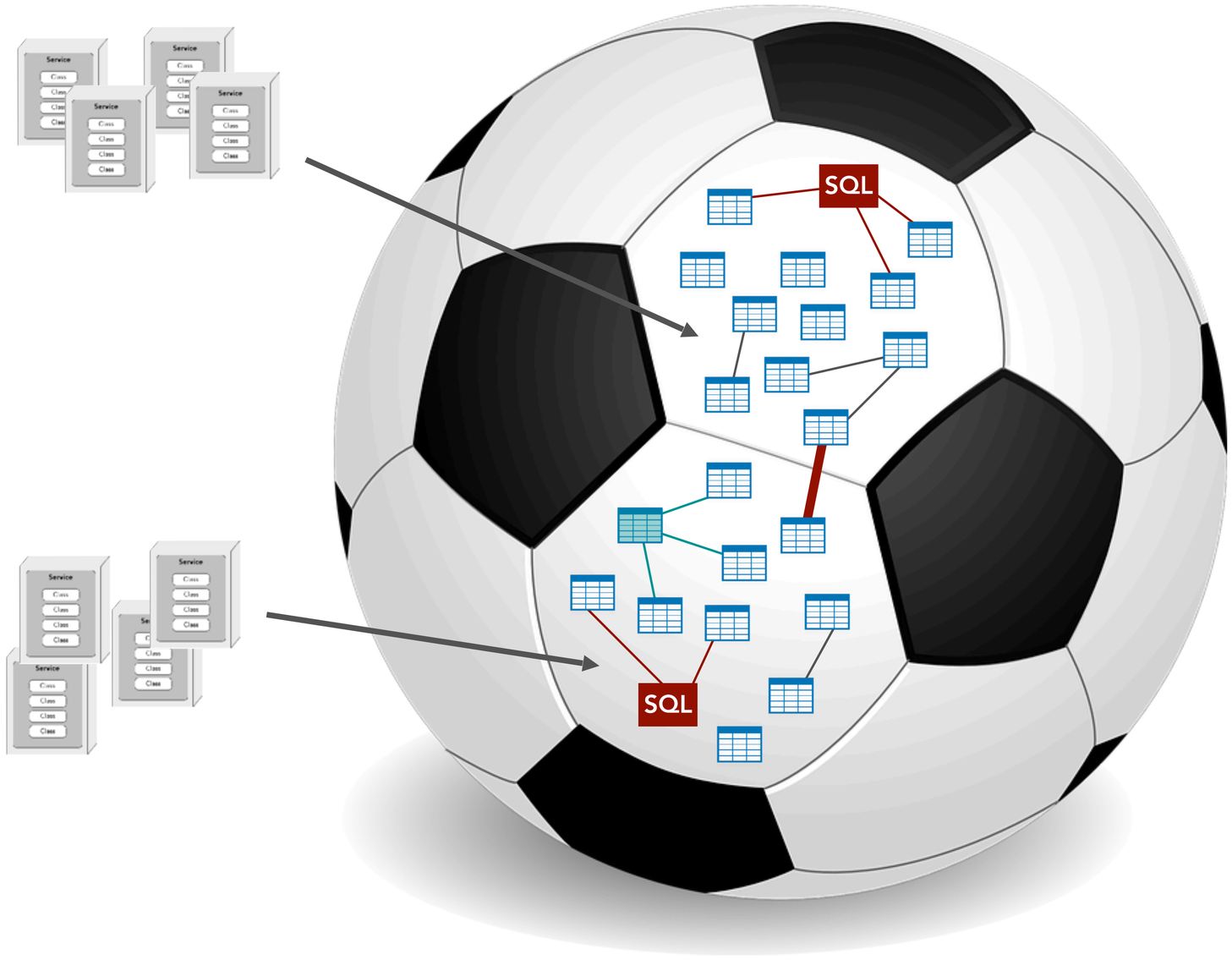


data domains

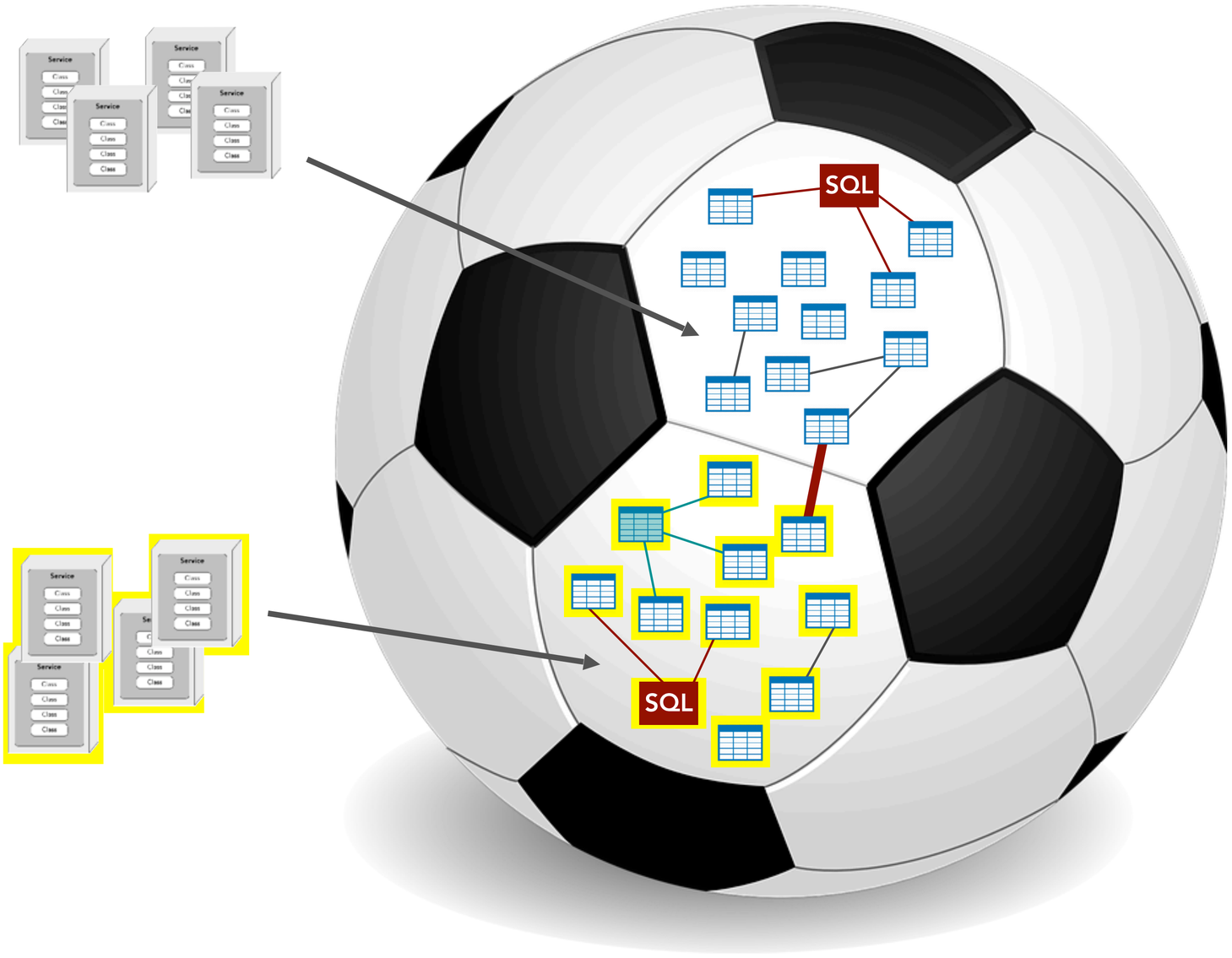


data domains

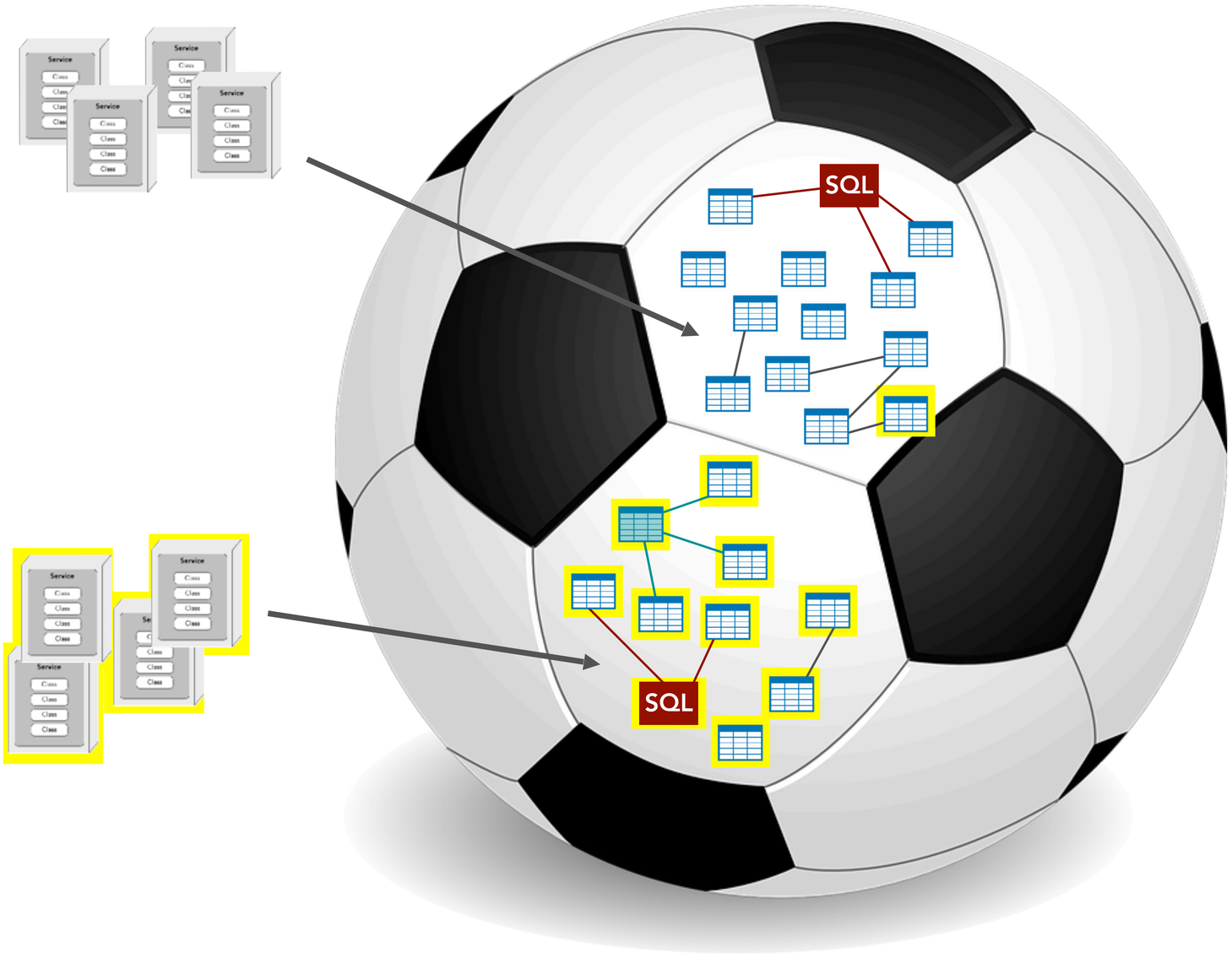
data domains



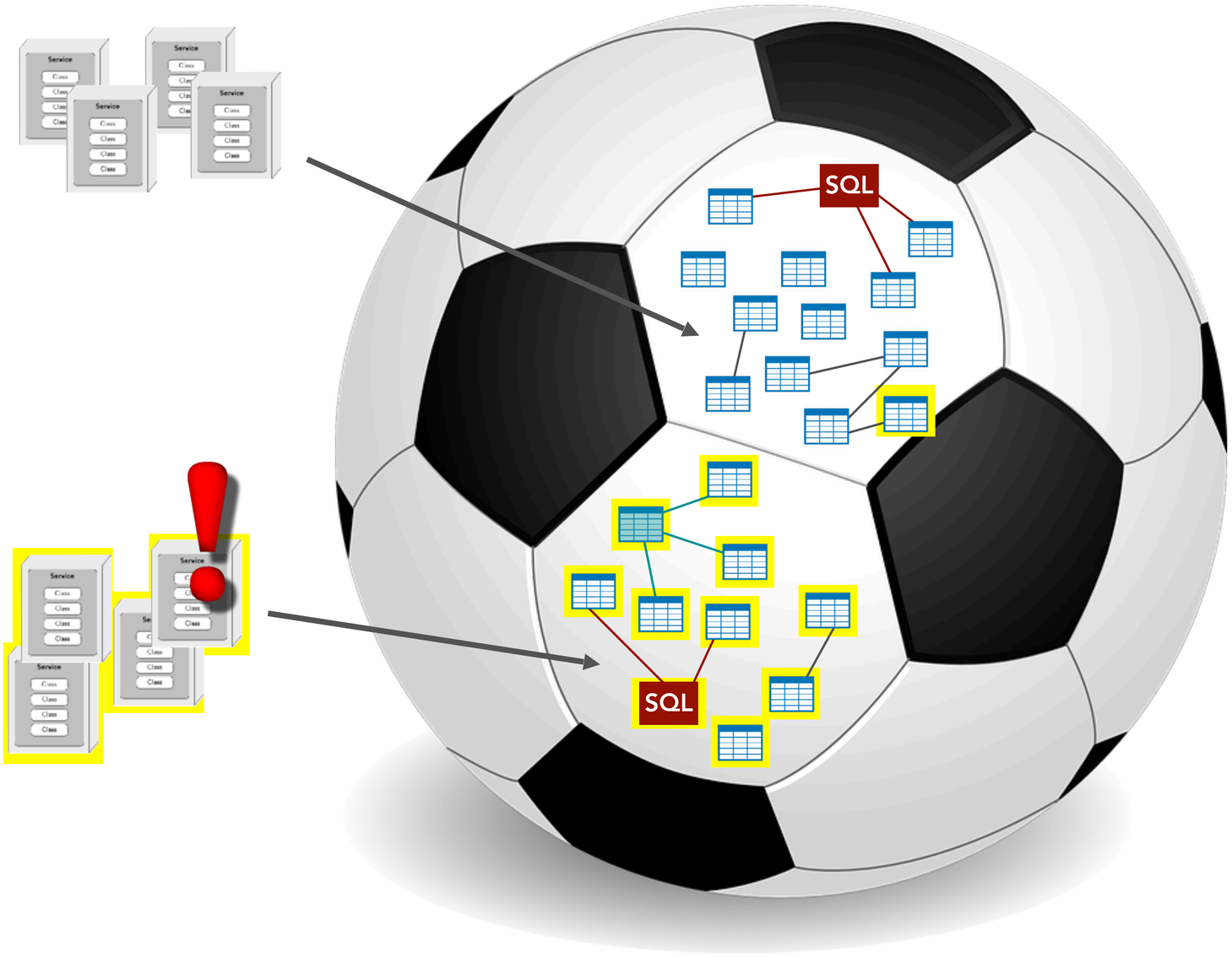
data domains



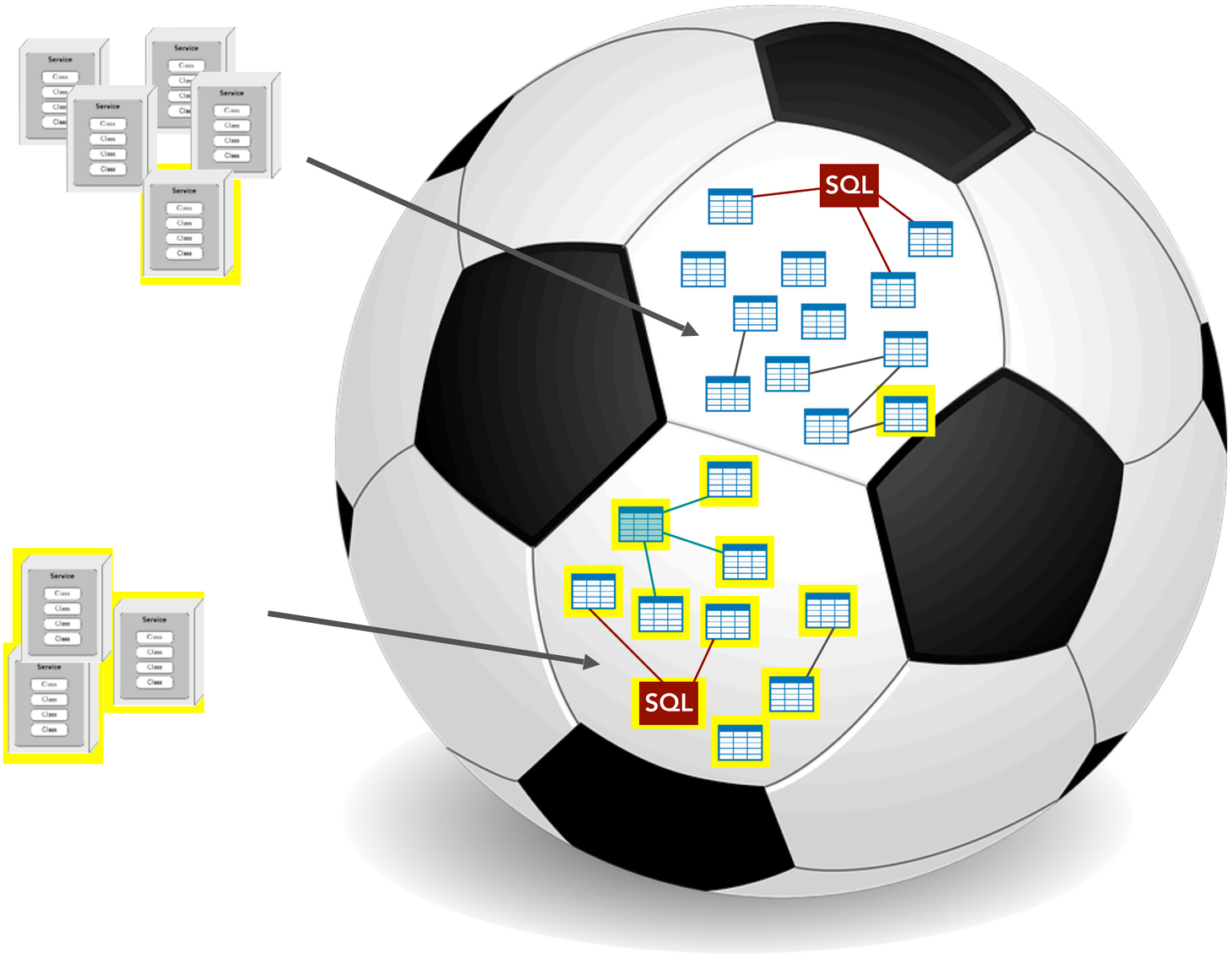
data domains



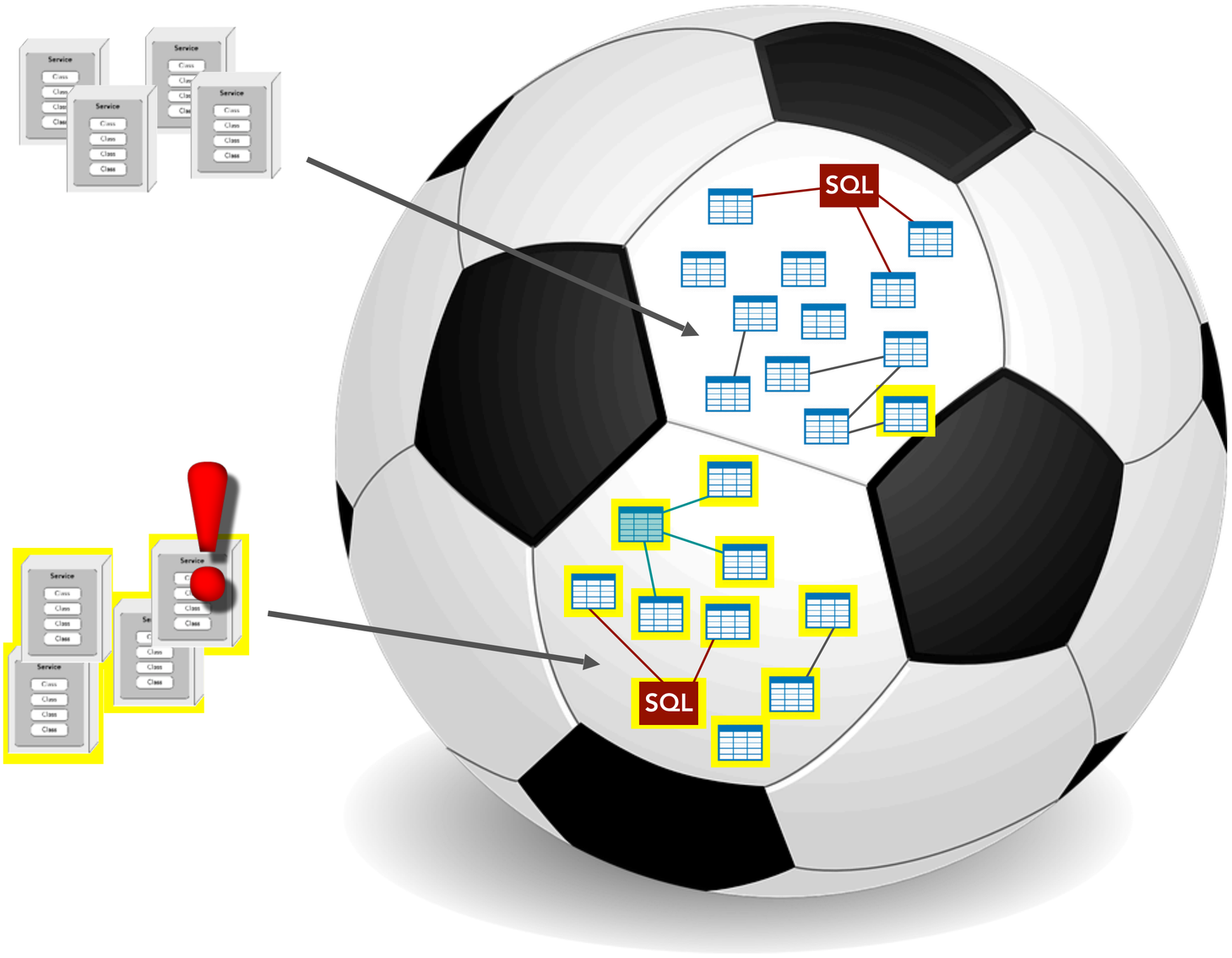
data domains



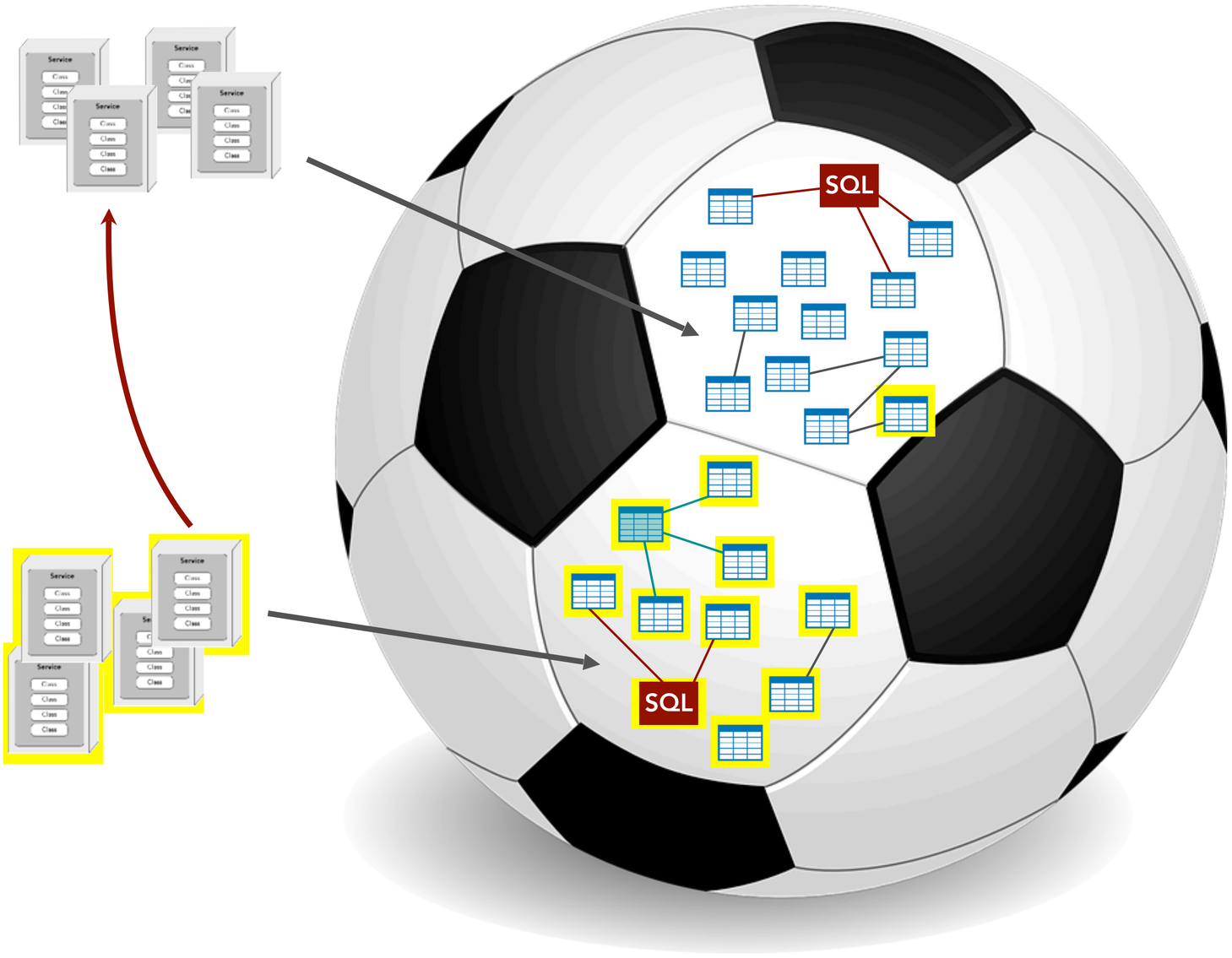
data domains



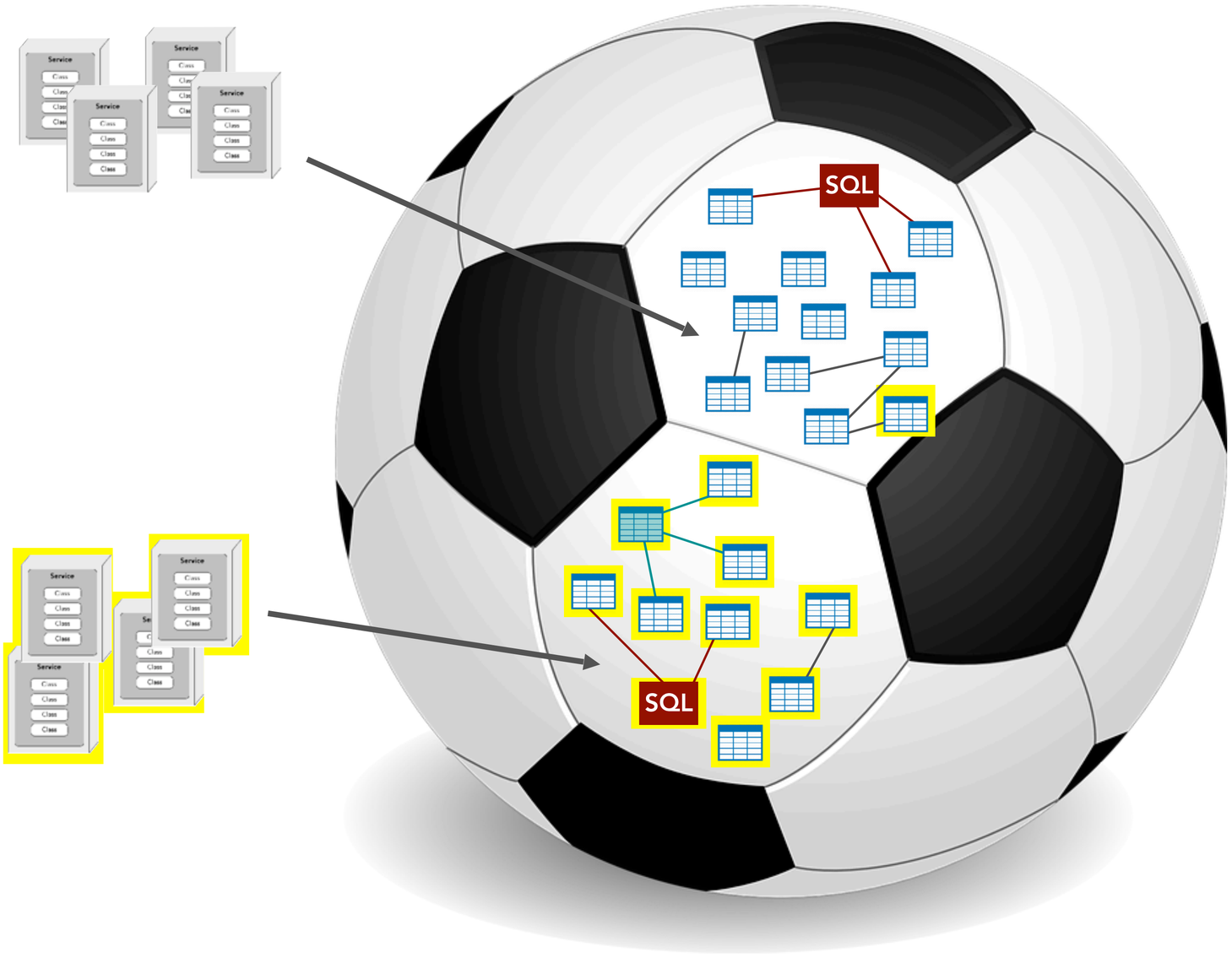
data domains



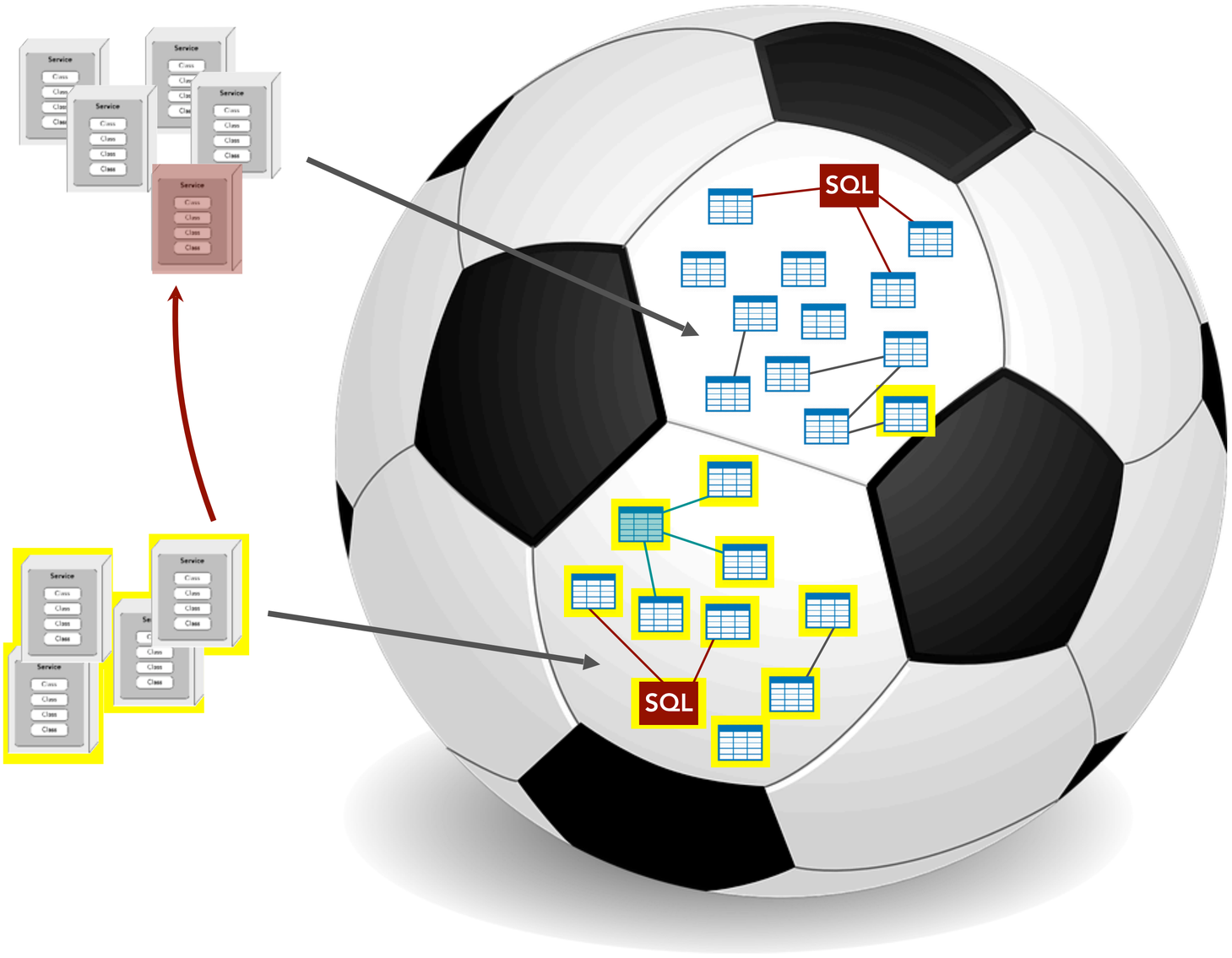
data domains



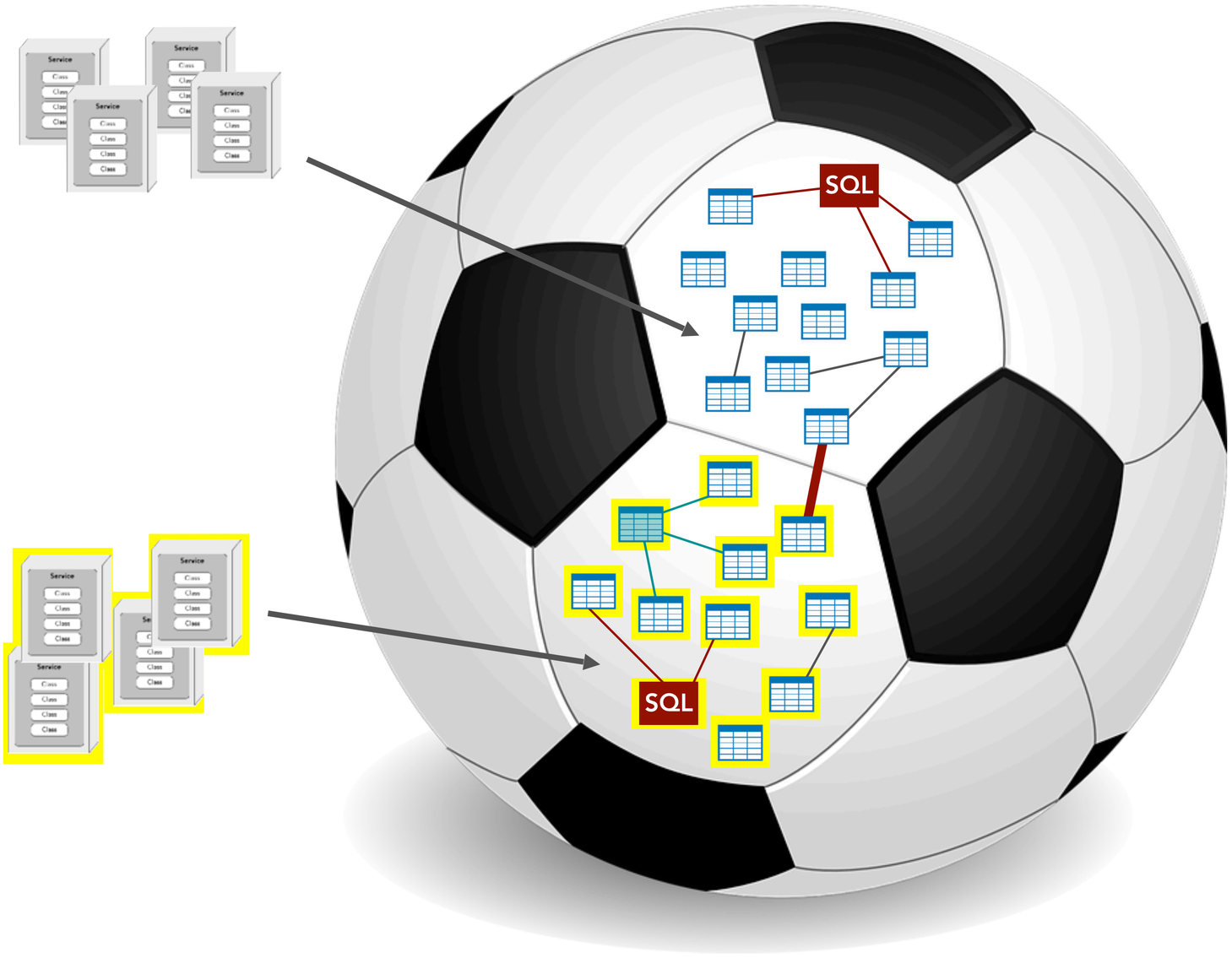
data domains



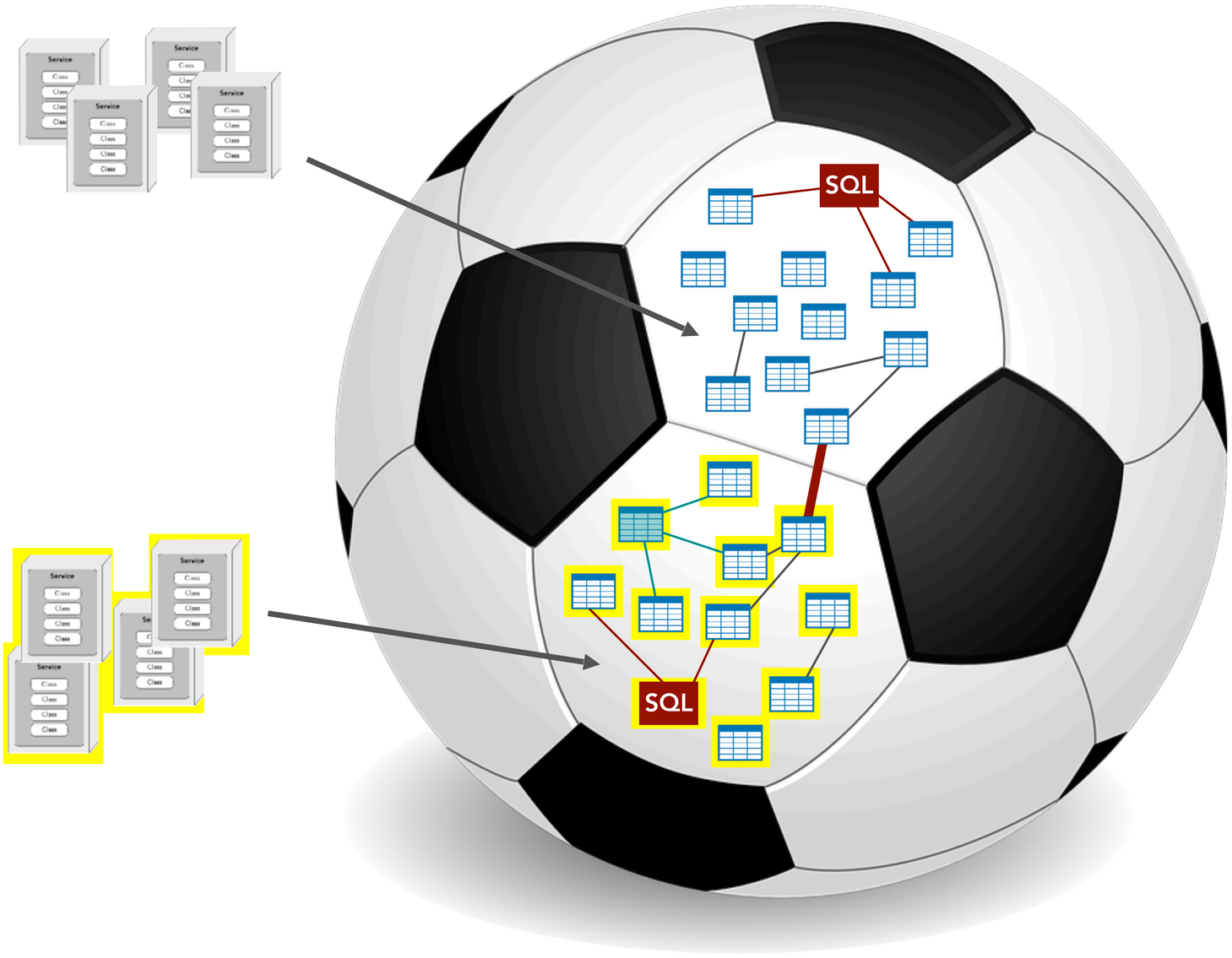
data domains



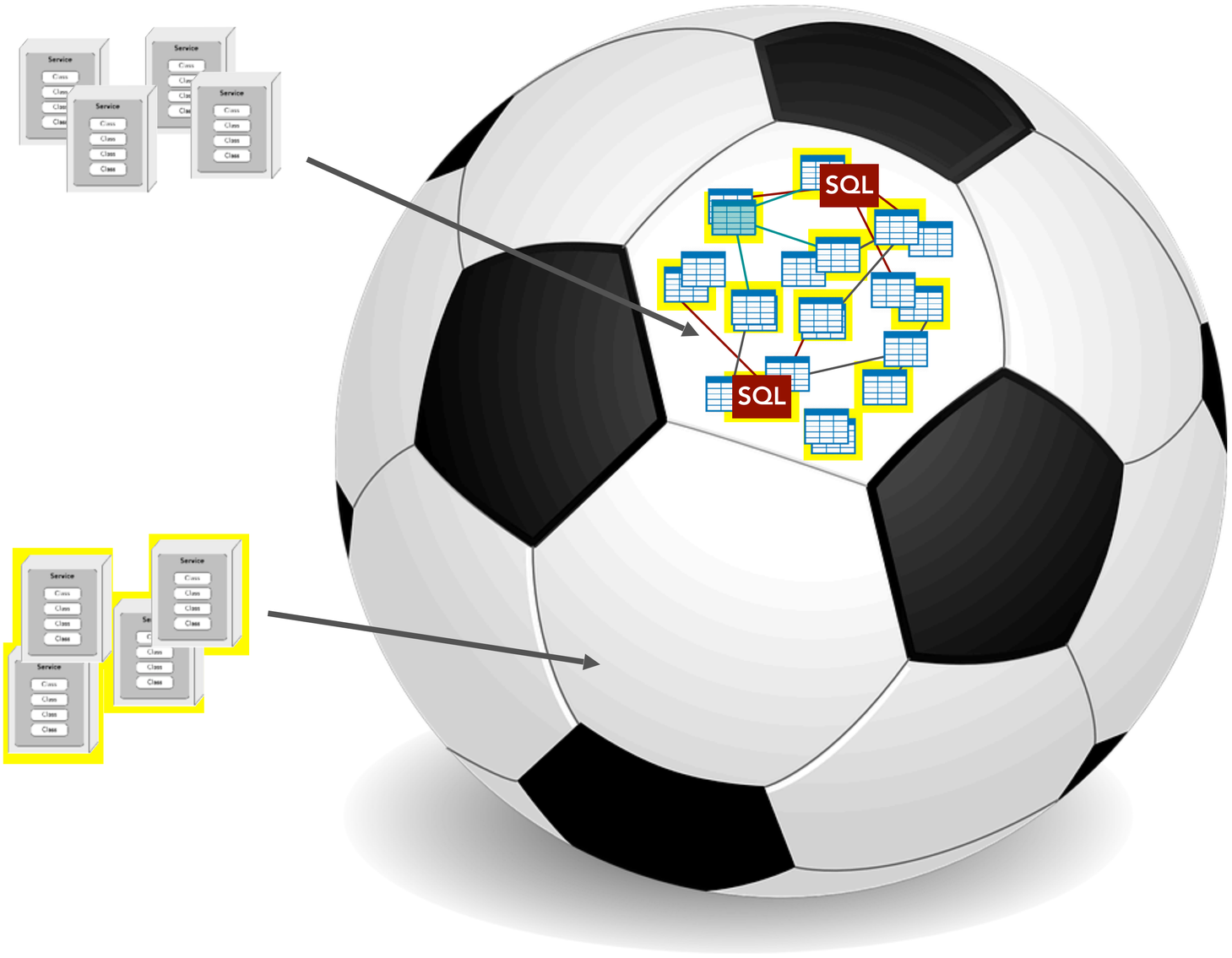
data domains



data domains



data domains

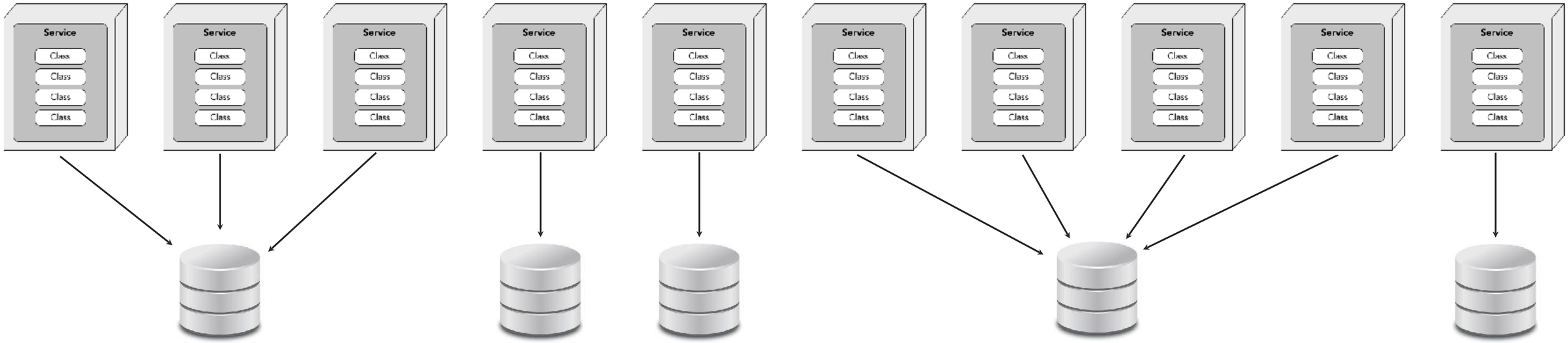


data domains

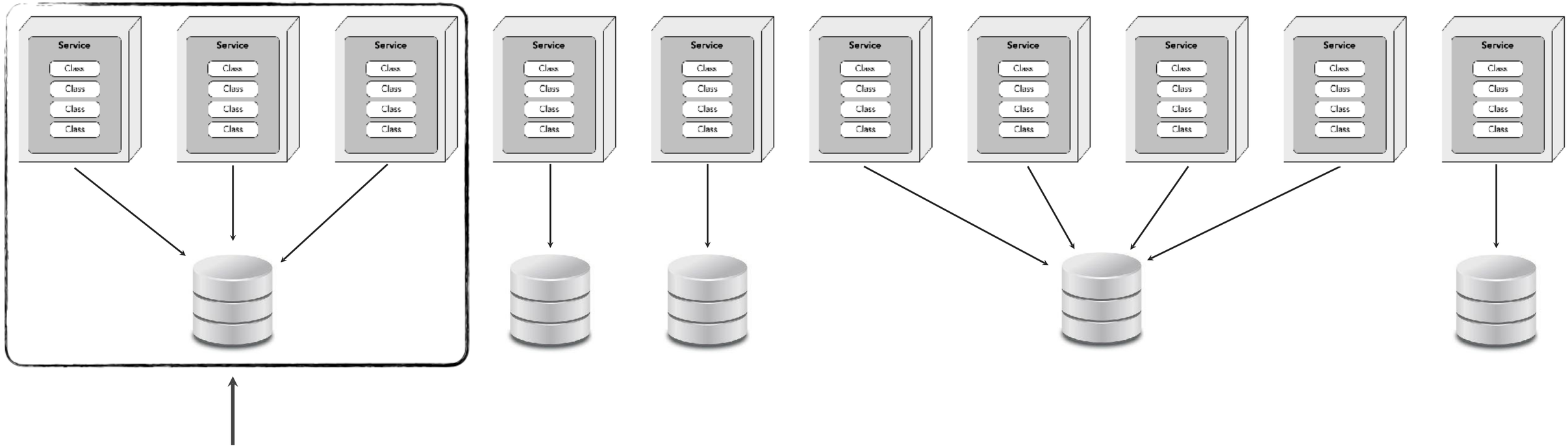
data domains

Defining a Broader Bounded Context

bounded context

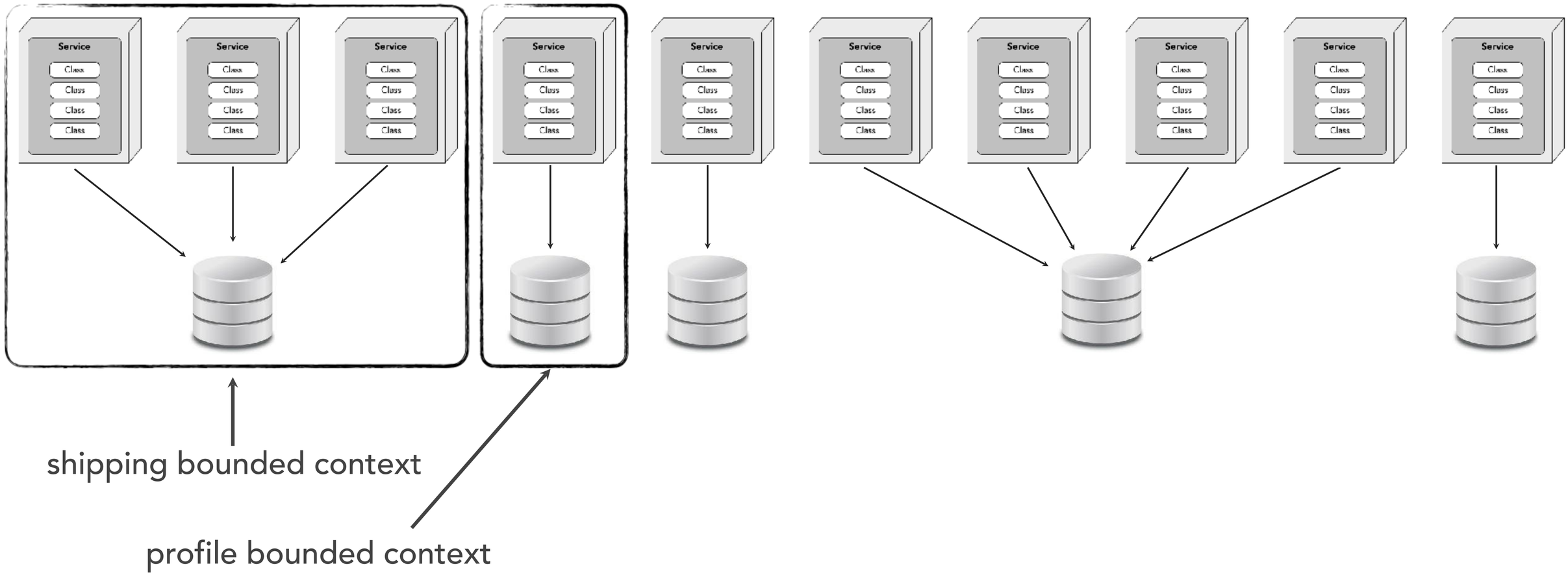


bounded context

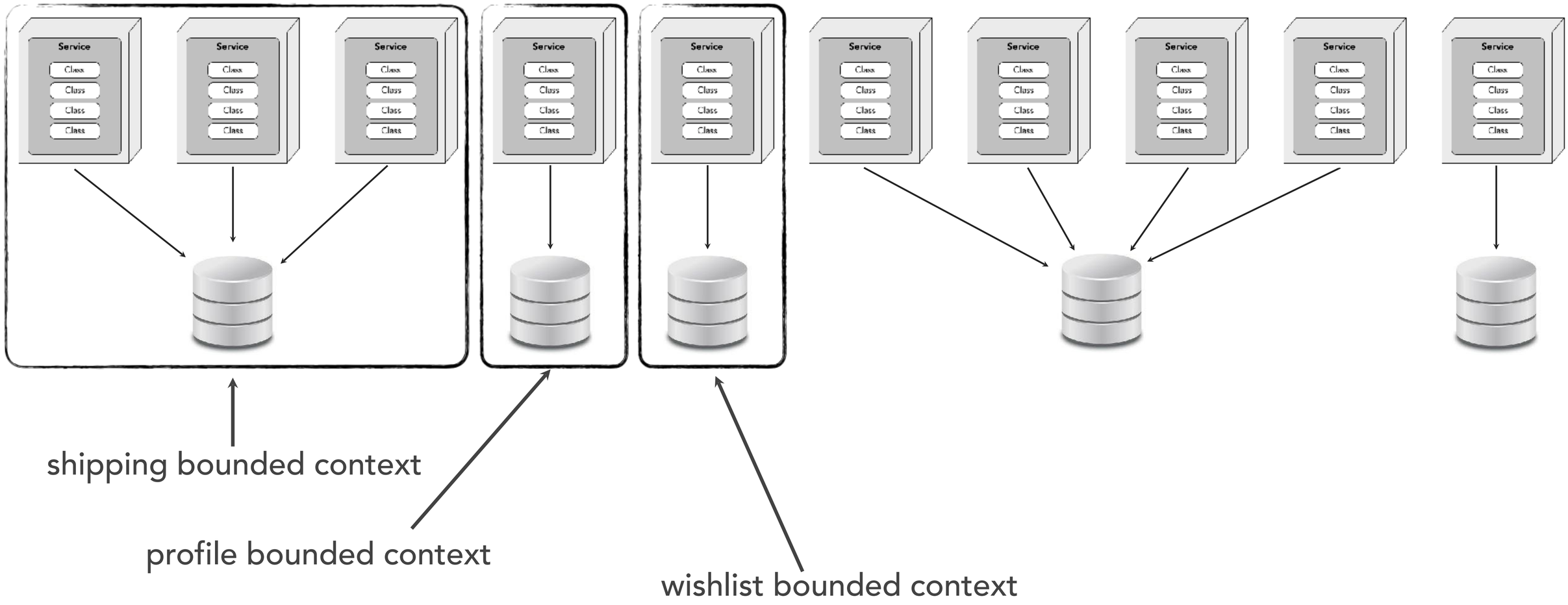


shipping bounded context

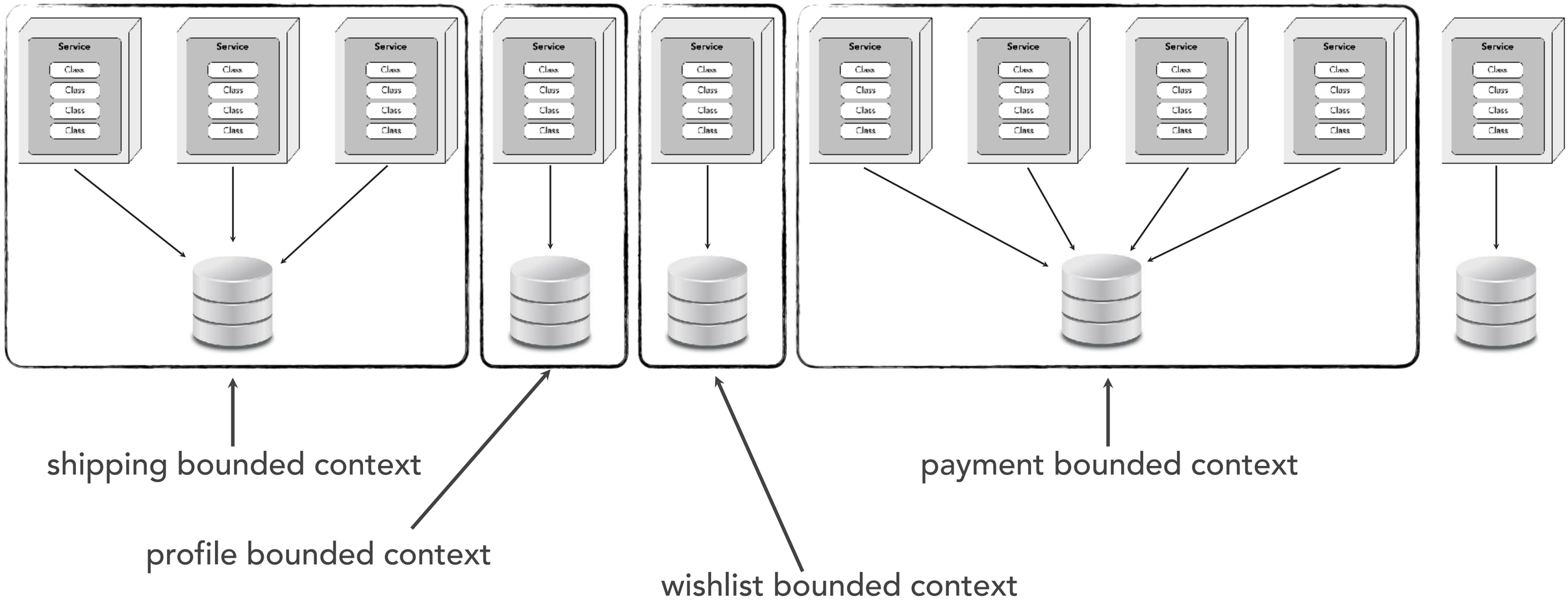
bounded context



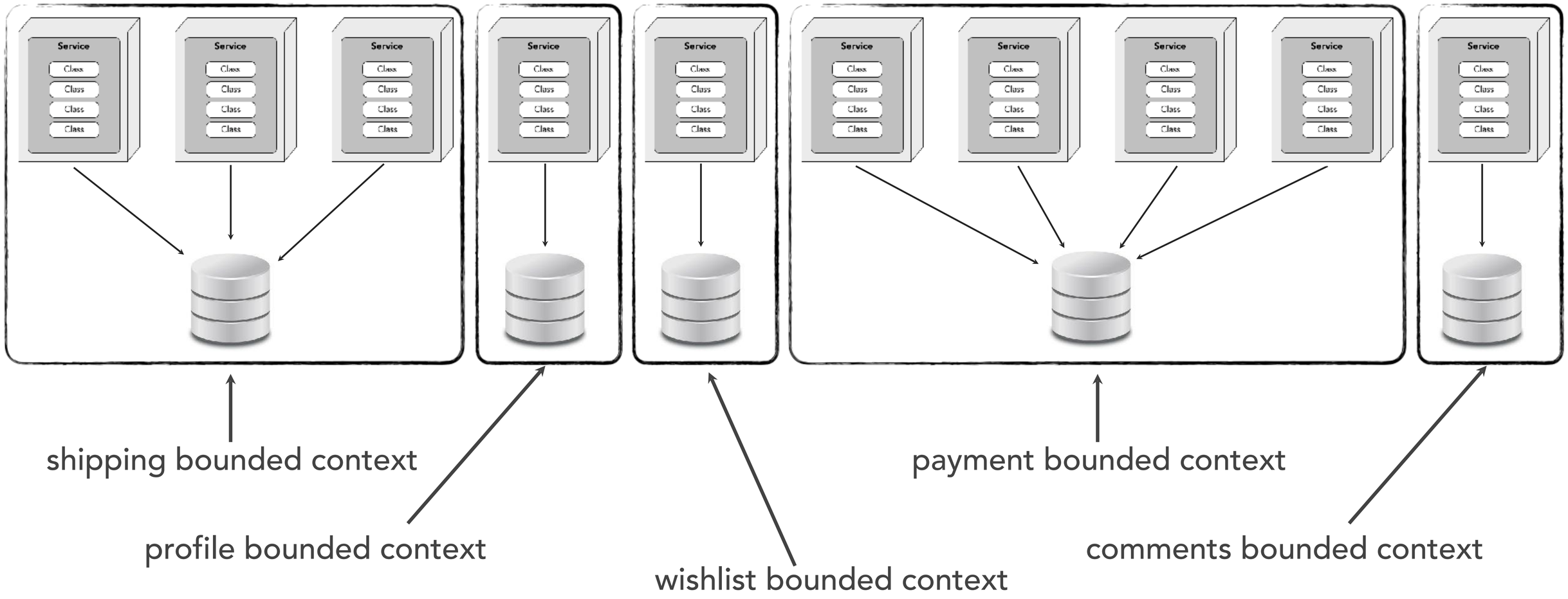
bounded context



bounded context



bounded context



bounded context

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface BoundedContext {
    public Context value();

    public enum Context {
        SHIPPING,
        PAYMENT,
        WISHLIST,
        COMMENTS,
        PROFILE
    }
}
```

bounded context

```
@ServiceEntrypoint
@FunctionalService
@BoundedContext(Context.PAYMENT)
public class CreditCardServiceAPI {
    ...
}
```

bounded context

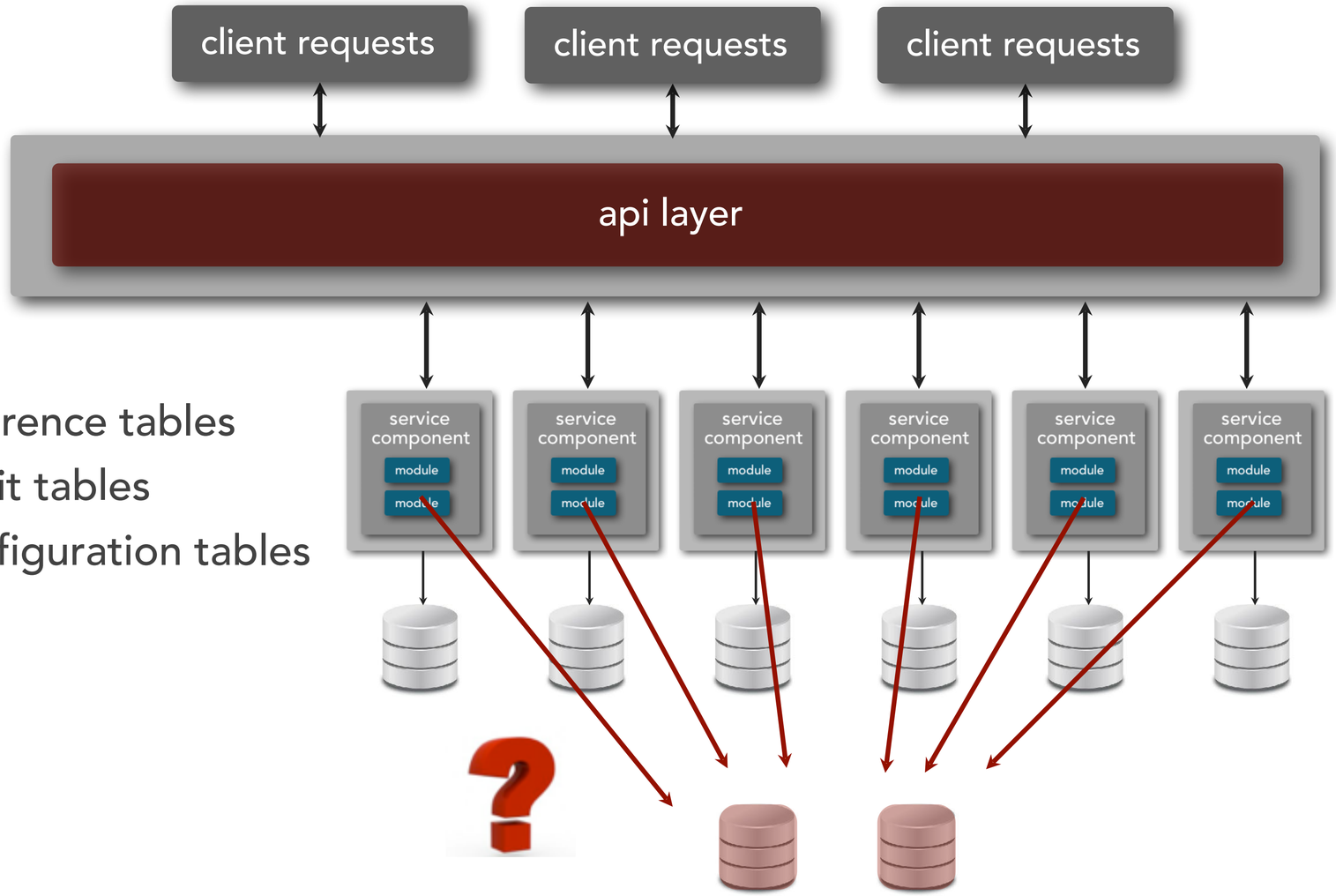
```
@ServiceEntrypoint
@FunctionalService
@BoundedContext(Context.PAYMENT)
public class CreditCardServiceAPI {
    ...
}

@ServiceEntrypoint
@FunctionalService
@BoundedContext(Context.PAYMENT)
public class GiftCardServiceAPI {
    ...
}
```

bounded context

```
@ServiceEntrypoint
@FunctionalService
@BoundedContext(Context.PAYMENT)
public class CreditCardServiceAPI {
    ...
}

@ServiceEntrypoint
@FunctionalService
@BoundedContext(Context.PAYMENT)
public class GiftCardServiceAPI {
    ...
}
```

Managing Common Data

managing common data

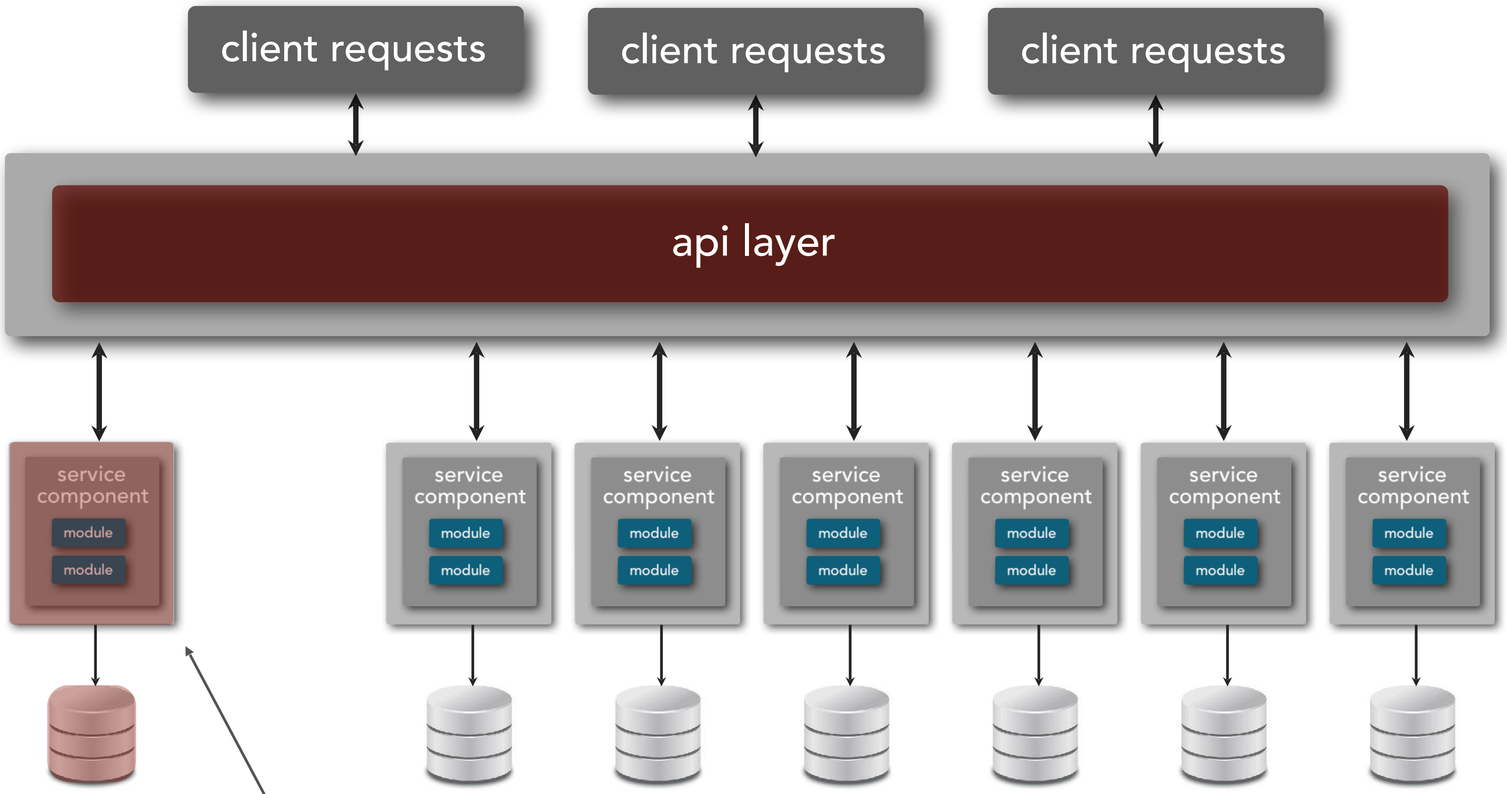


reference tables

audit tables

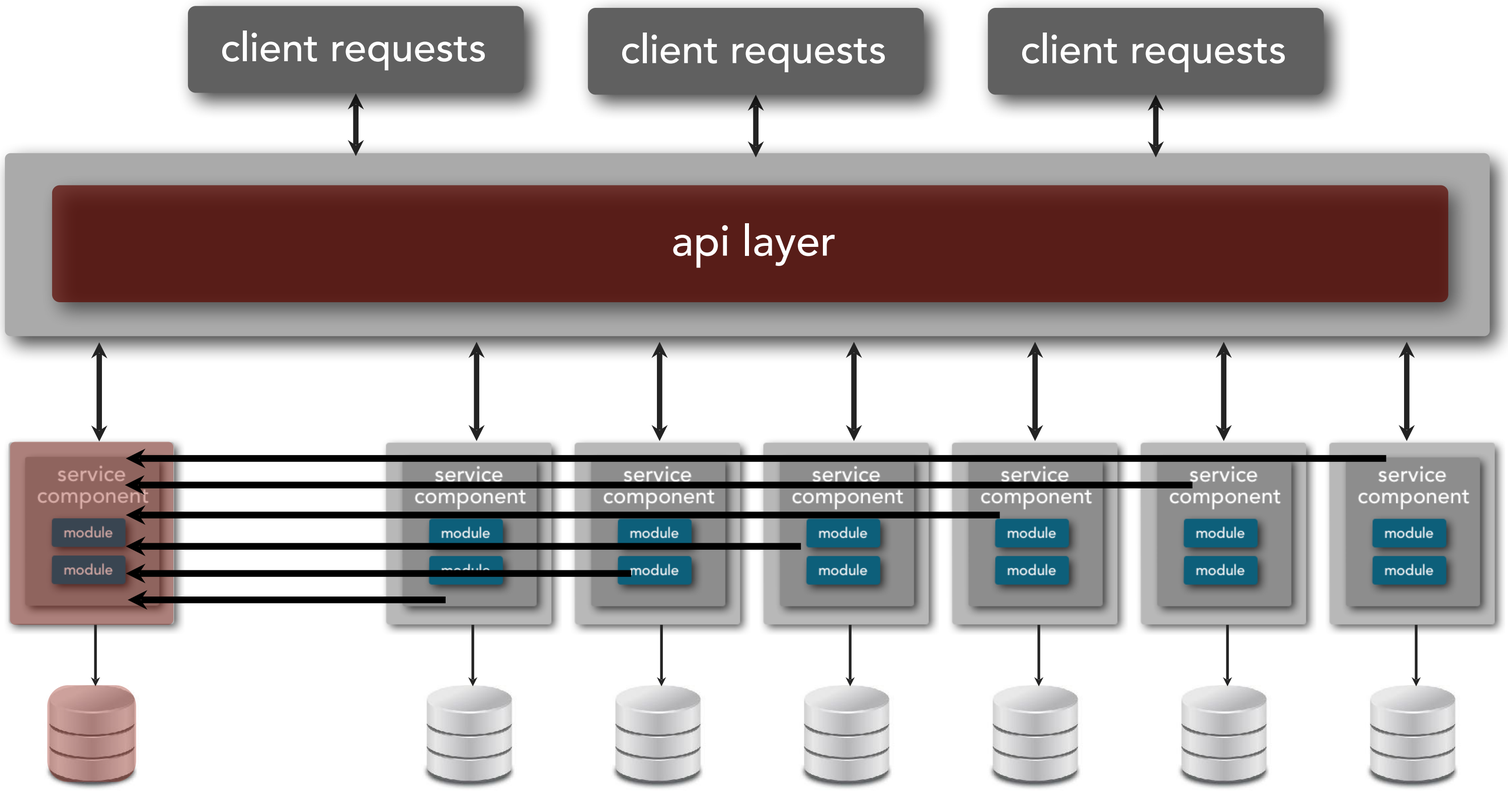
configuration tables

managing common data



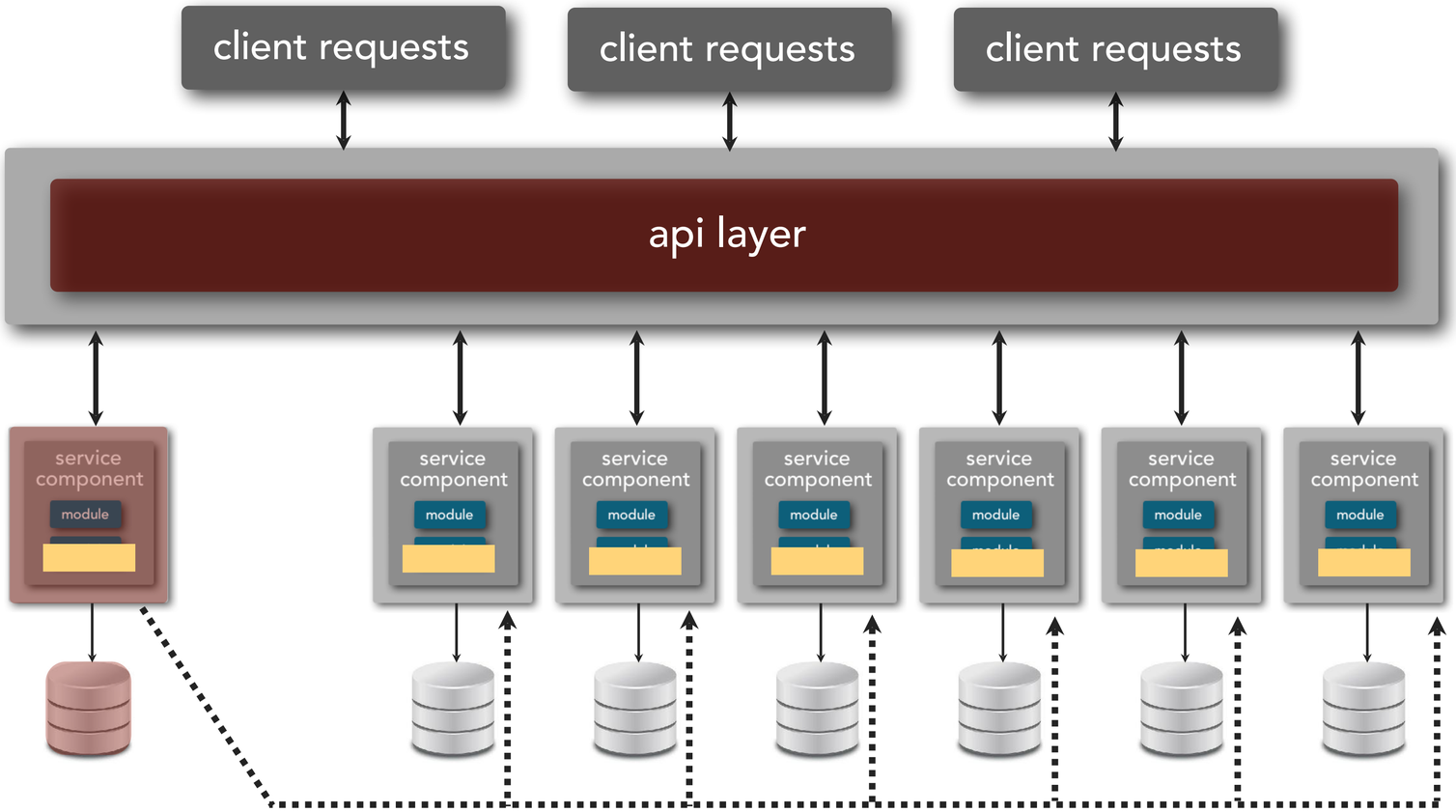
one service per common data domain
(ref, config, audit, etc.)

managing common data



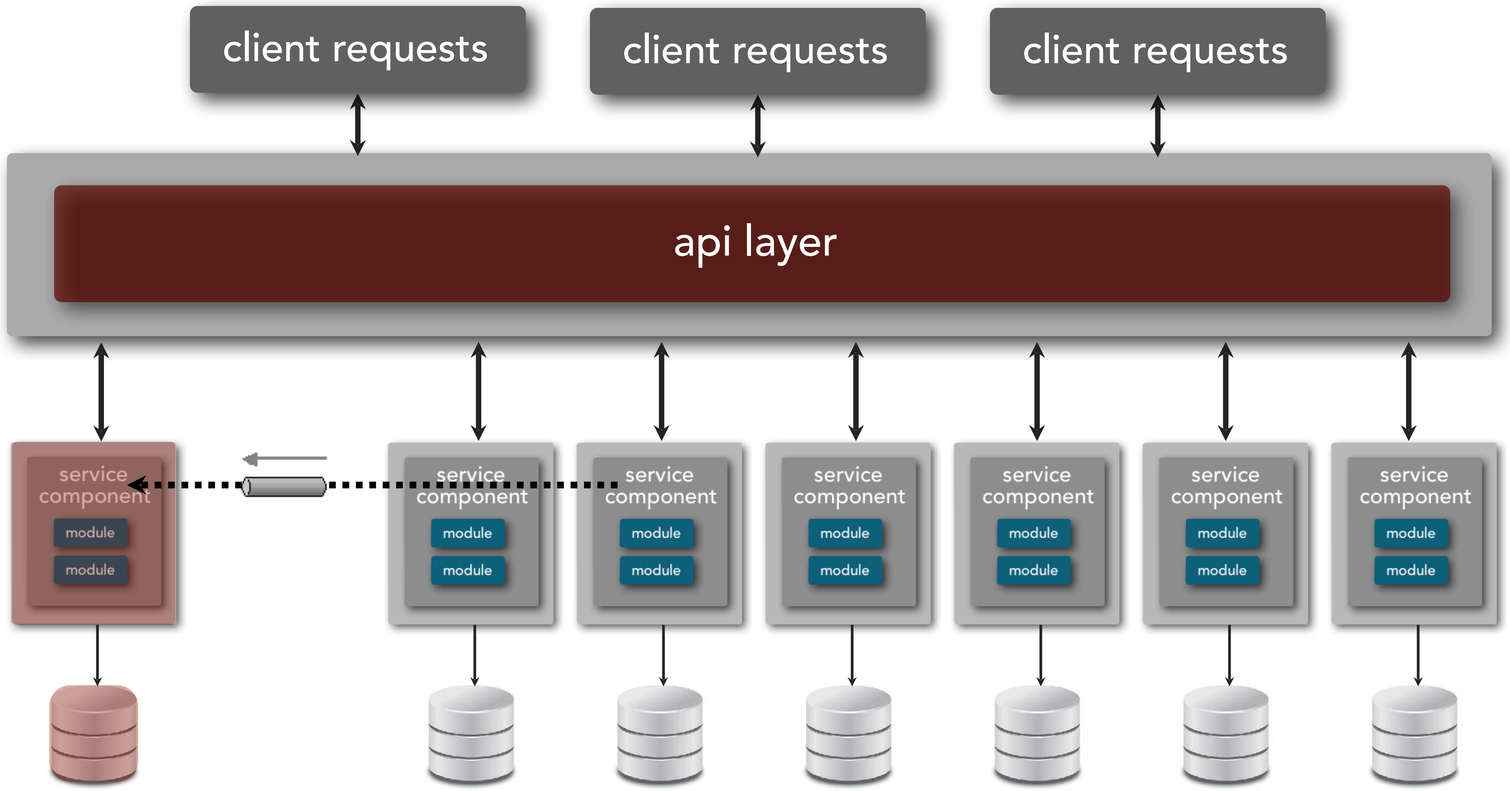
inter-service communication for reads

managing common data



replicated cache for reads

managing common data

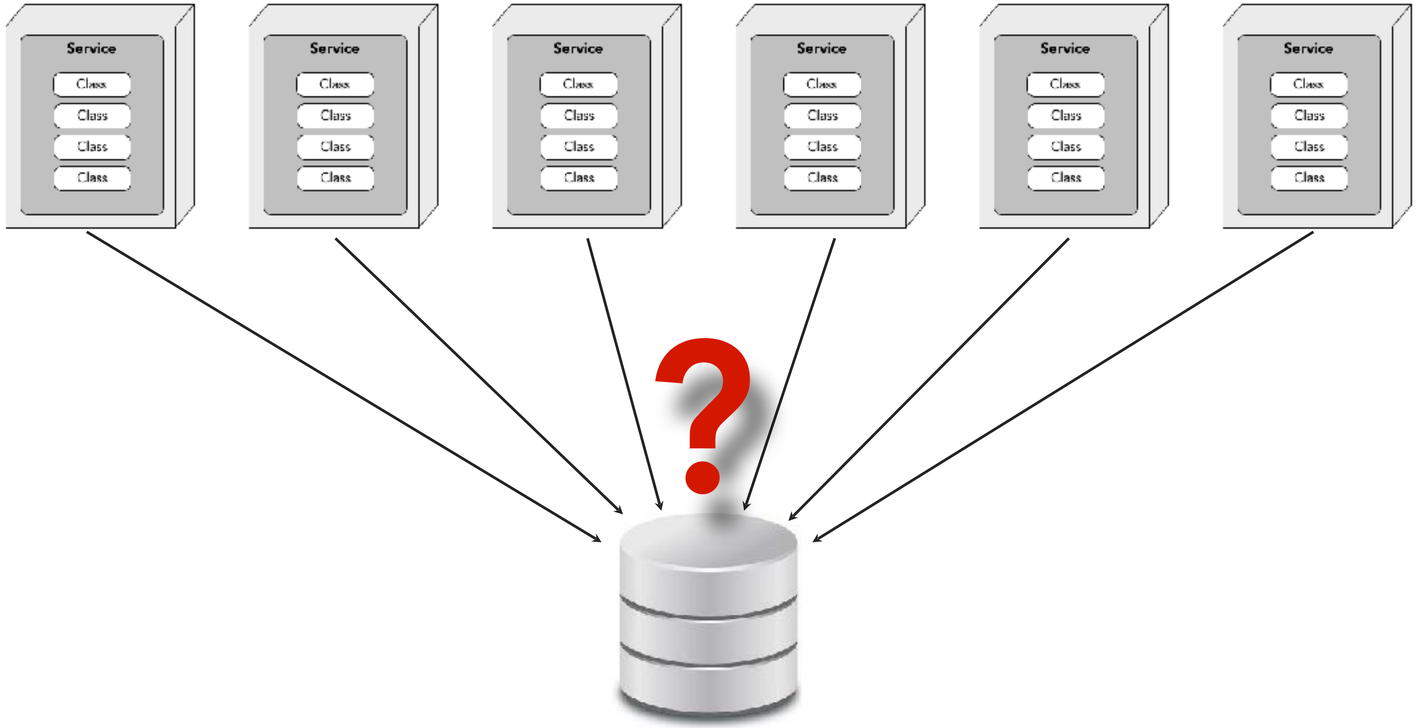


async messaging for writes

Data Decomposition Drivers

breaking apart data

"when should I consider breaking apart my data?"



breaking apart data

"when should I consider breaking apart my data?"

database granularity drivers

breaking apart data

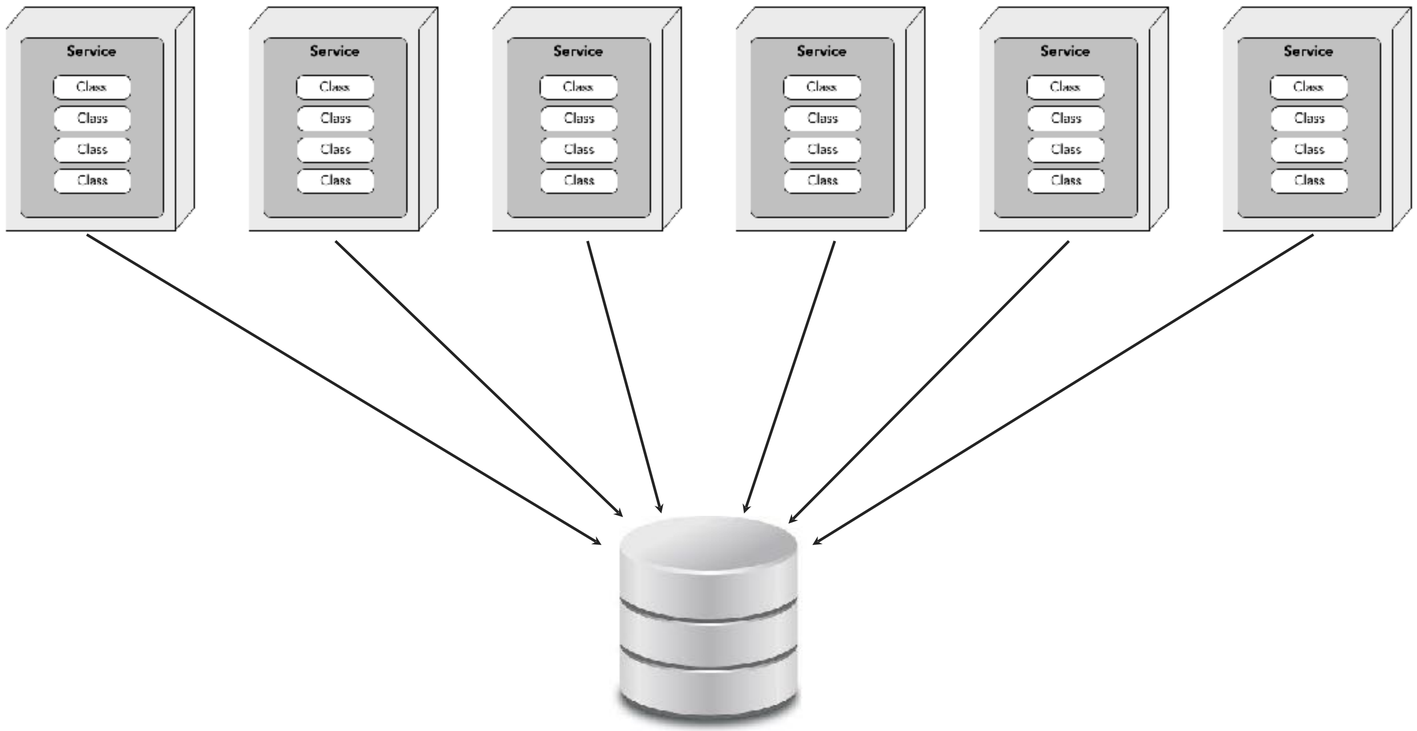
"when should I consider breaking apart my data?"

database granularity drivers

change
control

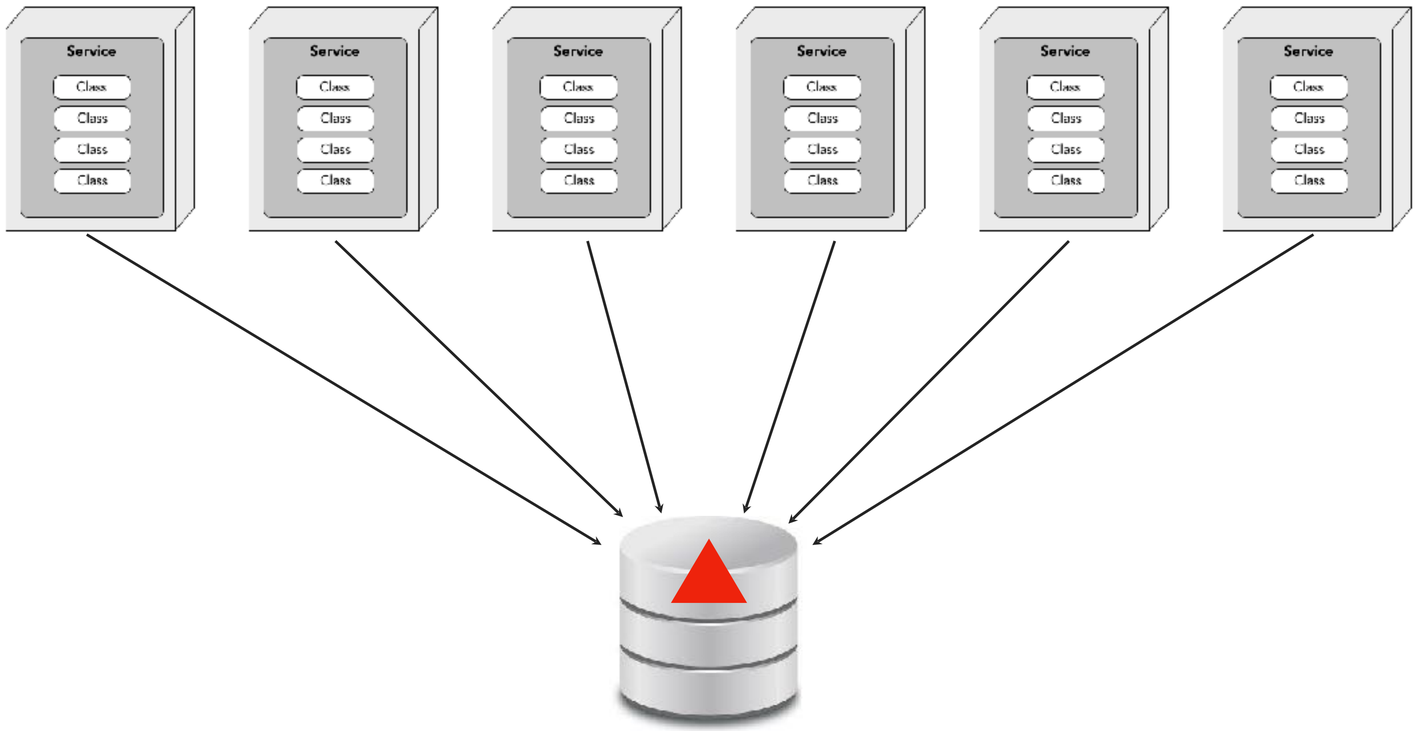
breaking apart data

change control



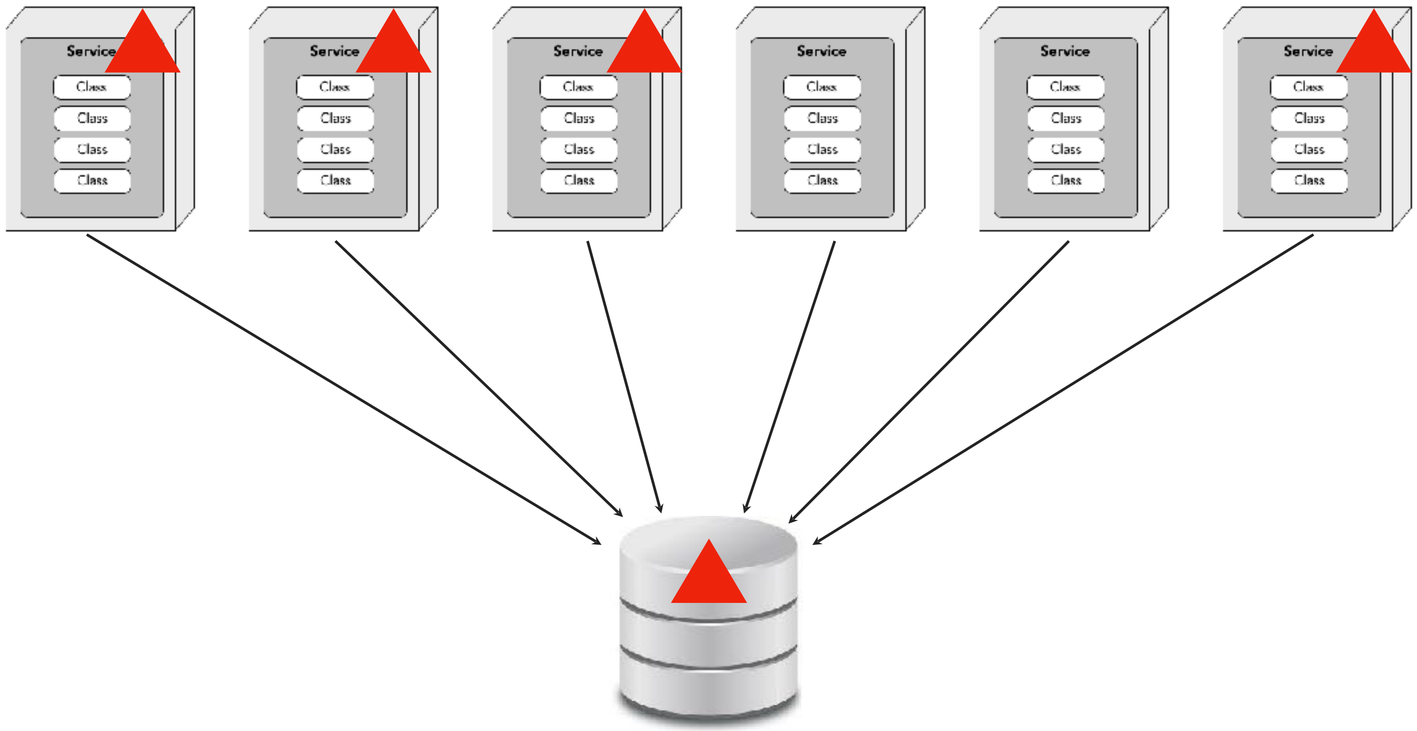
breaking apart data

change control



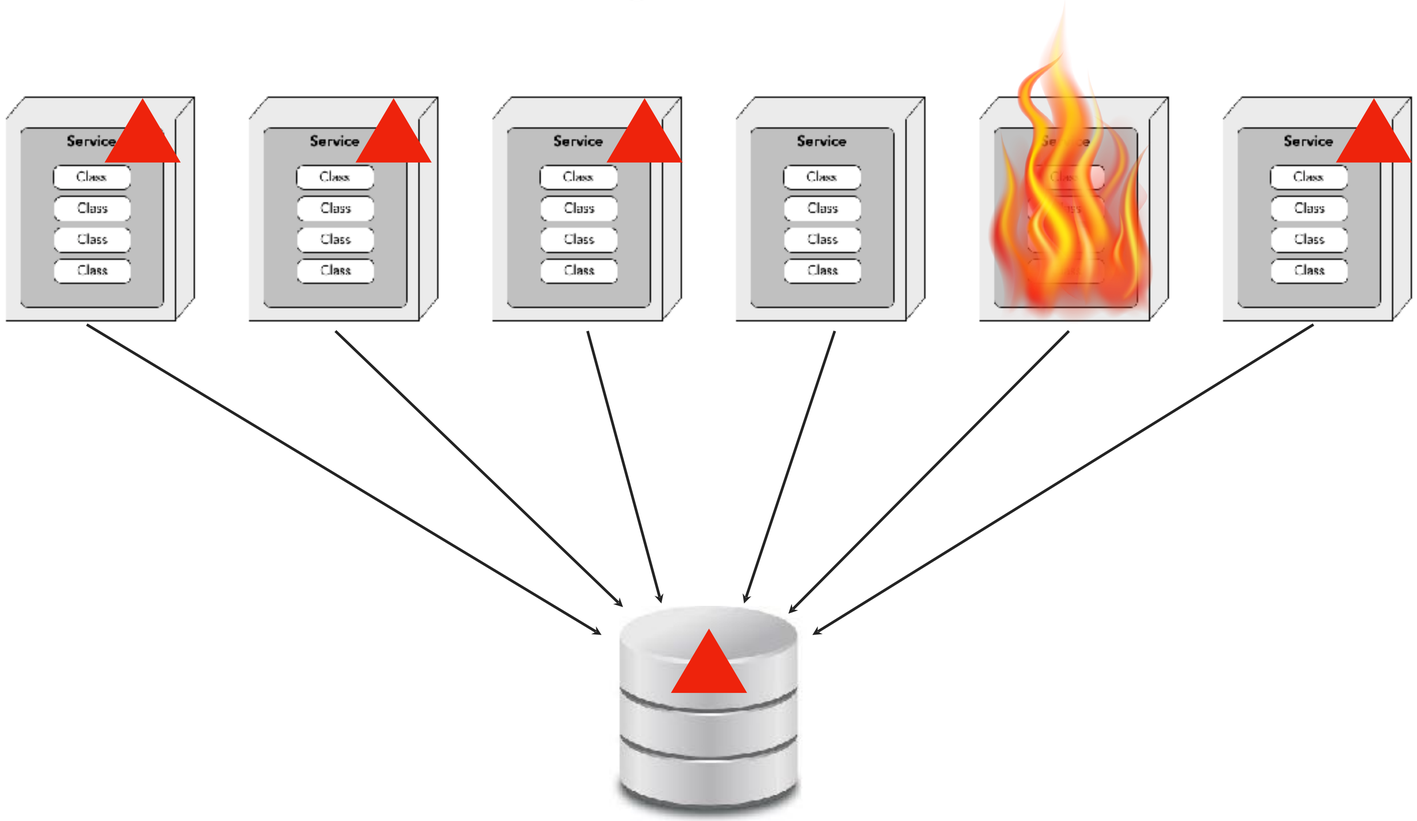
breaking apart data

change control



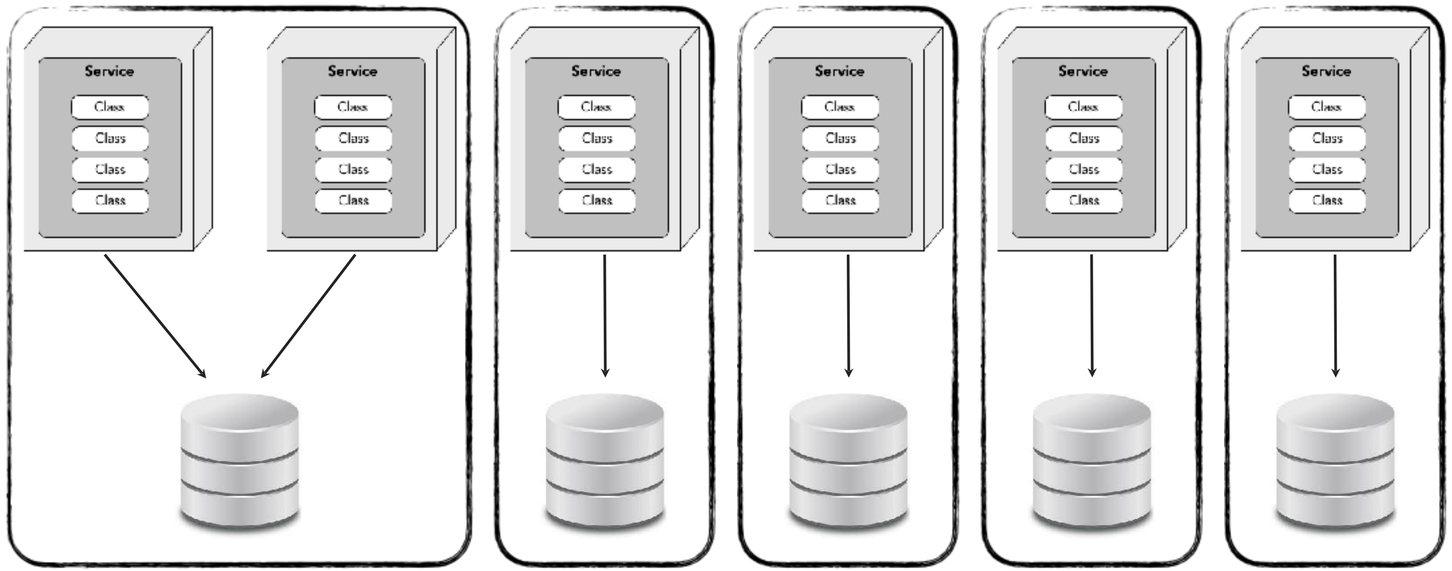
breaking apart data

change control



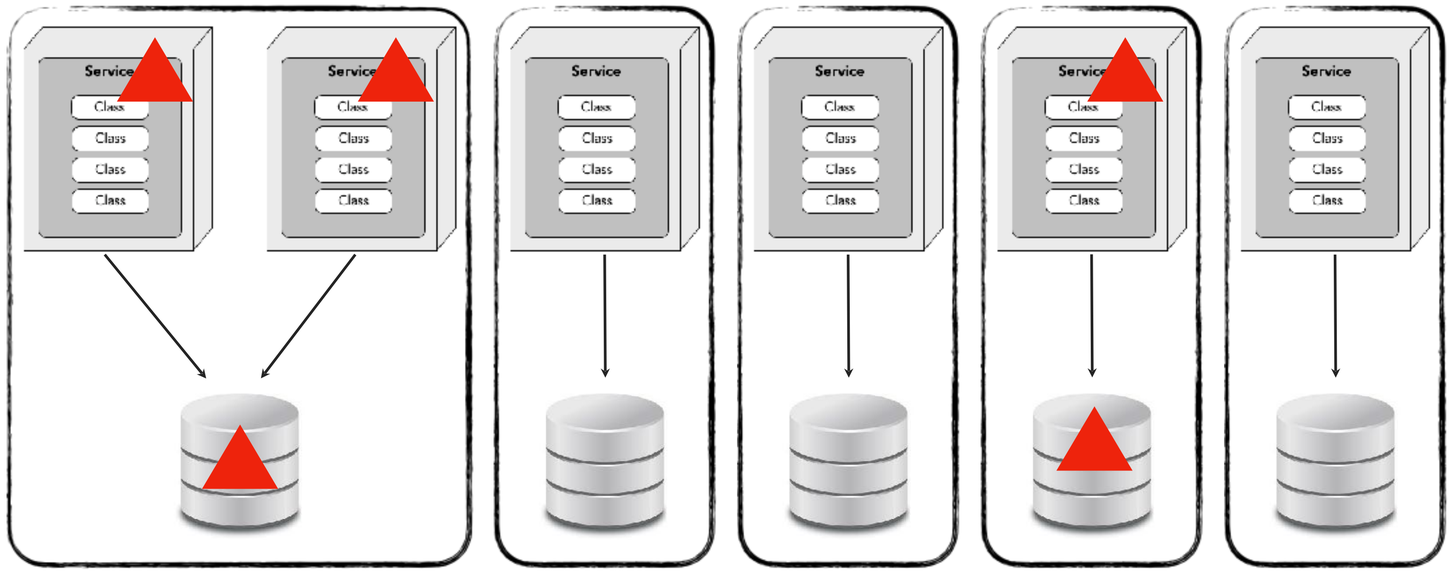
breaking apart data

change control



breaking apart data

change control



breaking apart data

"when should I consider breaking apart my data?"

database granularity drivers

change
control

breaking apart data

"when should I consider breaking apart my data?"

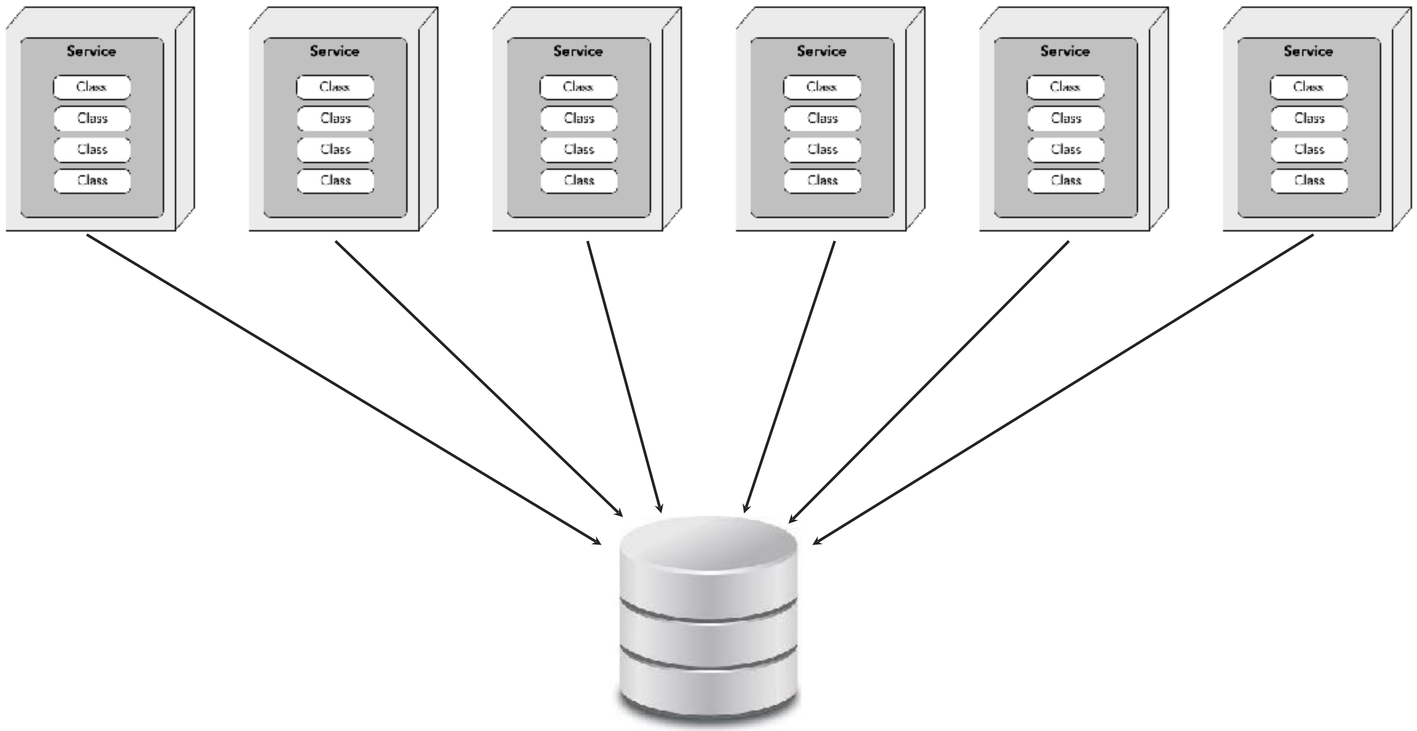
database granularity drivers

change control

database connections

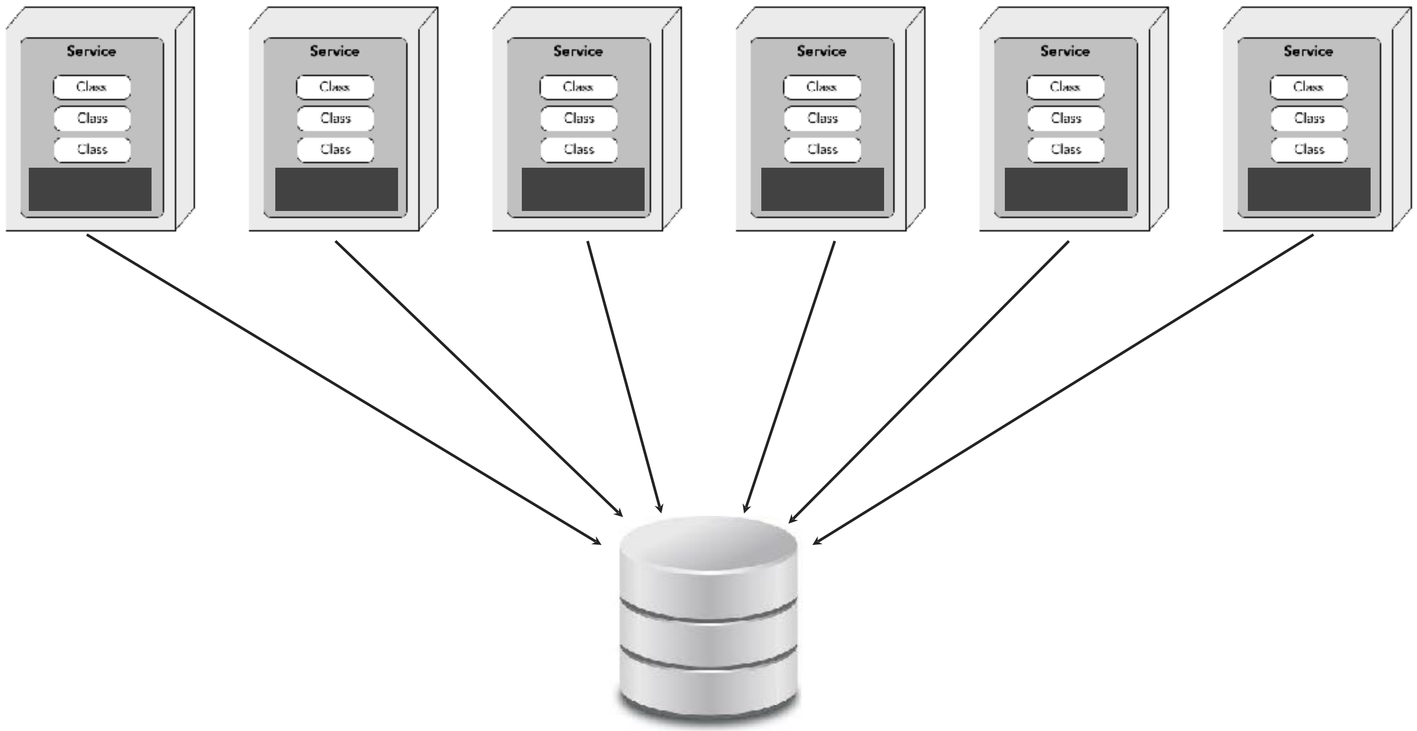
breaking apart data

database connections



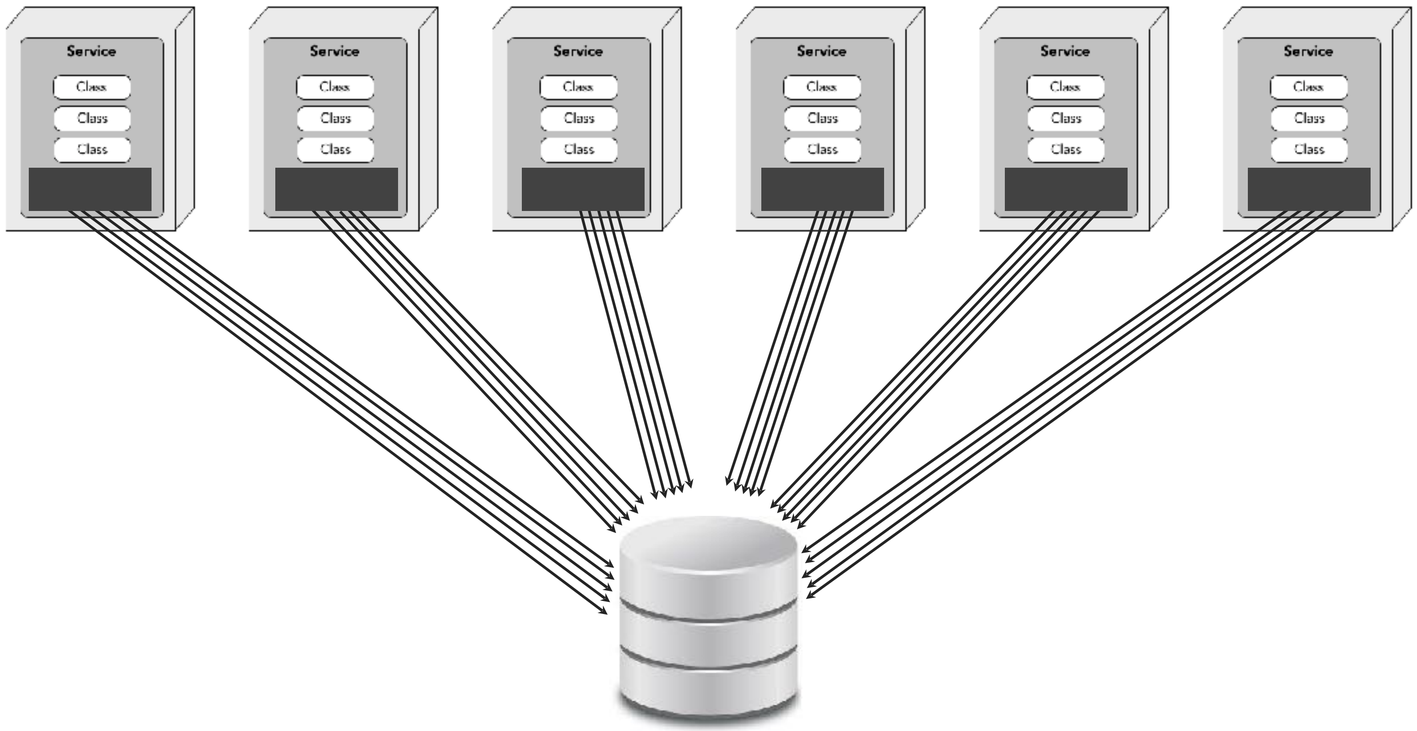
breaking apart data

database connections



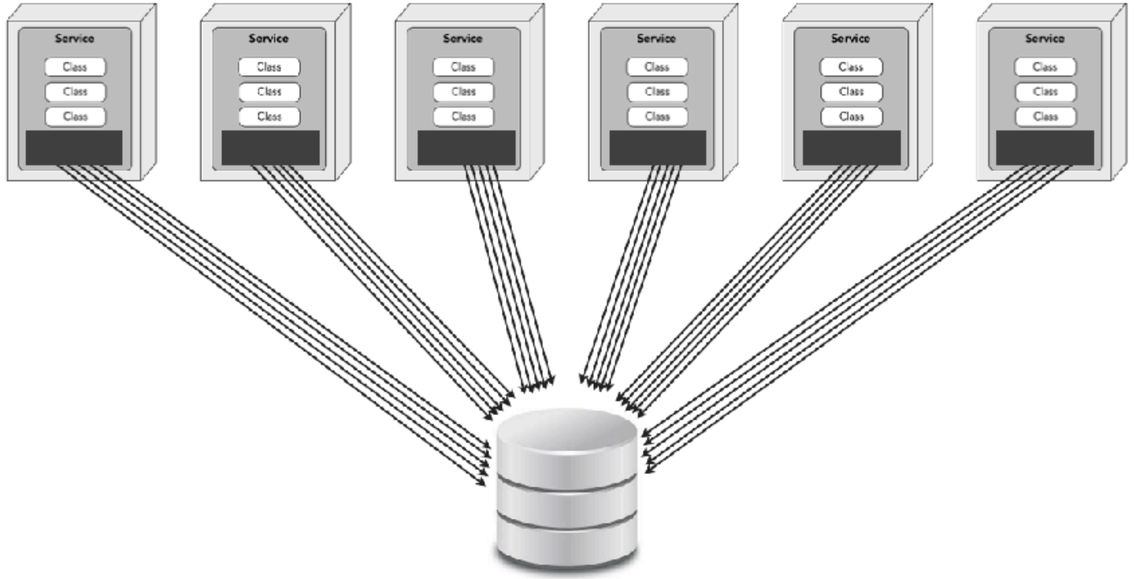
breaking apart data

database connections



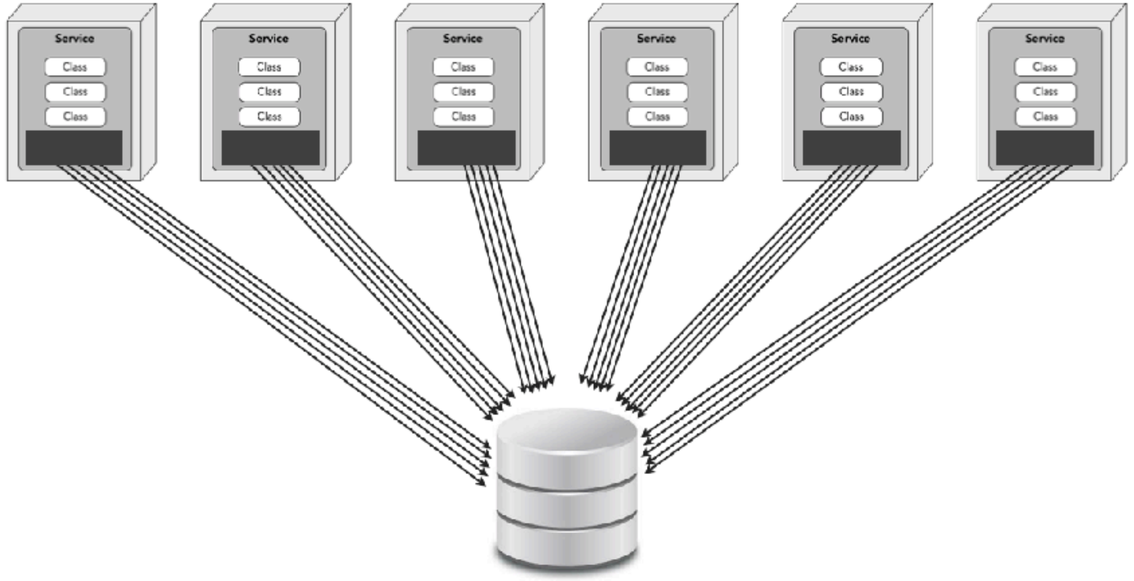
breaking apart data

database connections



breaking apart data

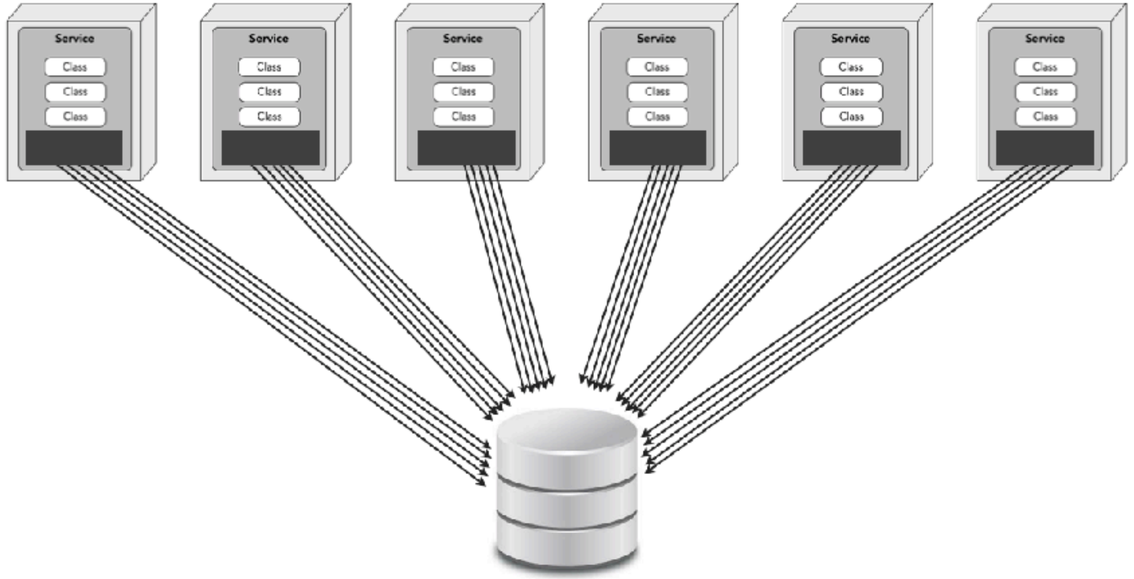
database connections



monolithic application: 200 connections

breaking apart data

database connections

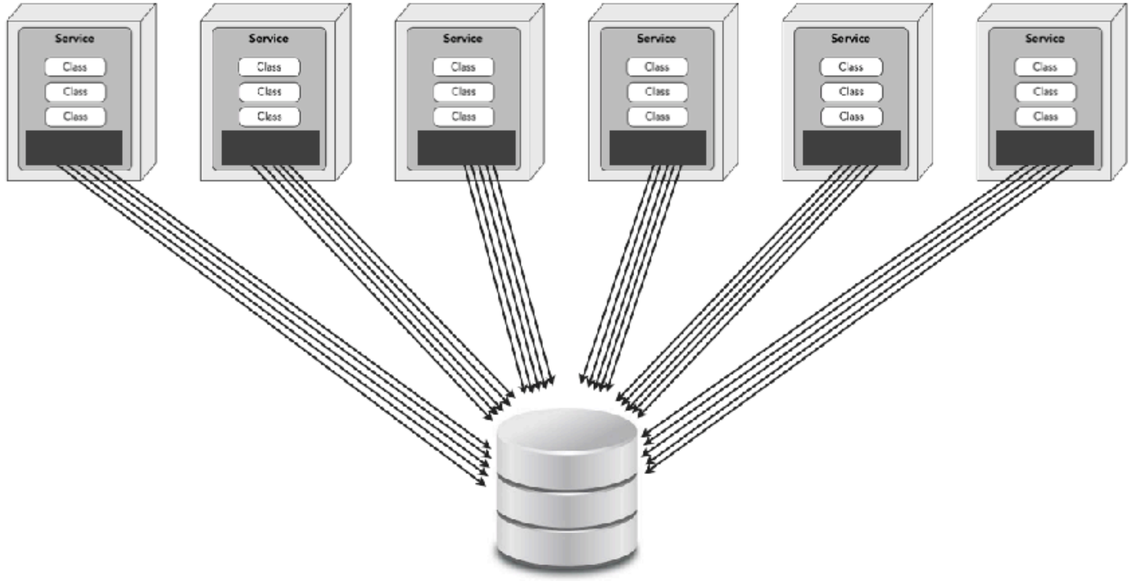


monolithic application: 200 connections

distributed services: 50

breaking apart data

database connections



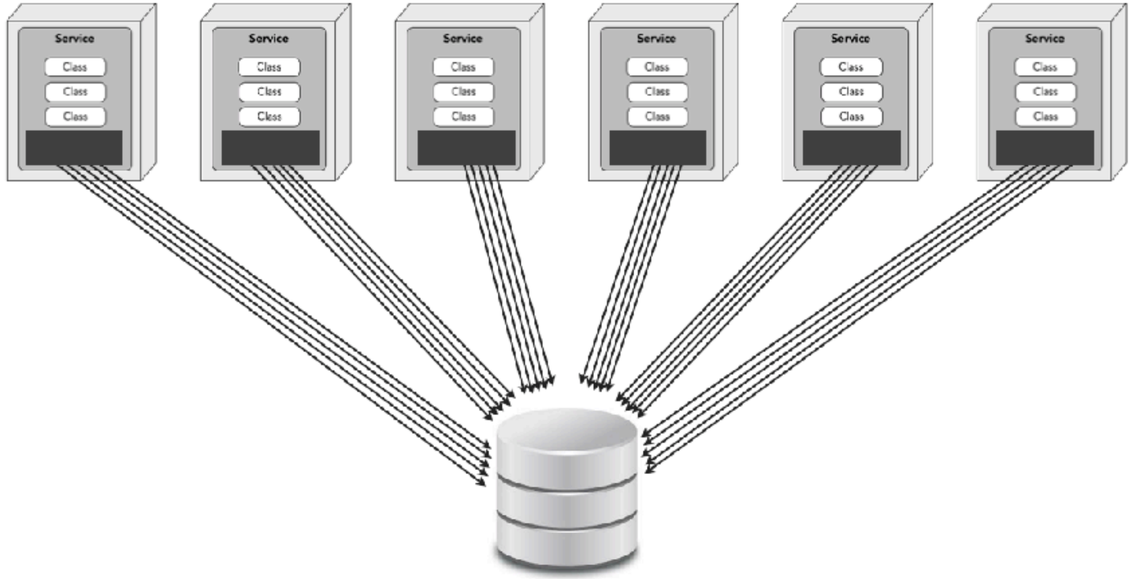
monolithic application: 200 connections

distributed services: 50

connections per service: 10

breaking apart data

database connections



monolithic application: 200 connections

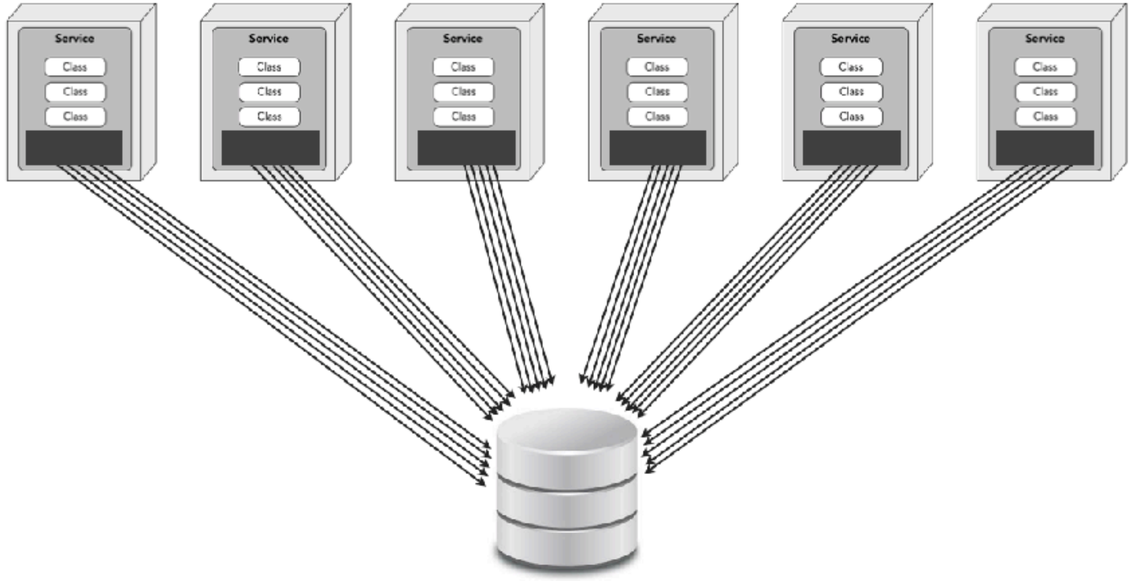
distributed services: 50

connections per service: 10

service instances (min): 2

breaking apart data

database connections



monolithic application: 200 connections

distributed services: 50

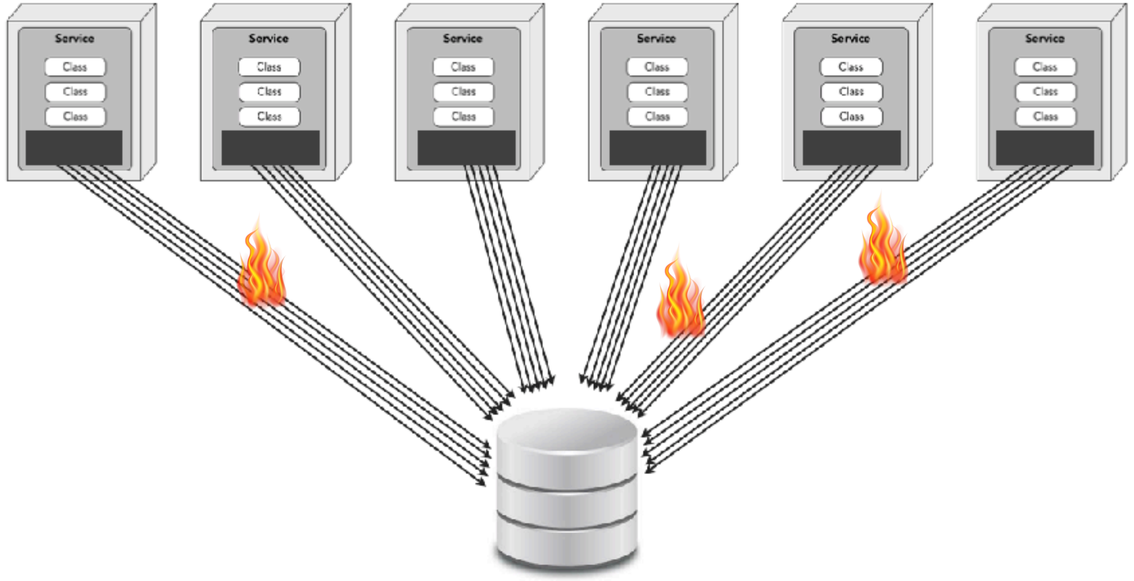
connections per service: 10

service instances (min): 2

distributed connections: 1000 connections

breaking apart data

database connections



monolithic application: 200 connections

distributed services: 50

connections per service: 10

service instances (min): 2

distributed connections: 1000 connections

breaking apart data

"when should I consider breaking apart my data?"

database granularity drivers

change control

database connections

breaking apart data

"when should I consider breaking apart my data?"

database granularity drivers

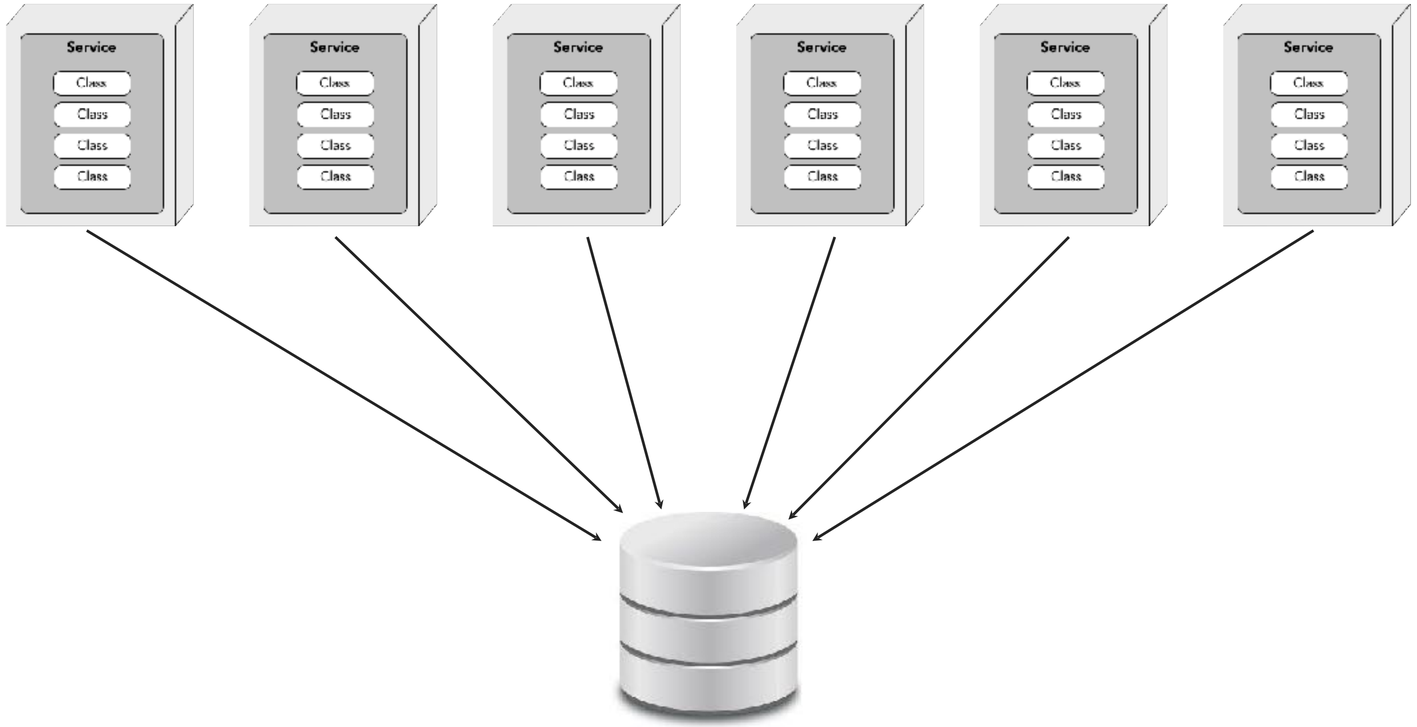
change control

database connections

database scalability

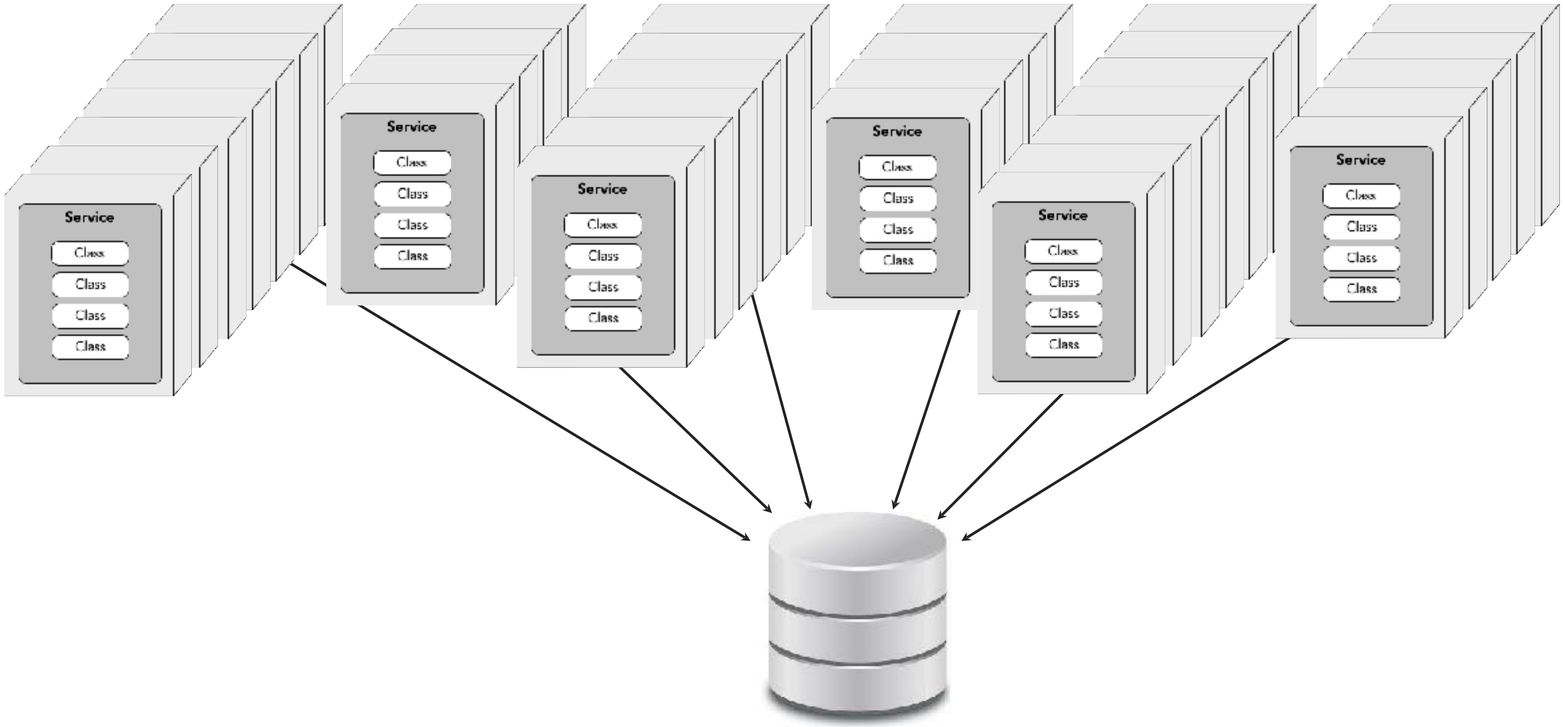
breaking apart data

database scalability



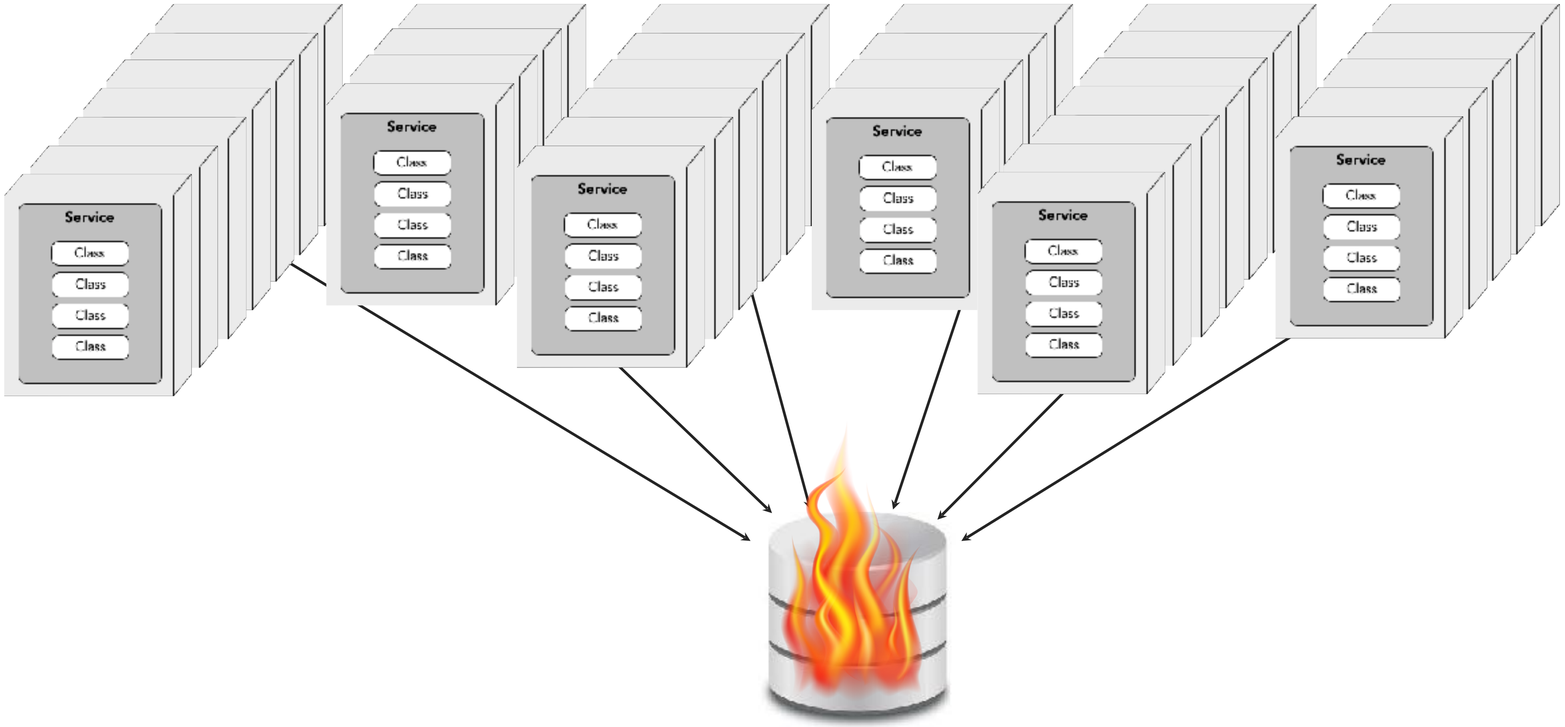
breaking apart data

database scalability



breaking apart data

database scalability



breaking apart data

"when should I consider breaking apart my data?"

database granularity drivers

change control

database connections

database scalability

breaking apart data

"when should I consider breaking apart my data?"

database granularity drivers

change control

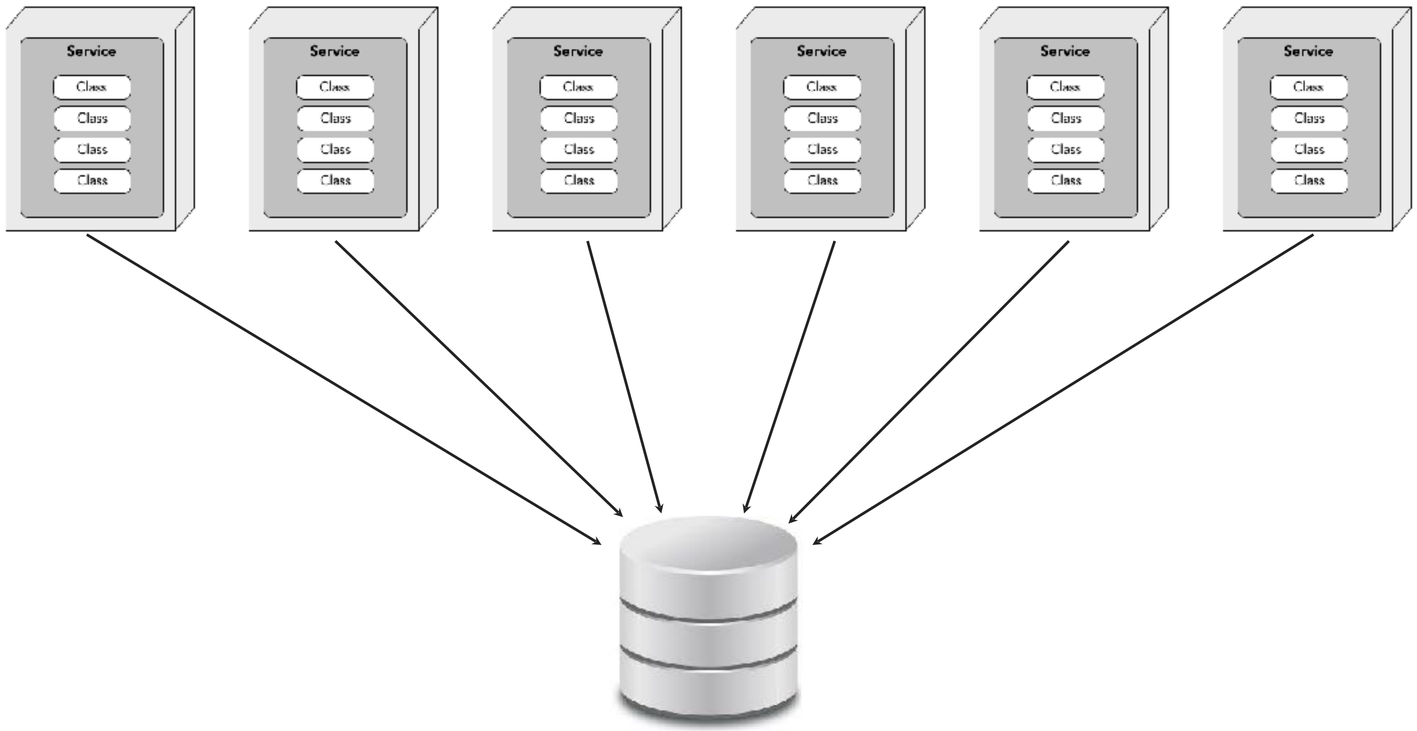
database connections

database scalability

fault tolerance

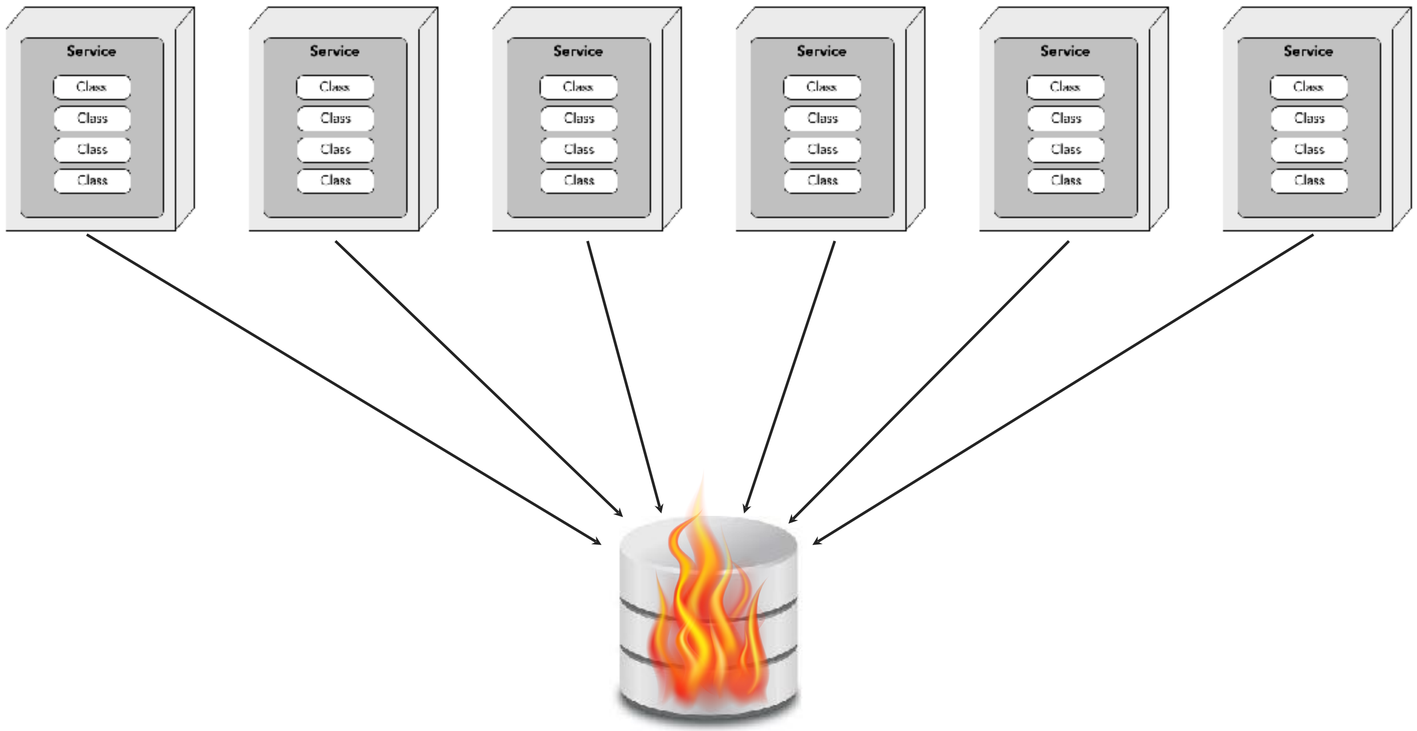
breaking apart data

fault tolerance



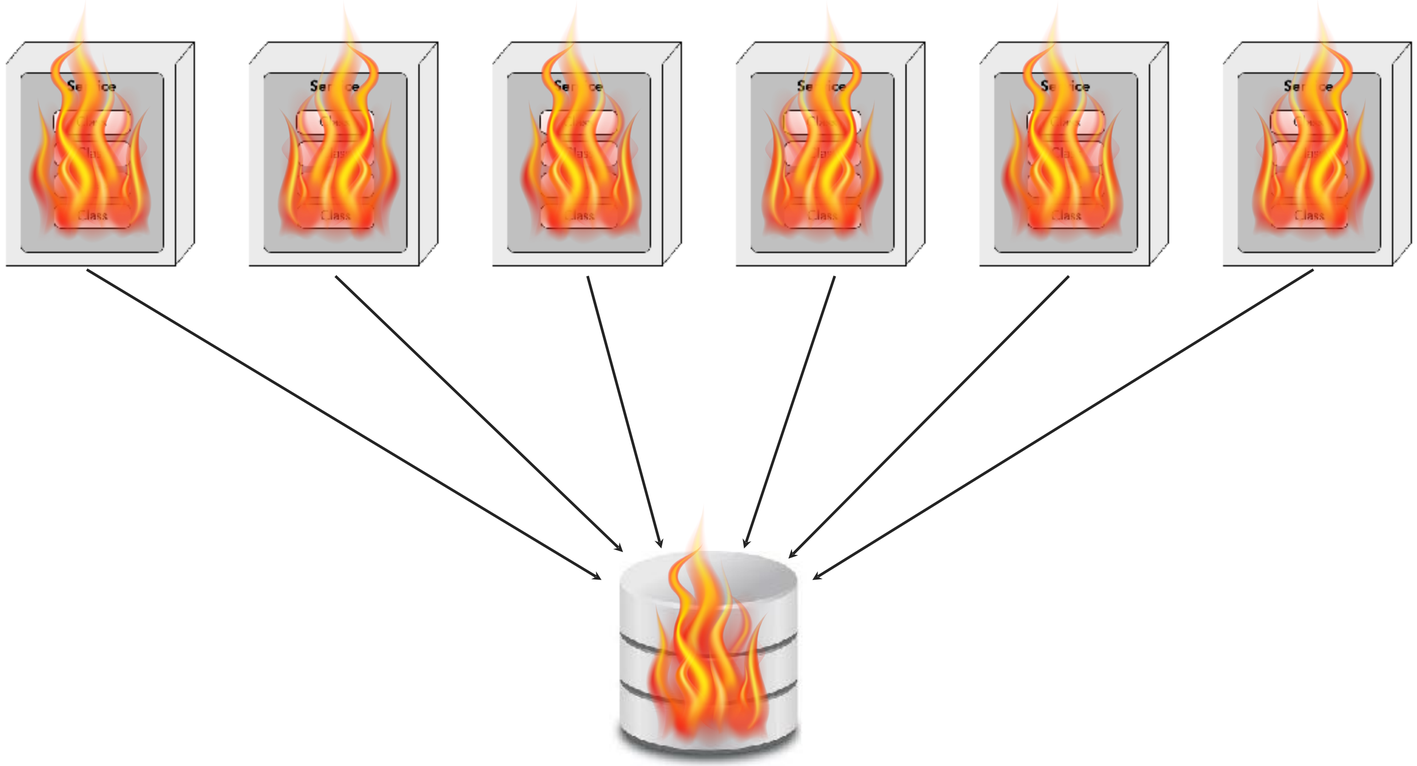
breaking apart data

fault tolerance



breaking apart data

fault tolerance



breaking apart data

"when should I consider breaking apart my data?"

database granularity drivers

change control

database connections

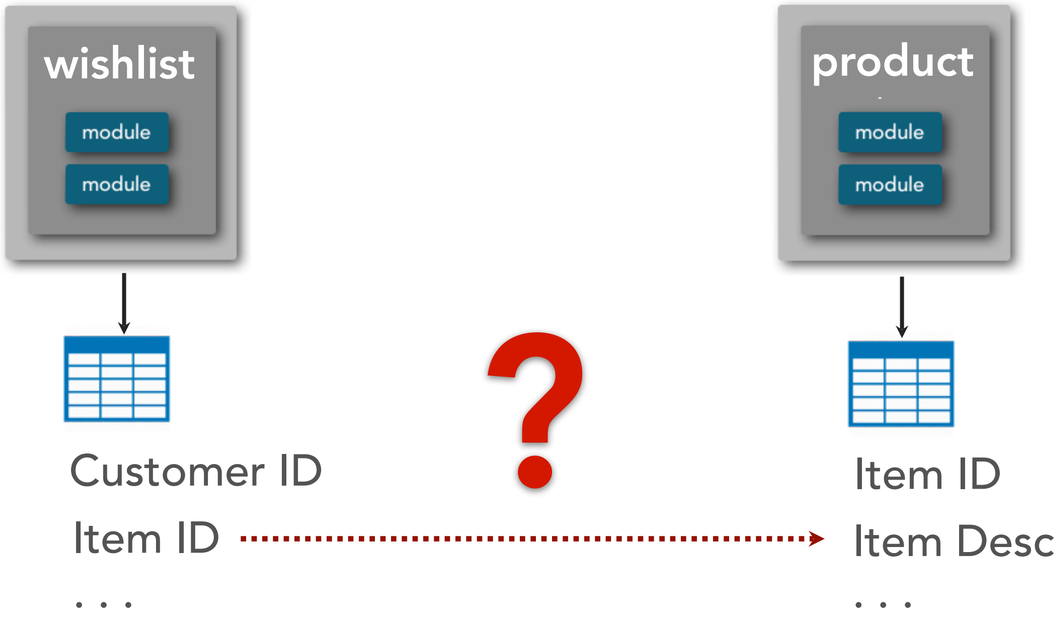
database scalability

fault tolerance

Data Ownership and Access

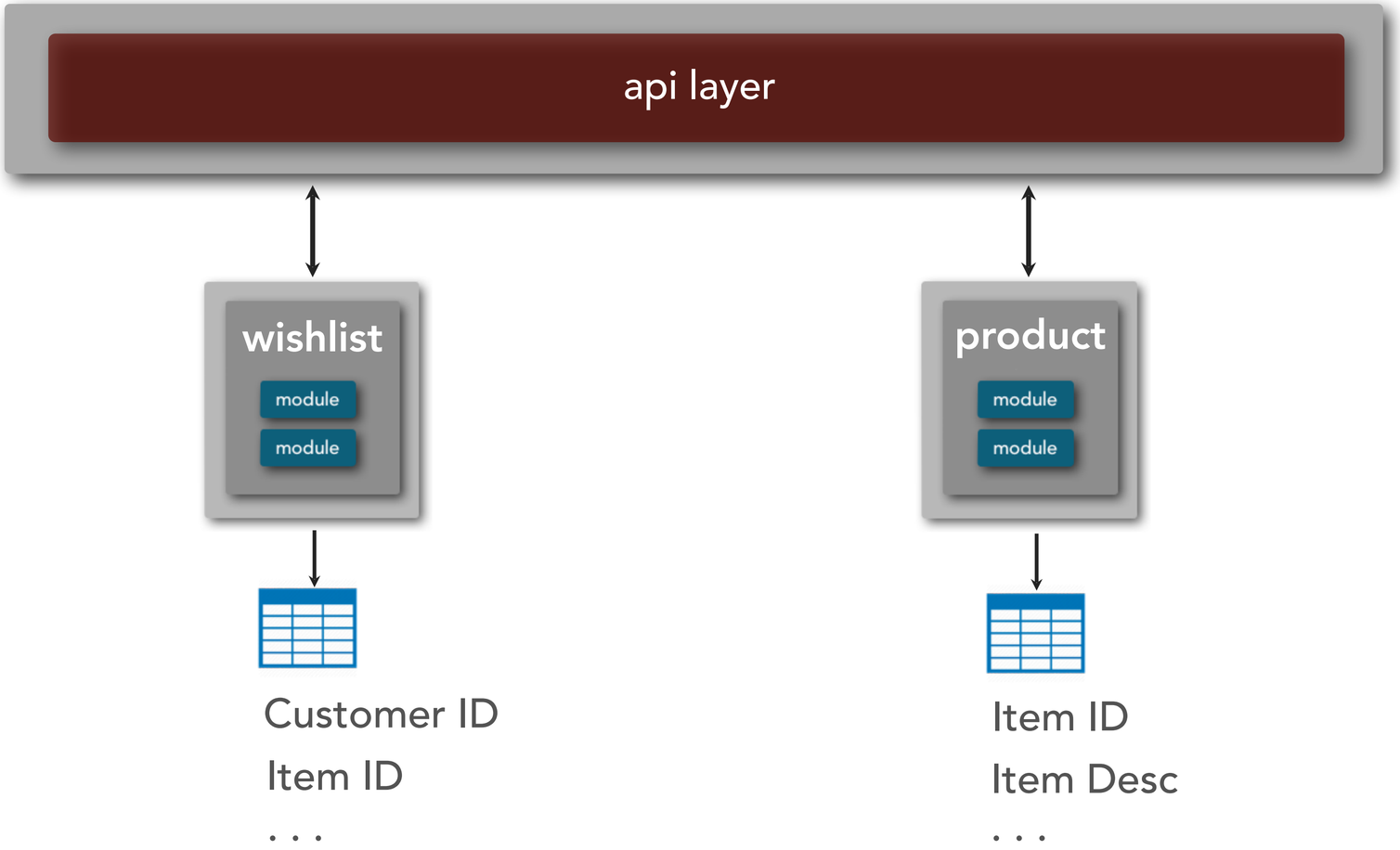
data access and sharing

"once I break apart my data, how to I access data I don't own?"



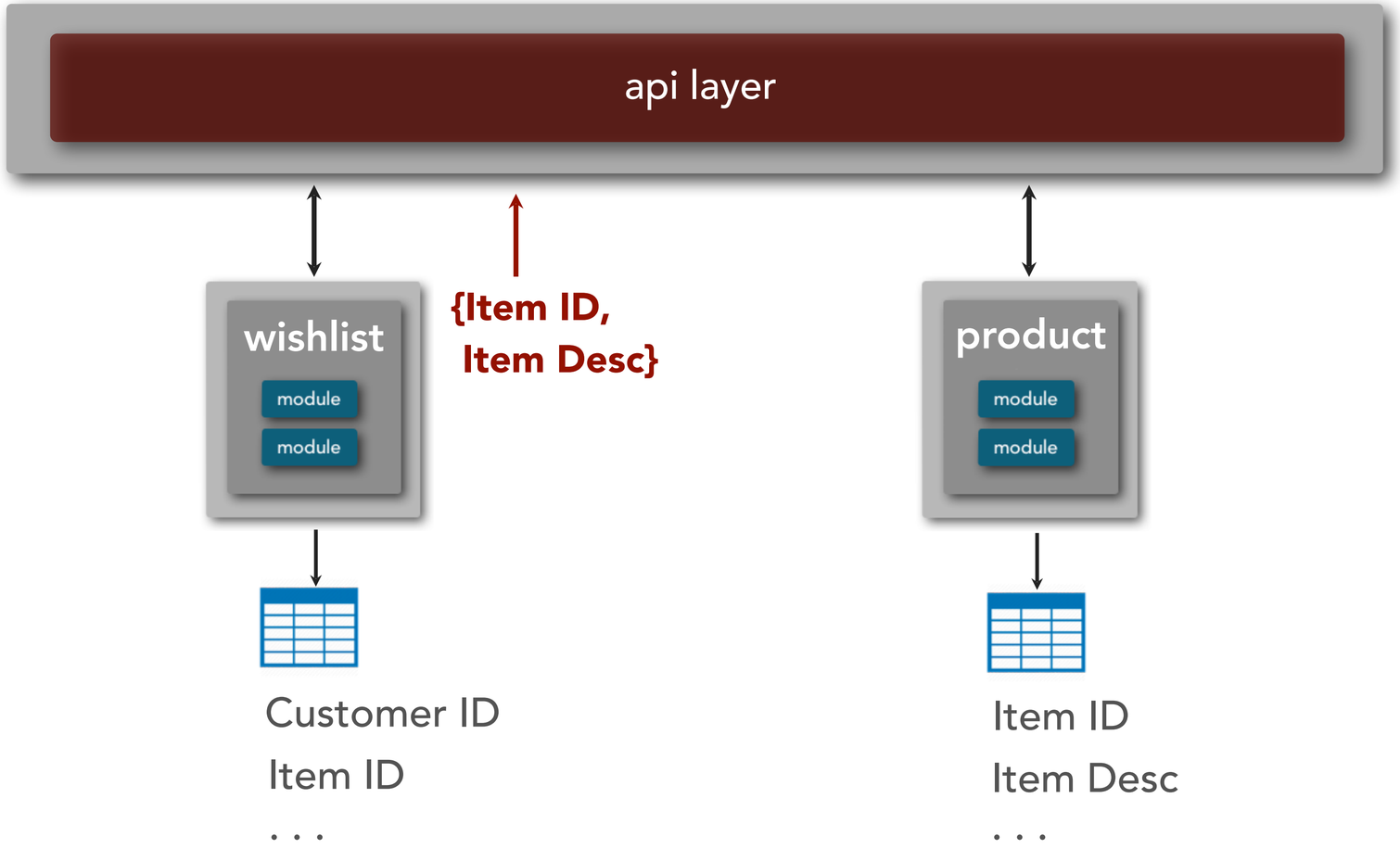
data access and sharing

the issue...



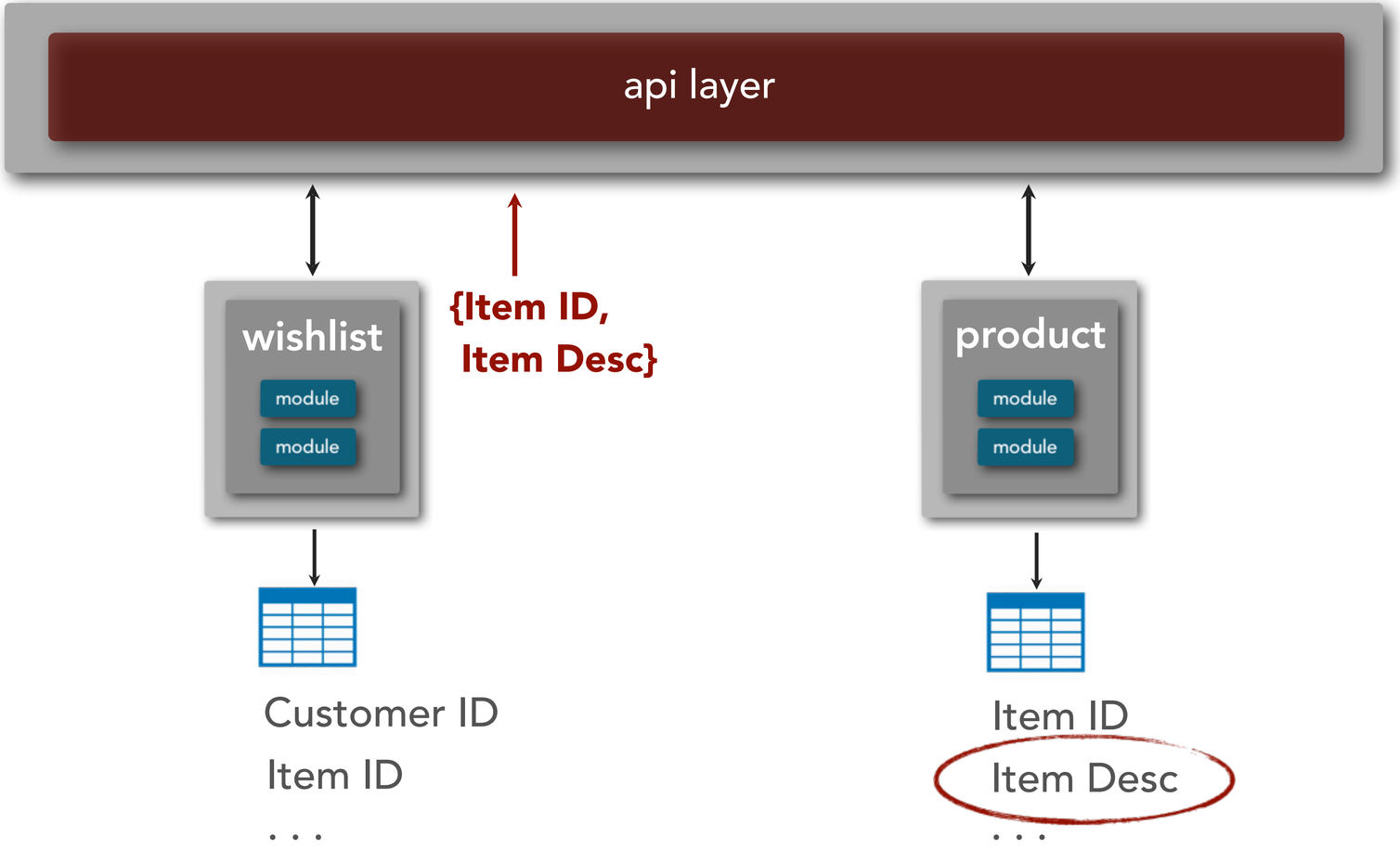
data access and sharing

the issue...



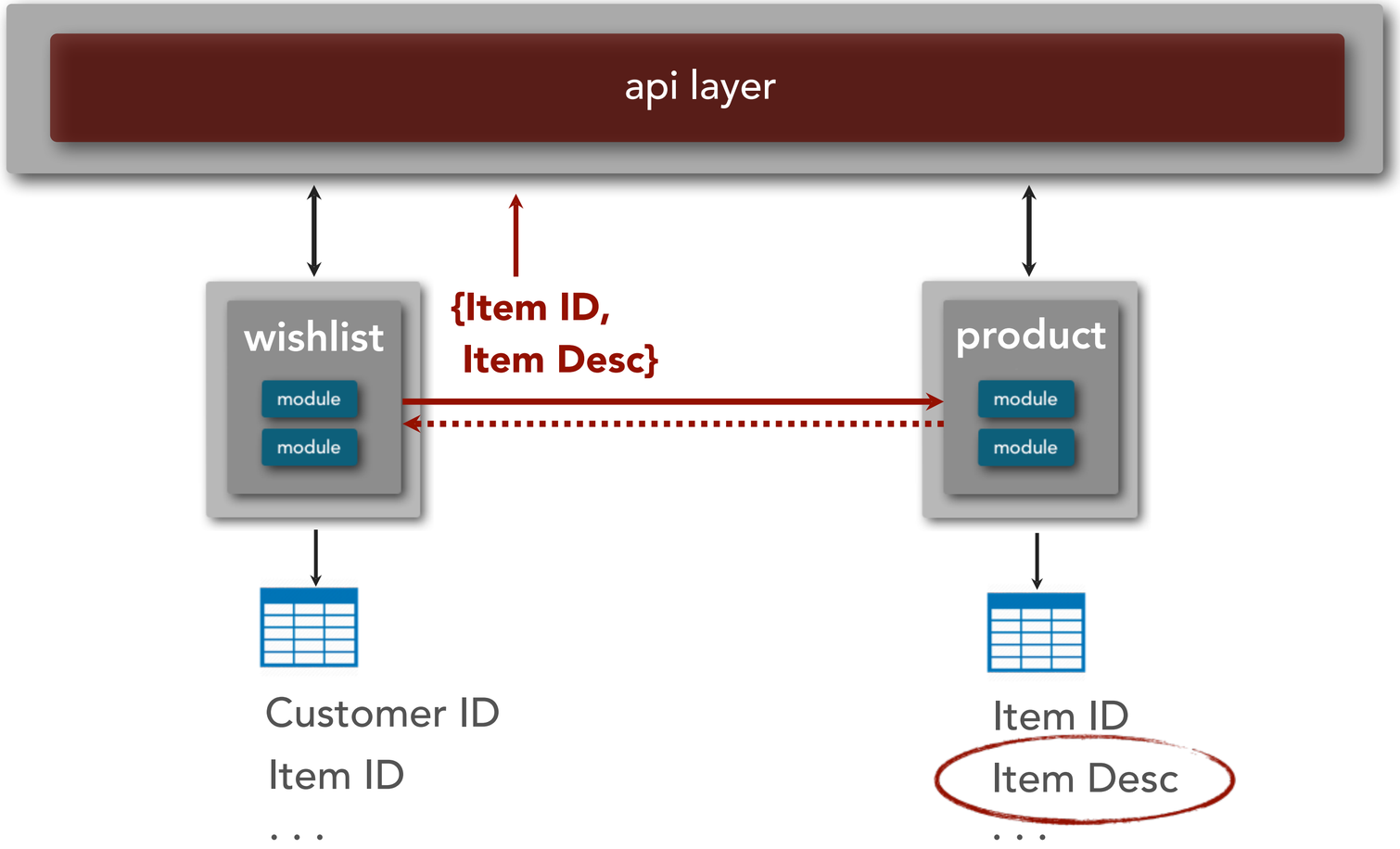
data access and sharing

the issue...



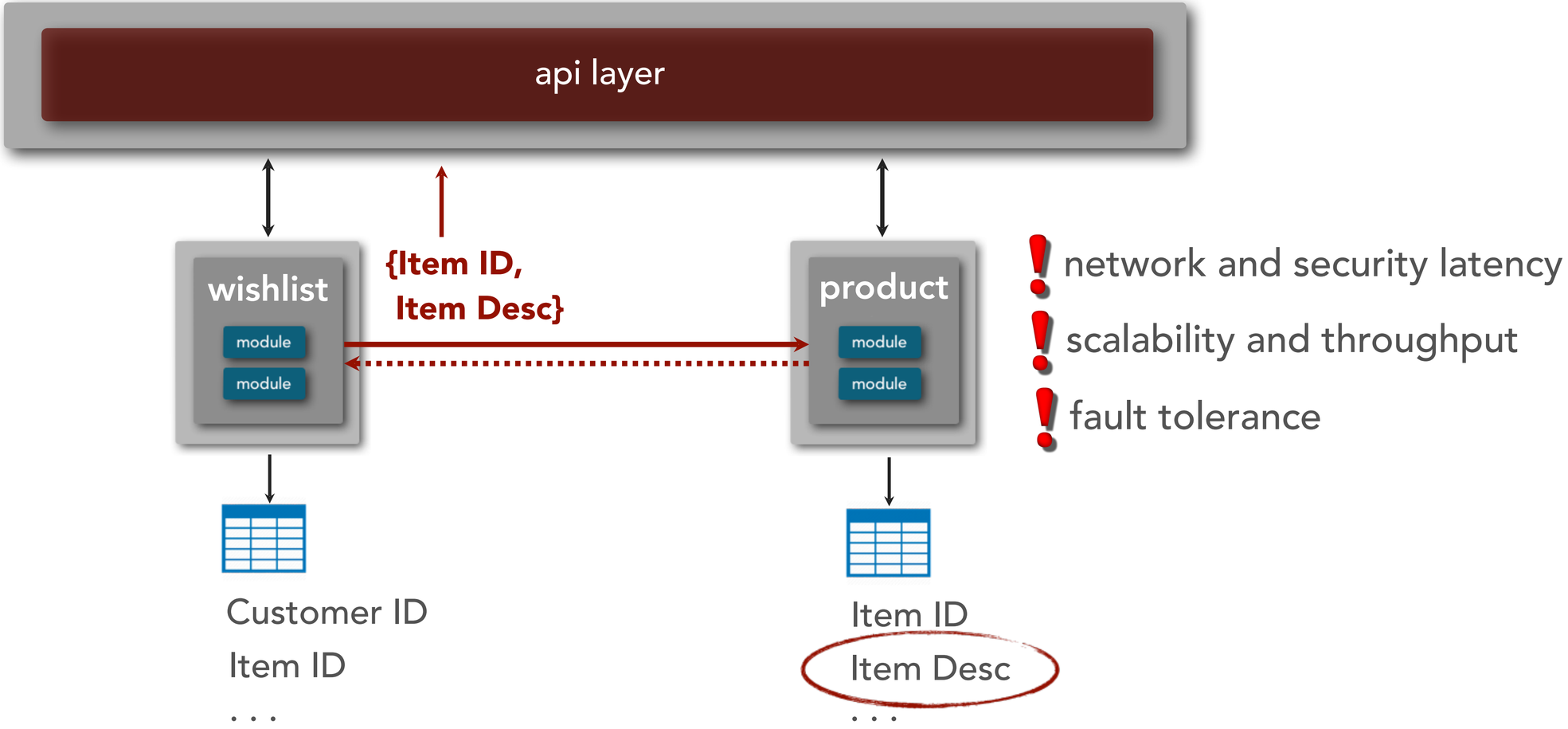
data access and sharing

option 1: interservice communication



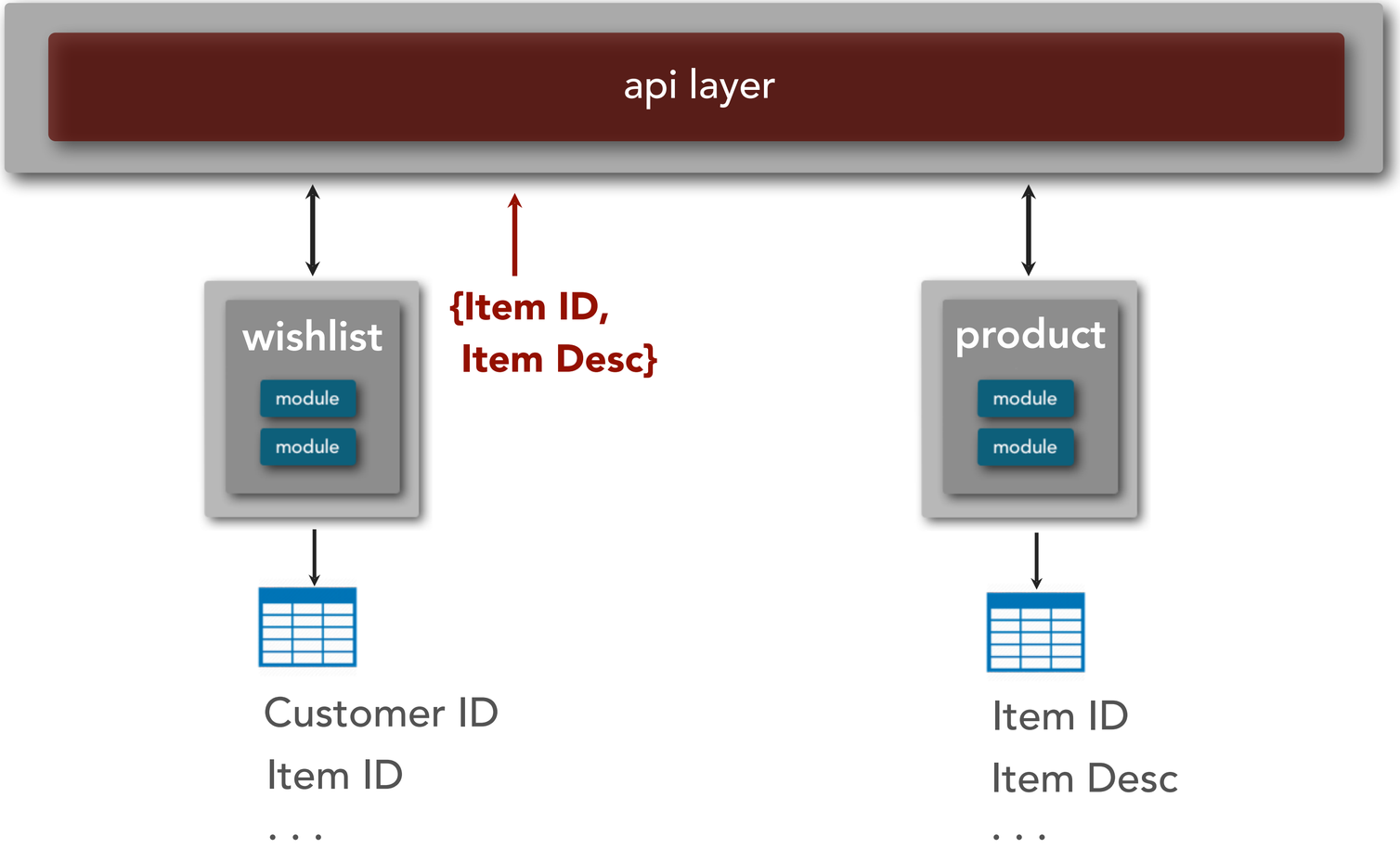
data access and sharing

option 1: interservice communication



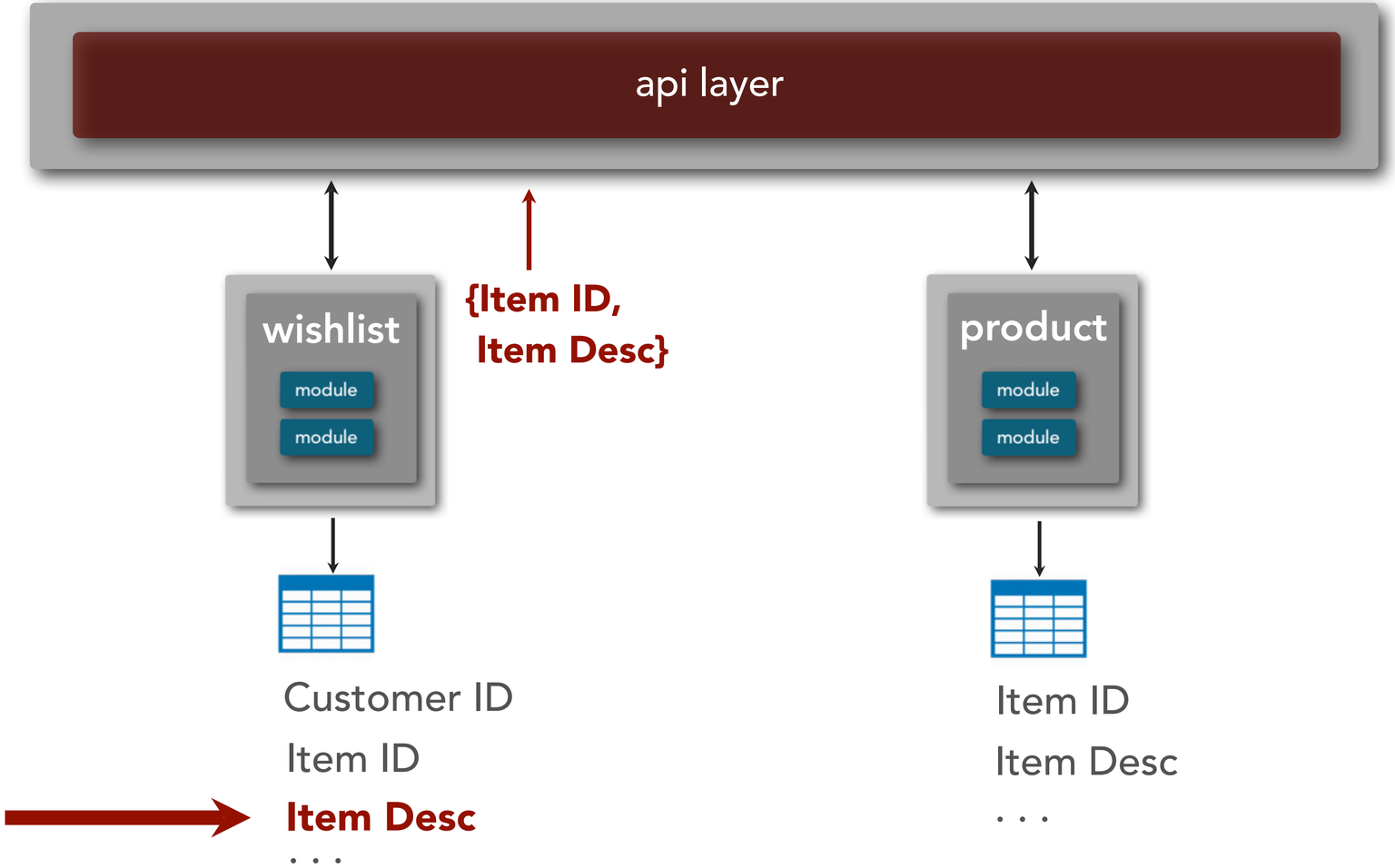
data access and sharing

option 2: data replication



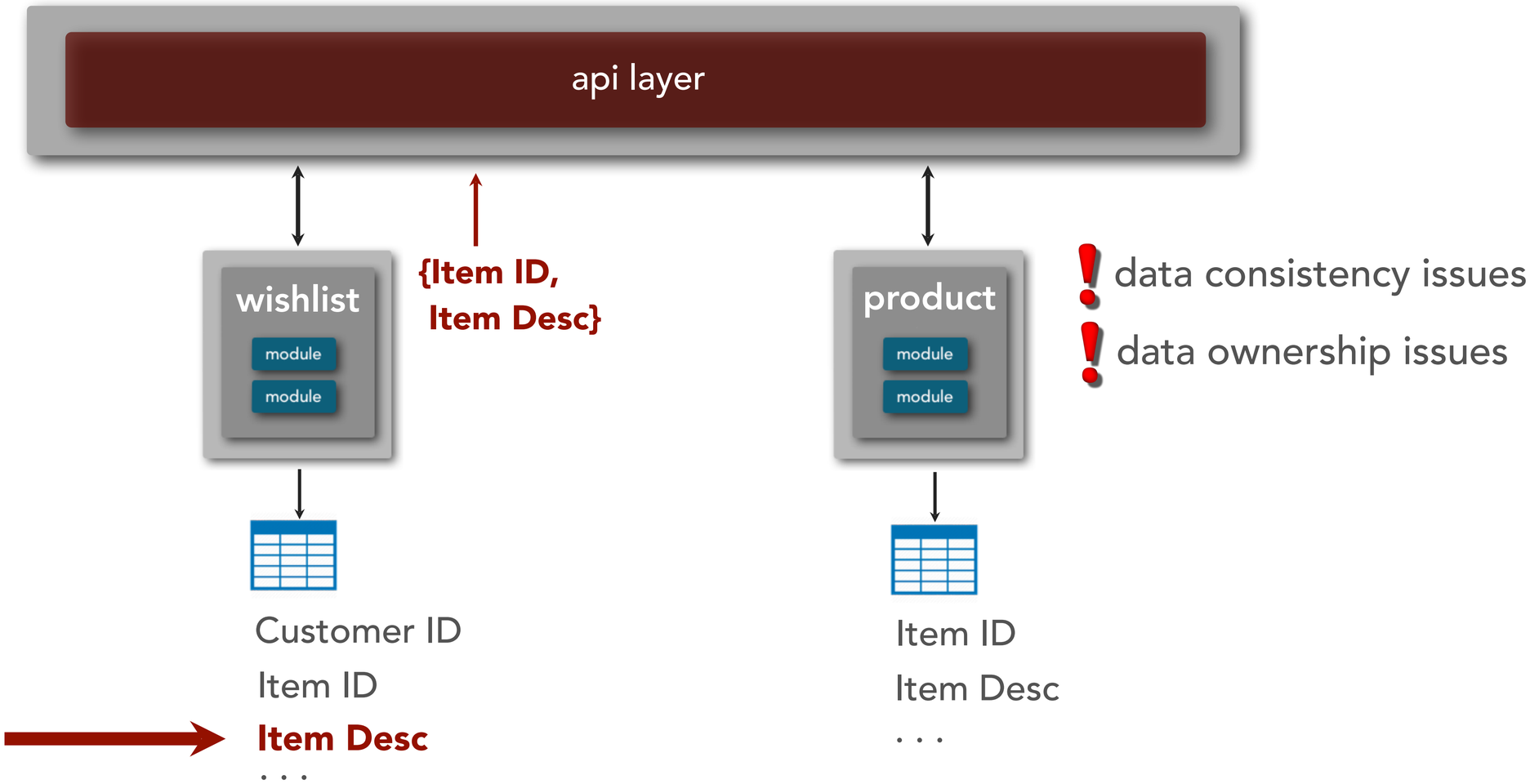
data access and sharing

option 2: data replication



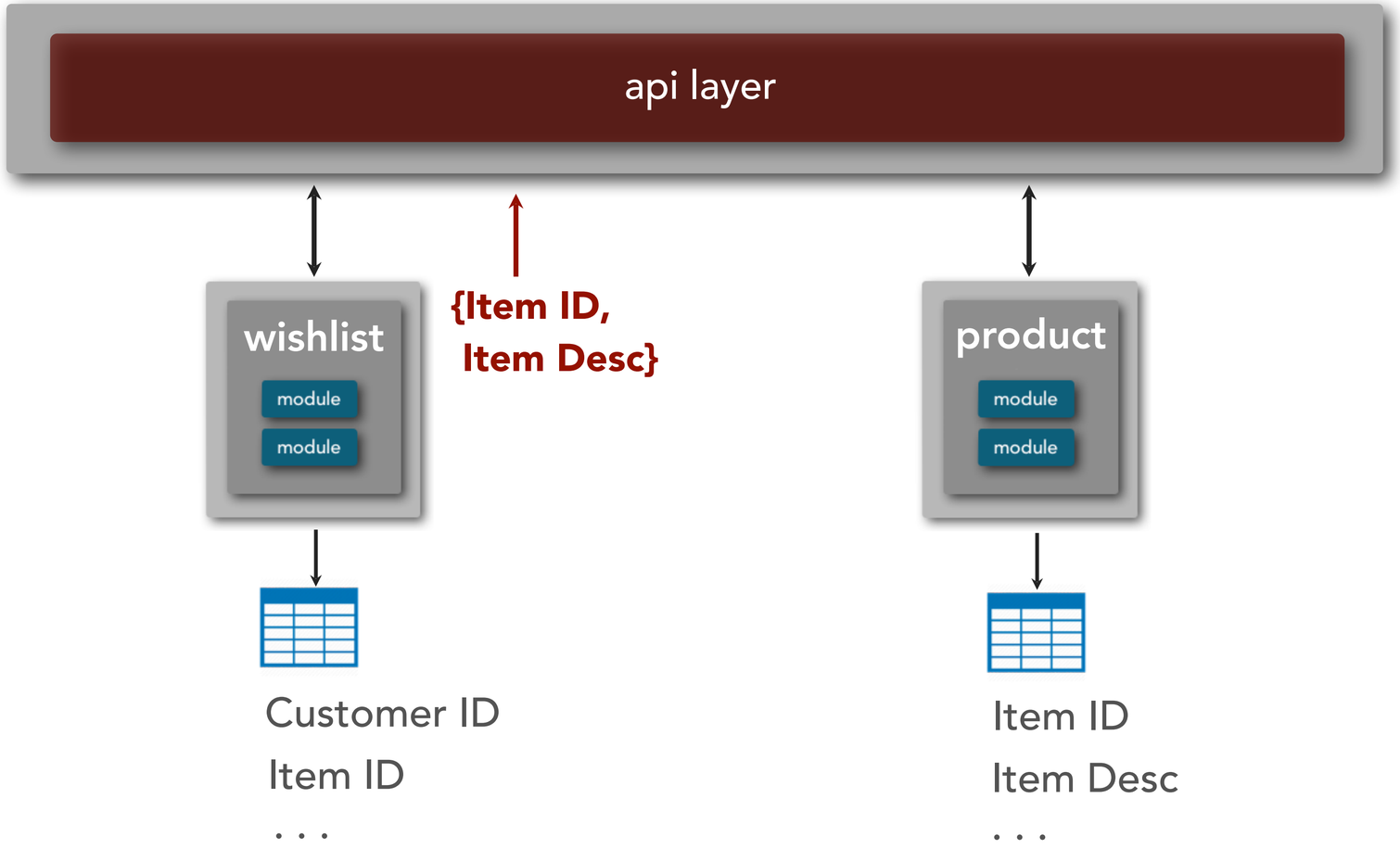
data access and sharing

option 2: data replication



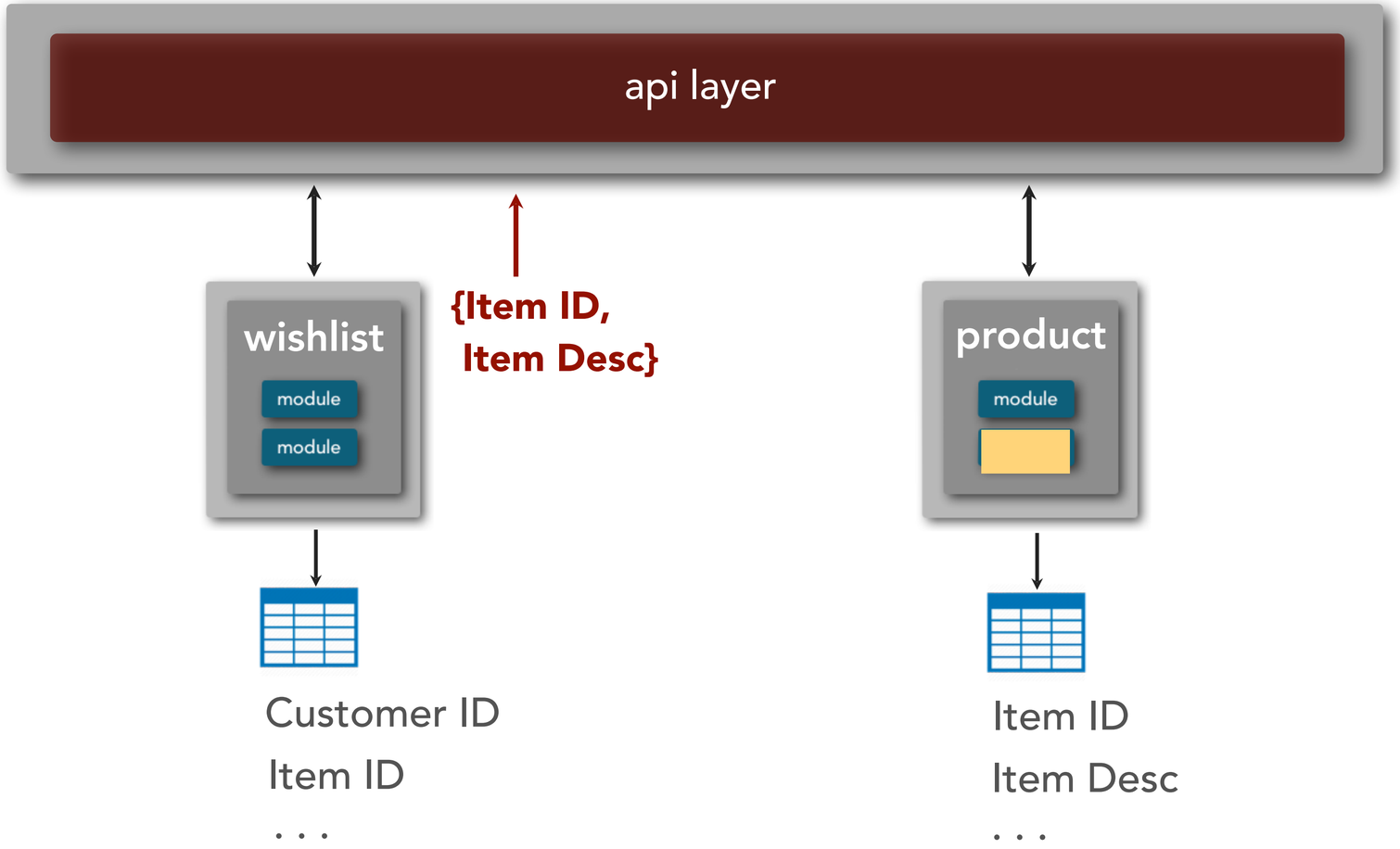
data access and sharing

option 3: in-memory replicated cache



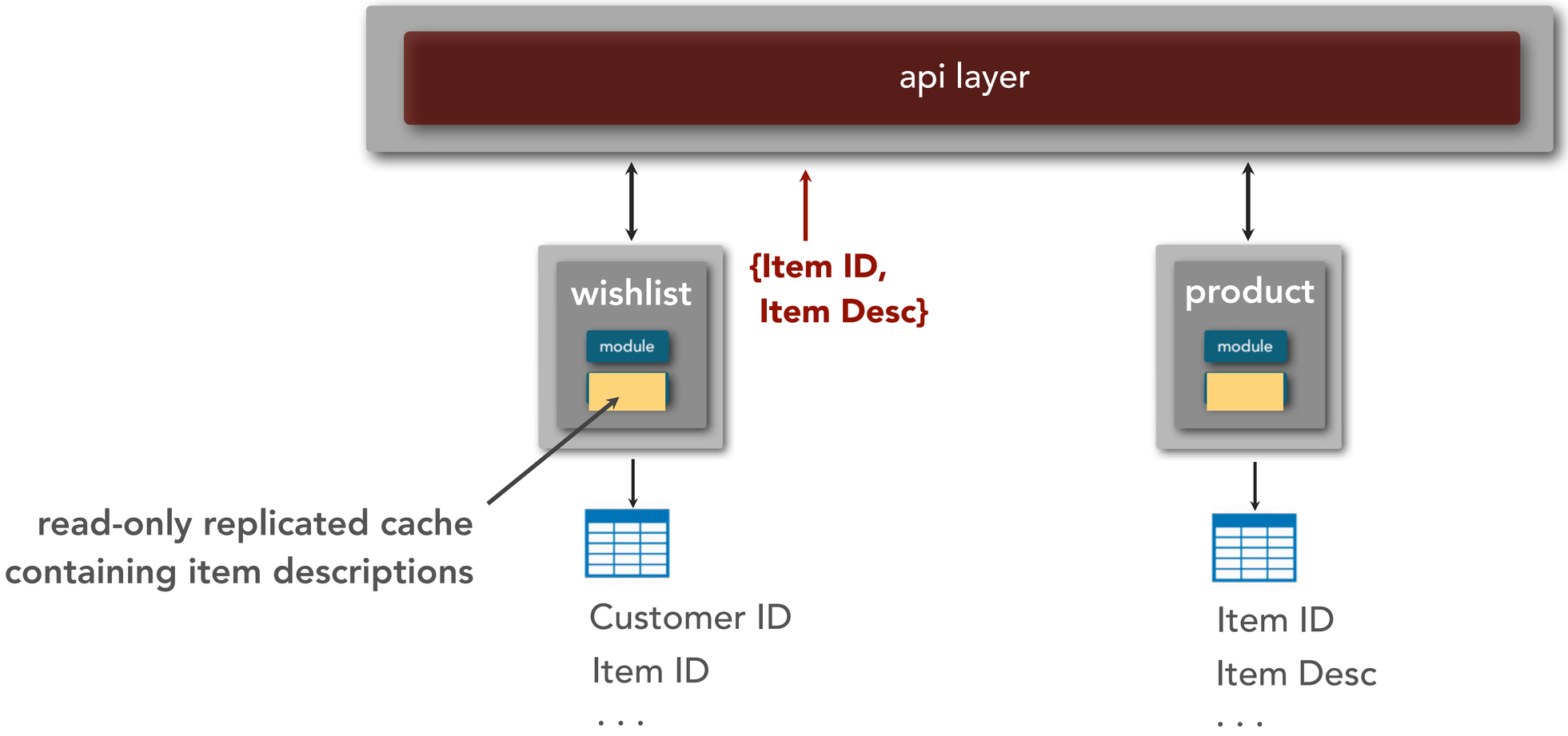
data access and sharing

option 3: in-memory replicated cache



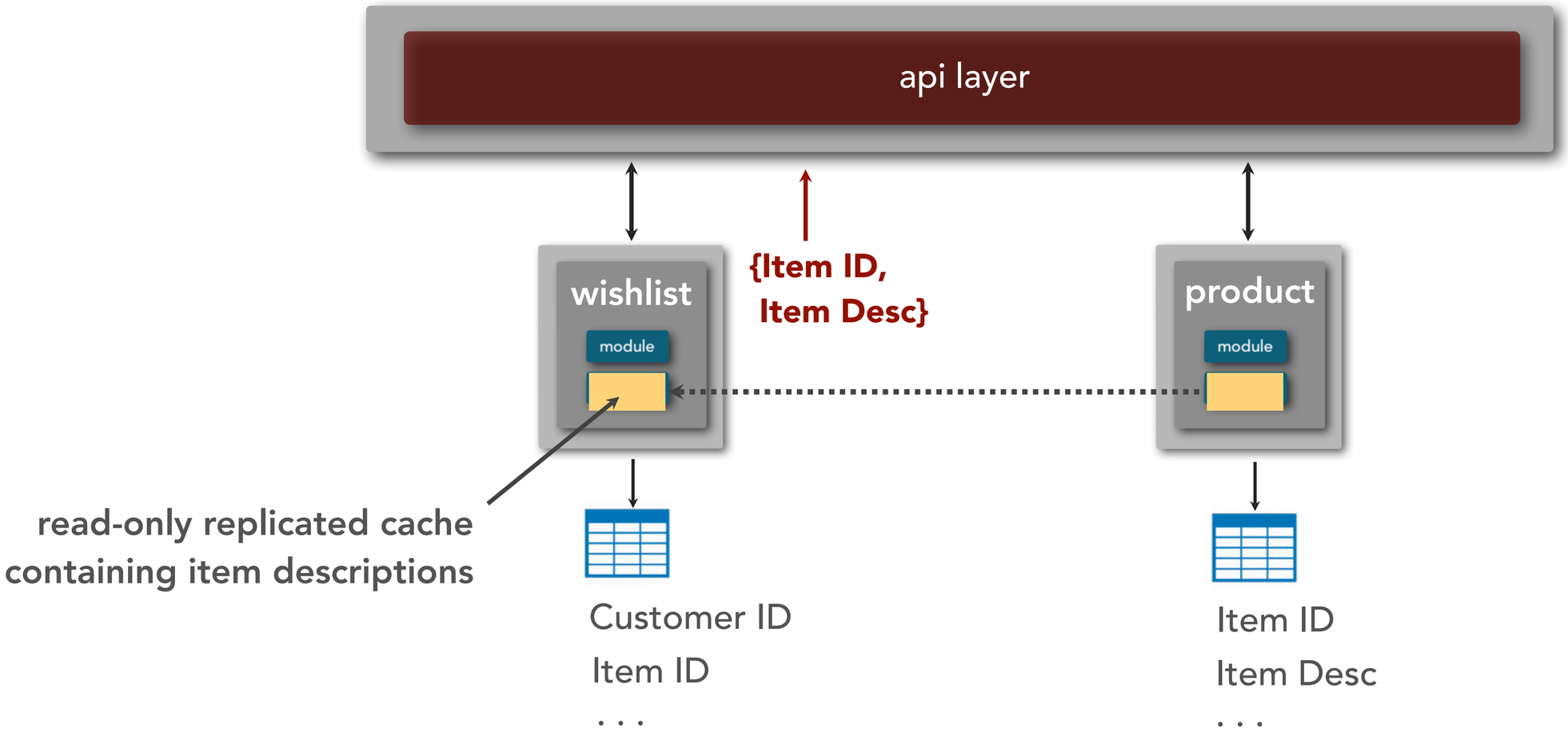
data access and sharing

option 3: in-memory replicated cache



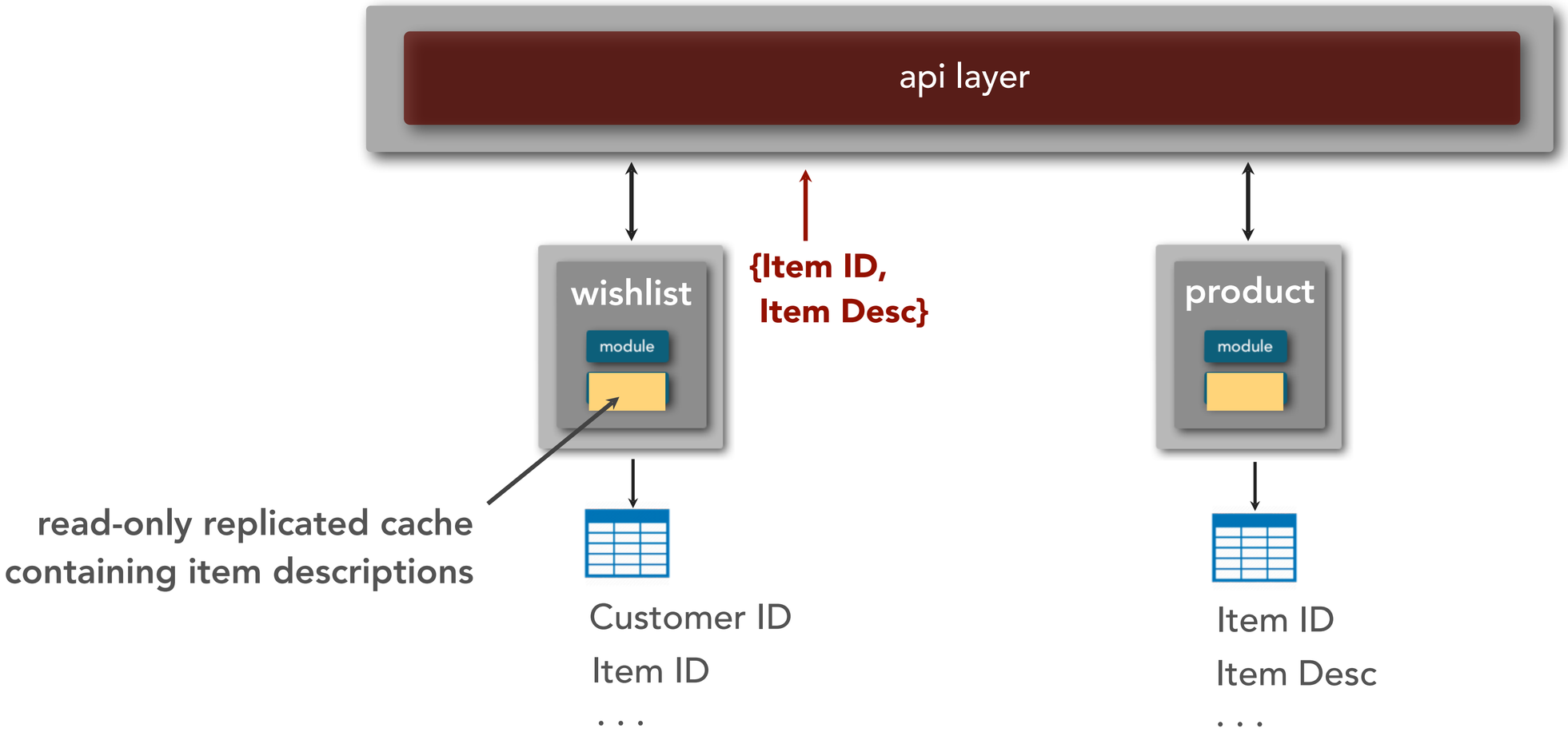
data access and sharing

option 3: in-memory replicated cache



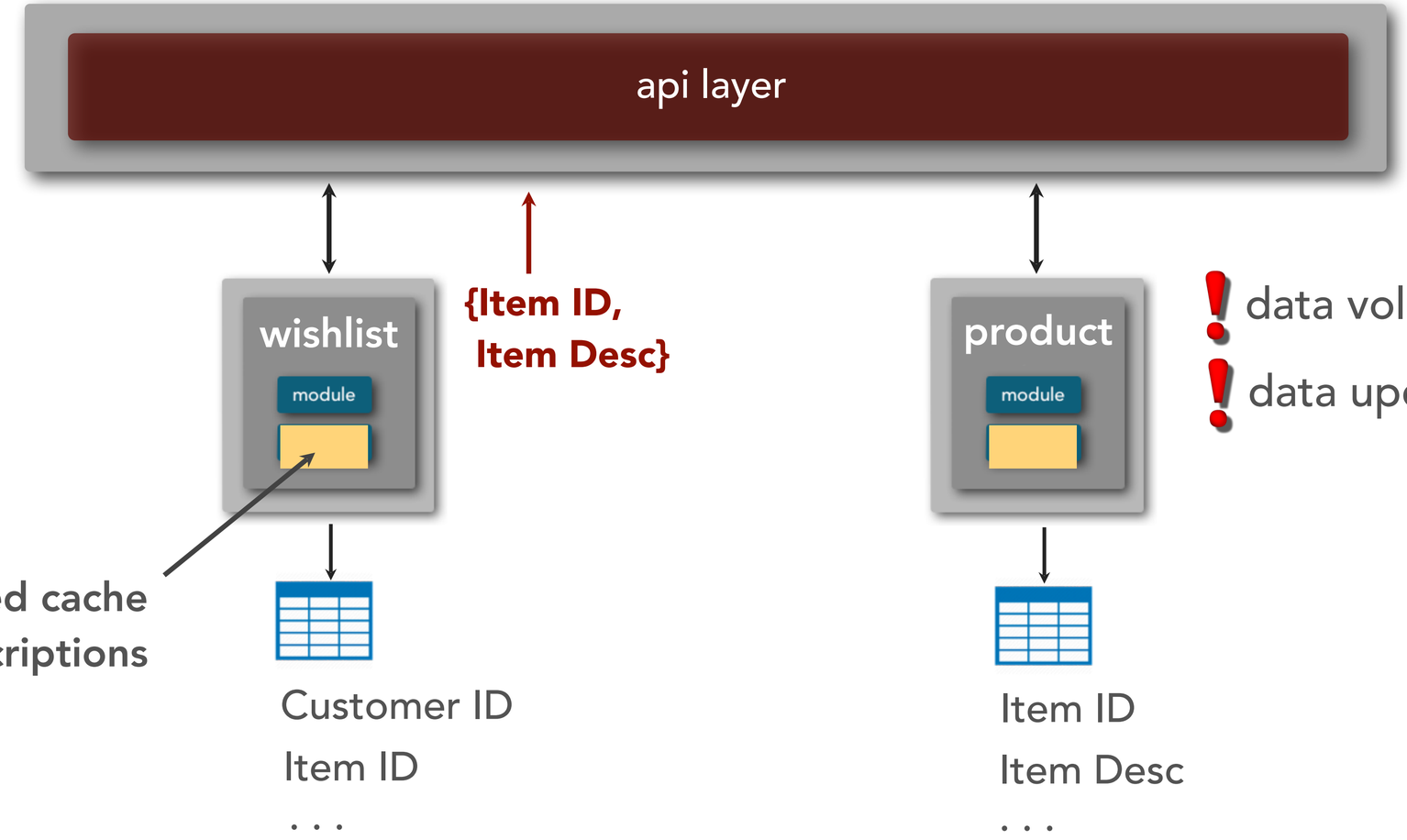
data access and sharing

option 3: in-memory replicated cache



data access and sharing

option 3: in-memory replicated cache

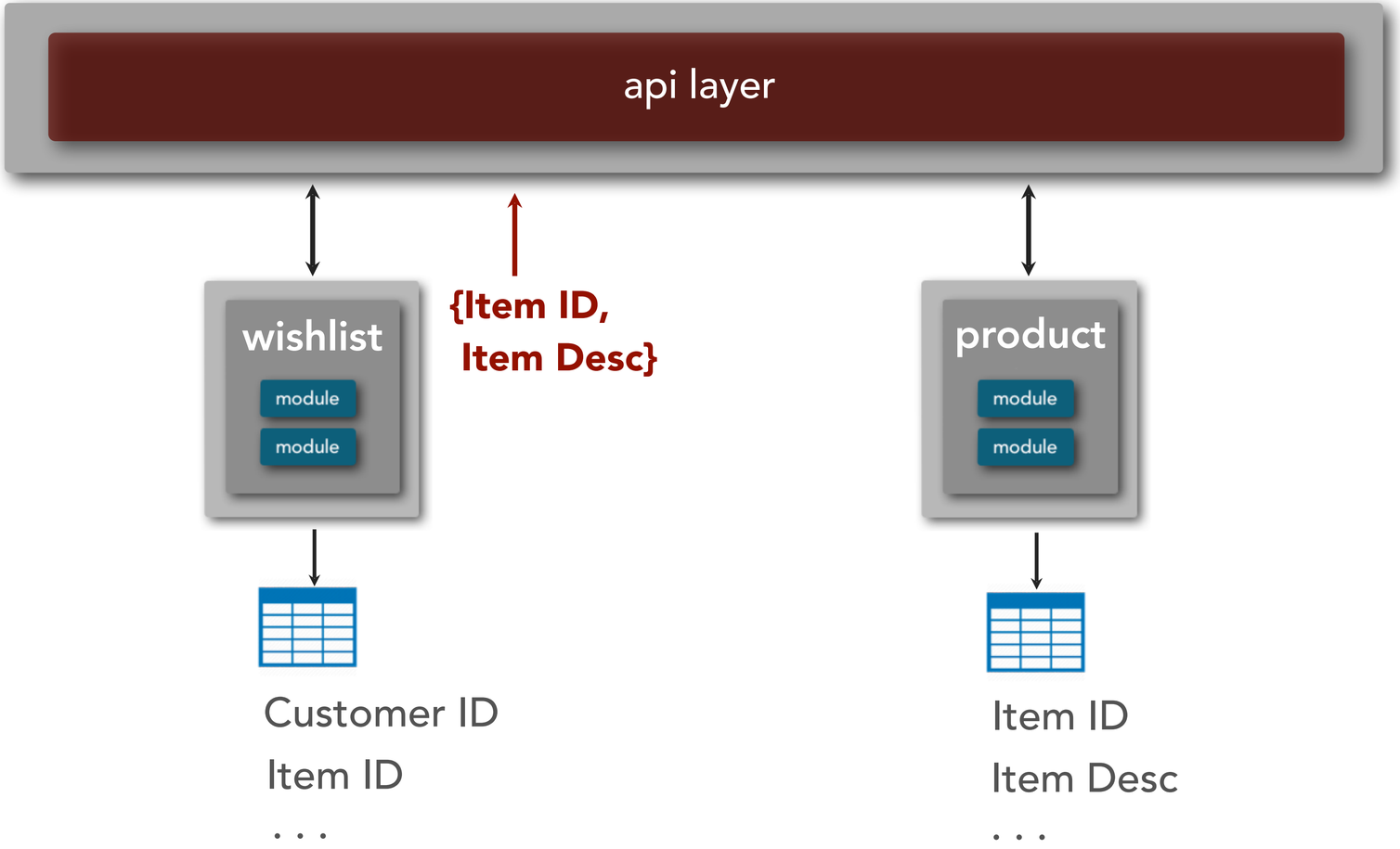


! data volume issues

! data update rate issues

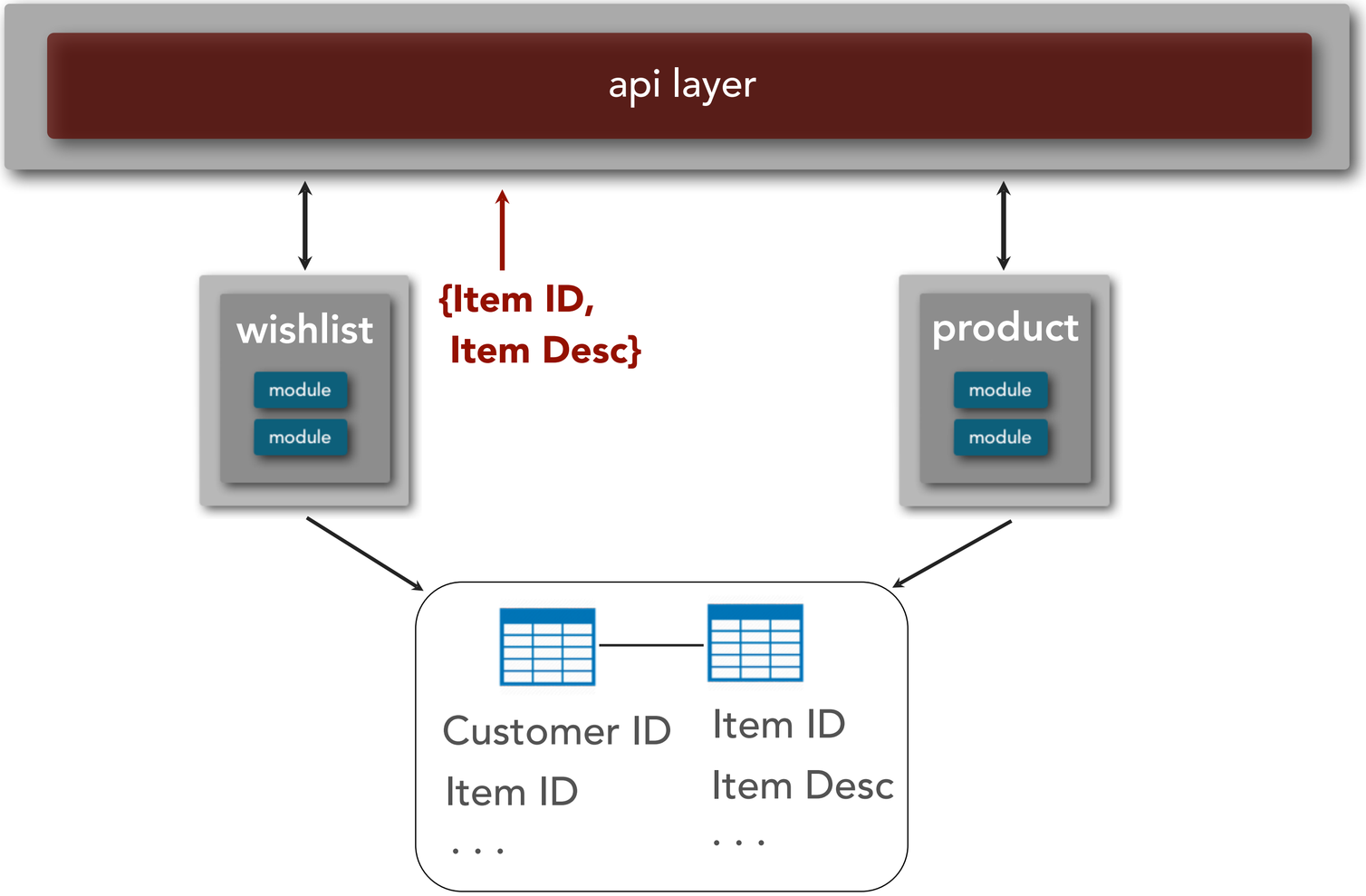
data access and sharing

option 4: data domain



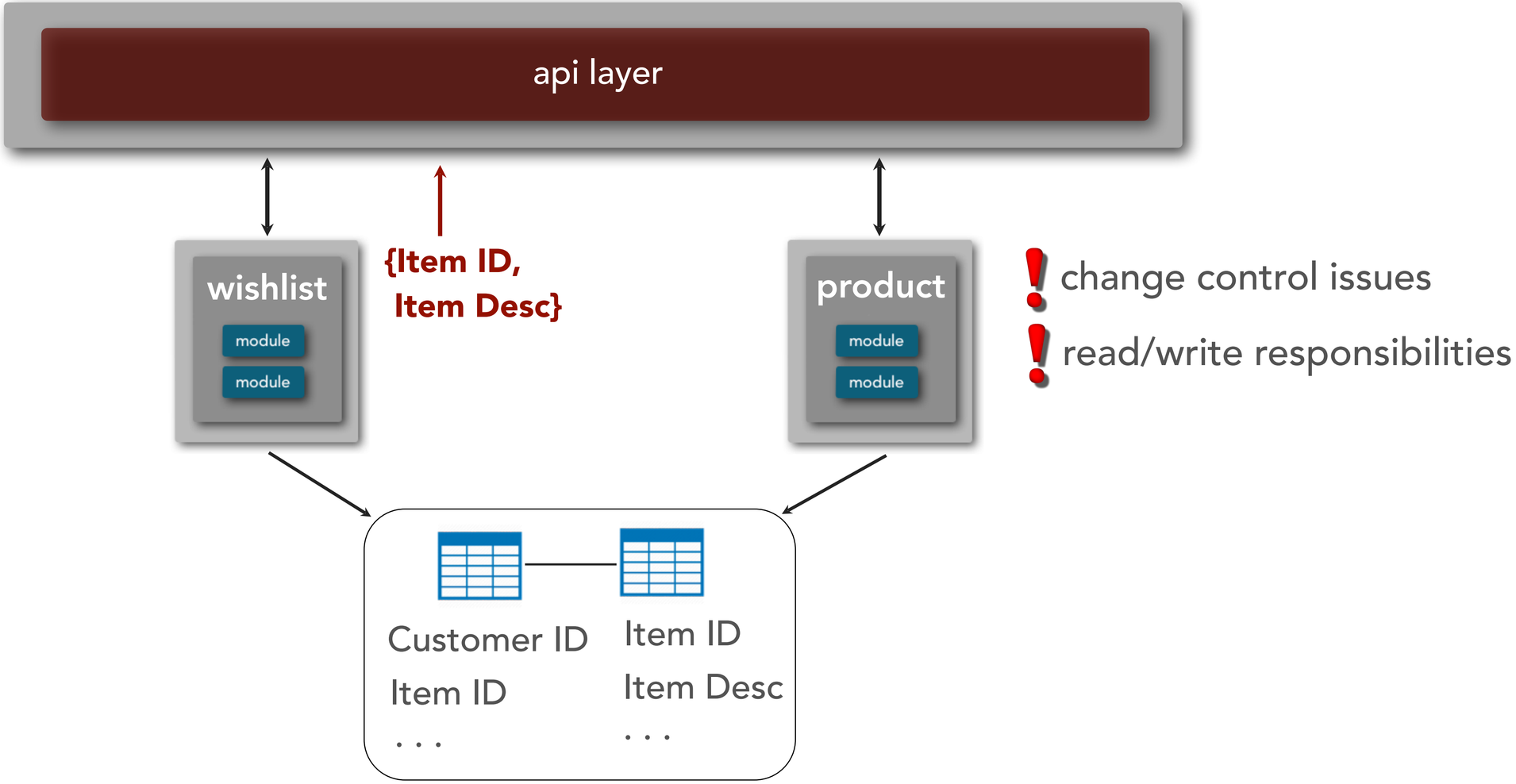
data access and sharing

option 4: data domain



data access and sharing

option 4: data domain



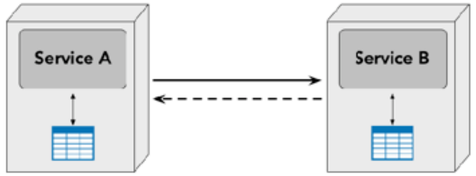
! change control issues

! read/write responsibilities

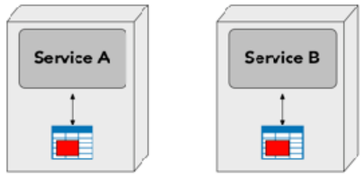
data access and sharing

which one is better?

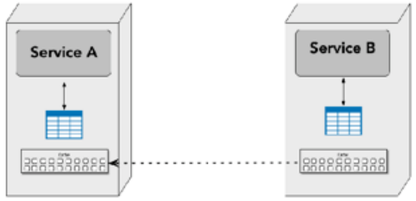
option 1:
interservice
communication



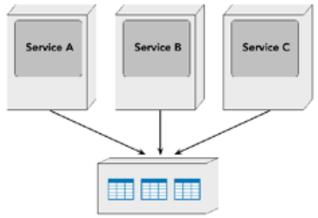
option 2:
data schema
replication



option 3:
in-memory
replicated cache



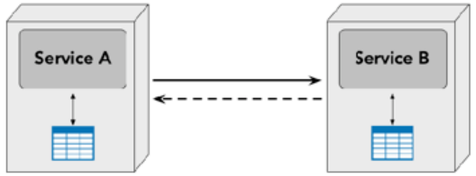
option 4:
data domains
(shared tables)



data access and sharing

which one is better?

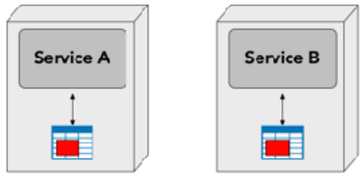
option 1:
interservice
communication



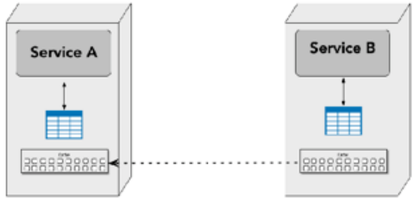
✓ large data volume

✓ low responsiveness

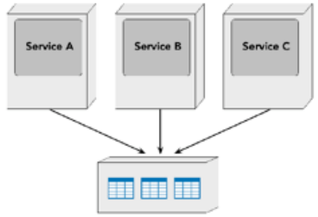
option 2:
data schema
replication



option 3:
in-memory
replicated cache



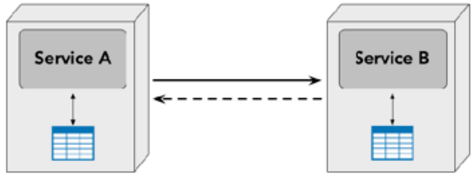
option 4:
data domains
(shared tables)



data access and sharing

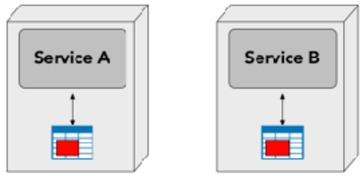
which one is better?

option 1:
interservice
communication



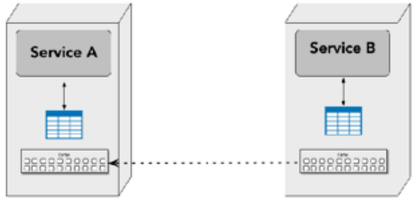
- ✓ large data volume
- ✓ low responsiveness

option 2:
data schema
replication

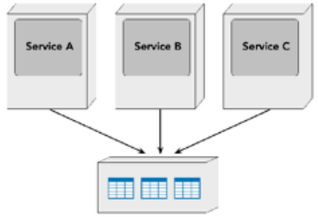


- ✓ reporting
- ✓ data aggregation

option 3:
in-memory
replicated cache



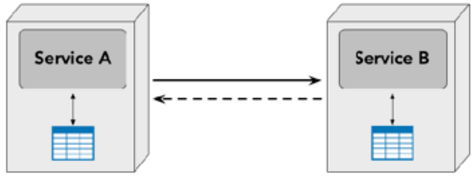
option 4:
data domains
(shared tables)



data access and sharing

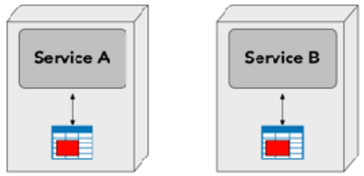
which one is better?

option 1:
interservice
communication



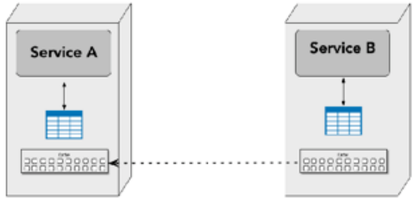
- ✓ large data volume
- ✓ low responsiveness

option 2:
data schema
replication



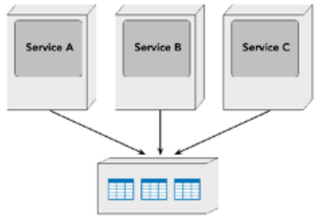
- ✓ reporting
- ✓ data aggregation

option 3:
in-memory
replicated cache



- ✓ low data volume
- ✓ high responsiveness

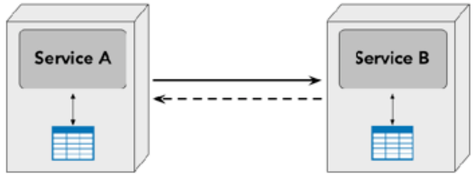
option 4:
data domains
(shared tables)



data access and sharing

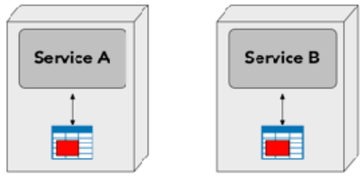
which one is better?

option 1:
interservice
communication



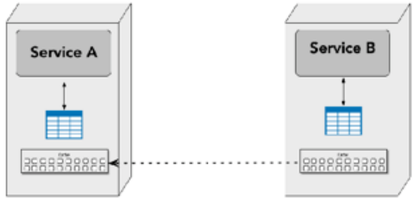
- ✓ large data volume
- ✓ low responsiveness

option 2:
data schema
replication



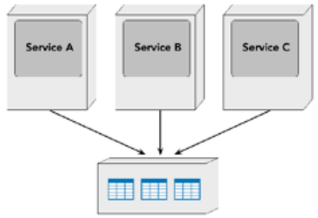
- ✓ reporting
- ✓ data aggregation

option 3:
in-memory
replicated cache



- ✓ low data volume
- ✓ high responsiveness

option 4:
data domains
(shared tables)

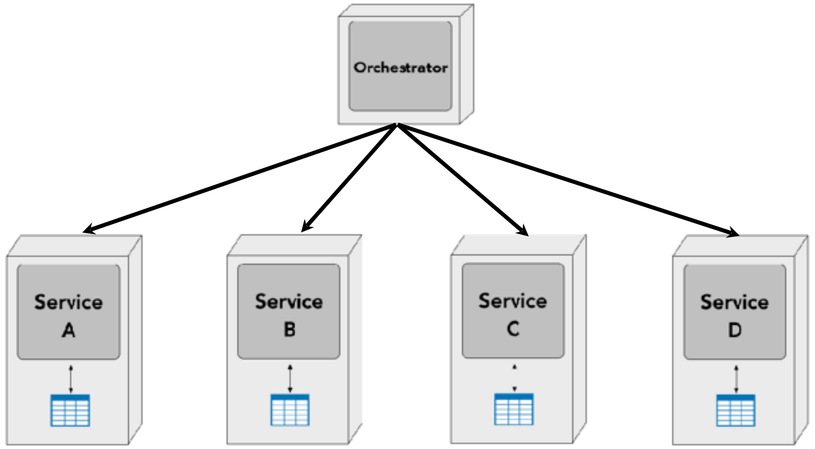


- ✓ multiple owners (write)
- ✓ high responsiveness

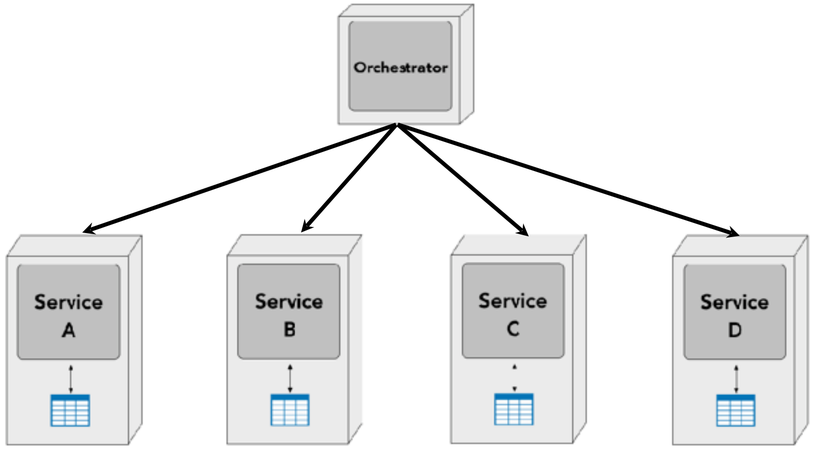
Managing Workflows

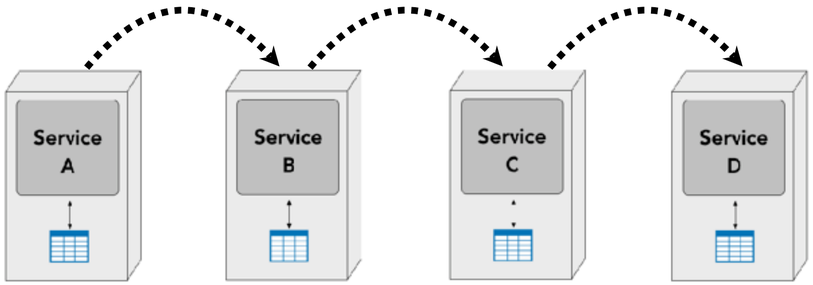
managing workflows

managing workflows

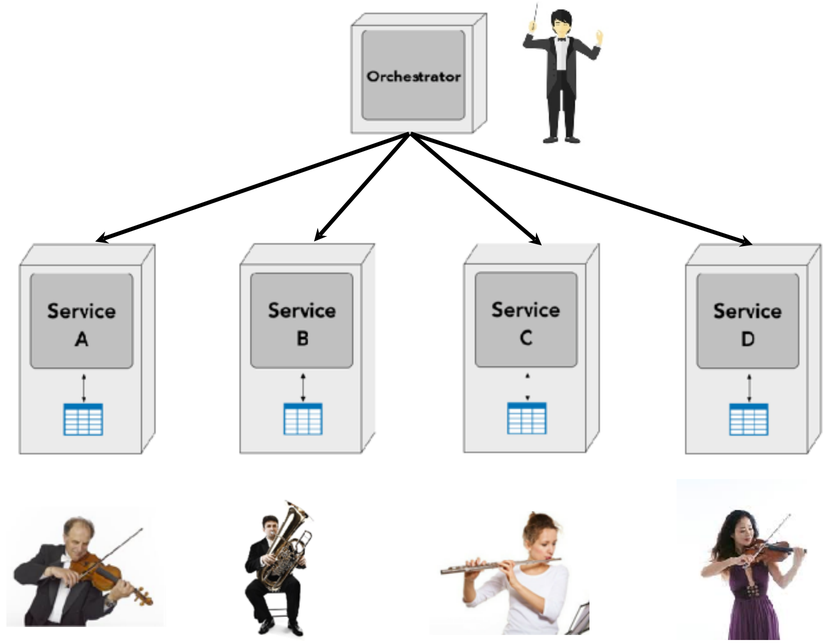


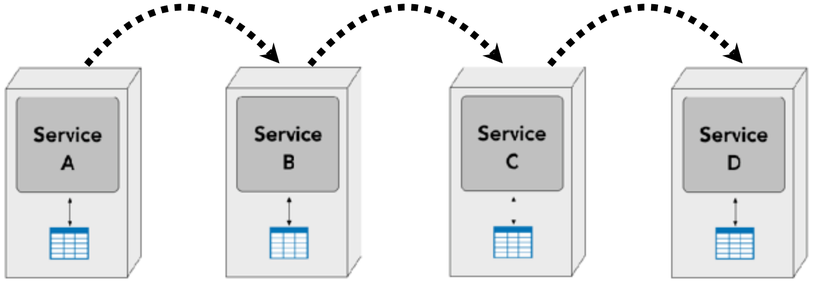
managing workflows



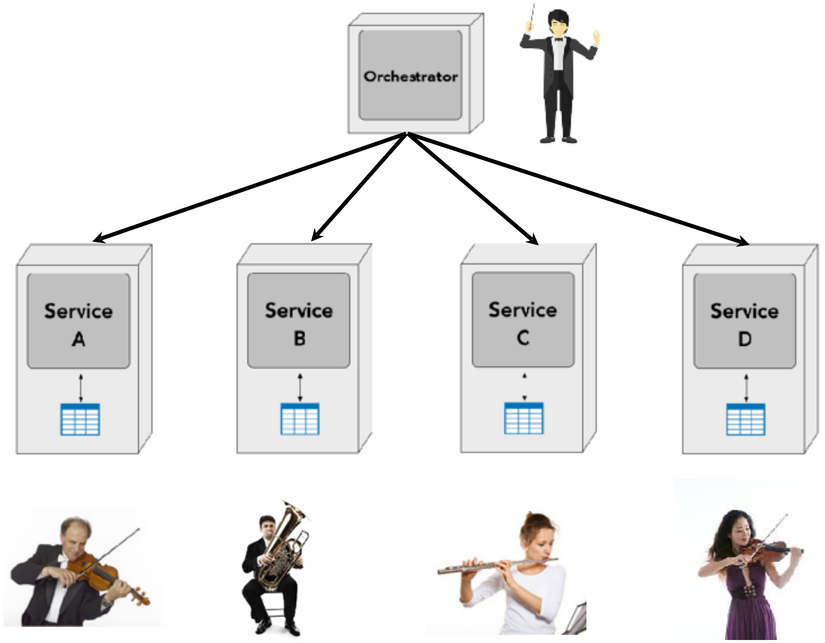


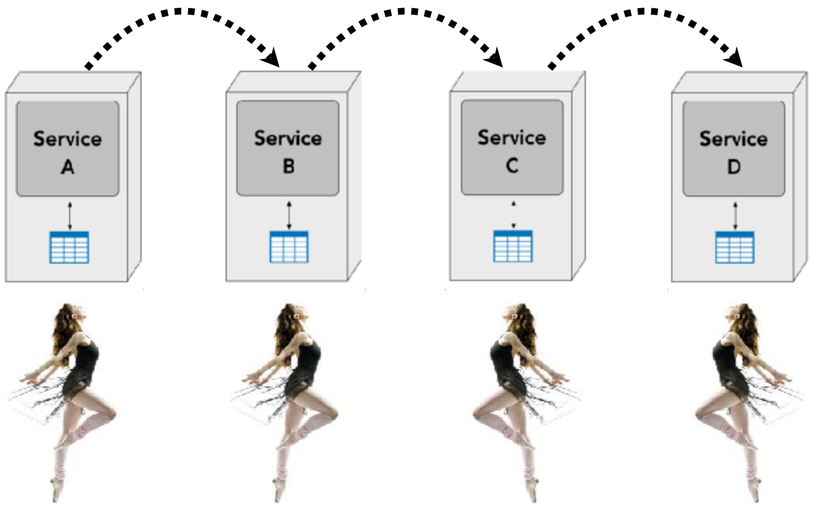
managing workflows



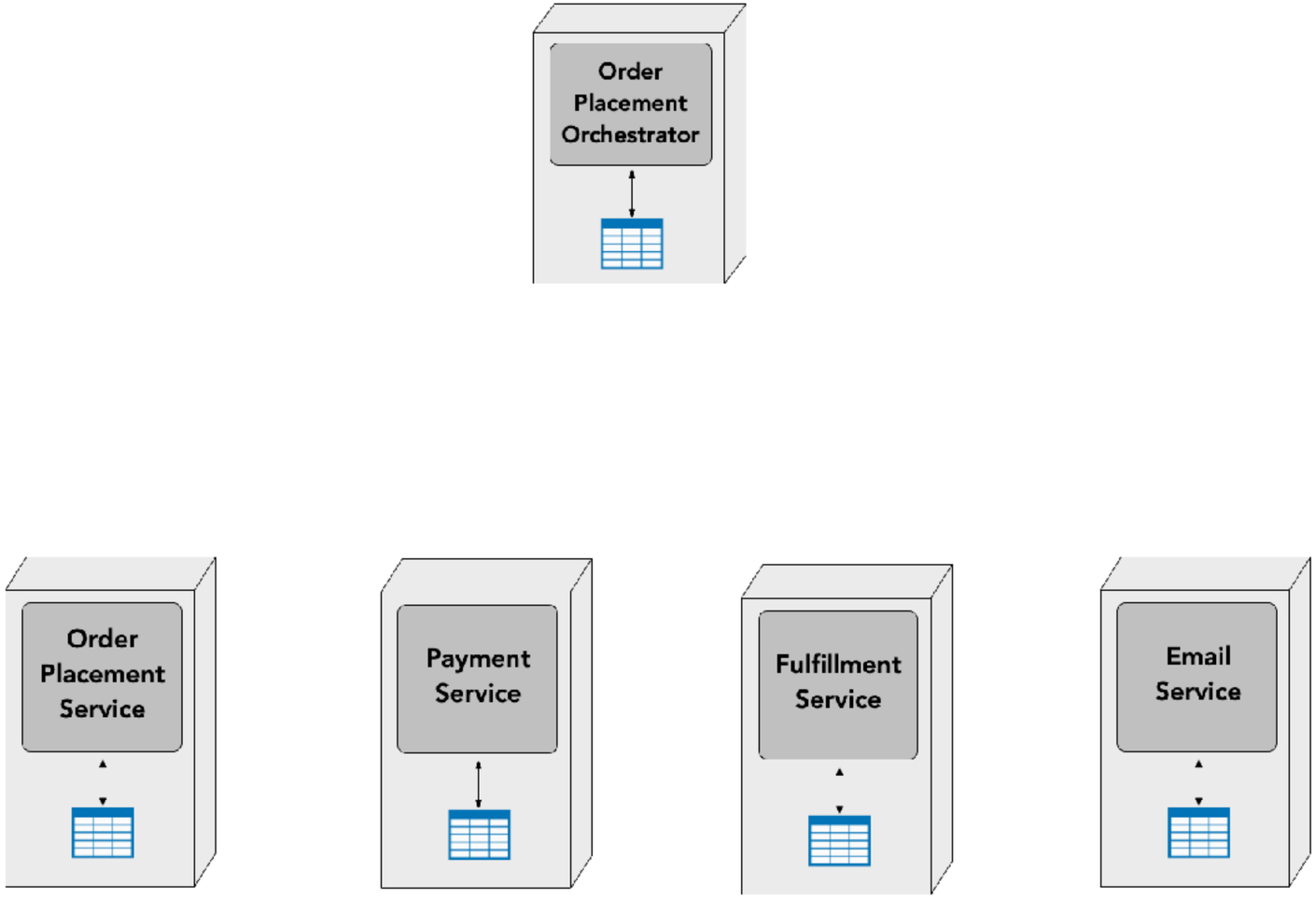


managing workflows

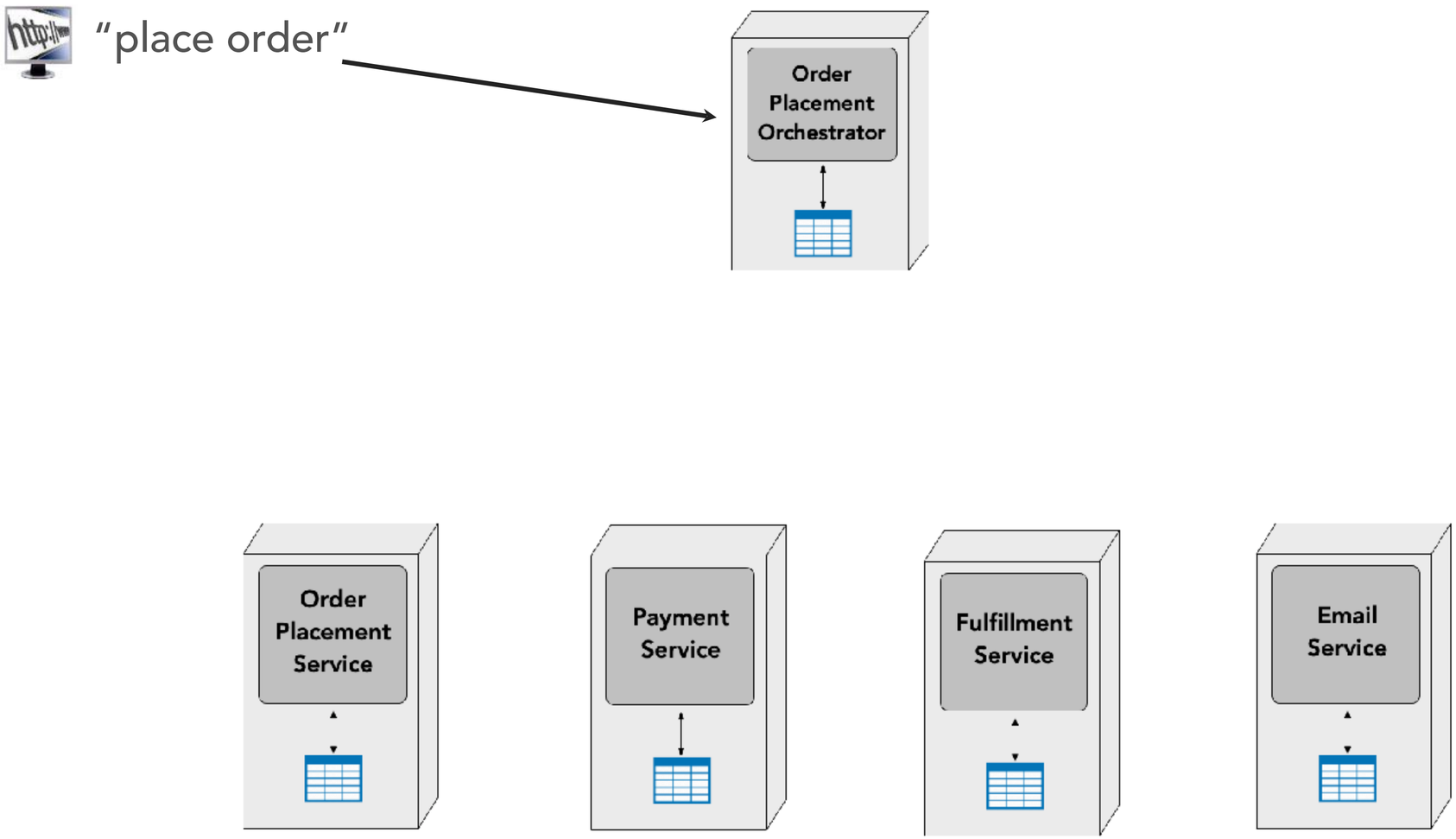




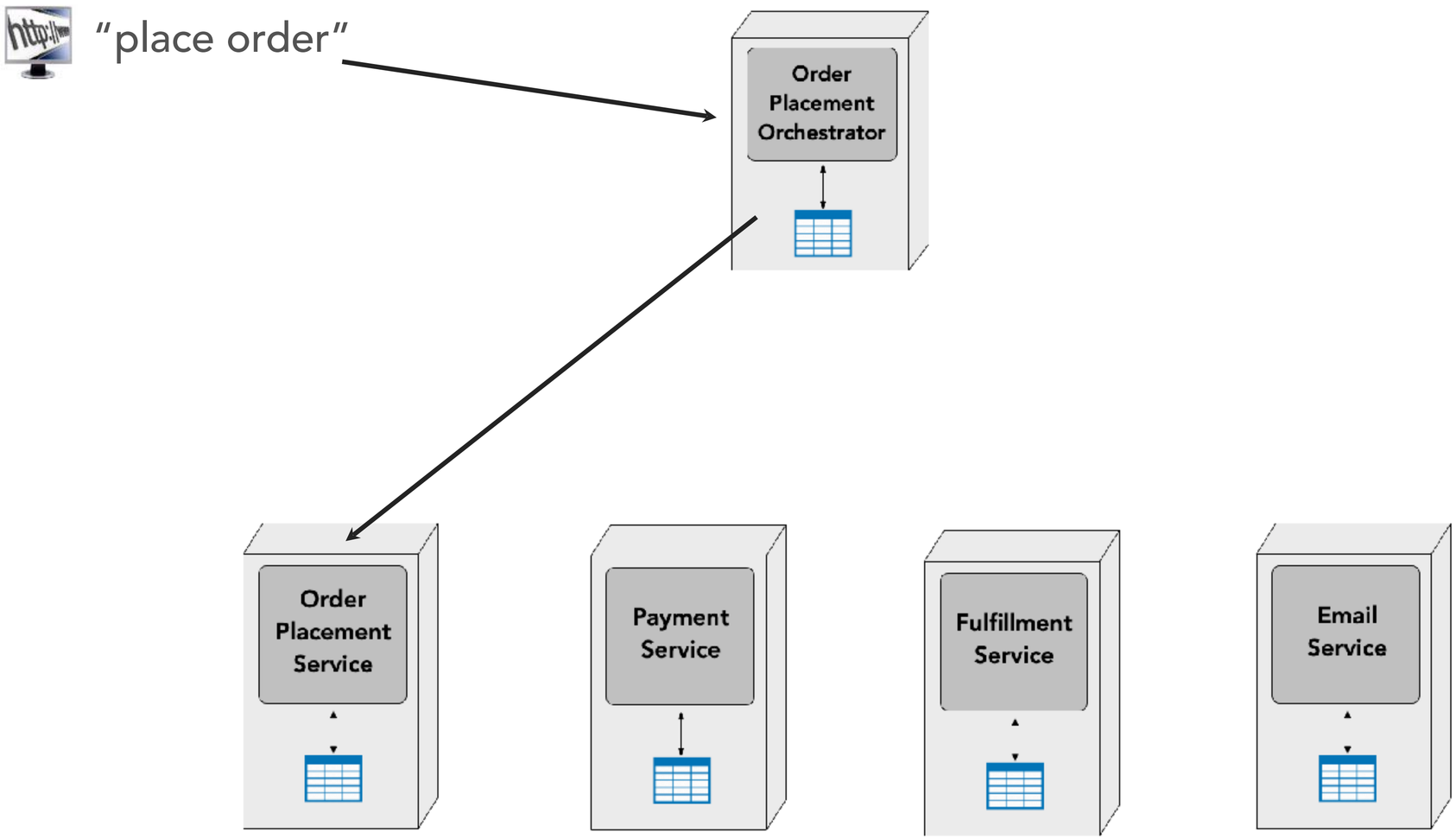
managing workflows



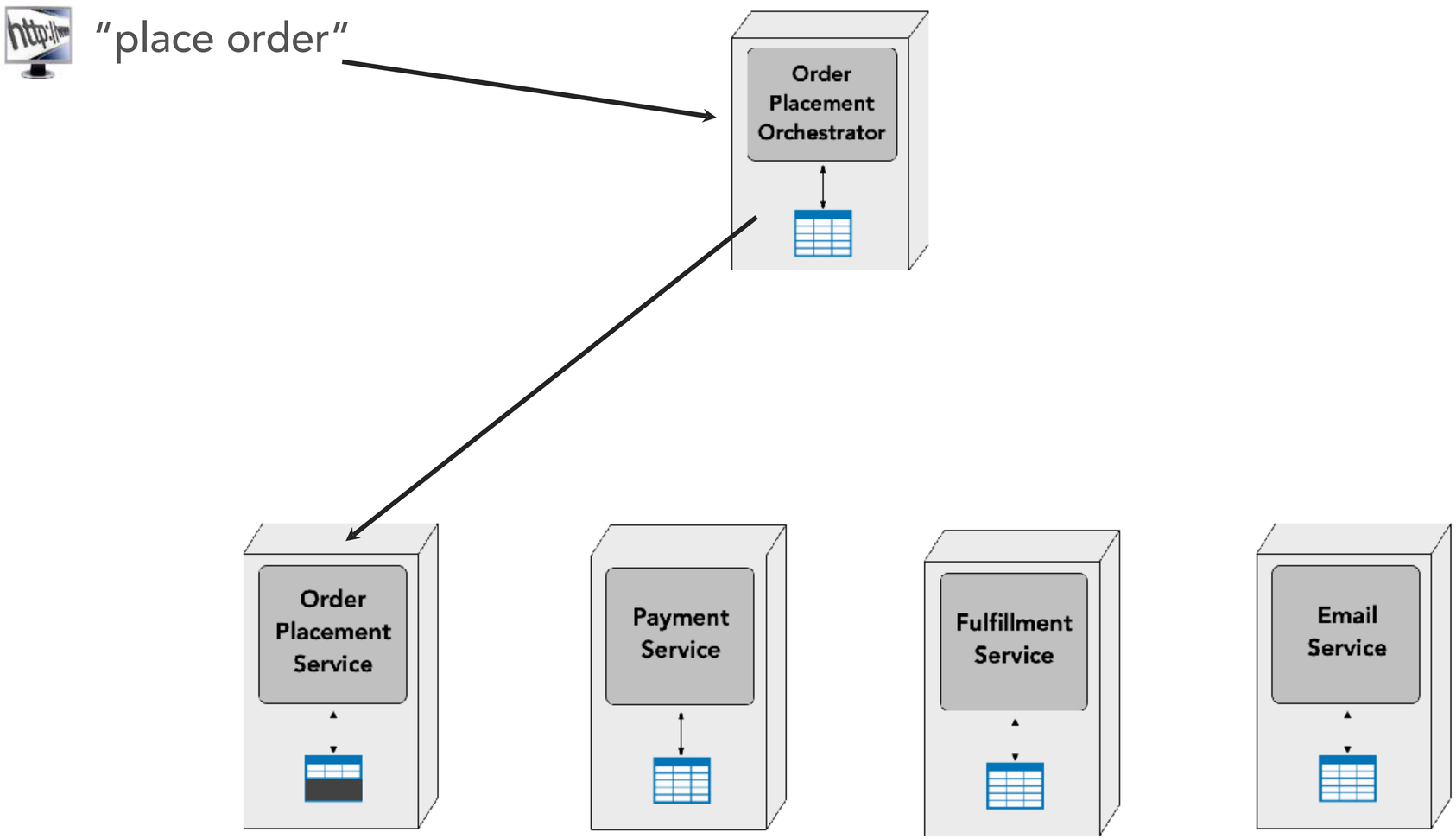
managing workflows



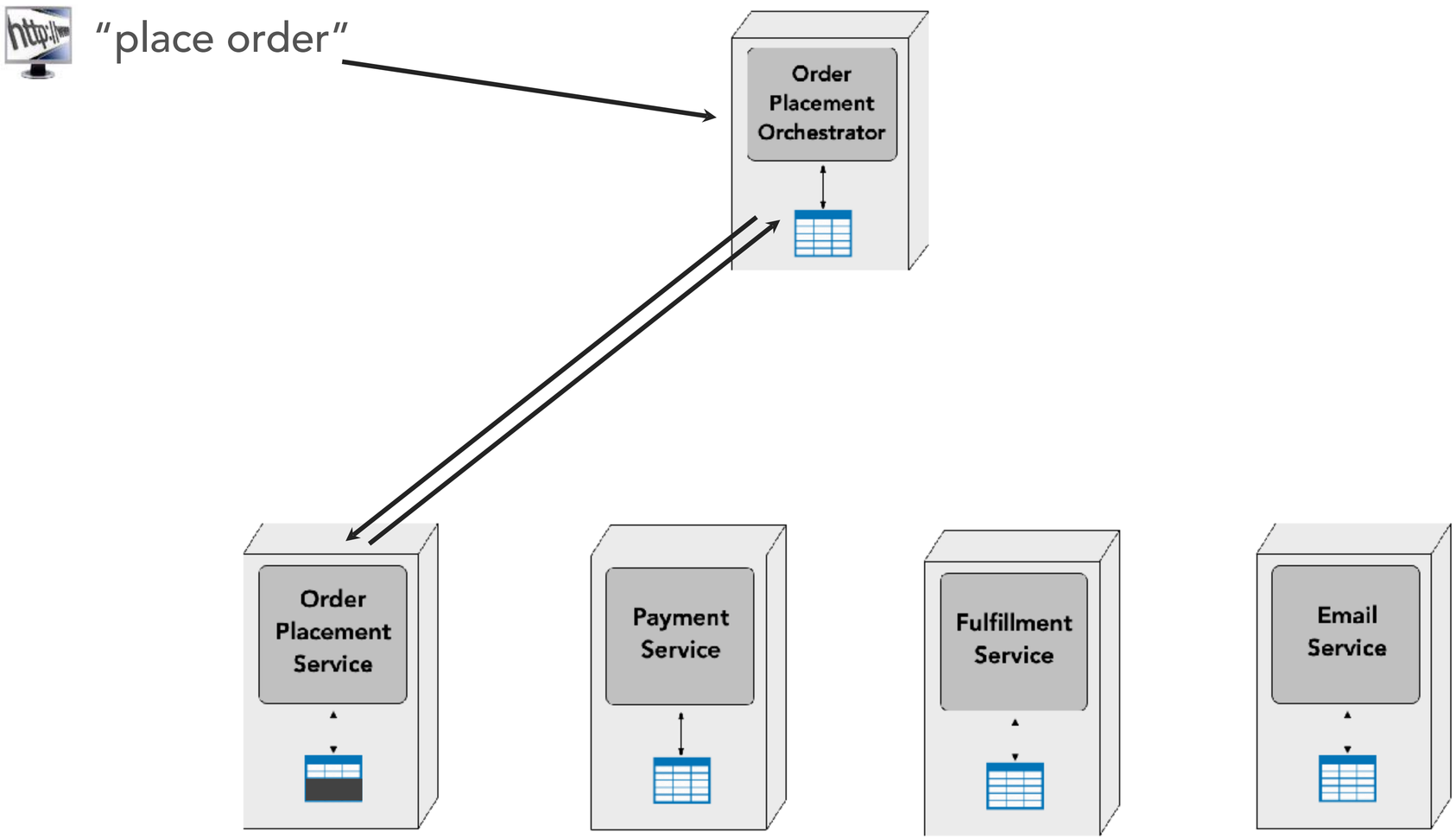
managing workflows



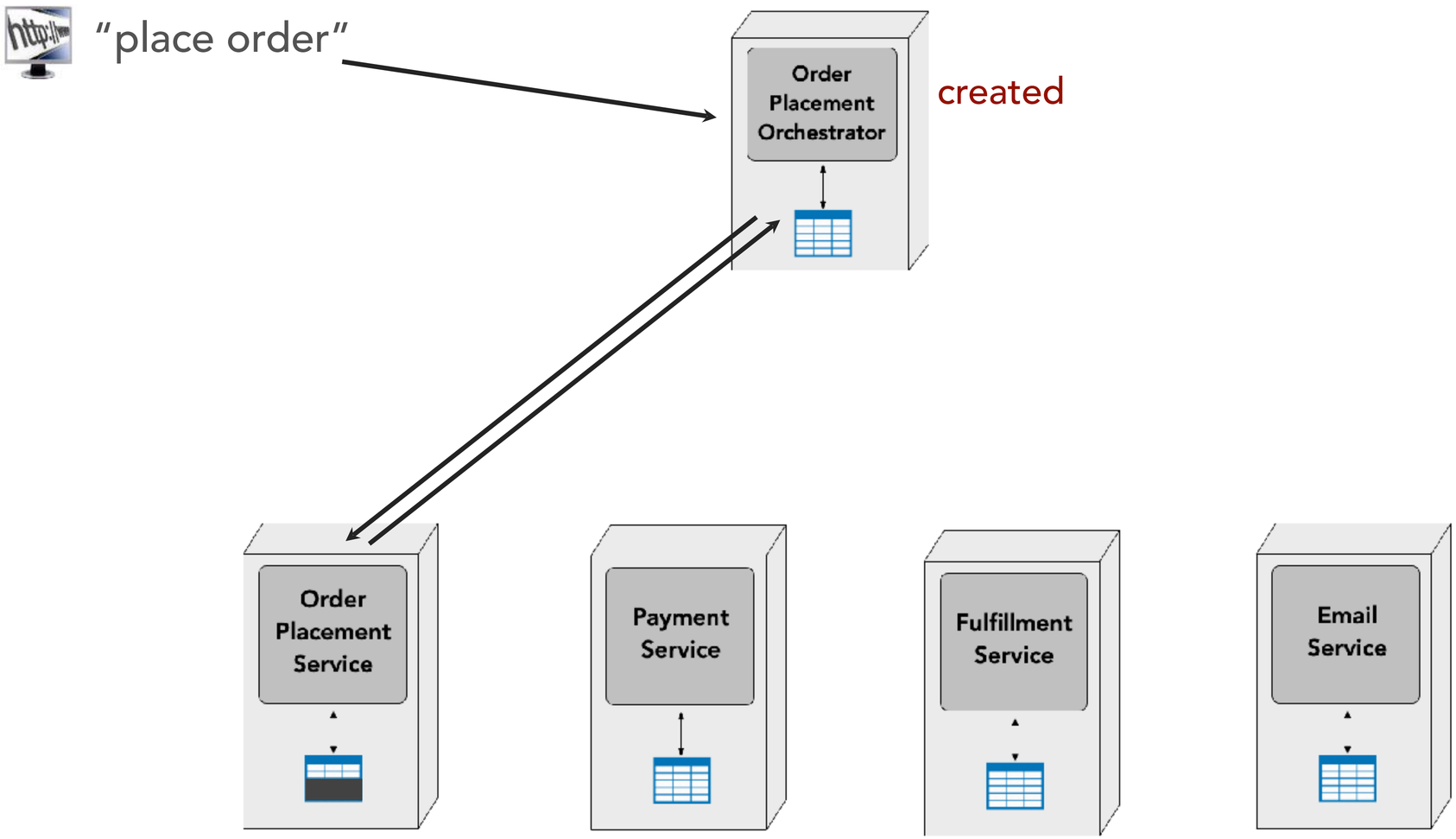
managing workflows



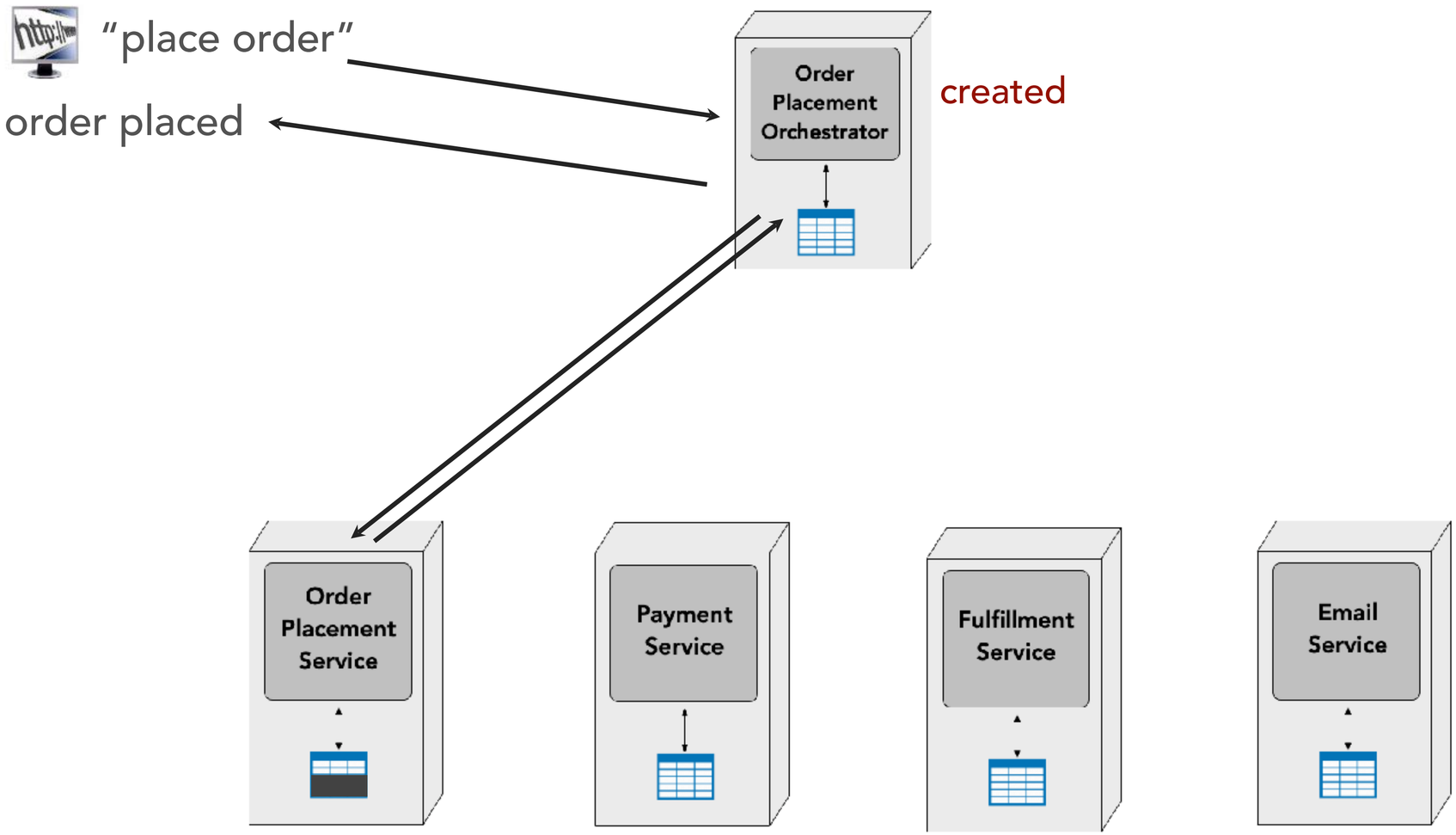
managing workflows



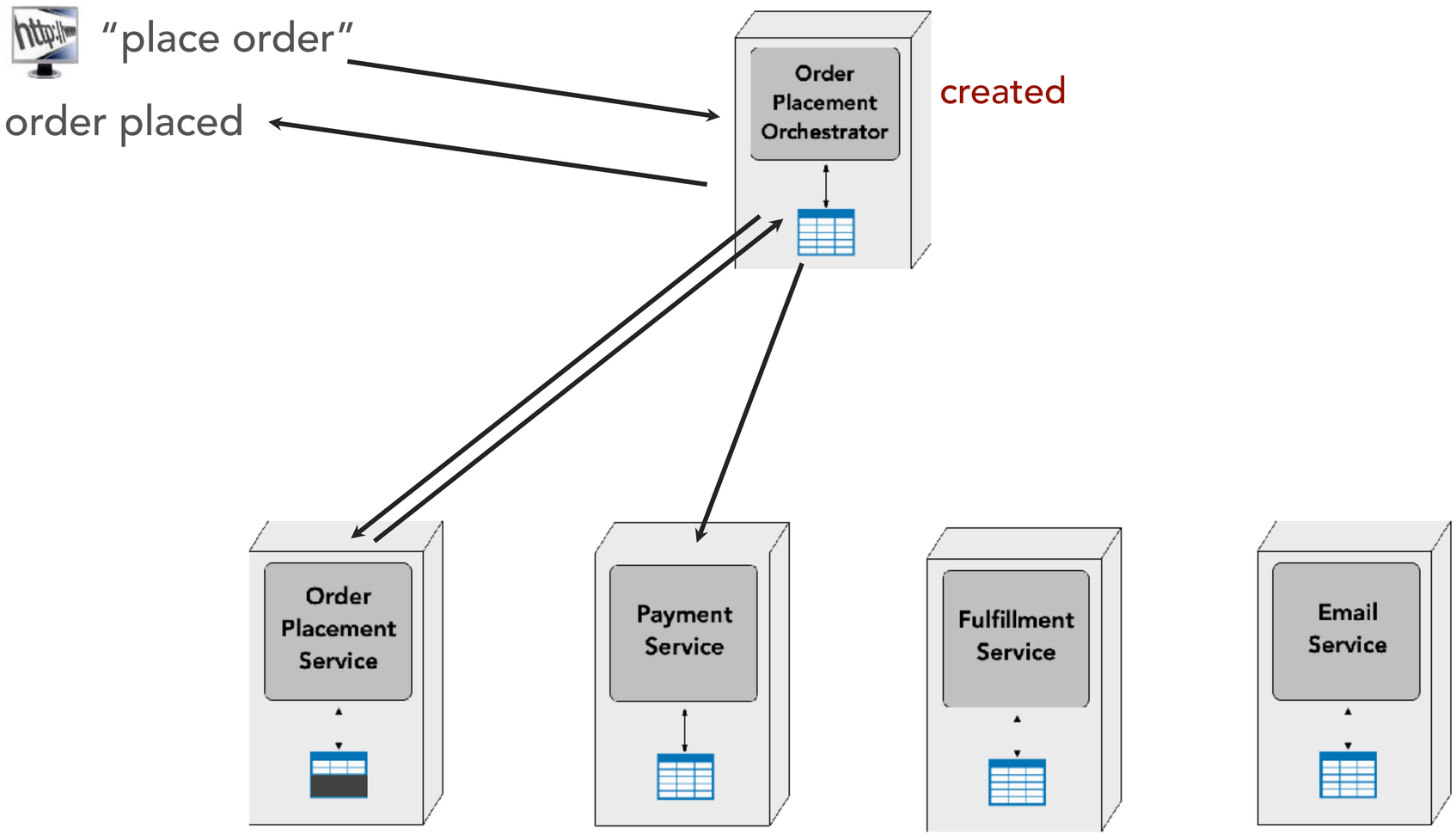
managing workflows



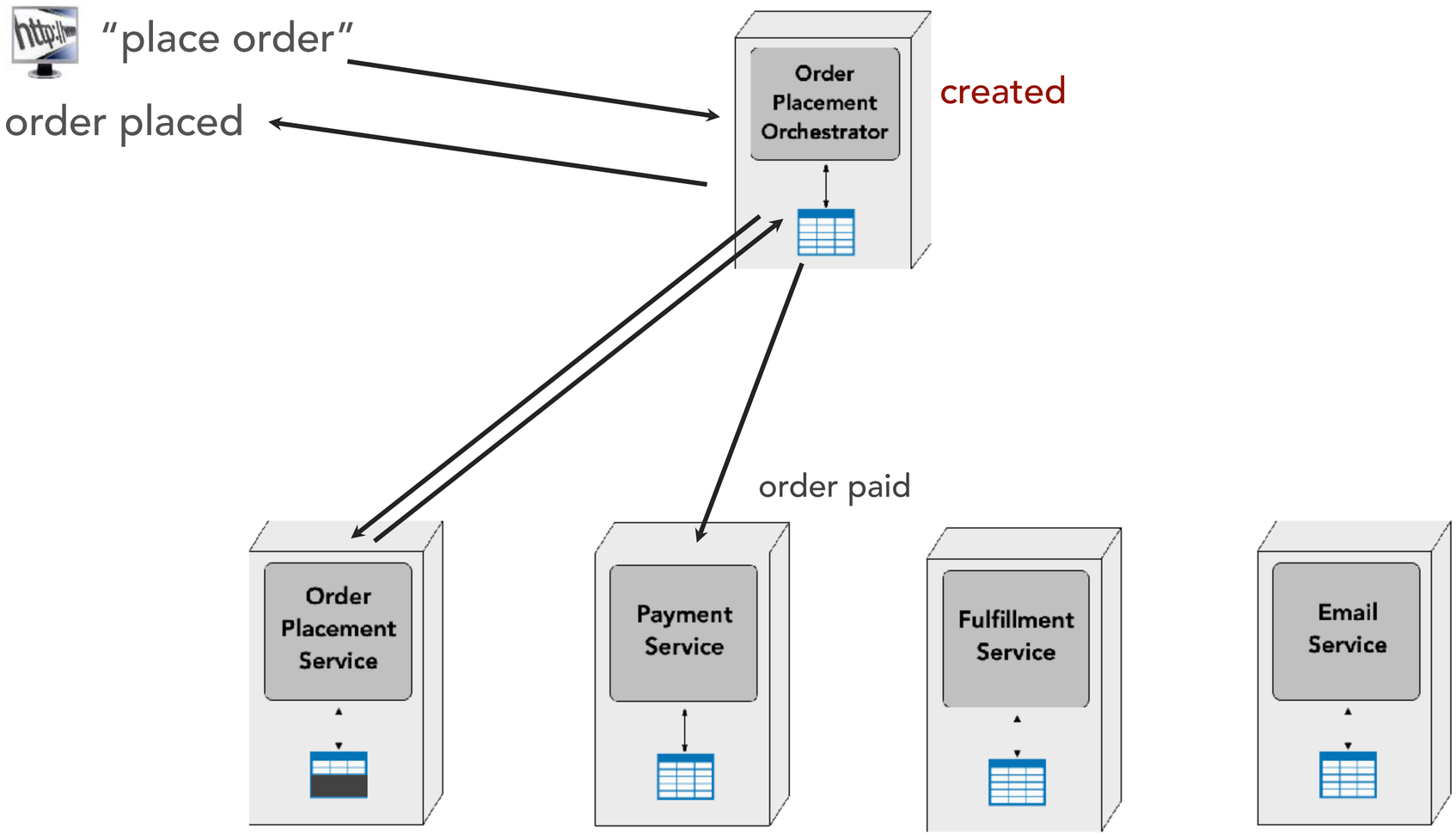
managing workflows



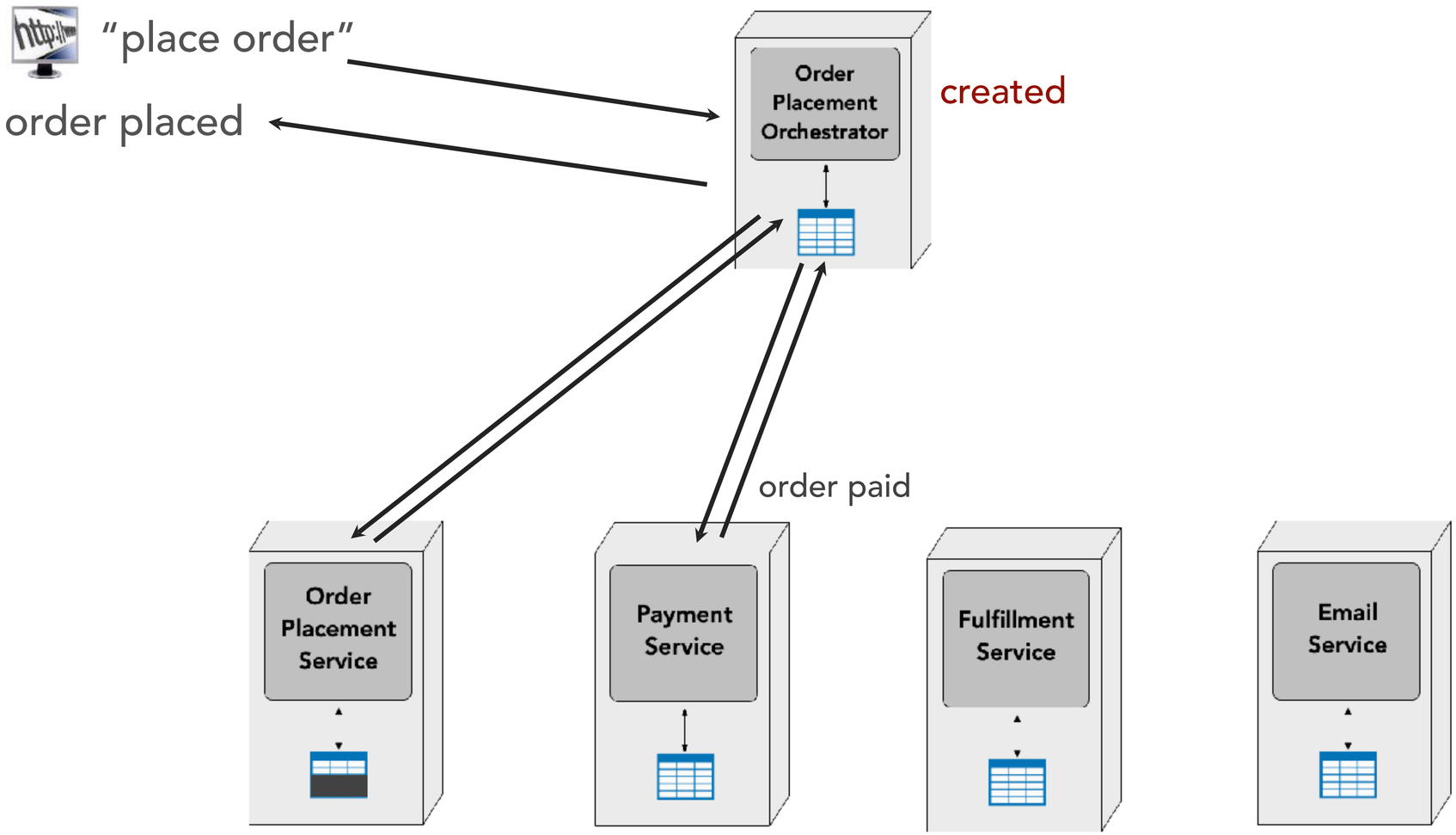
managing workflows



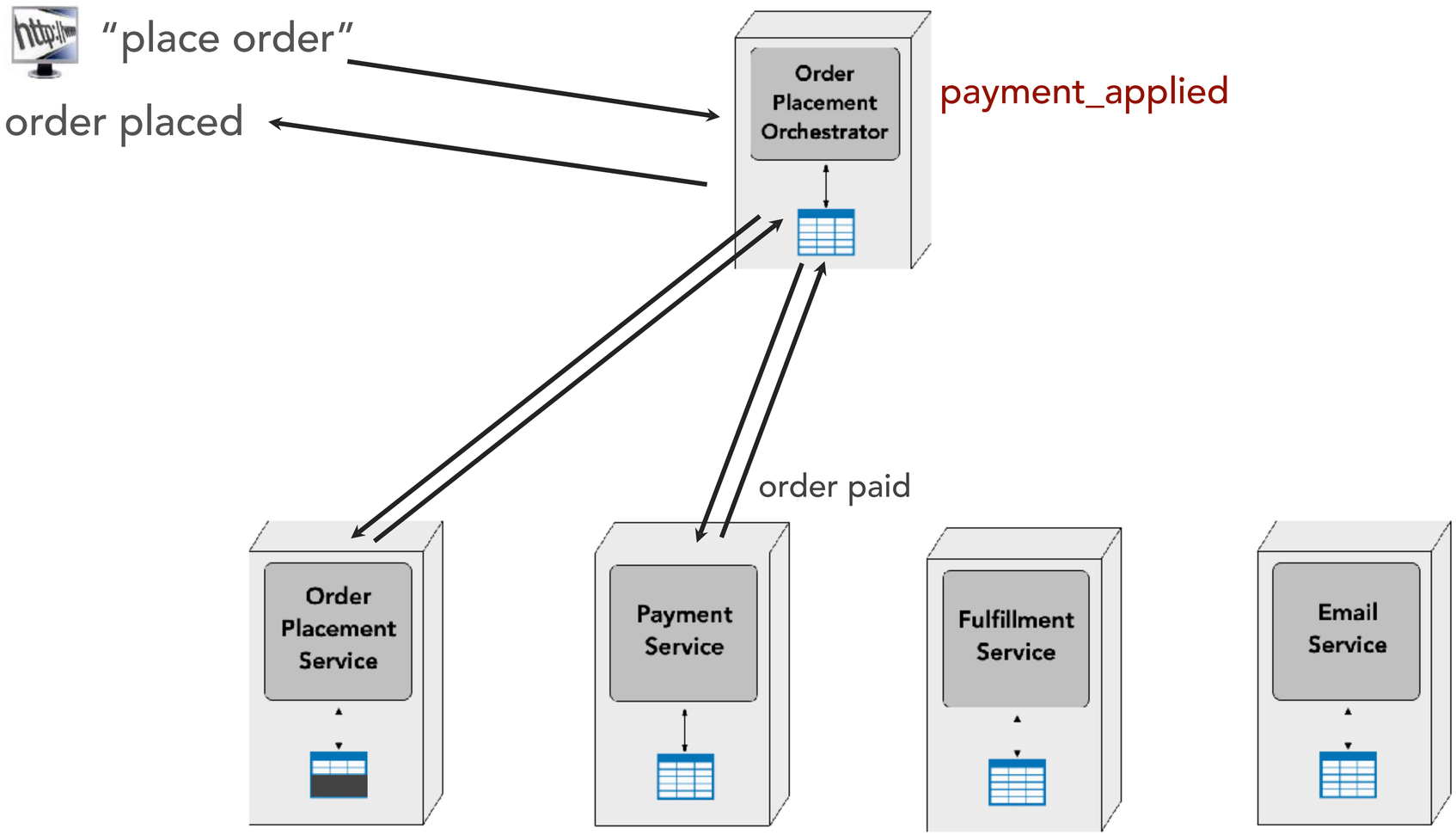
managing workflows



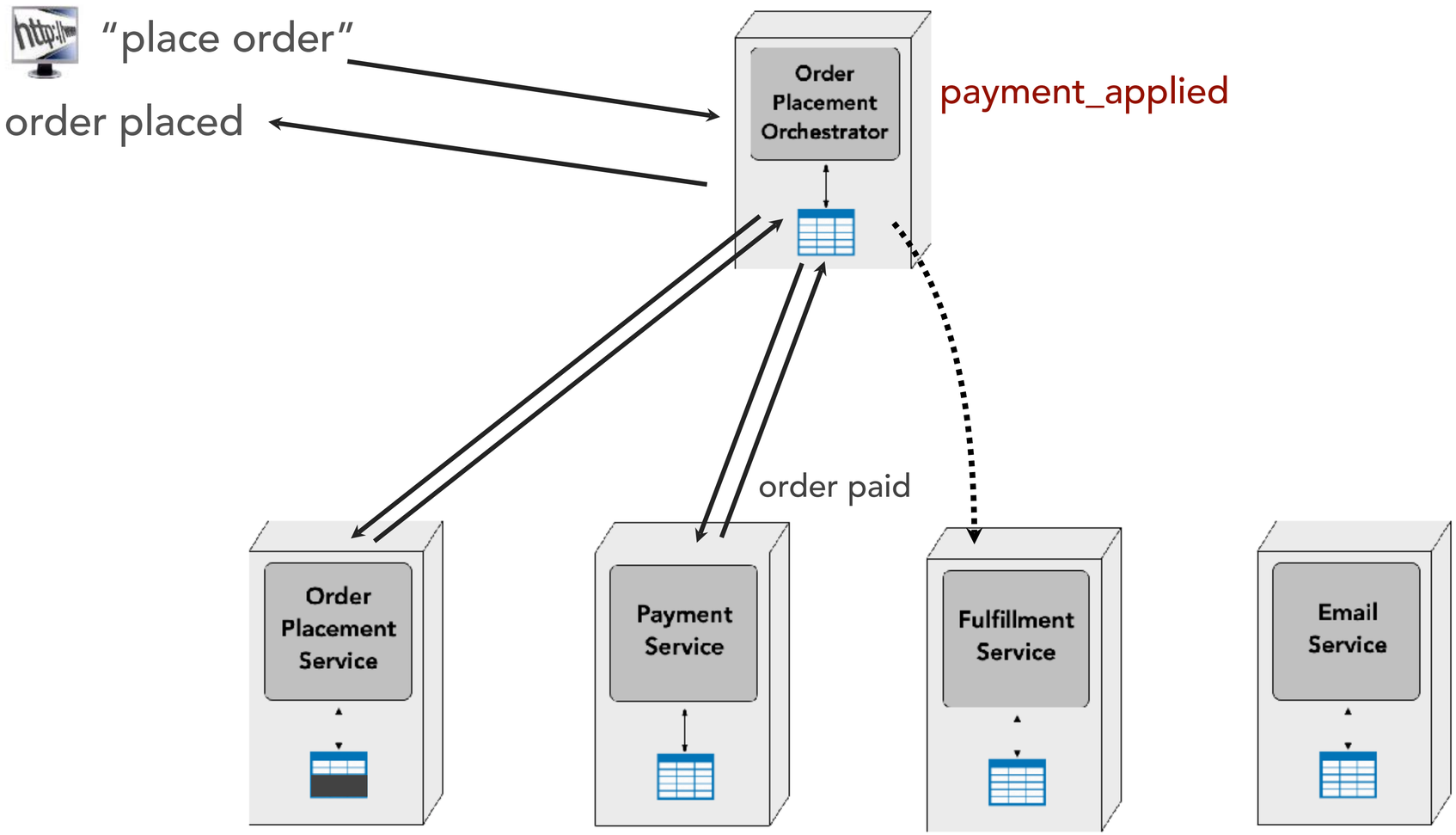
managing workflows



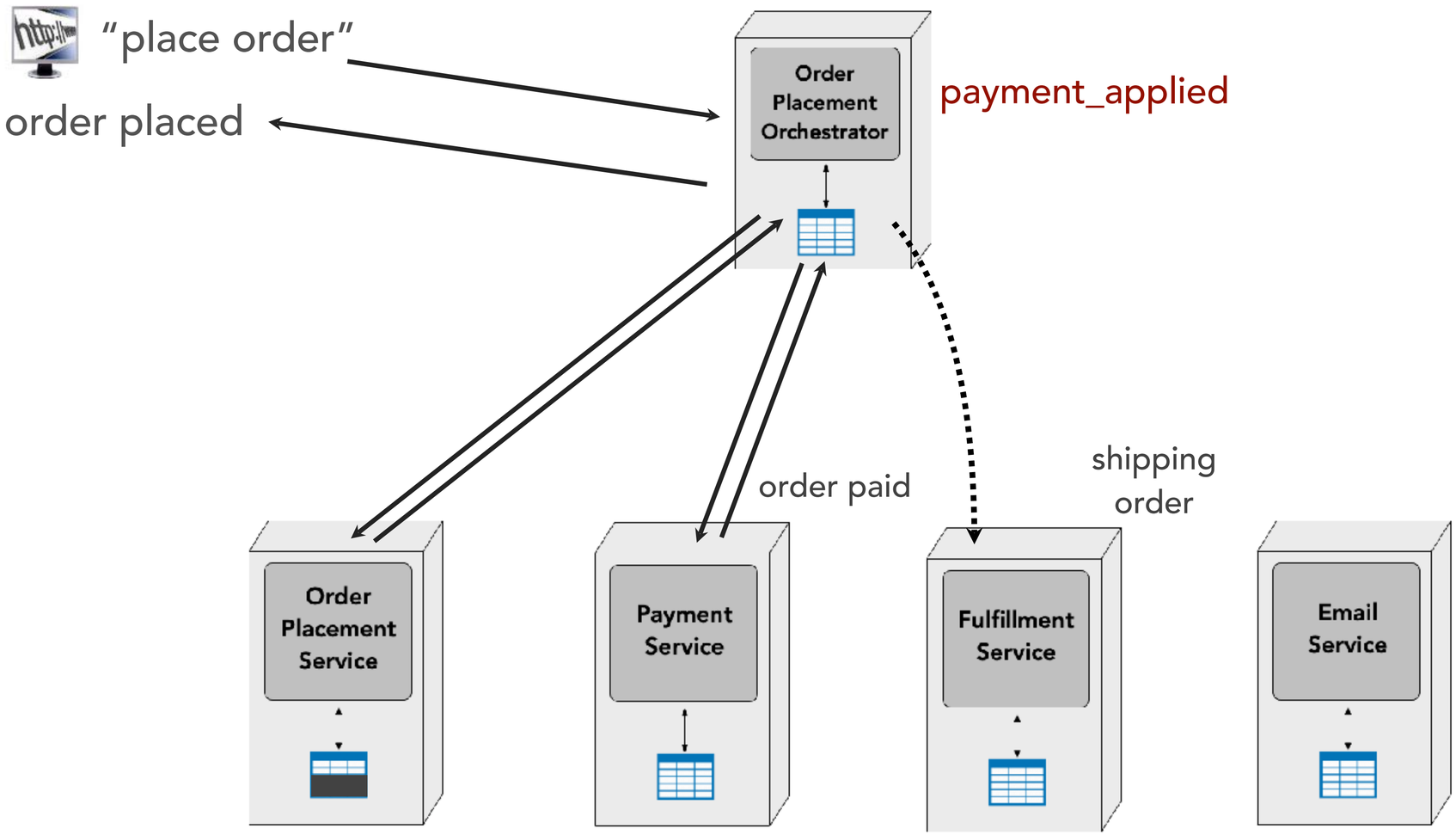
managing workflows



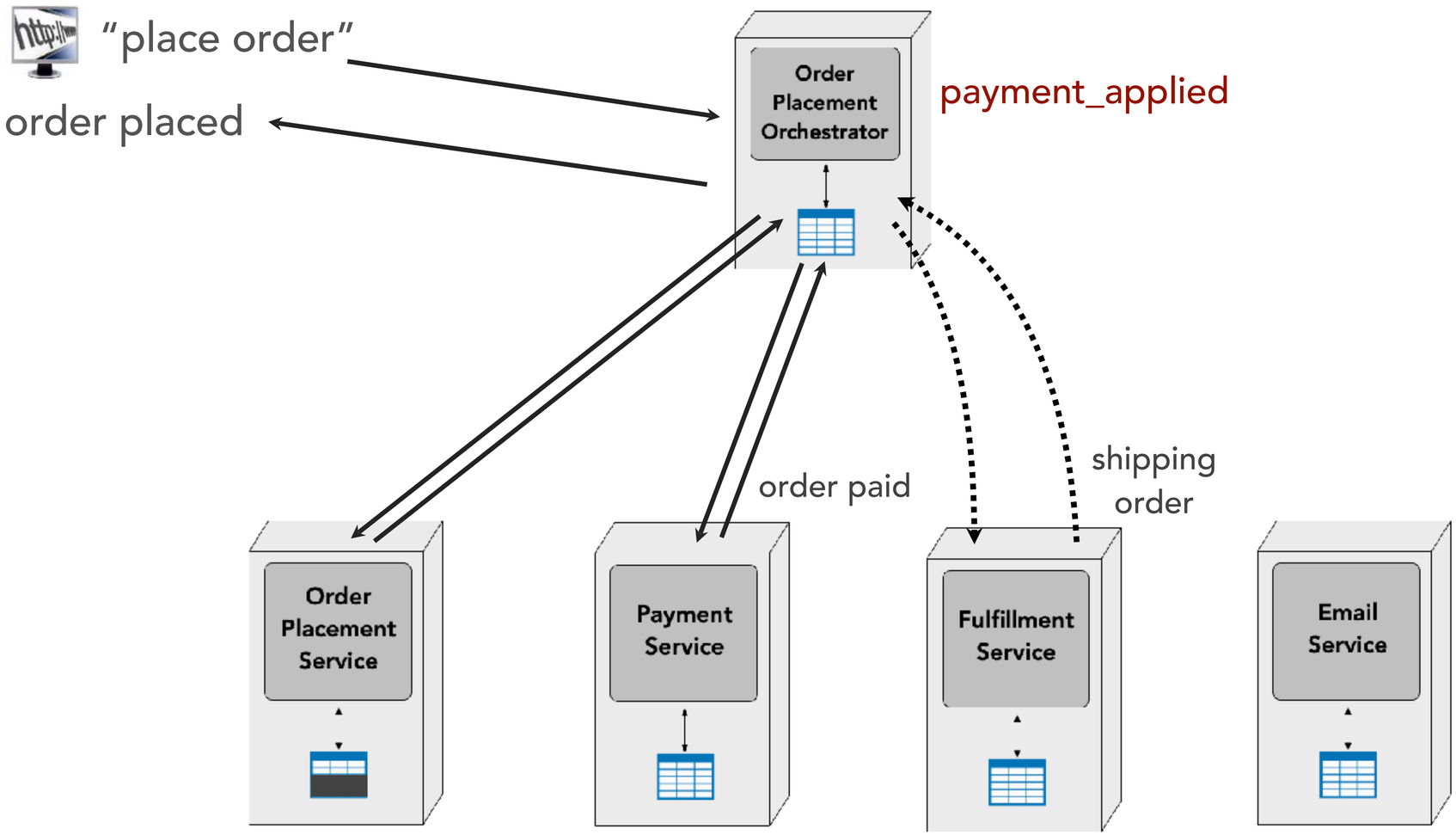
managing workflows



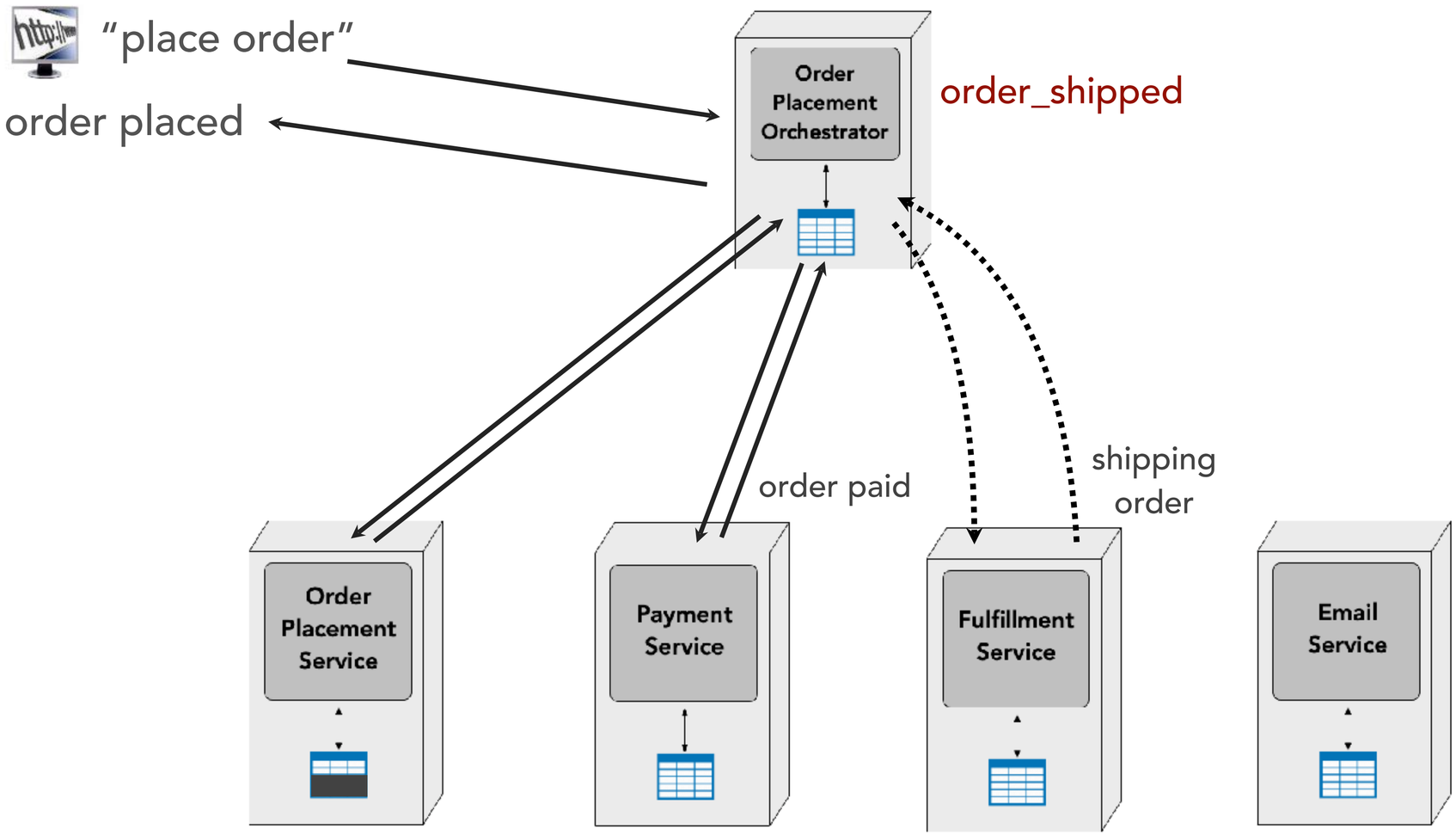
managing workflows



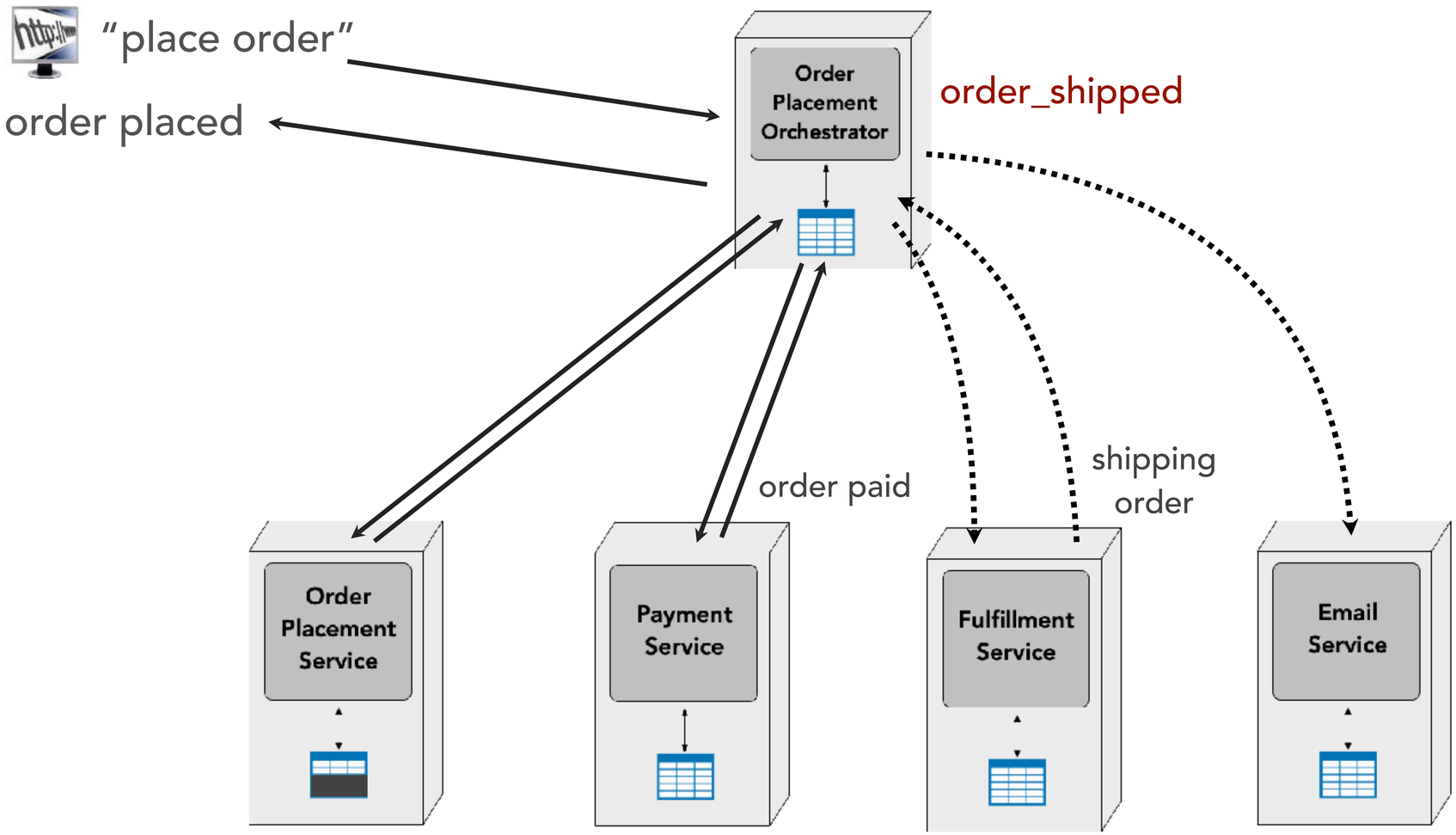
managing workflows



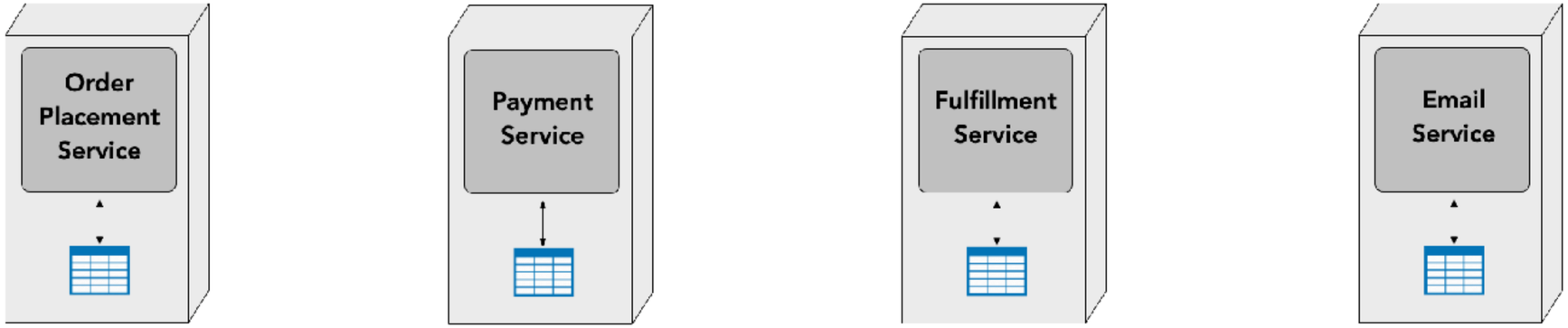
managing workflows



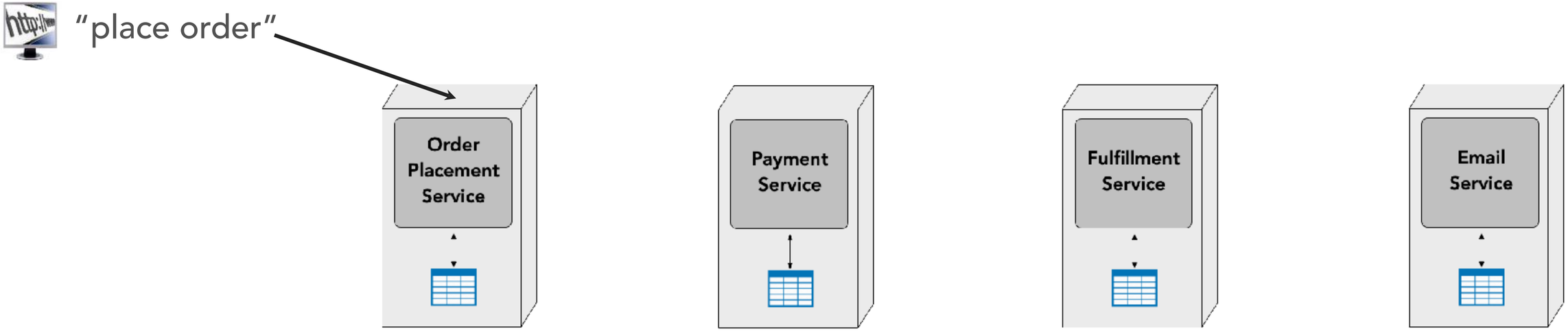
managing workflows



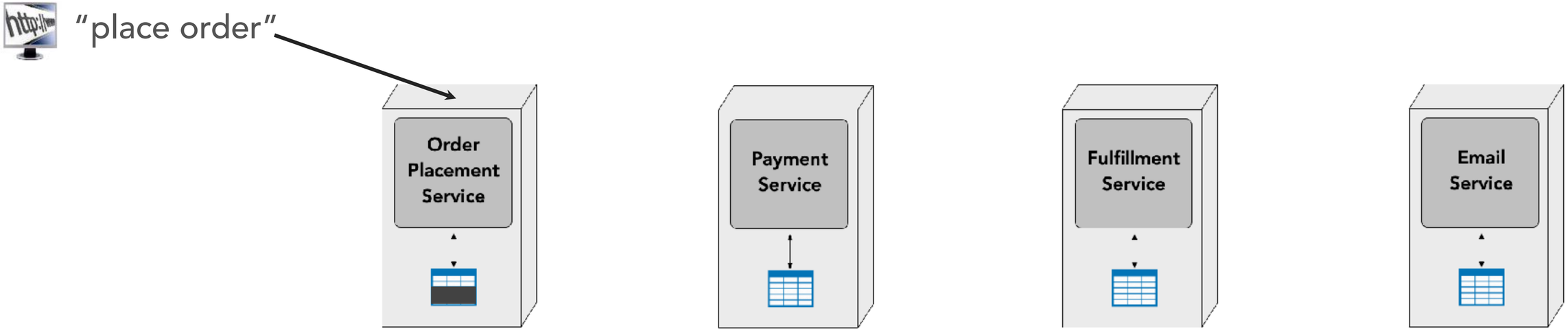
managing workflows



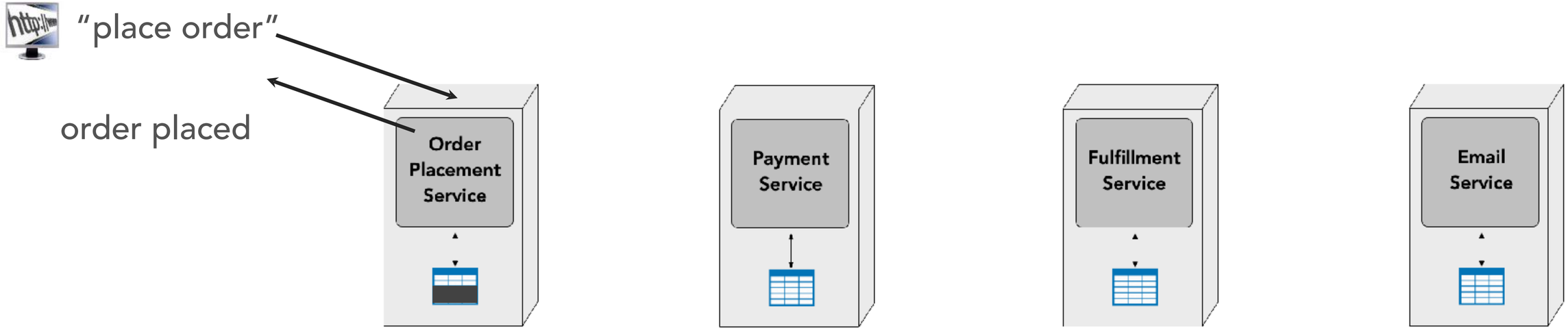
managing workflows



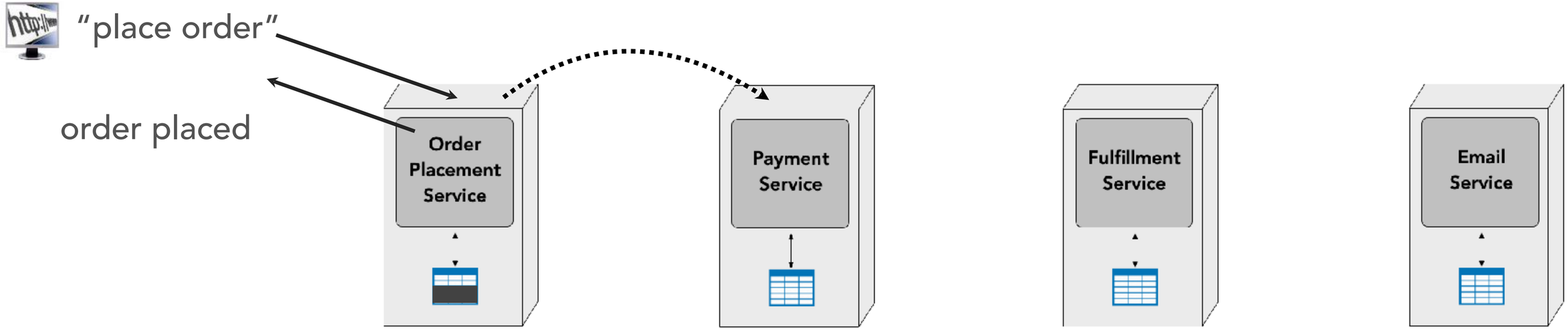
managing workflows



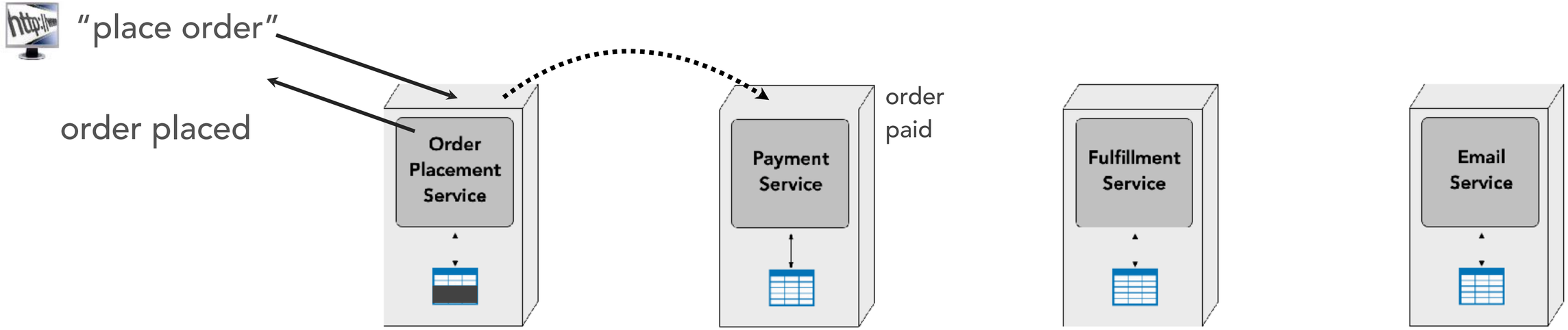
managing workflows



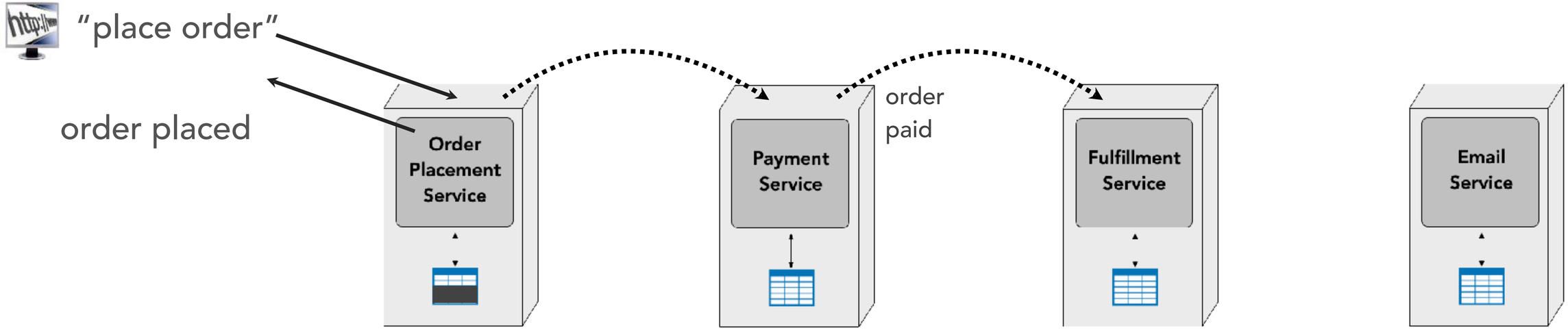
managing workflows



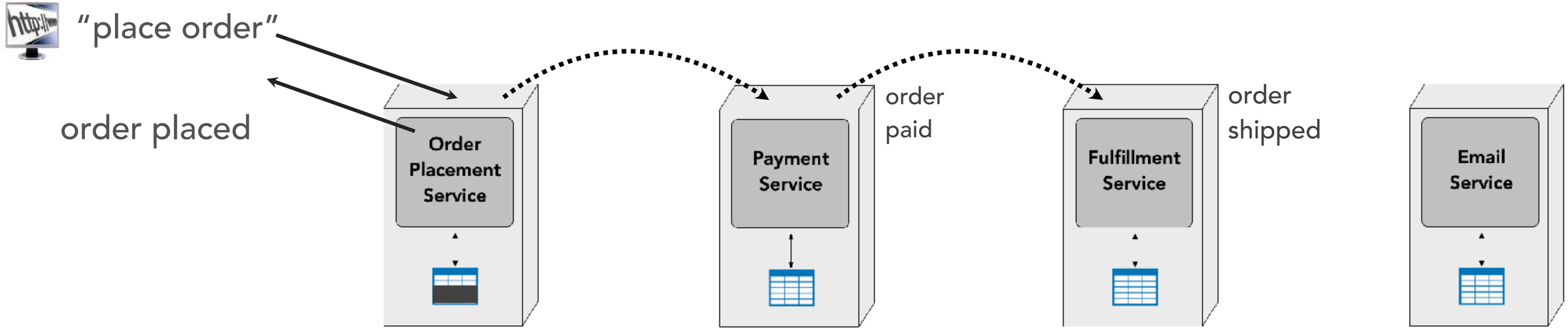
managing workflows



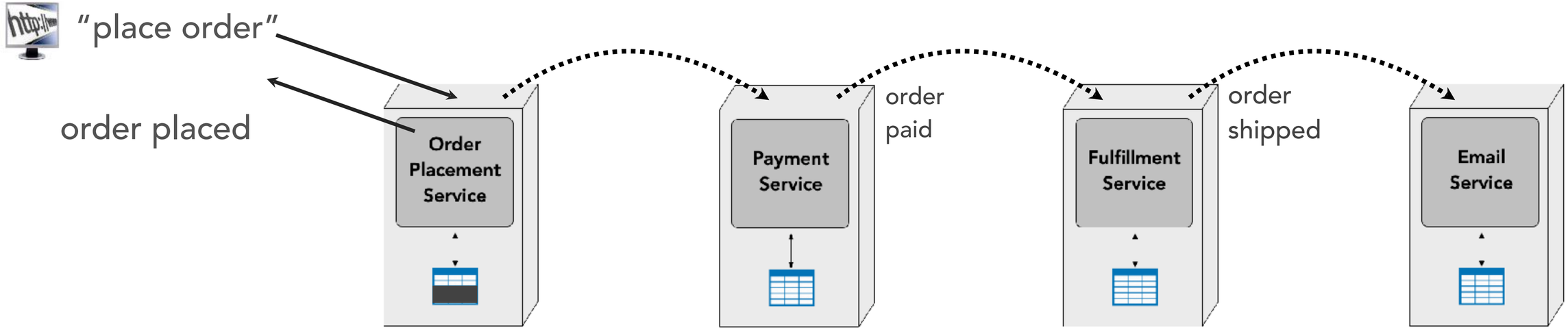
managing workflows



managing workflows

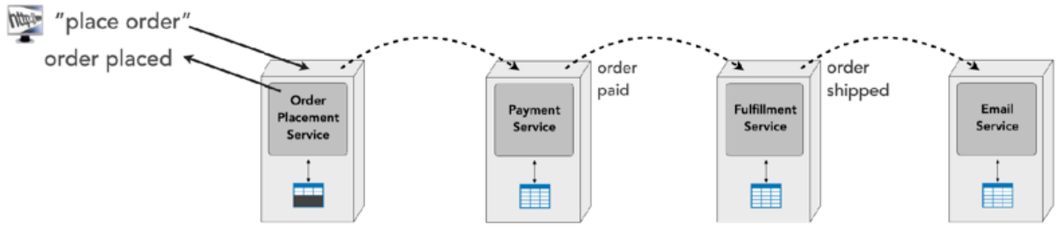


managing workflows

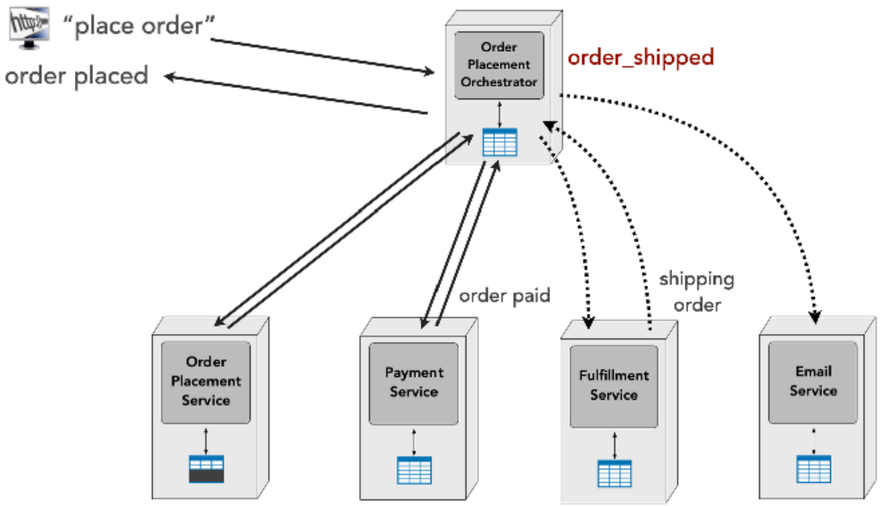


managing workflows

choreography

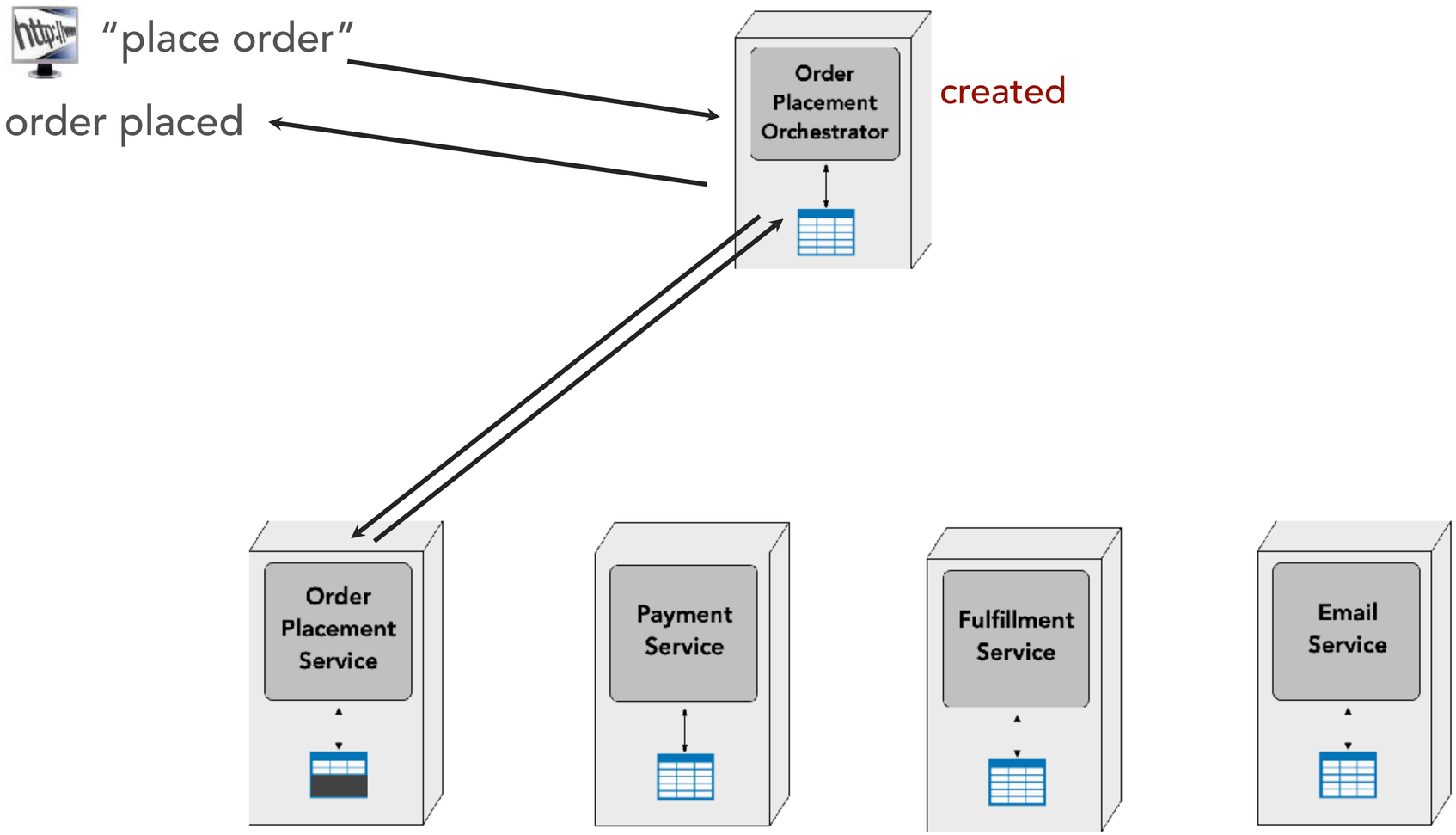


orchestration



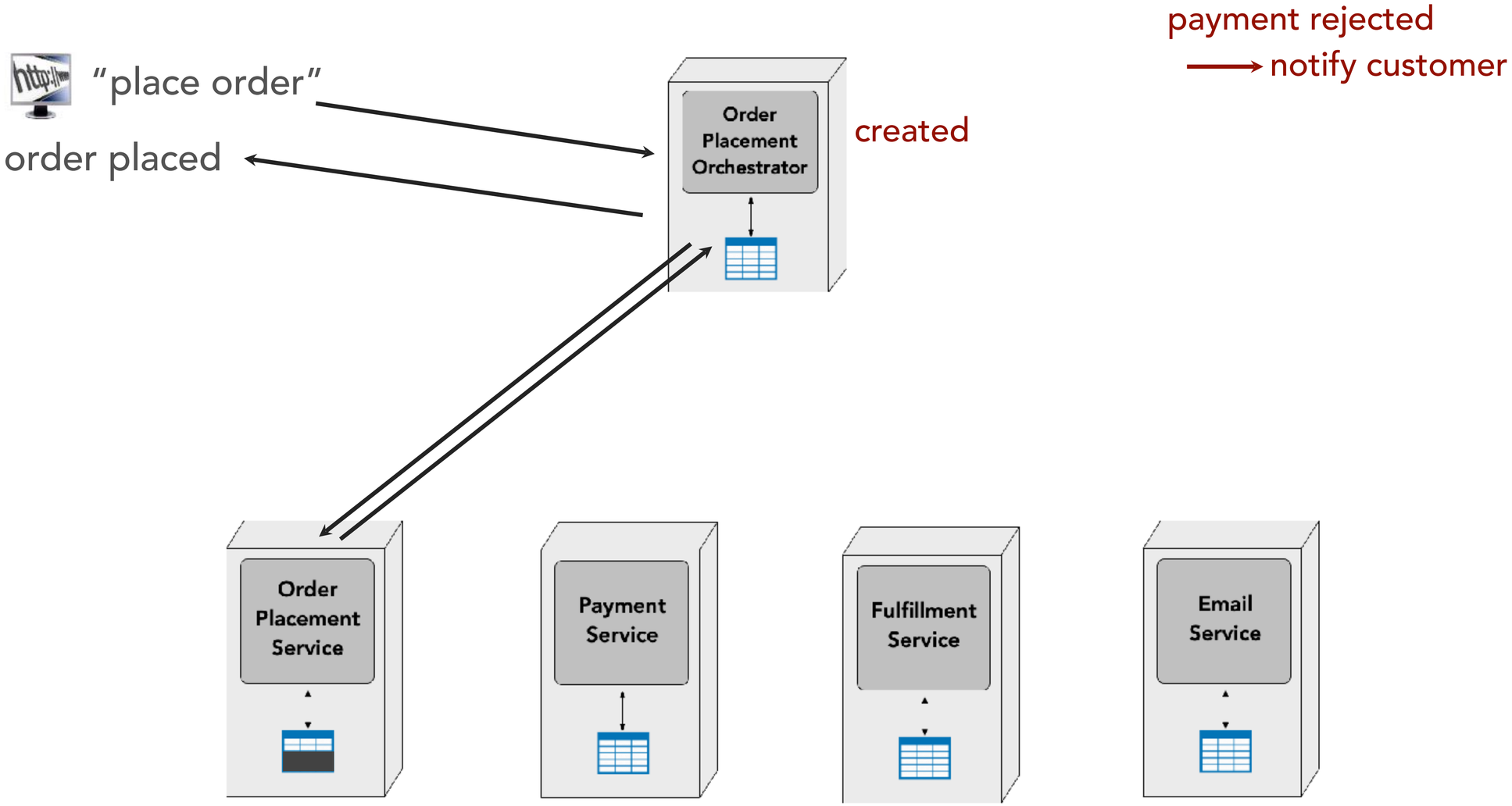
managing workflows

error handling



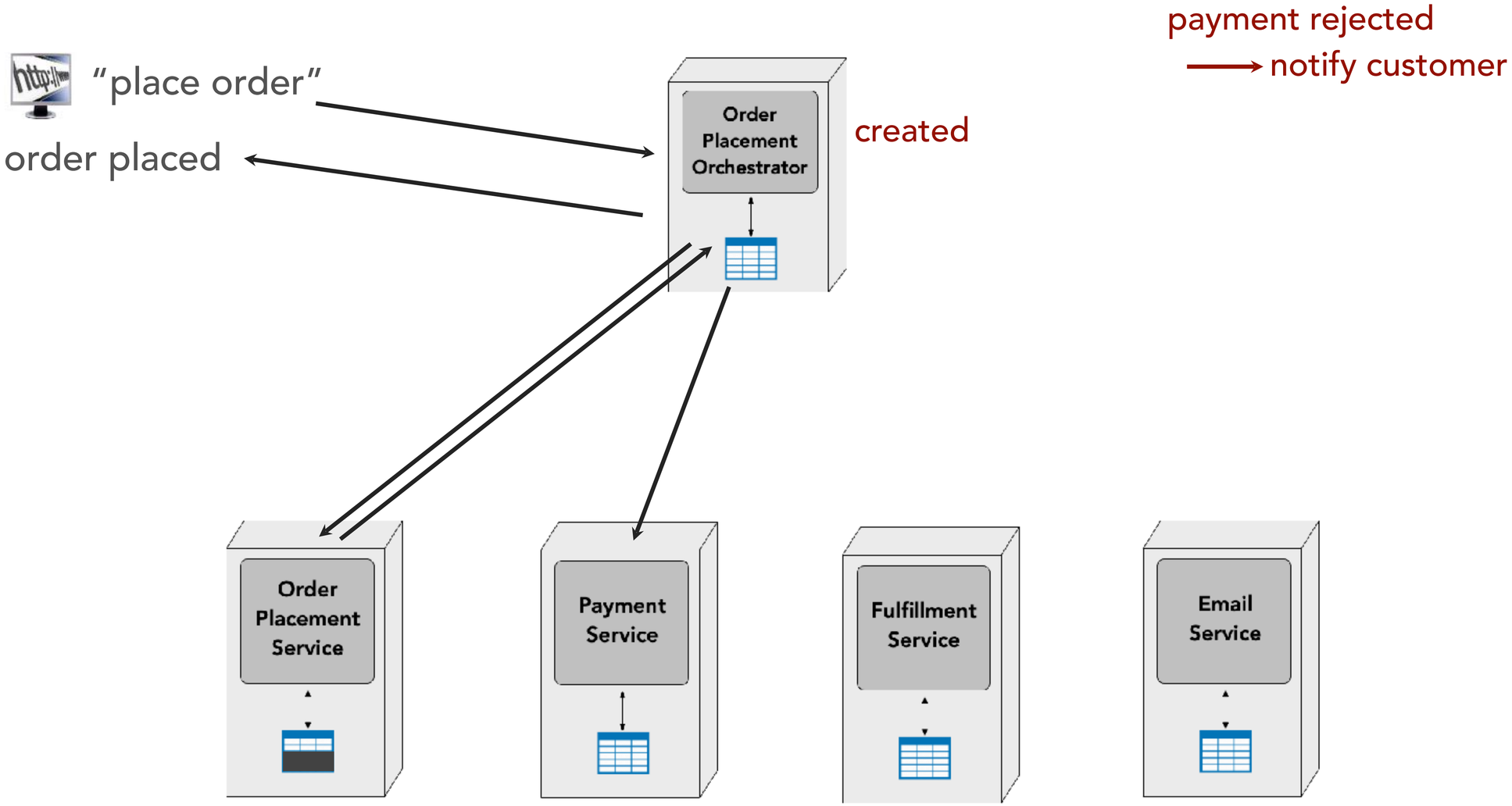
managing workflows

error handling



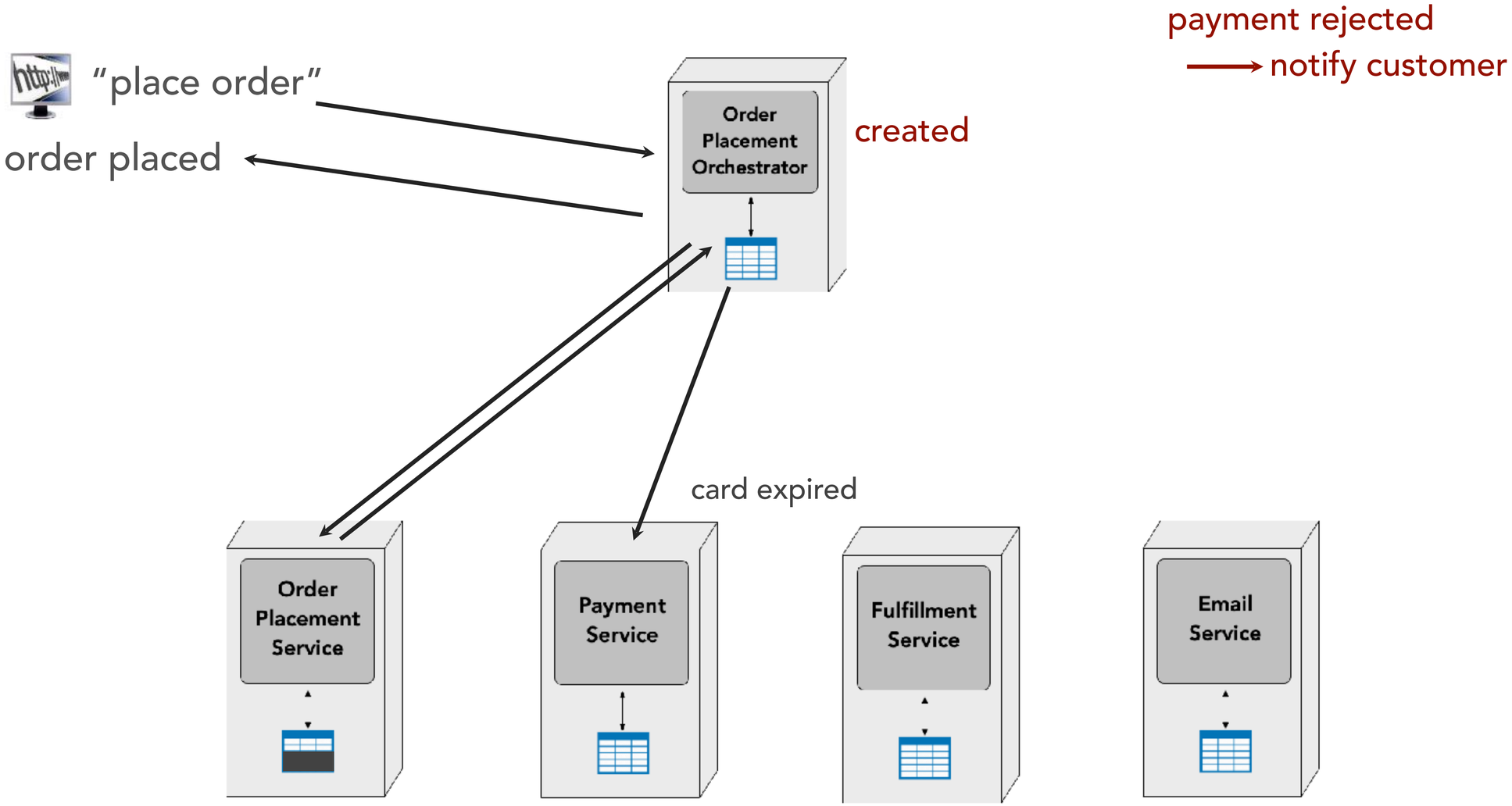
managing workflows

error handling



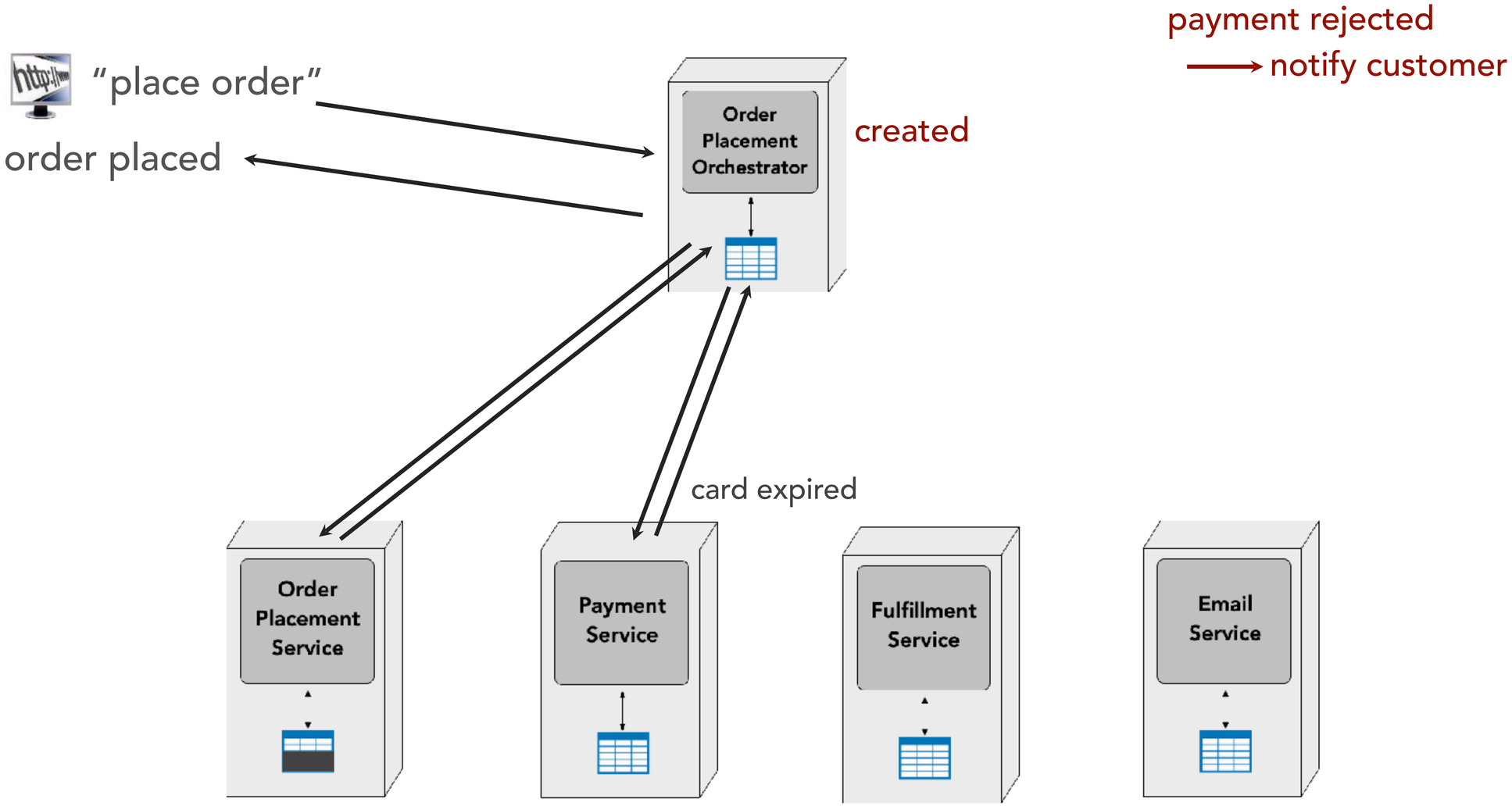
managing workflows

error handling



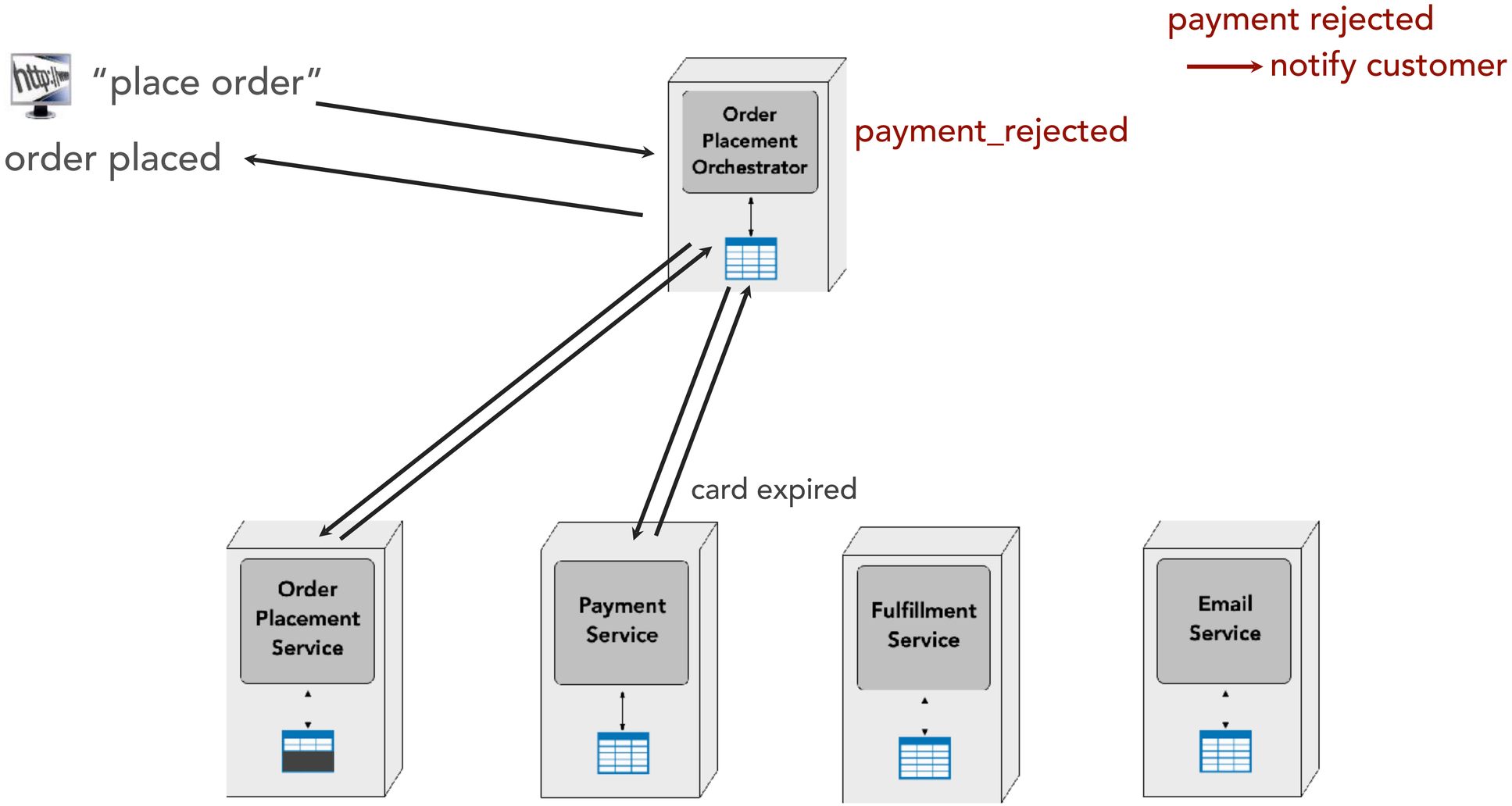
managing workflows

error handling



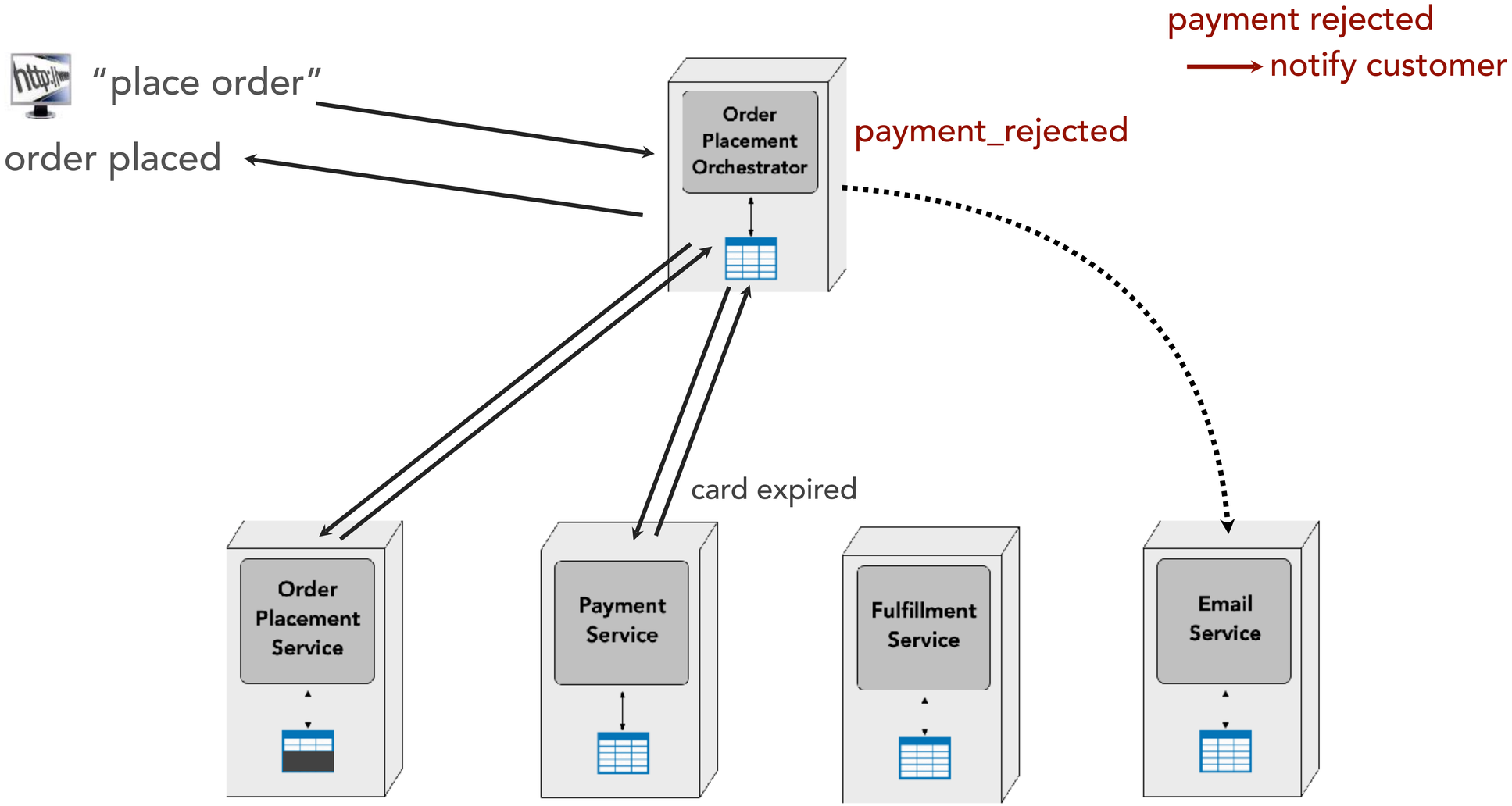
managing workflows

error handling



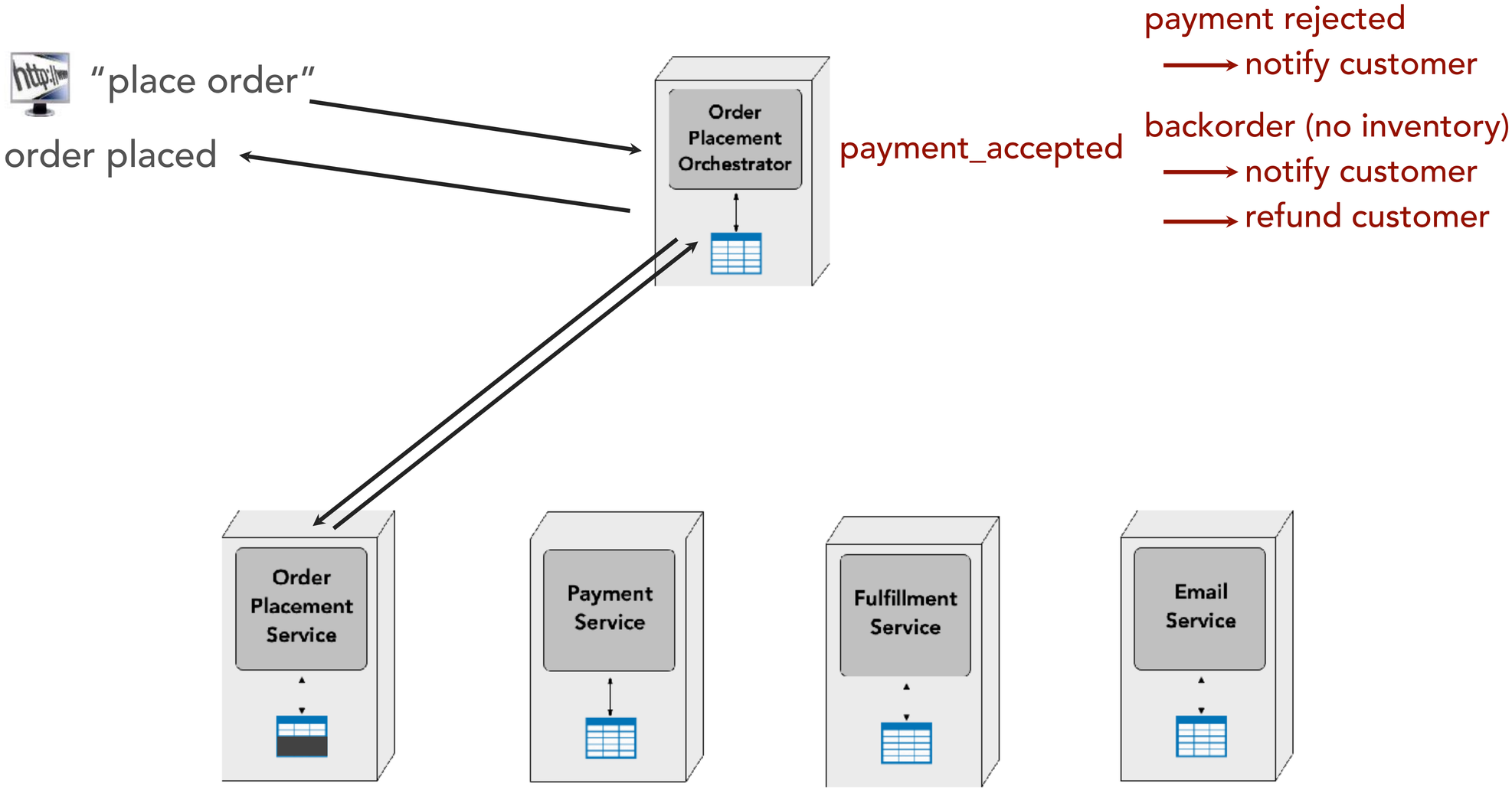
managing workflows

error handling



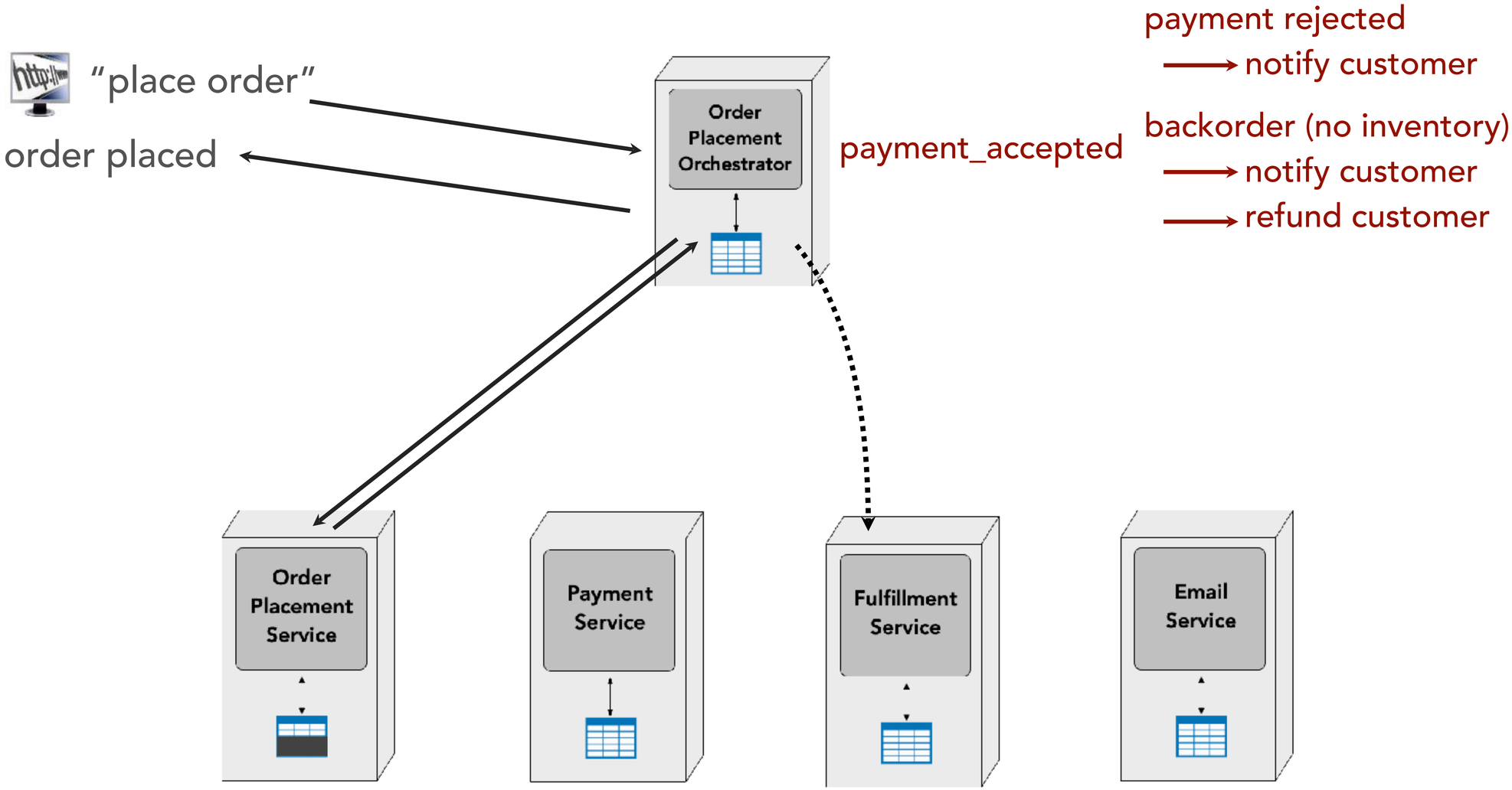
managing workflows

error handling



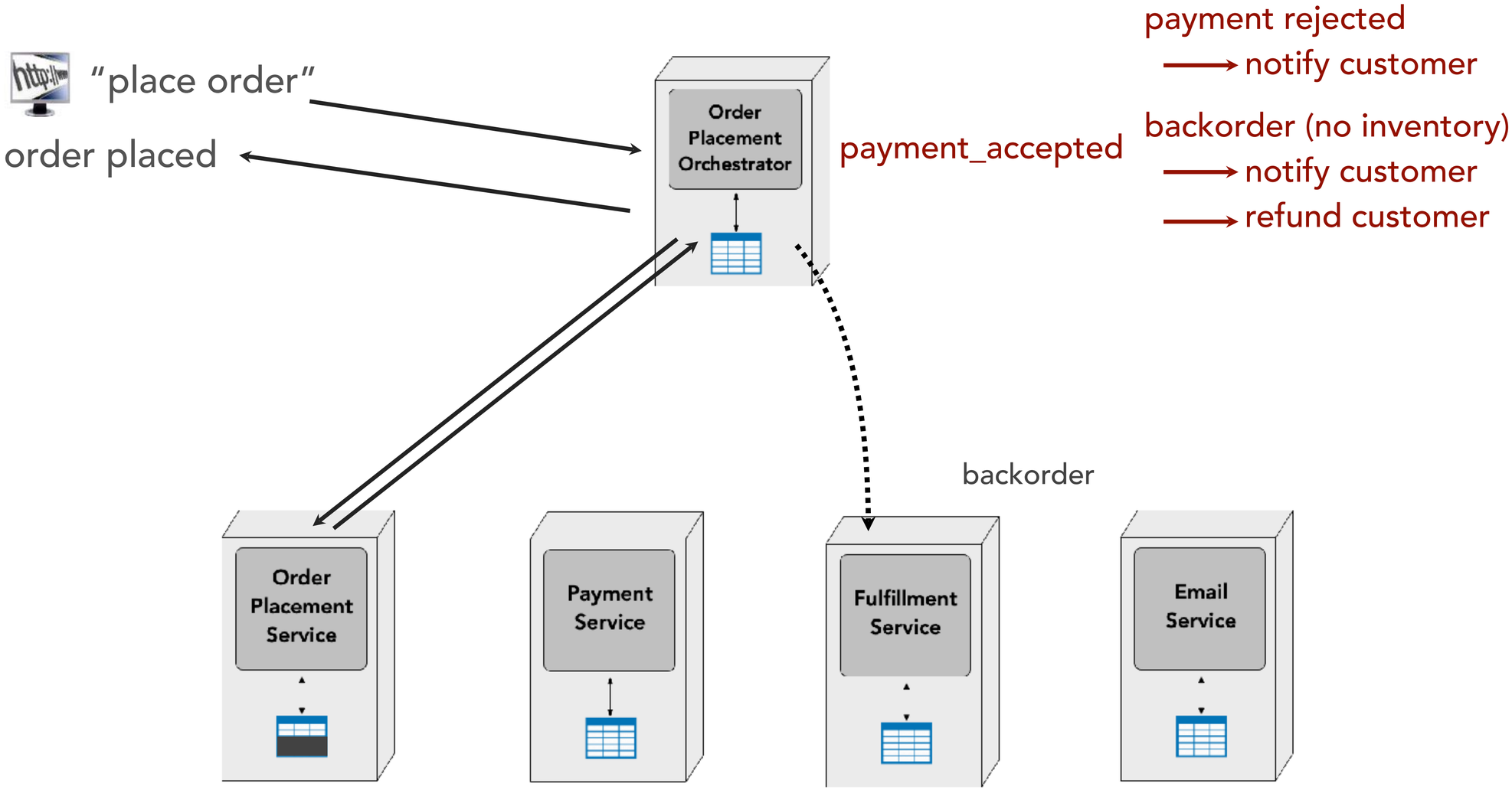
managing workflows

error handling



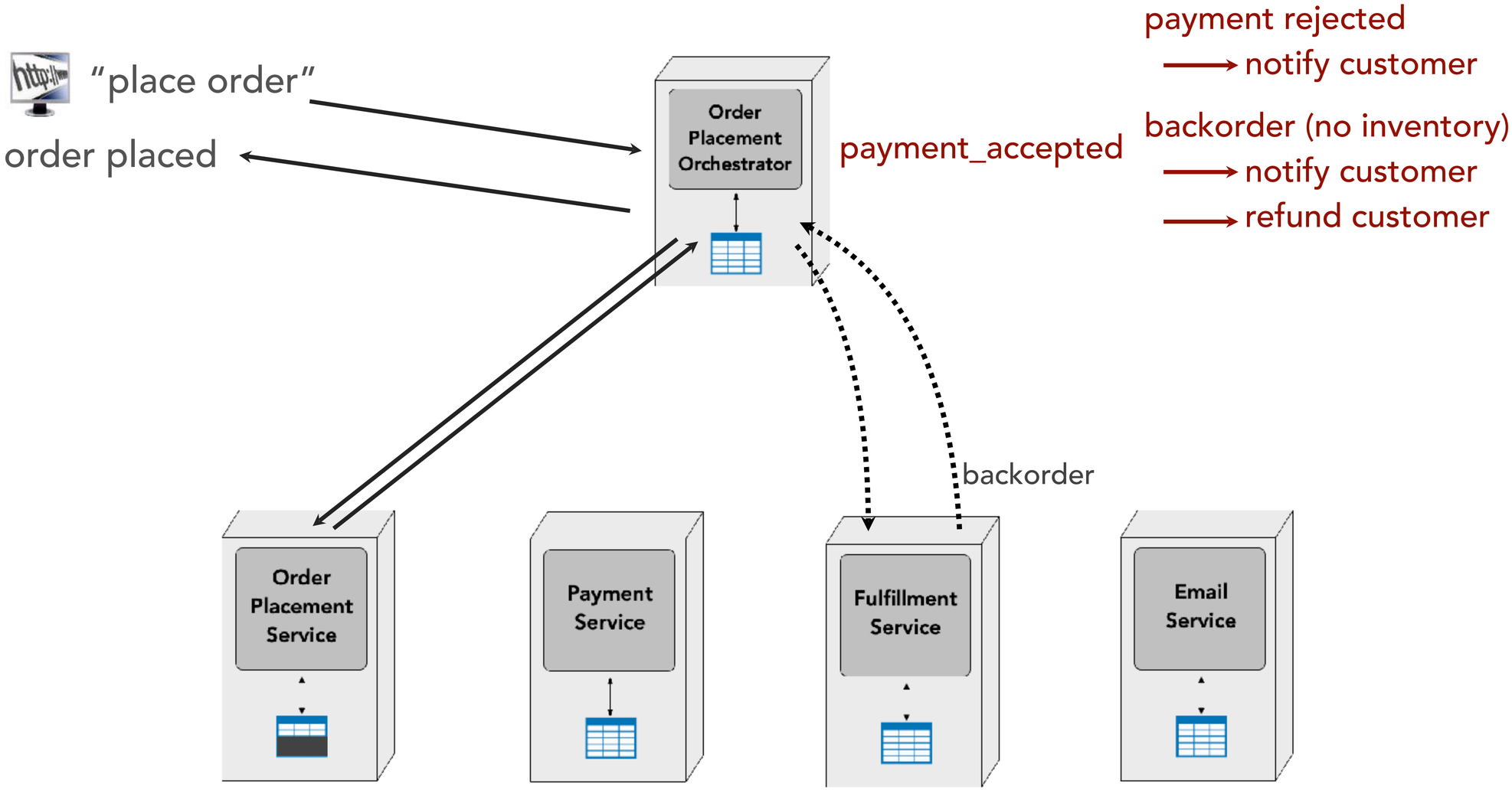
managing workflows

error handling



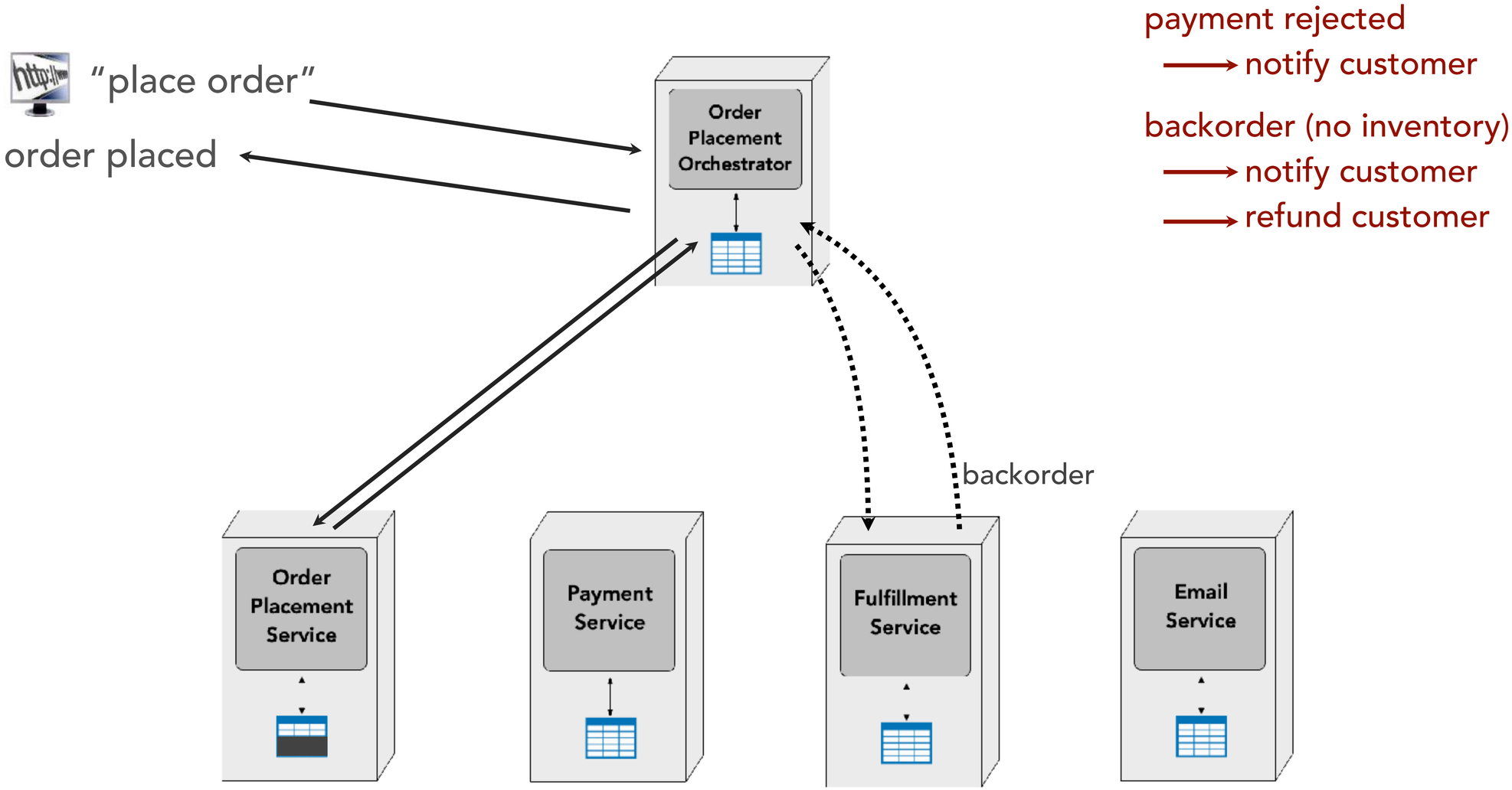
managing workflows

error handling



managing workflows

error handling

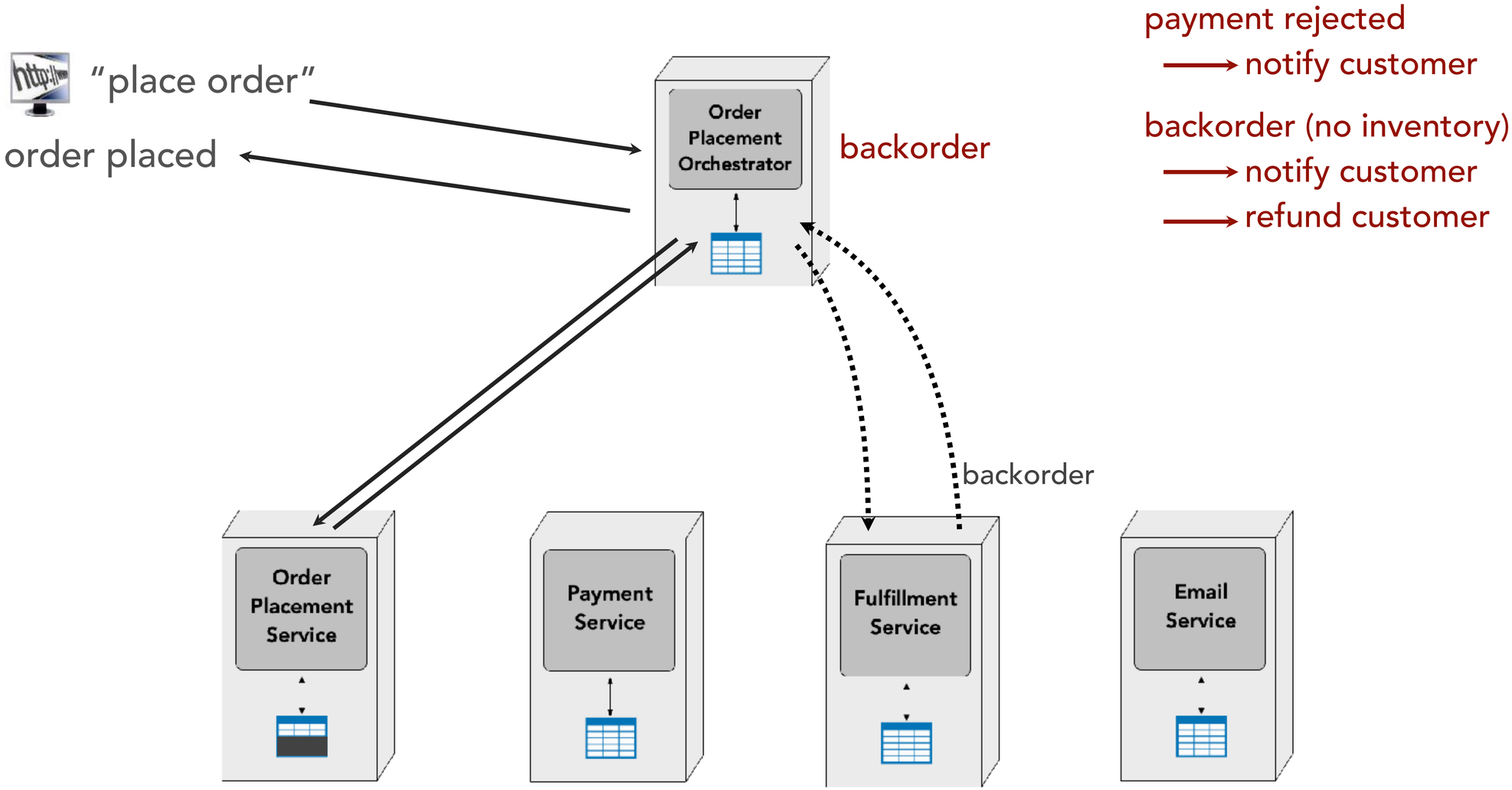


payment rejected
→ notify customer

backorder (no inventory)
→ notify customer
→ refund customer

managing workflows

error handling

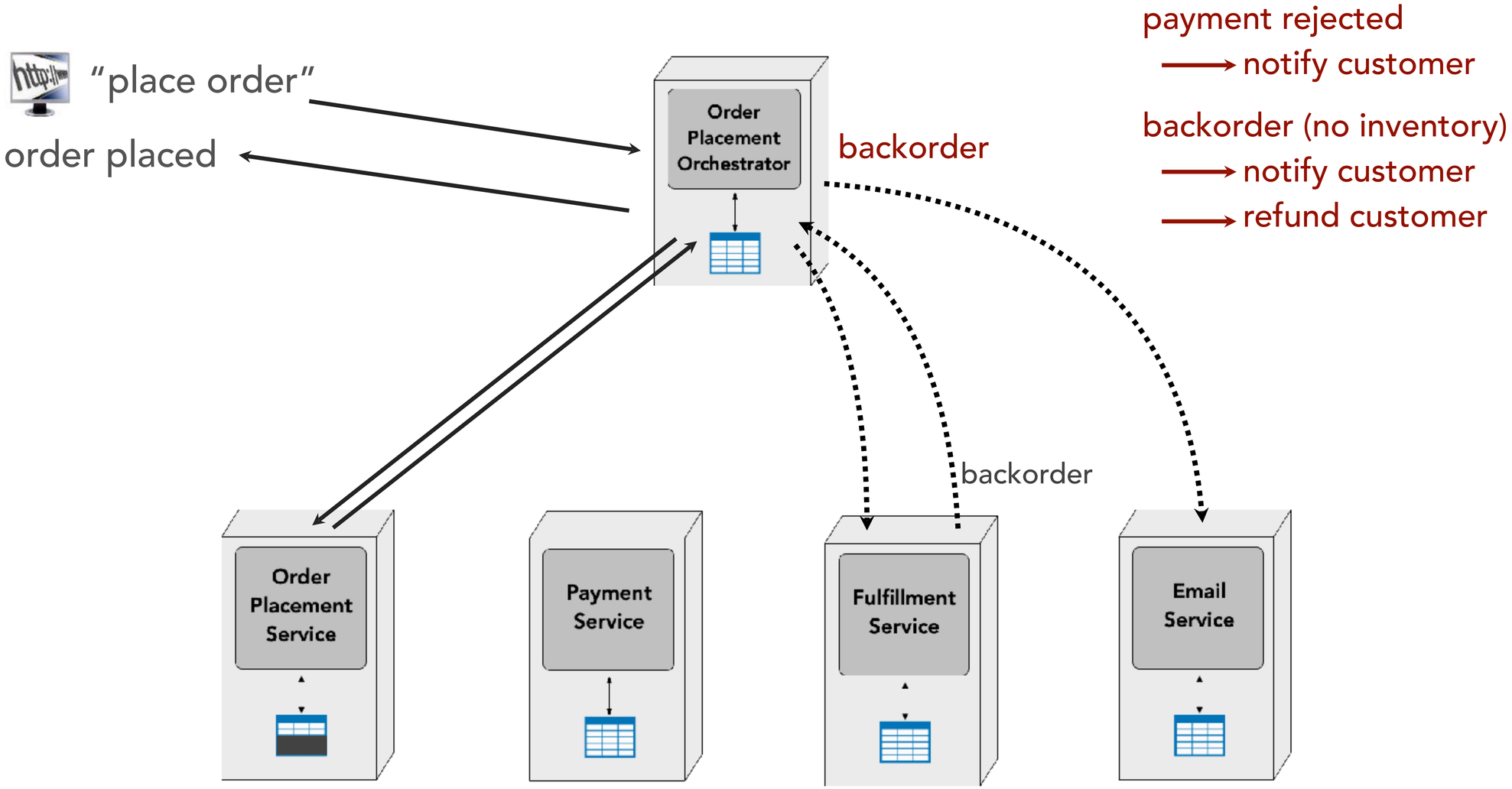


payment rejected
→ notify customer

backorder (no inventory)
→ notify customer
→ refund customer

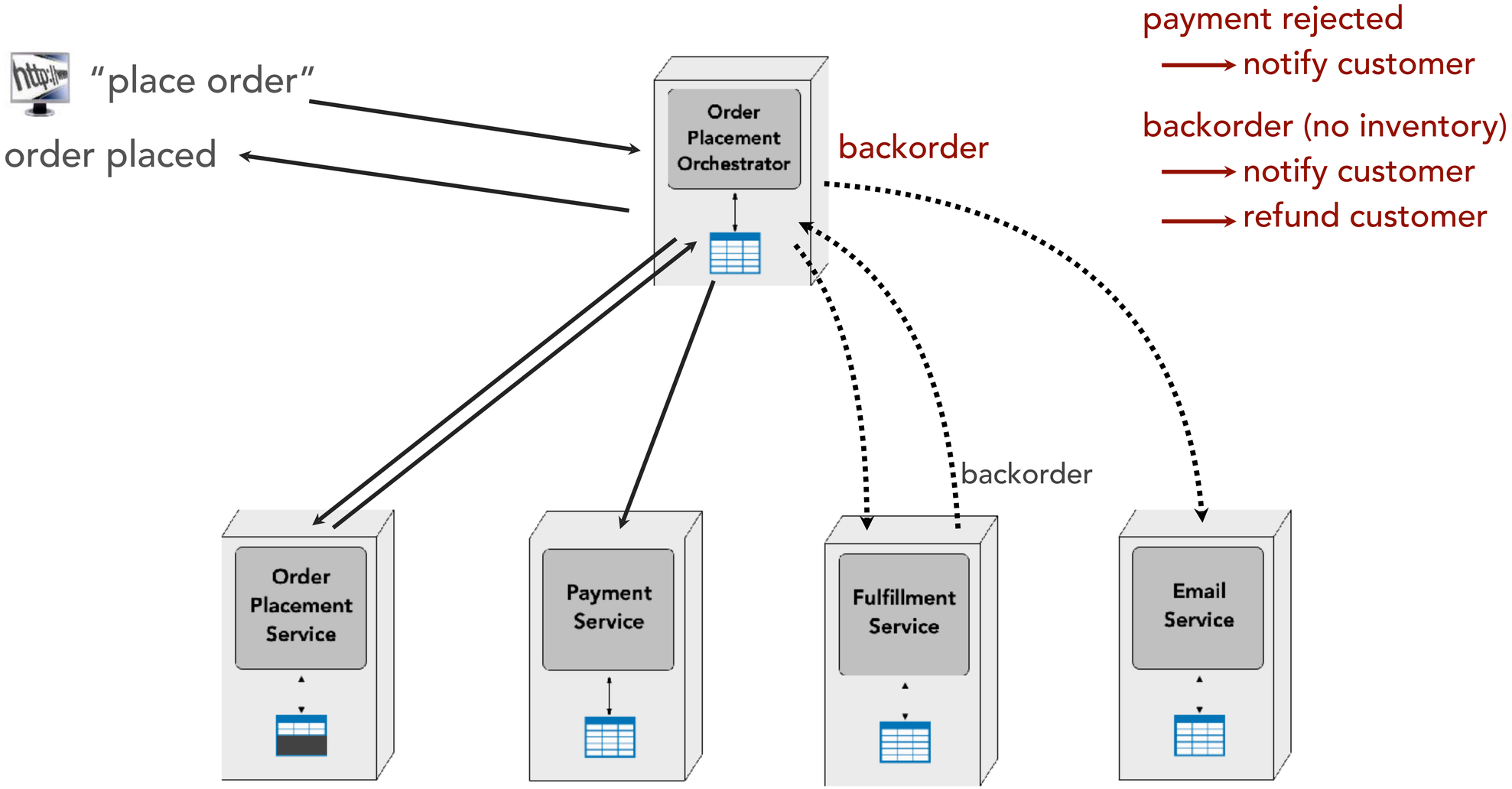
managing workflows

error handling



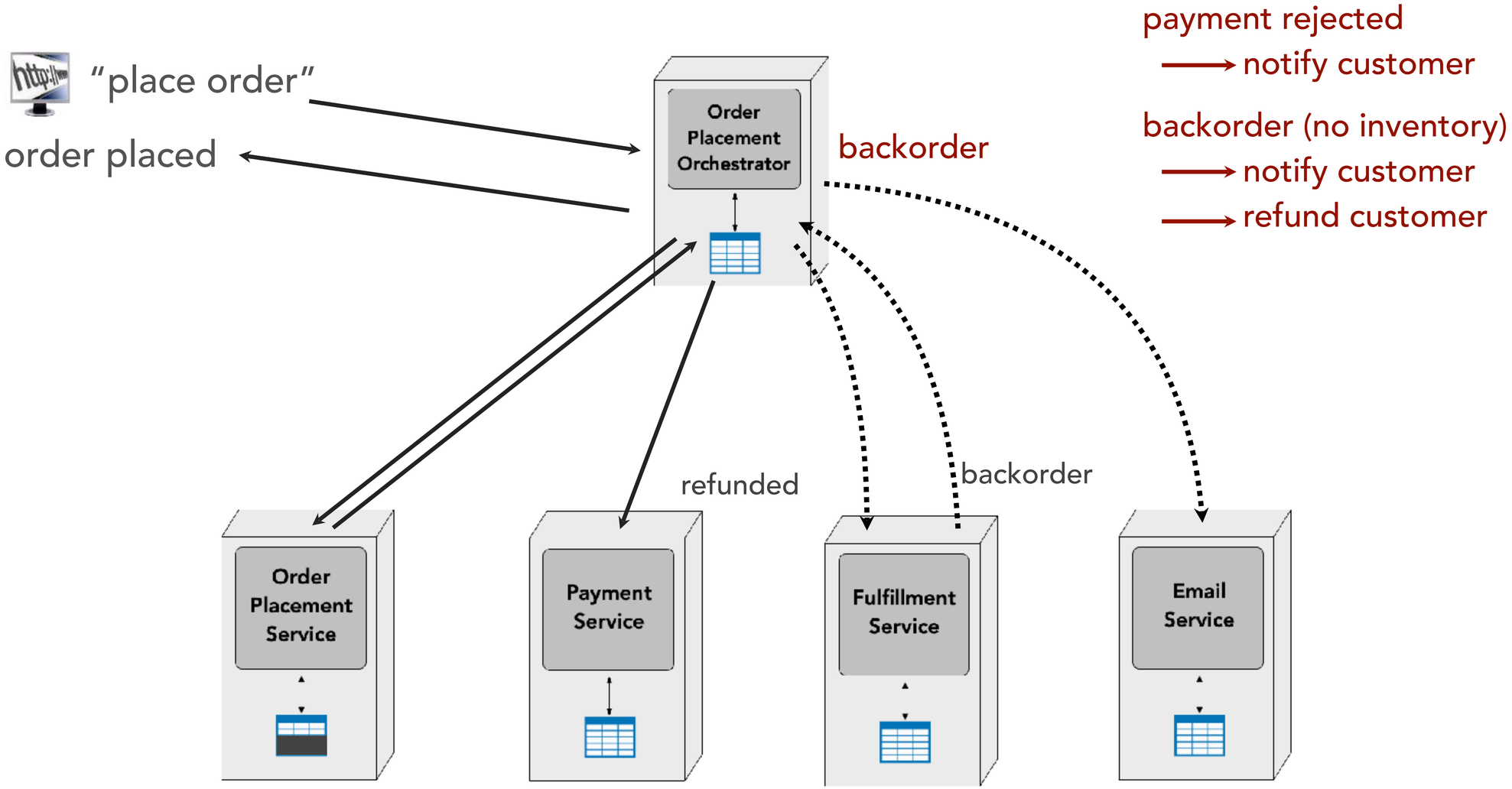
managing workflows

error handling



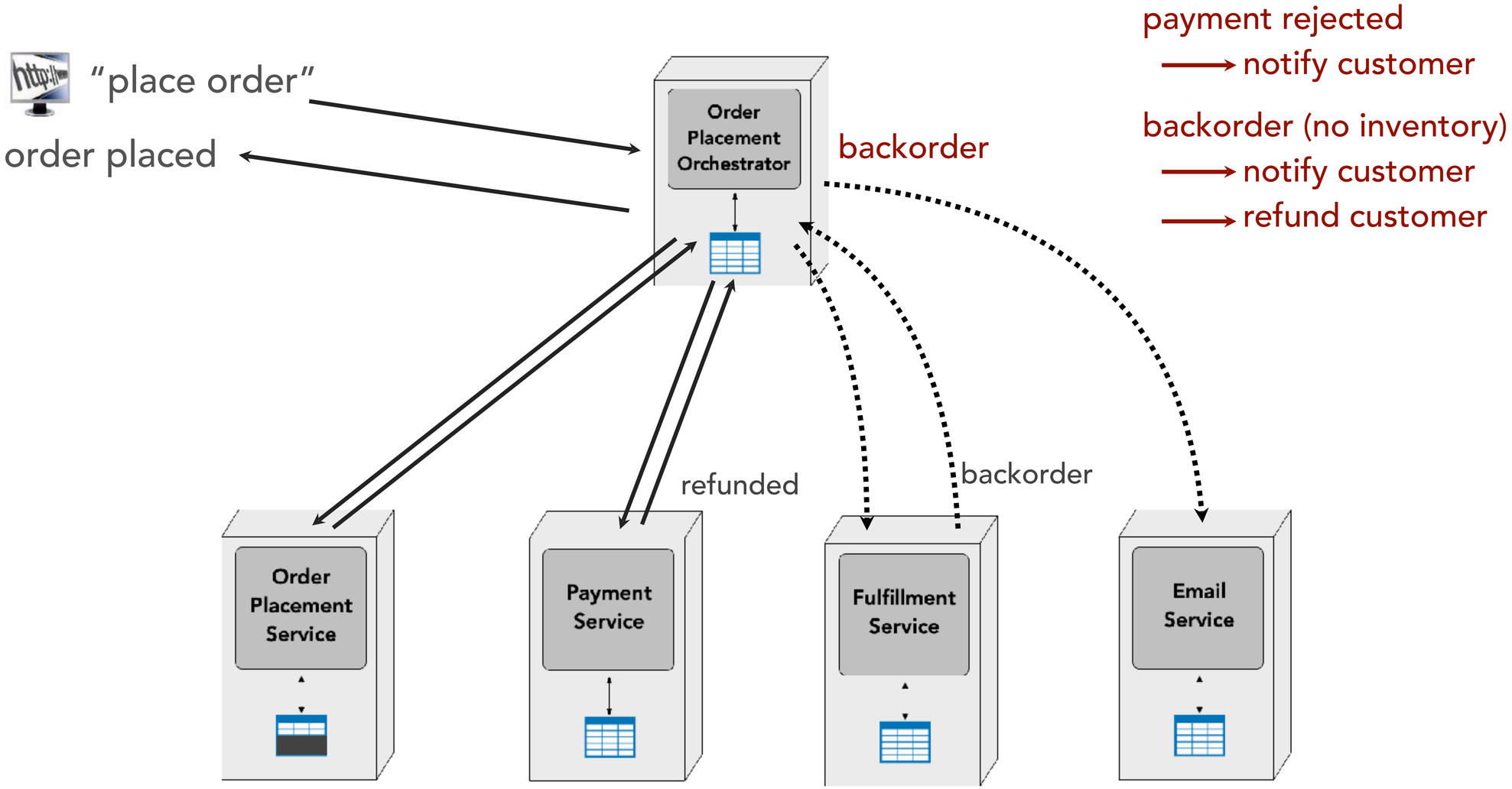
managing workflows

error handling



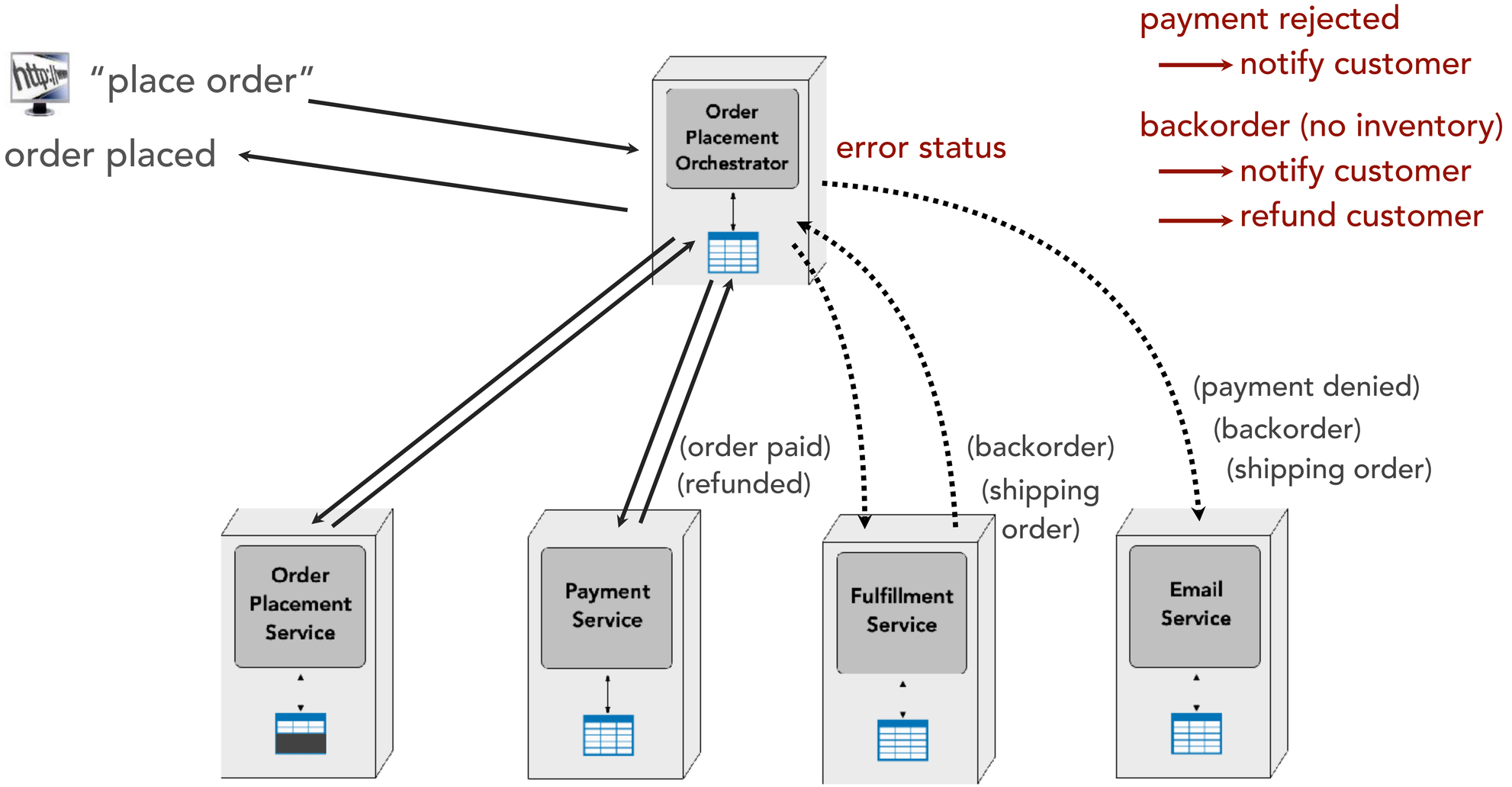
managing workflows

error handling



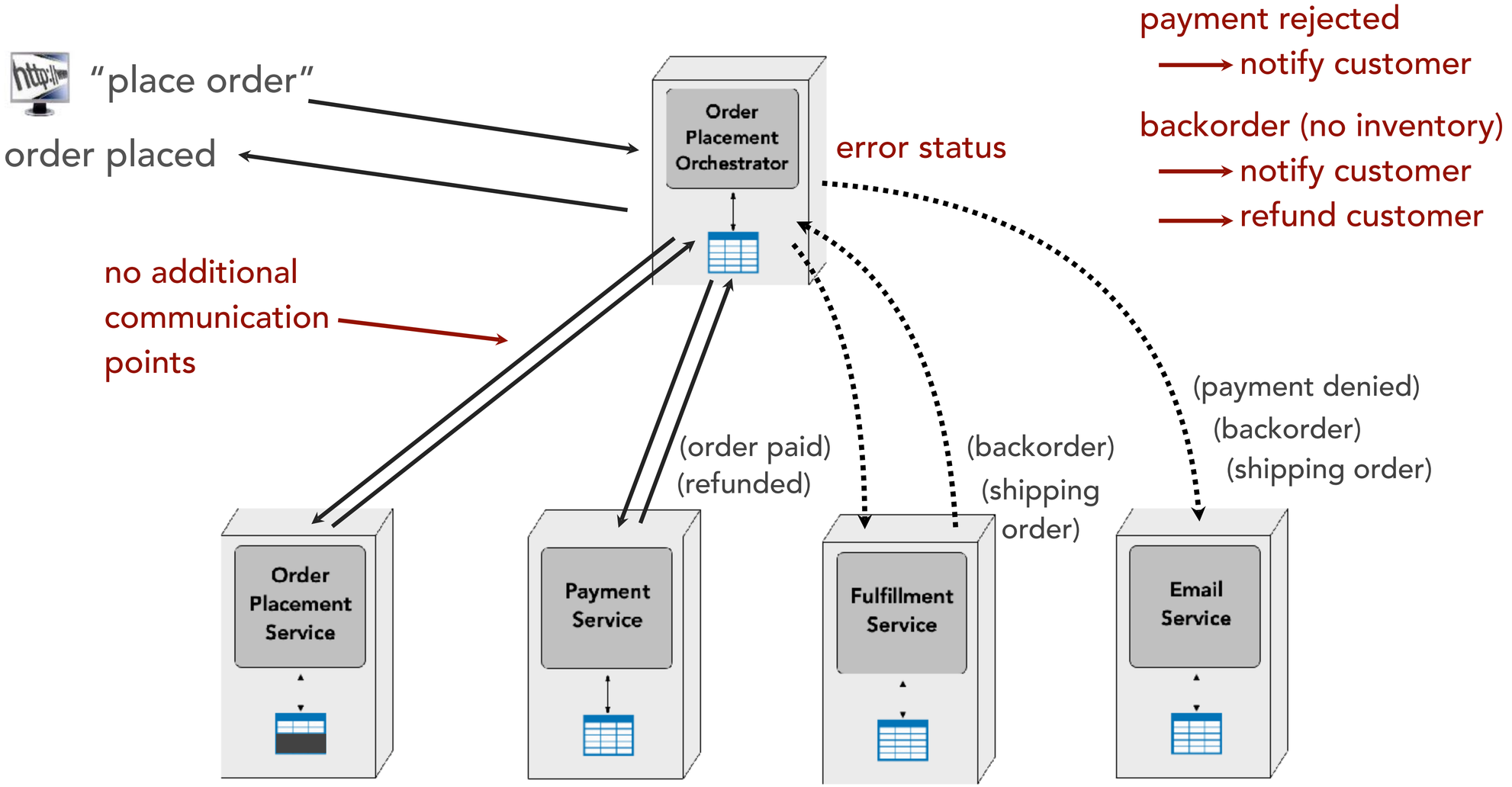
managing workflows

error handling



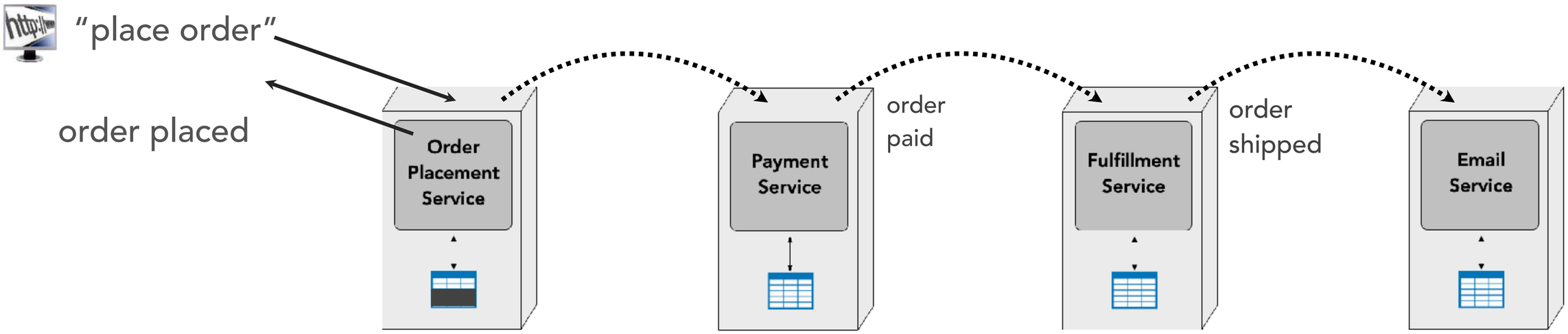
managing workflows

error handling



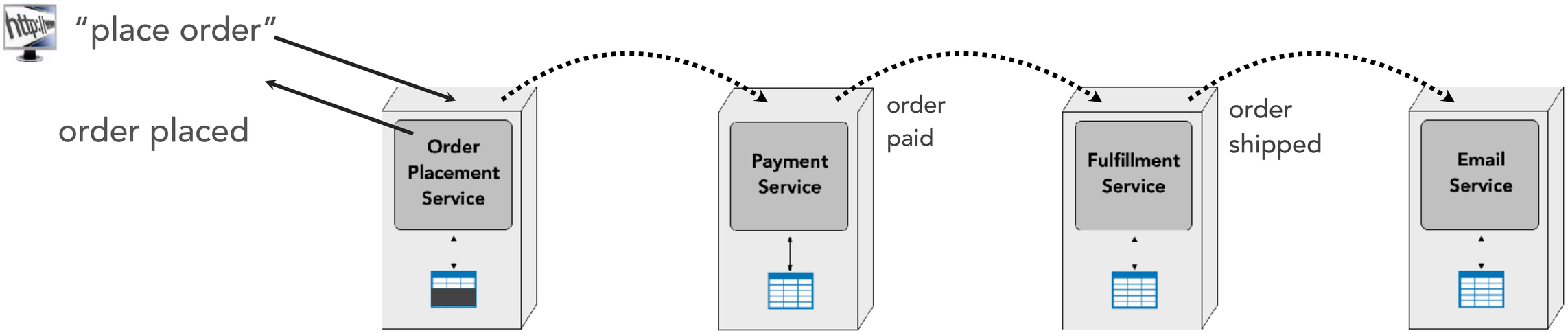
managing workflows

error handling



managing workflows

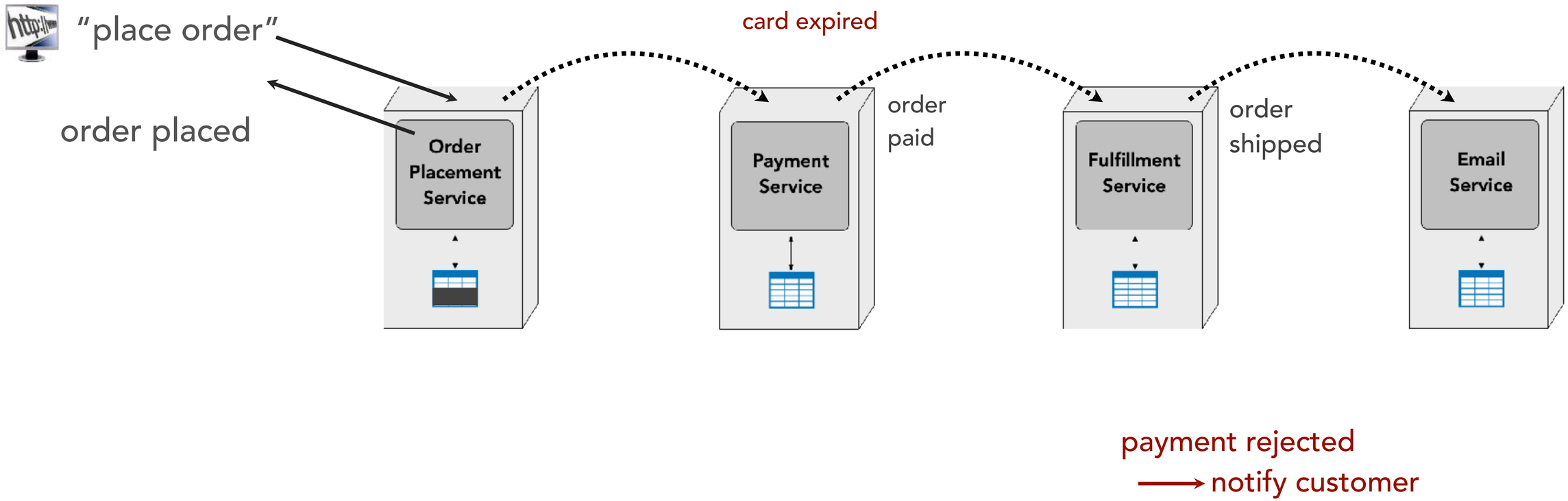
error handling



payment rejected
→ notify customer

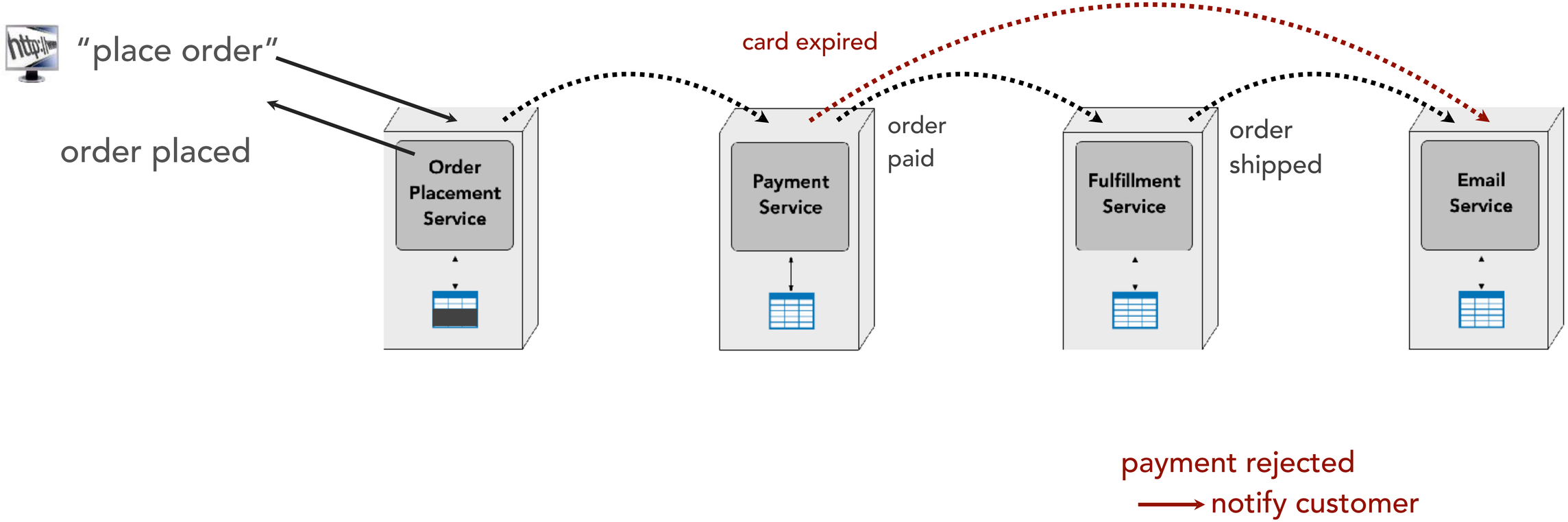
managing workflows

error handling



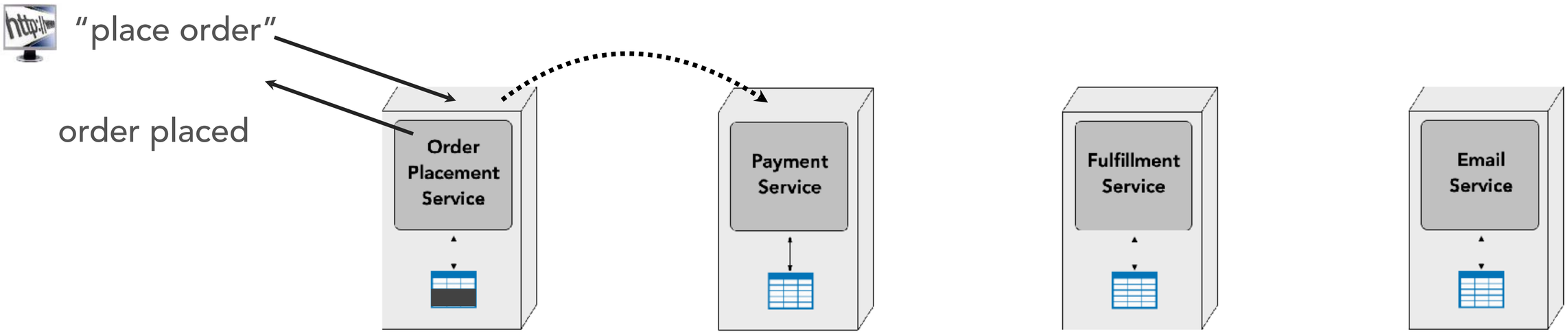
managing workflows

error handling



managing workflows

error handling



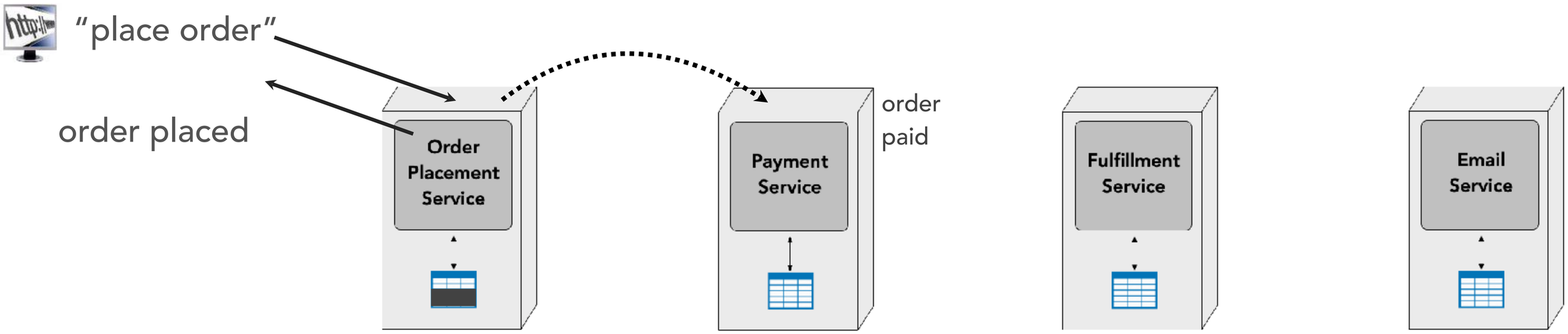
backorder (no inventory)

→ notify customer

→ refund customer

managing workflows

error handling



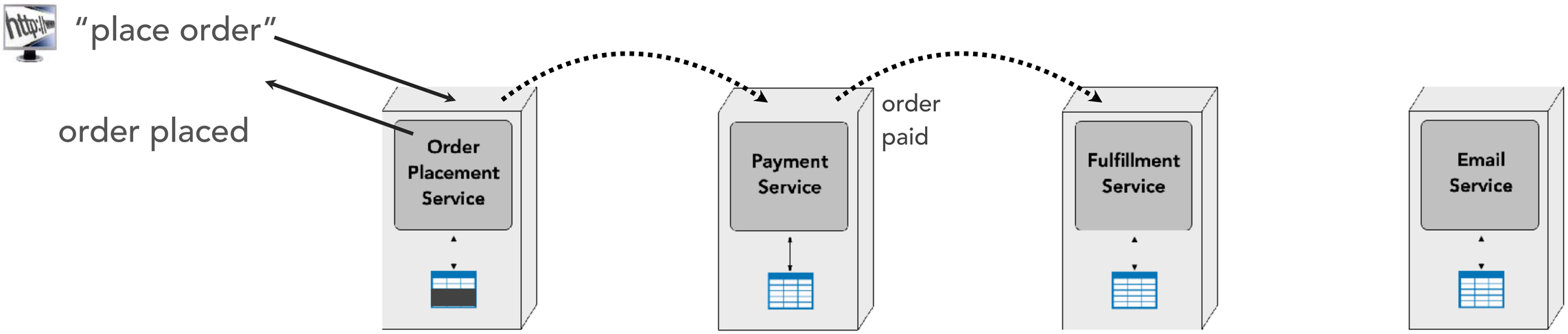
backorder (no inventory)

→ notify customer

→ refund customer

managing workflows

error handling



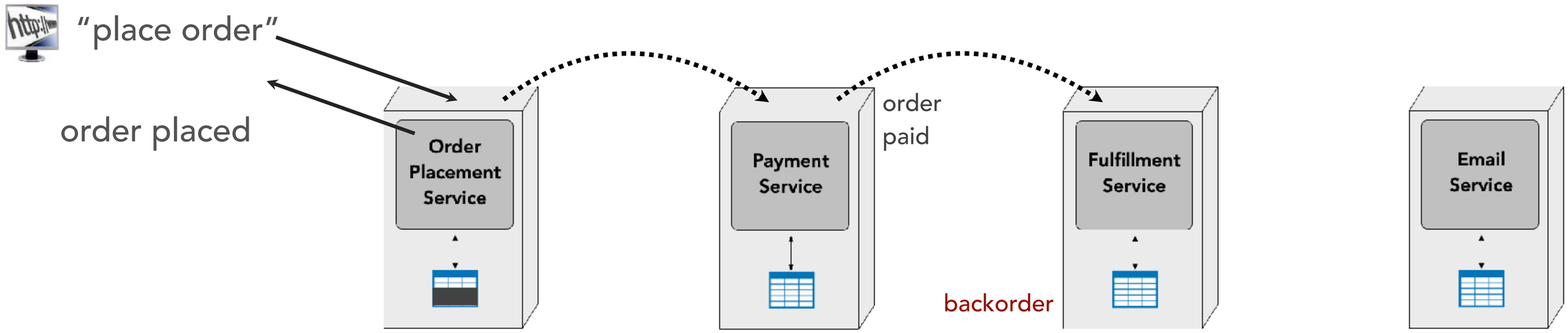
backorder (no inventory)

→ notify customer

→ refund customer

managing workflows

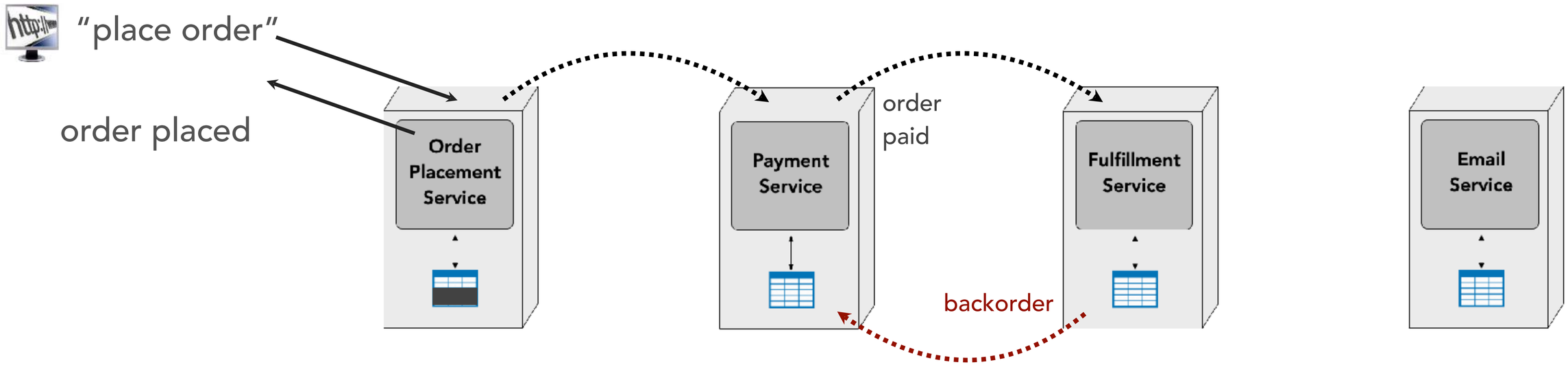
error handling



backorder (no inventory)
→ notify customer
→ refund customer

managing workflows

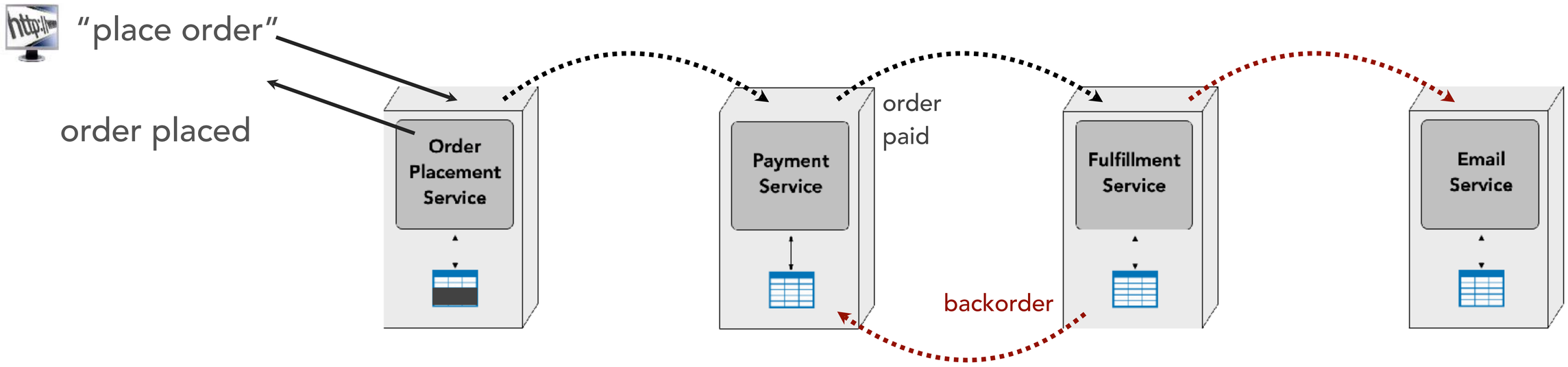
error handling



backorder (no inventory)
→ notify customer
→ refund customer

managing workflows

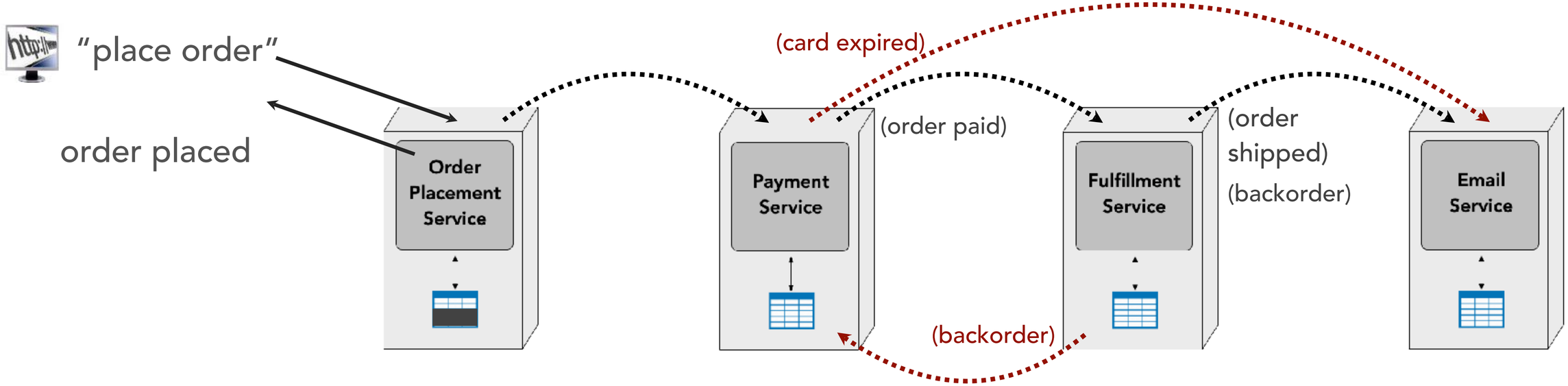
error handling



backorder (no inventory)
→ notify customer
→ refund customer

managing workflows

error handling

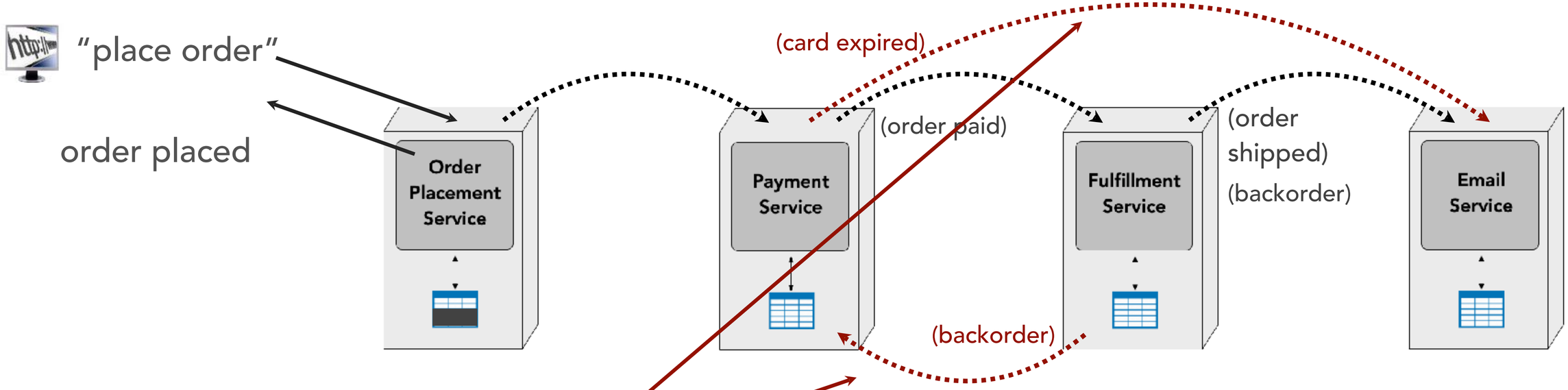


payment rejected
→ notify customer

backorder (no inventory)
→ notify customer
→ refund customer

managing workflows

error handling



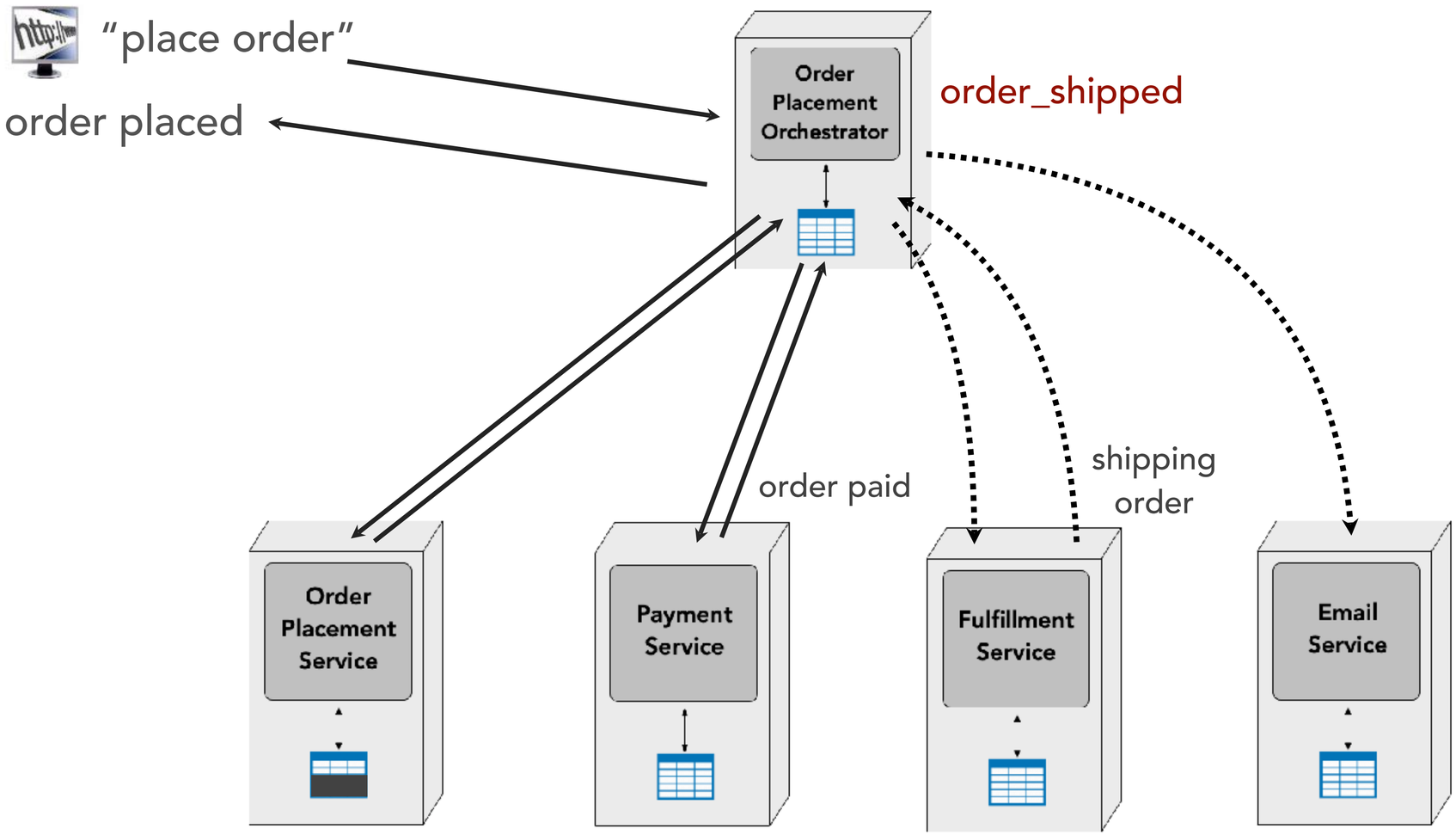
additional communication points

payment rejected
→ notify customer

backorder (no inventory)
→ notify customer
→ refund customer

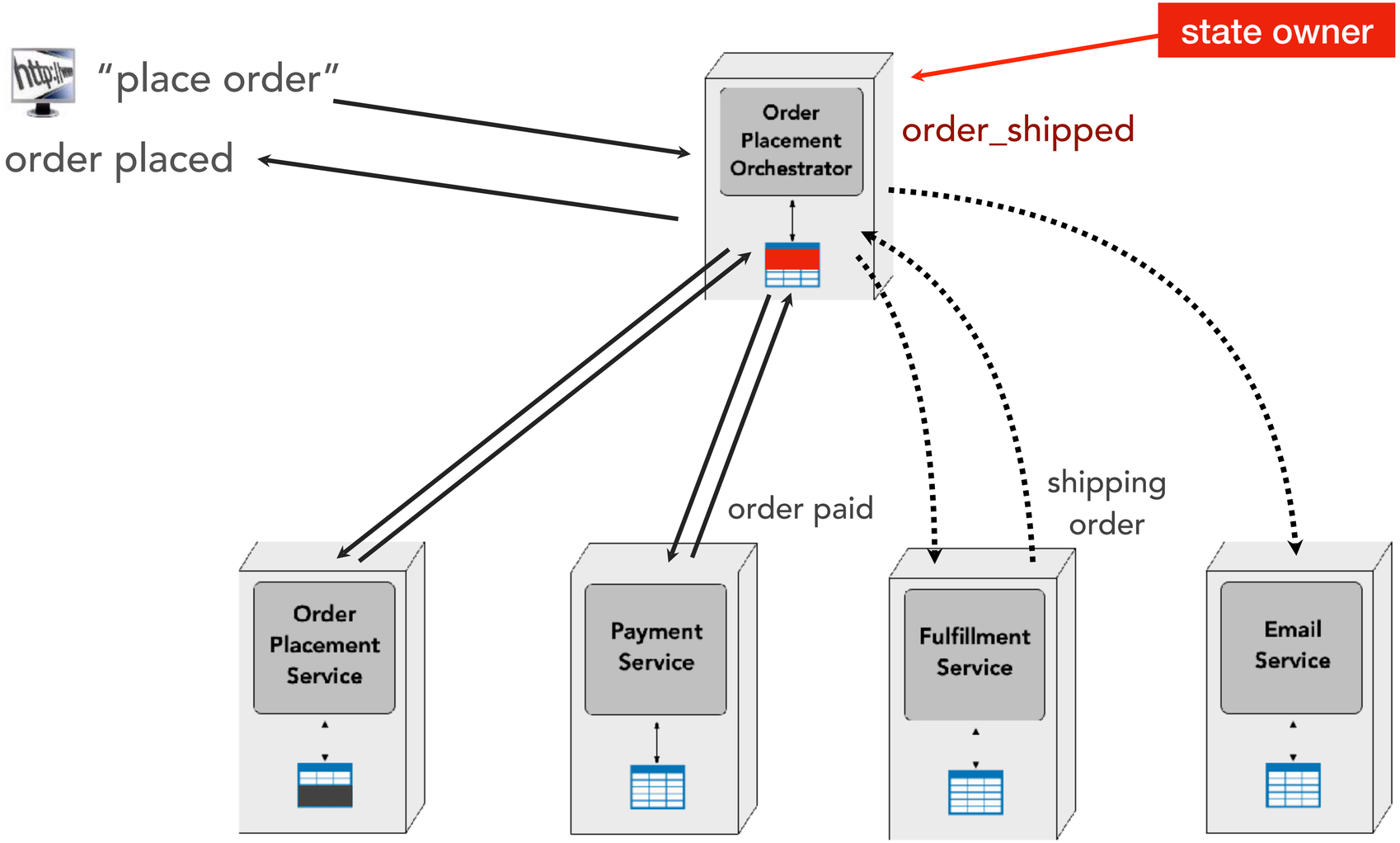
managing workflows

state owner



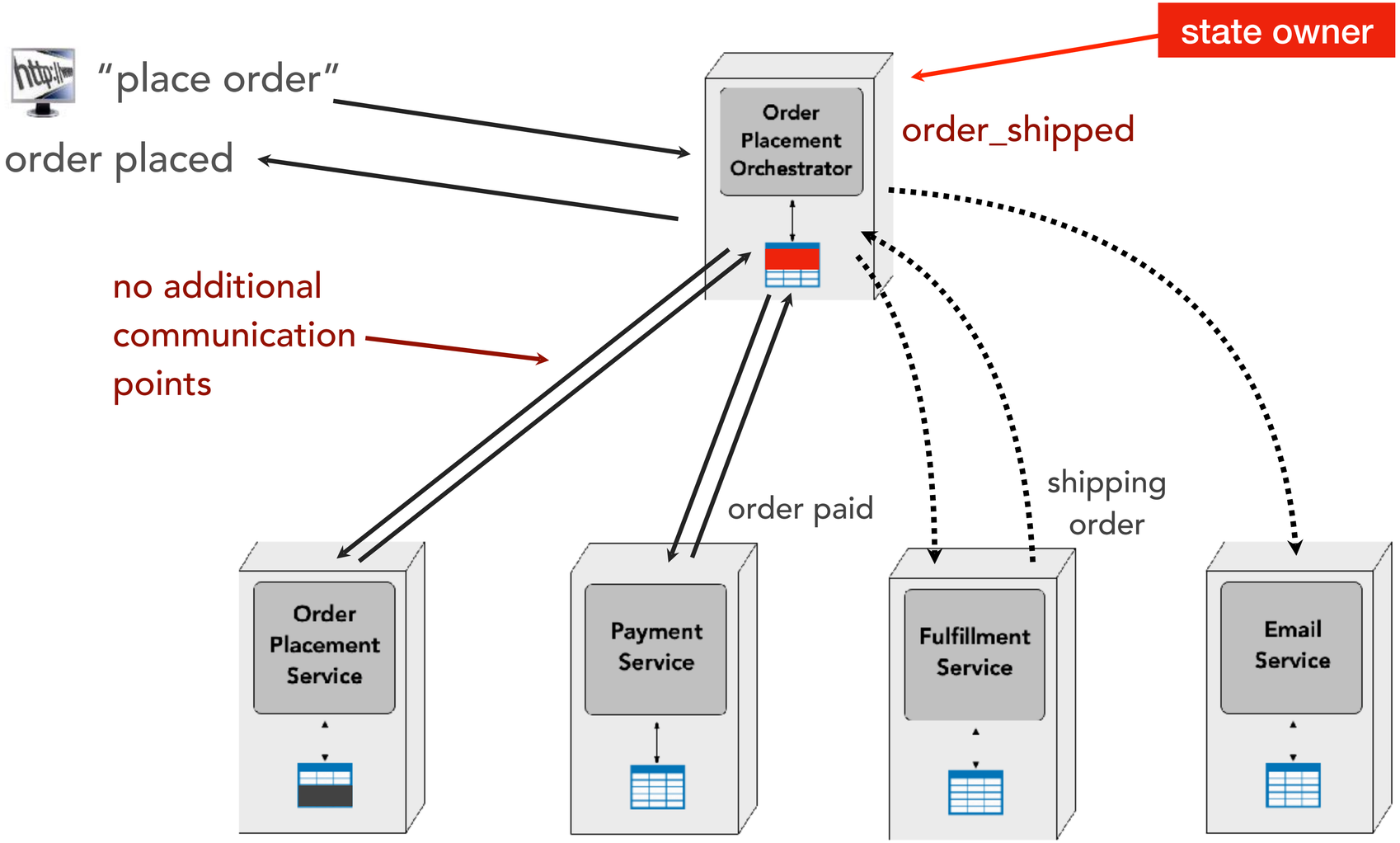
managing workflows

state owner



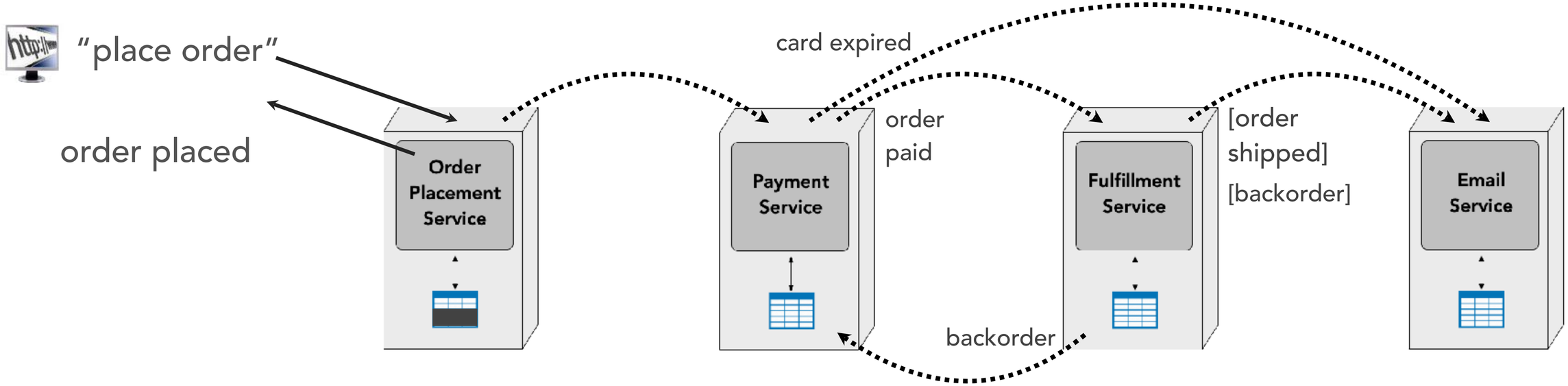
managing workflows

state owner



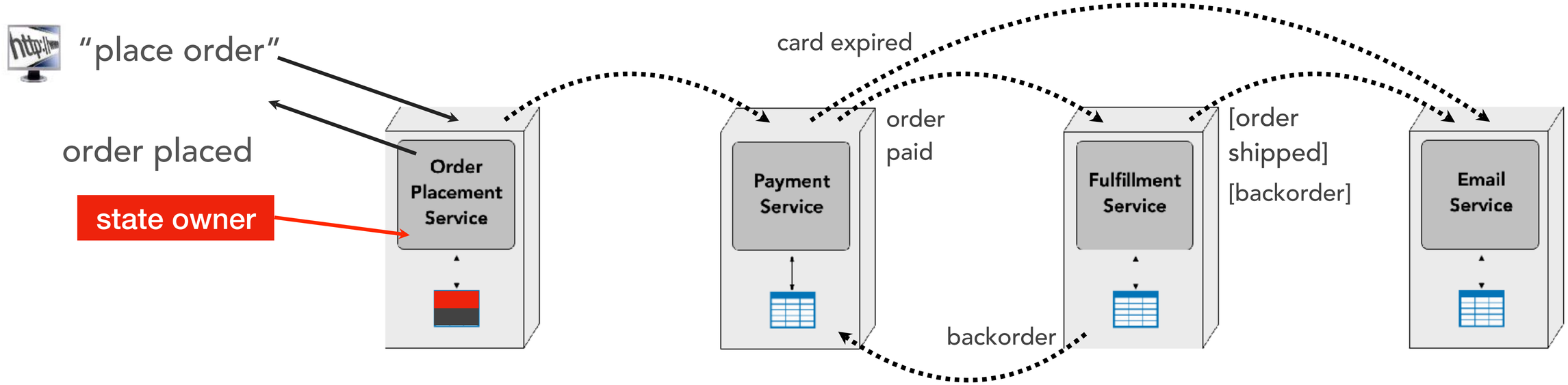
managing workflows

state owner



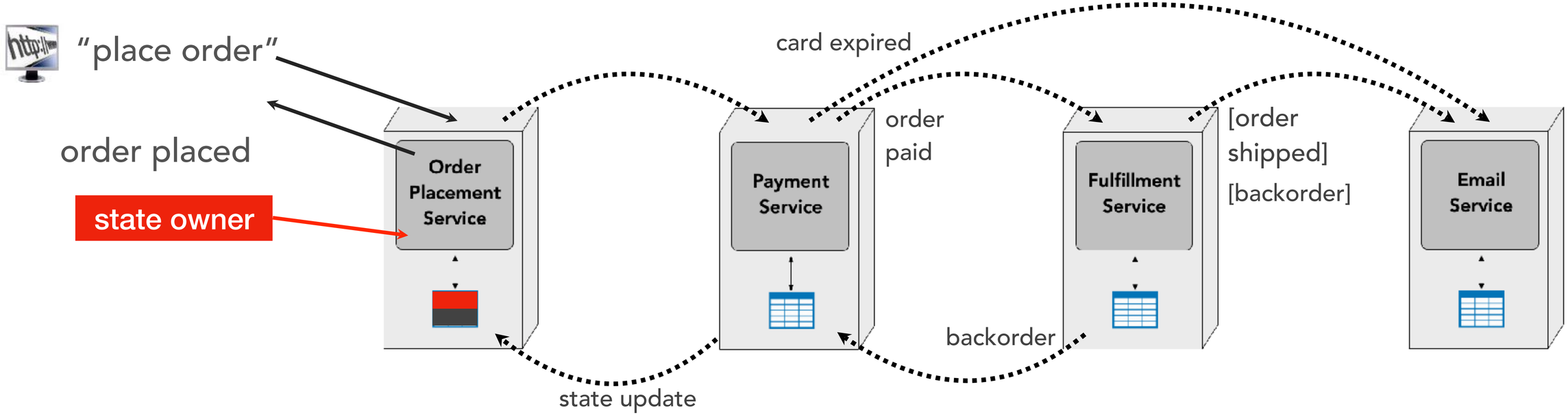
managing workflows

state owner



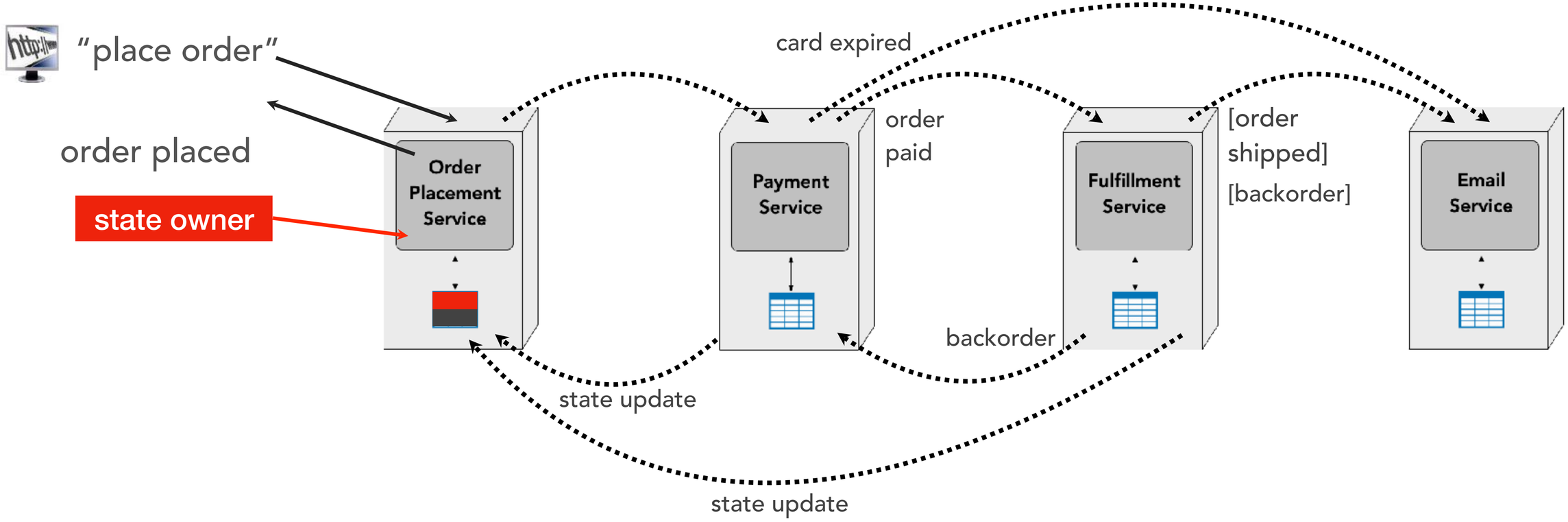
managing workflows

state owner

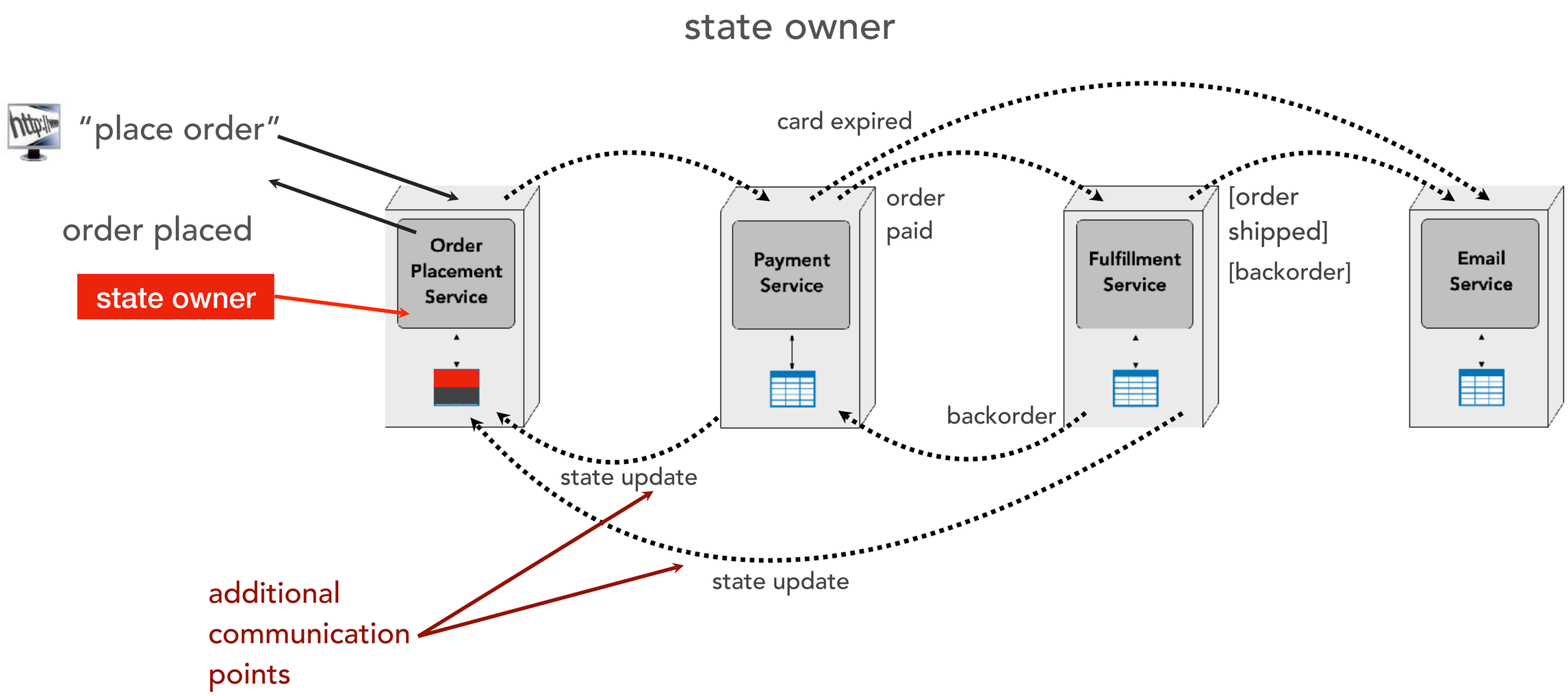


managing workflows

state owner

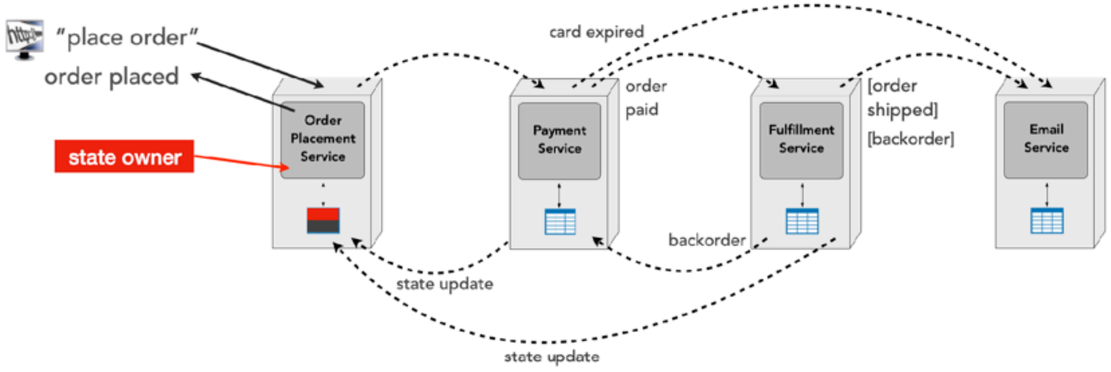


managing workflows

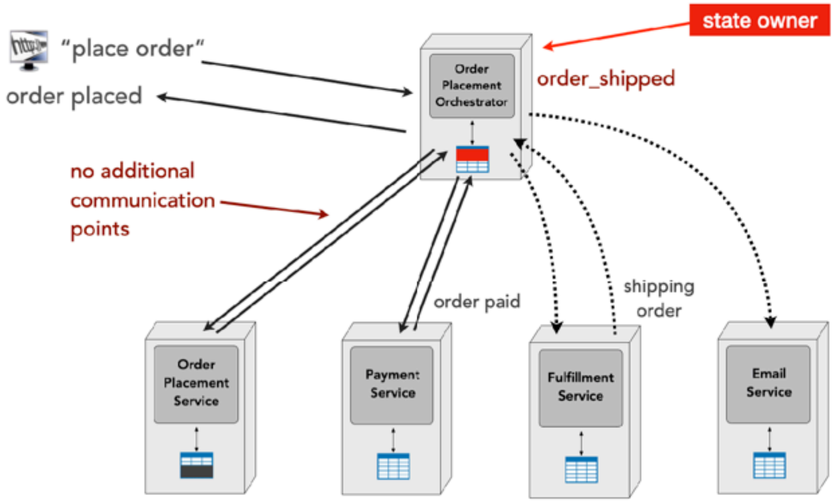


managing workflows

choreography



orchestration



managing workflows

finite state machine and state transitions

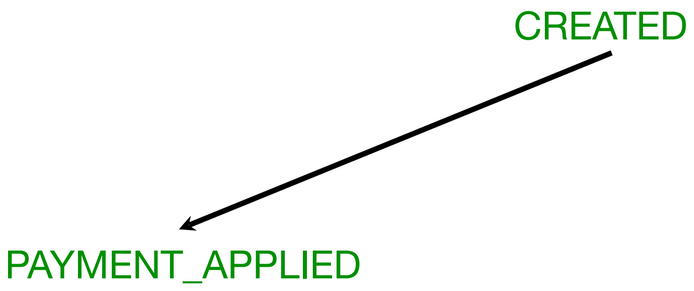
managing workflows

finite state machine and state transitions

CREATED

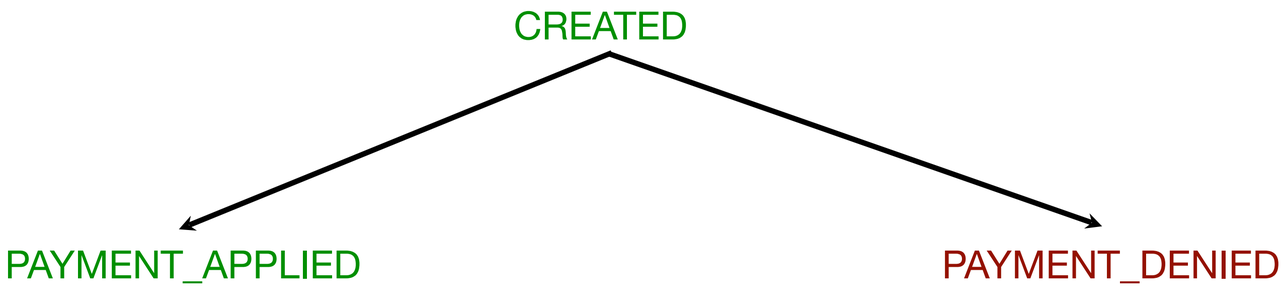
managing workflows

finite state machine and state transitions



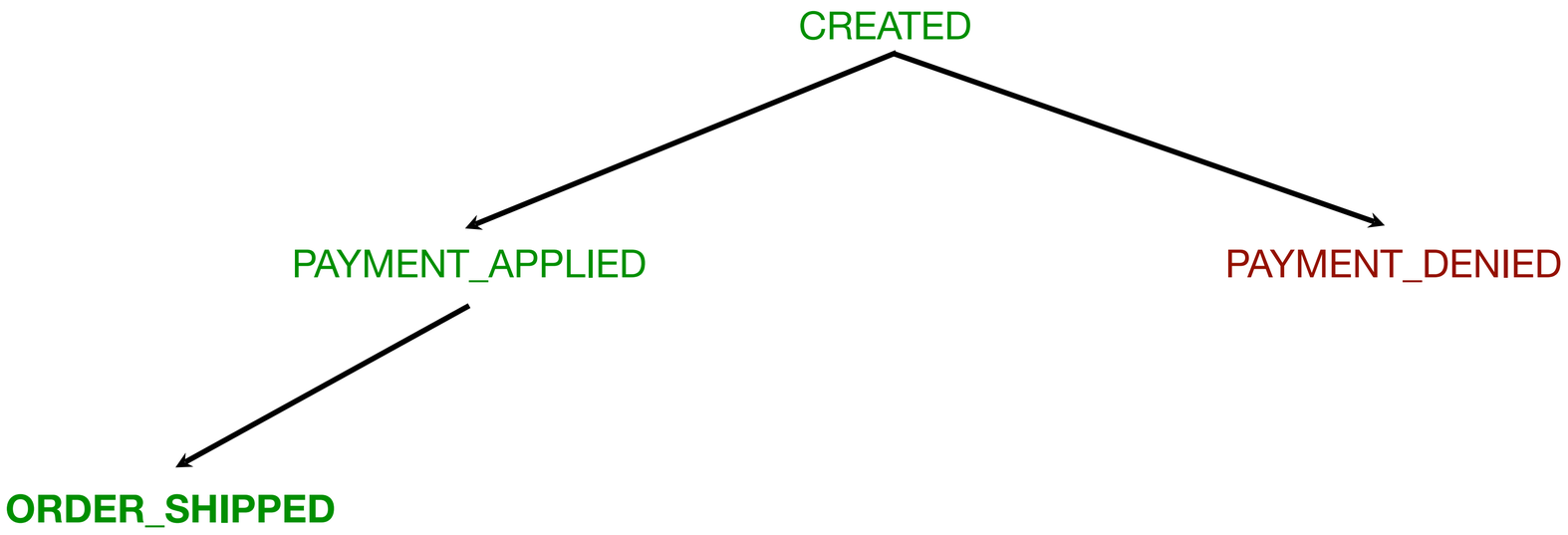
managing workflows

finite state machine and state transitions



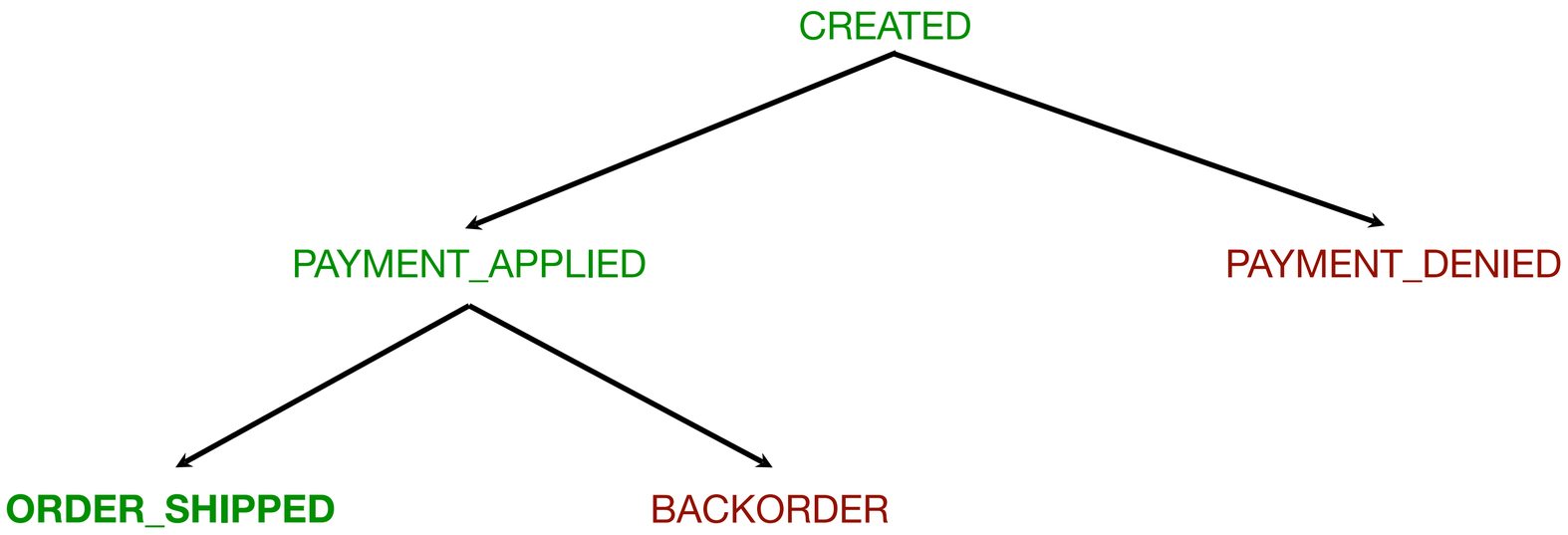
managing workflows

finite state machine and state transitions



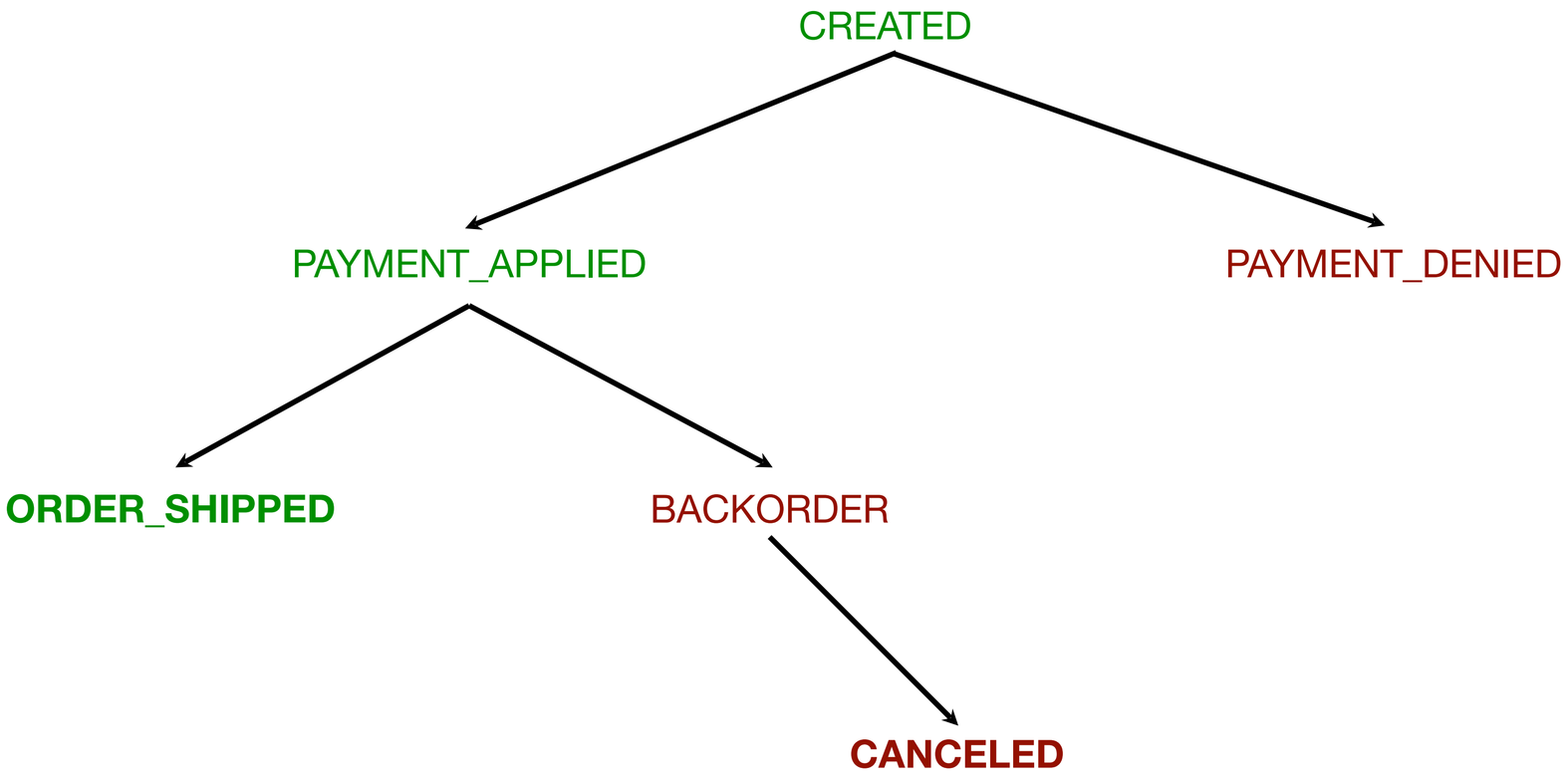
managing workflows

finite state machine and state transitions



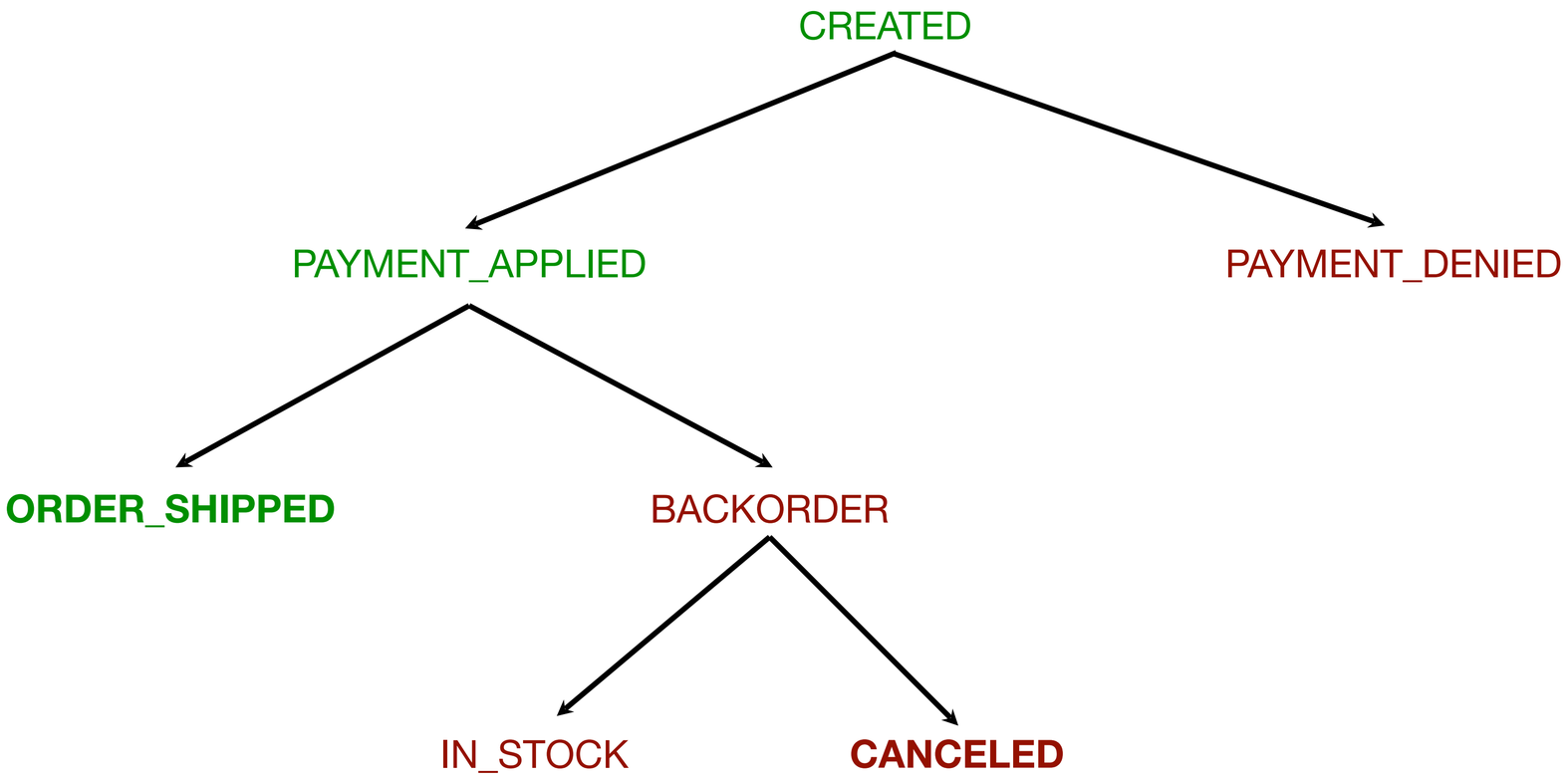
managing workflows

finite state machine and state transitions



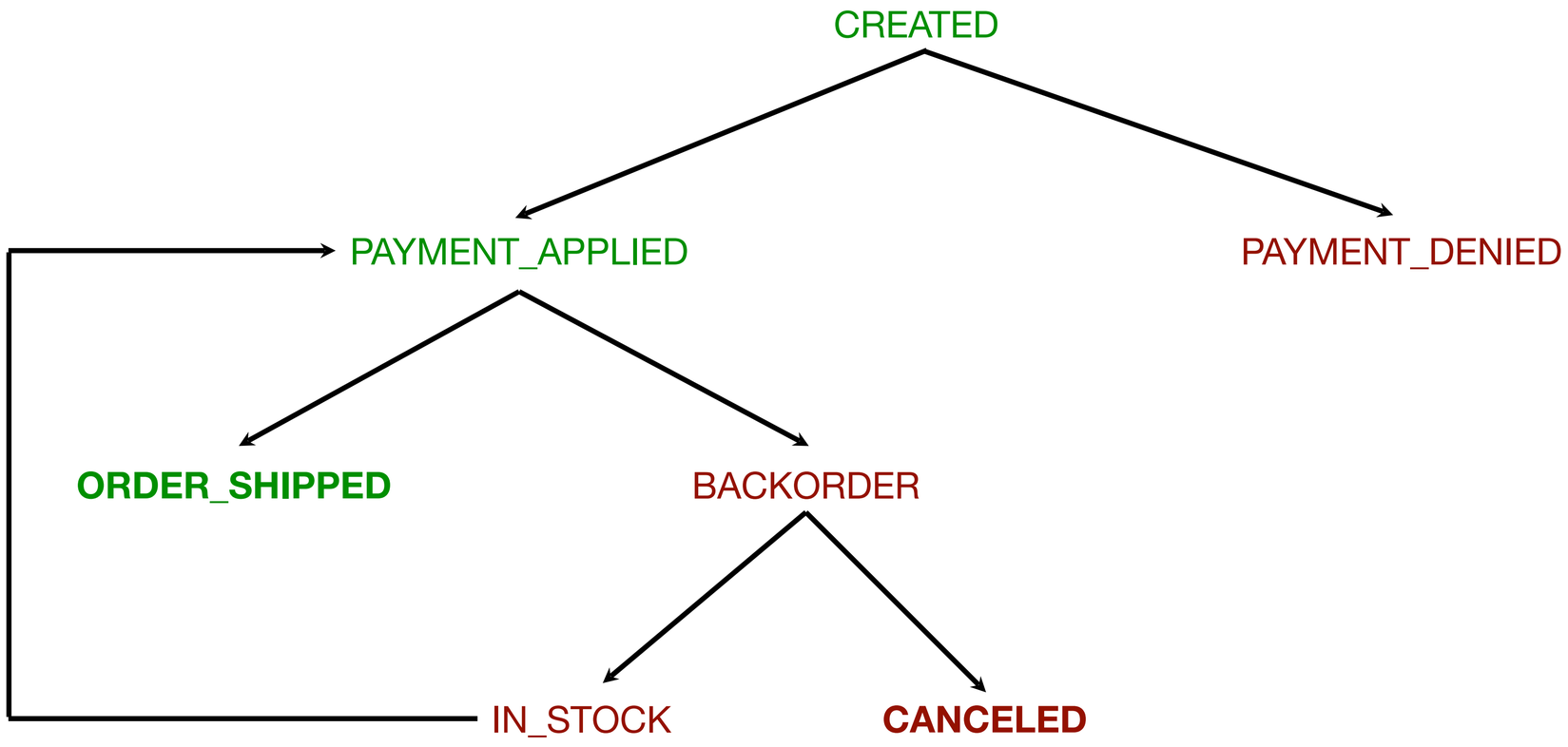
managing workflows

finite state machine and state transitions



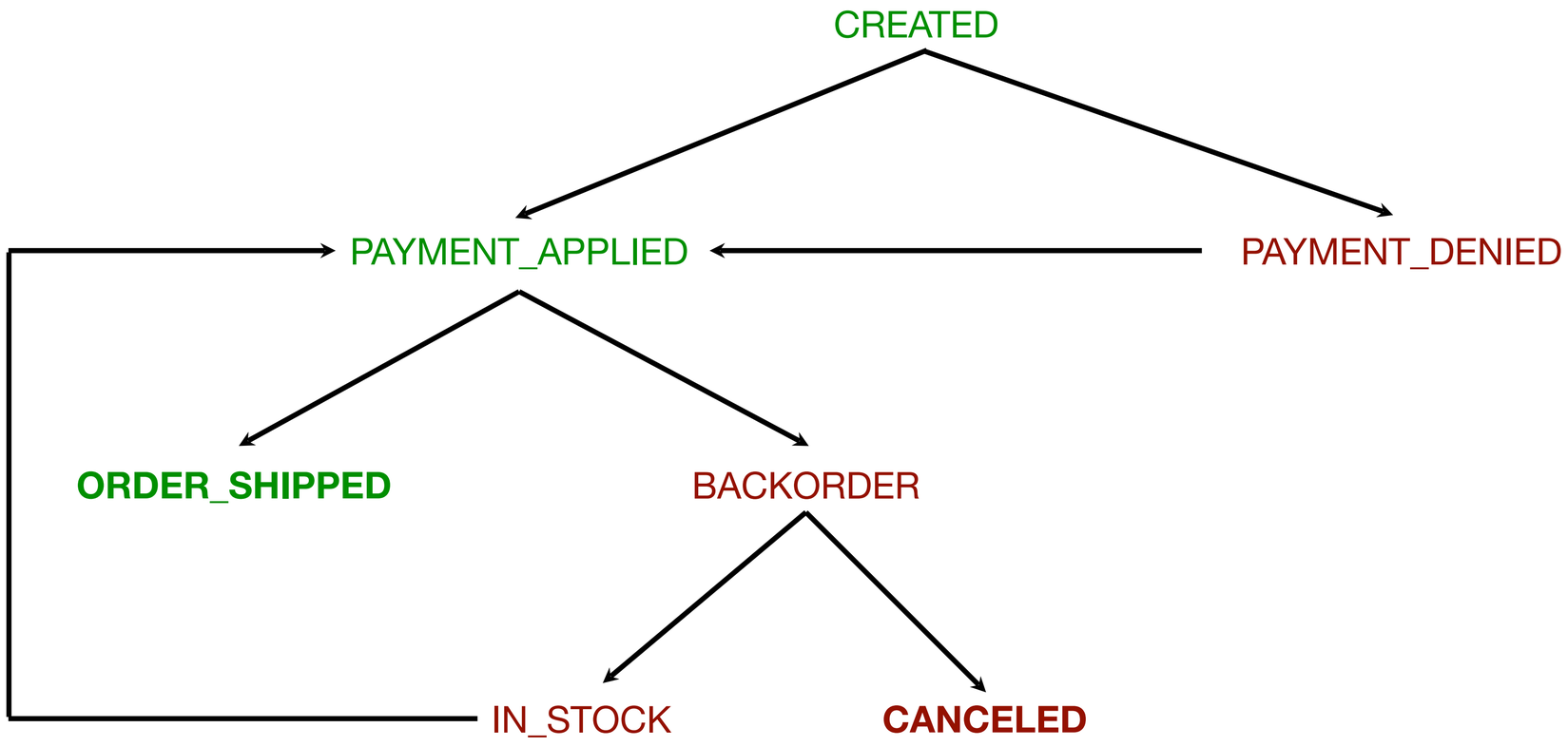
managing workflows

finite state machine and state transitions



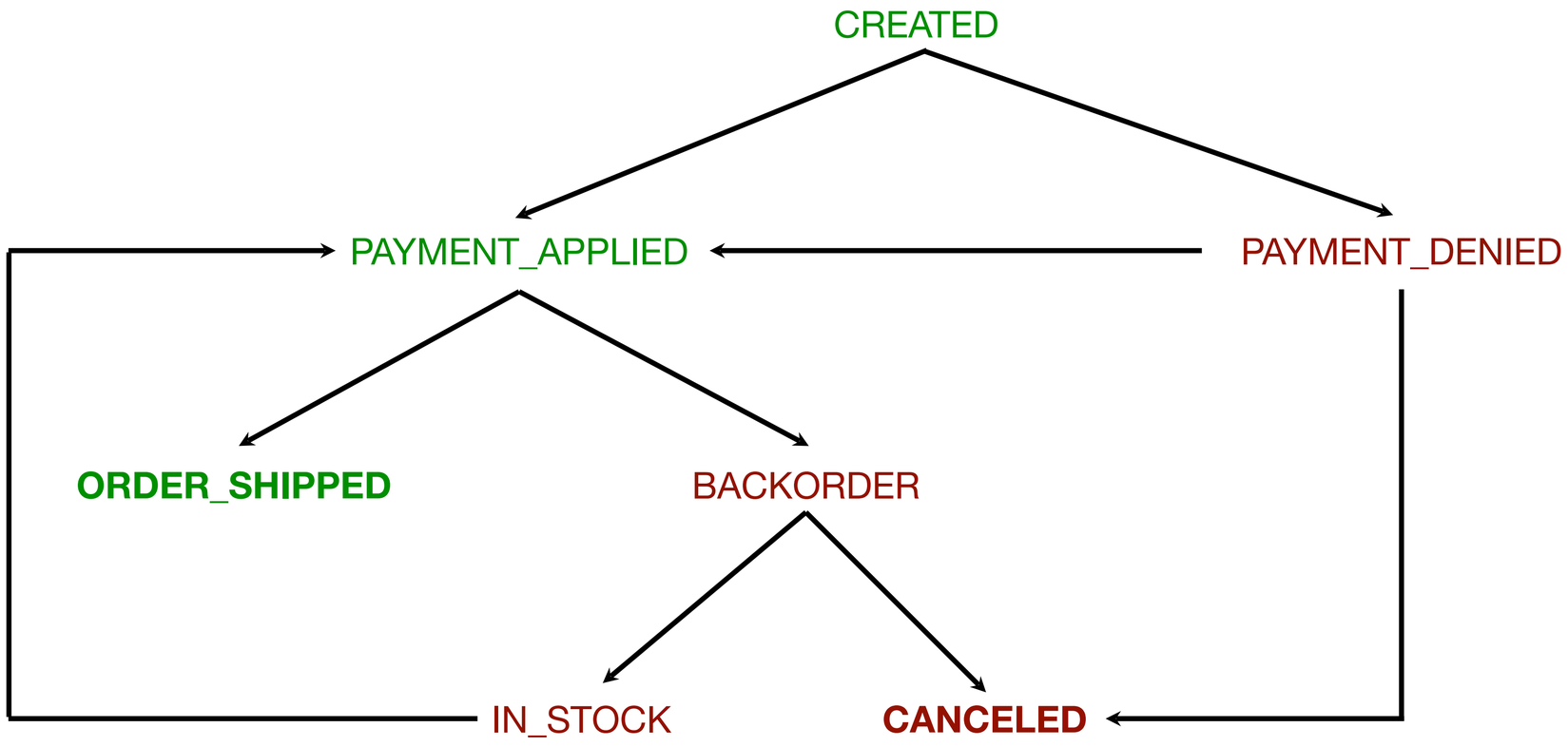
managing workflows

finite state machine and state transitions



managing workflows

finite state machine and state transitions



managing workflows

finite state machine and state transitions

Initiating State	Resulting State	Transition Action

managing workflows

finite state machine and state transitions

Initiating State	Resulting State	Transition Action
START →	CREATED	send to payment

managing workflows

finite state machine and state transitions

Initiating State	Resulting State	Transition Action
START →	CREATED	send to payment
CREATED →	PAYMENT_APPLIED	sent to fulfillment, notification
CREATED →	PAYMENT_DENIED	send to notification

managing workflows

finite state machine and state transitions

Initiating State	Resulting State	Transition Action
START →	CREATED	send to payment
CREATED →	PAYMENT_APPLIED	sent to fulfillment, notification
CREATED →	PAYMENT_DENIED	send to notification
PAYMENT_APPLIED →	**ORDER_SHIPPED**	send to notification
PAYMENT_APPLIED →	BACKORDER	send to notification, payment for refund

managing workflows

finite state machine and state transitions

Initiating State	Resulting State	Transition Action
START →	CREATED	send to payment
CREATED →	PAYMENT_APPLIED	sent to fulfillment, notification
CREATED →	PAYMENT_DENIED	send to notification
PAYMENT_APPLIED →	**ORDER_SHIPPED**	send to notification
PAYMENT_APPLIED →	BACKORDER	send to notification, payment for refund
PAYMENT_DENIED →	PAYMENT_APPLIED	sent to fulfillment, notification
PAYMENT_DENIED →	**CANCELED**	send to notification

managing workflows

finite state machine and state transitions

Initiating State	Resulting State	Transition Action
START →	CREATED	send to payment
CREATED →	PAYMENT_APPLIED	sent to fulfillment, notification
CREATED →	PAYMENT_DENIED	send to notification
PAYMENT_APPLIED →	**ORDER_SHIPPED**	send to notification
PAYMENT_APPLIED →	BACKORDER	send to notification, payment for refund
PAYMENT_DENIED →	PAYMENT_APPLIED	sent to fulfillment, notification
PAYMENT_DENIED →	**CANCELED**	send to notification
BACKORDER →	IN_STOCK	send to payment
BACKORDER →	**CANCELED**	send to notification

managing workflows

finite state machine and state transitions

Initiating State	Resulting State	Transition Action
START →	CREATED	send to payment
CREATED →	PAYMENT_APPLIED	sent to fulfillment, notification
CREATED →	PAYMENT_DENIED	send to notification
PAYMENT_APPLIED →	**ORDER_SHIPPED**	send to notification
PAYMENT_APPLIED →	BACKORDER	send to notification, payment for refund
PAYMENT_DENIED →	PAYMENT_APPLIED	sent to fulfillment, notification
PAYMENT_DENIED →	**CANCELED**	send to notification
BACKORDER →	IN_STOCK	send to payment
BACKORDER →	**CANCELED**	send to notification
IN_STOCK →	PAYMENT_APPLIED	sent to fulfillment, notification
IN_STOCK →	PAYMENT_DENIED	send to notification

managing workflows

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface Saga {
    public Transaction[] value();

    public enum Transaction {
        ORDER_PLACEMENT,
        ORDER_CANCEL,
        NEW_CUSTOMER
    }
}
```

managing workflows

```
[System.AttributeUsage(System.AttributeTargets.Class)]
class Saga : System.Attribute {

    public Transaction[] transaction;

    public enum Transaction {
        ORDER_PLACEMENT,
        ORDER_CANCEL,
        NEW_CUSTOMER
    };
}
```

managing workflows

managing workflows

```
@ServiceEntrypoint
@Saga(Transaction.ORDER_PLACEMENT)
public class OrderPlacementServiceAPI {
    ...
}
```

managing workflows

```
@ServiceEntrypoint
@Saga(Transaction.ORDER_PLACEMENT)
public class OrderPlacementServiceAPI {
    ...
}

@ServiceEntrypoint
@Saga({Transaction.ORDER_PLACEMENT,)
       Transaction.ORDER_CANCEL})
public class PaymentServiceAPI {
    ...
}
```

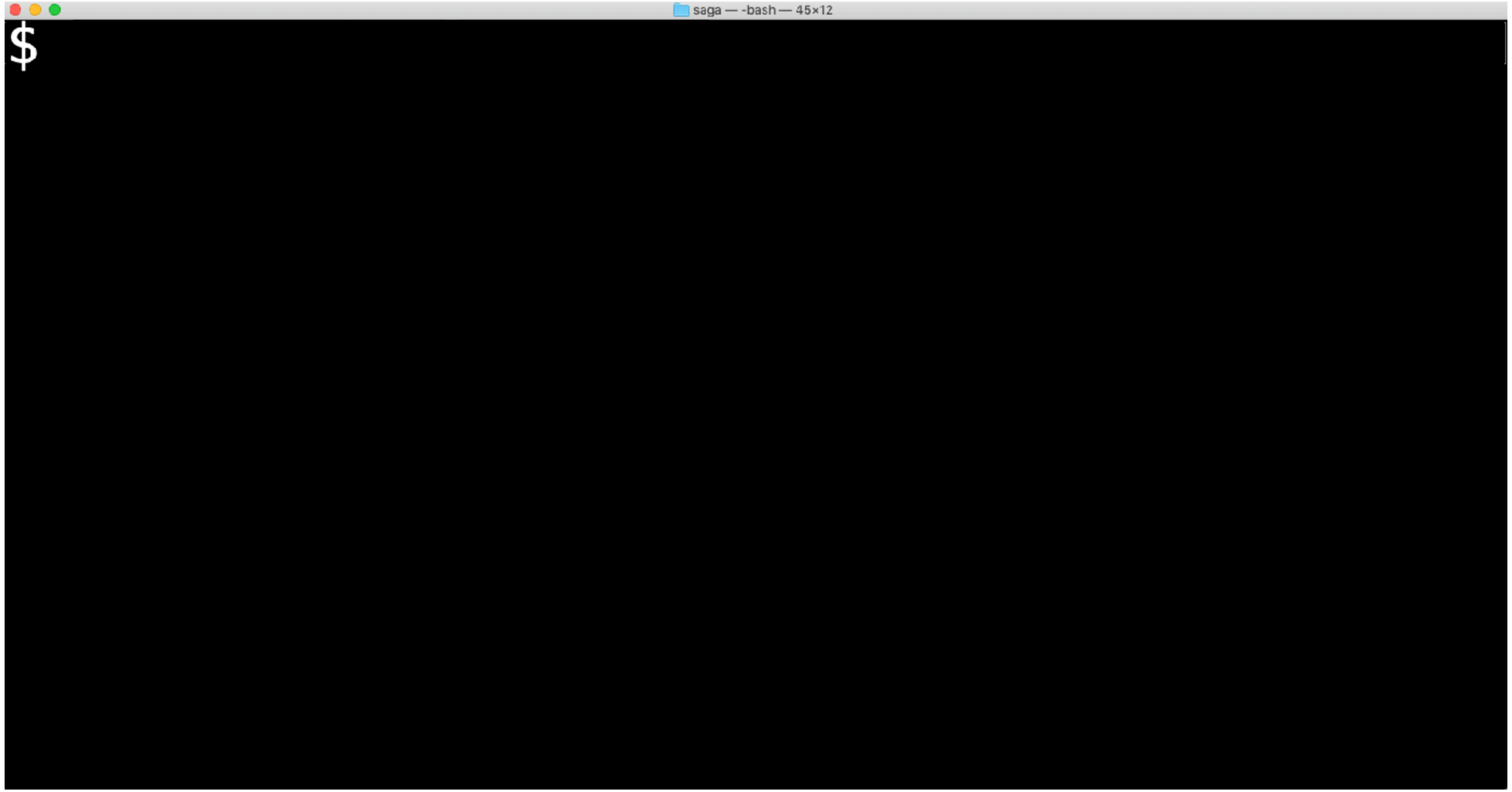
managing workflows

```
@ServiceEntrypoint
@Saga(Transaction.ORDER_PLACEMENT)
public class OrderPlacementServiceAPI {
    ...
}

@ServiceEntrypoint
@Saga({Transaction.ORDER_PLACEMENT,)
       Transaction.ORDER_CANCEL})
public class PaymentServiceAPI {
    ...
}
```

managing workflows

```
@ServiceEntrypoint
@Saga(Transaction.ORDER_PLACEMENT)
public class OrderPlacementServiceAPI {
    ...
}

@ServiceEntrypoint
@Saga({Transaction.ORDER_PLACEMENT,)
       Transaction.ORDER_CANCEL})
public class PaymentServiceAPI {
    ...
}
```

managing workflows



managing workflows

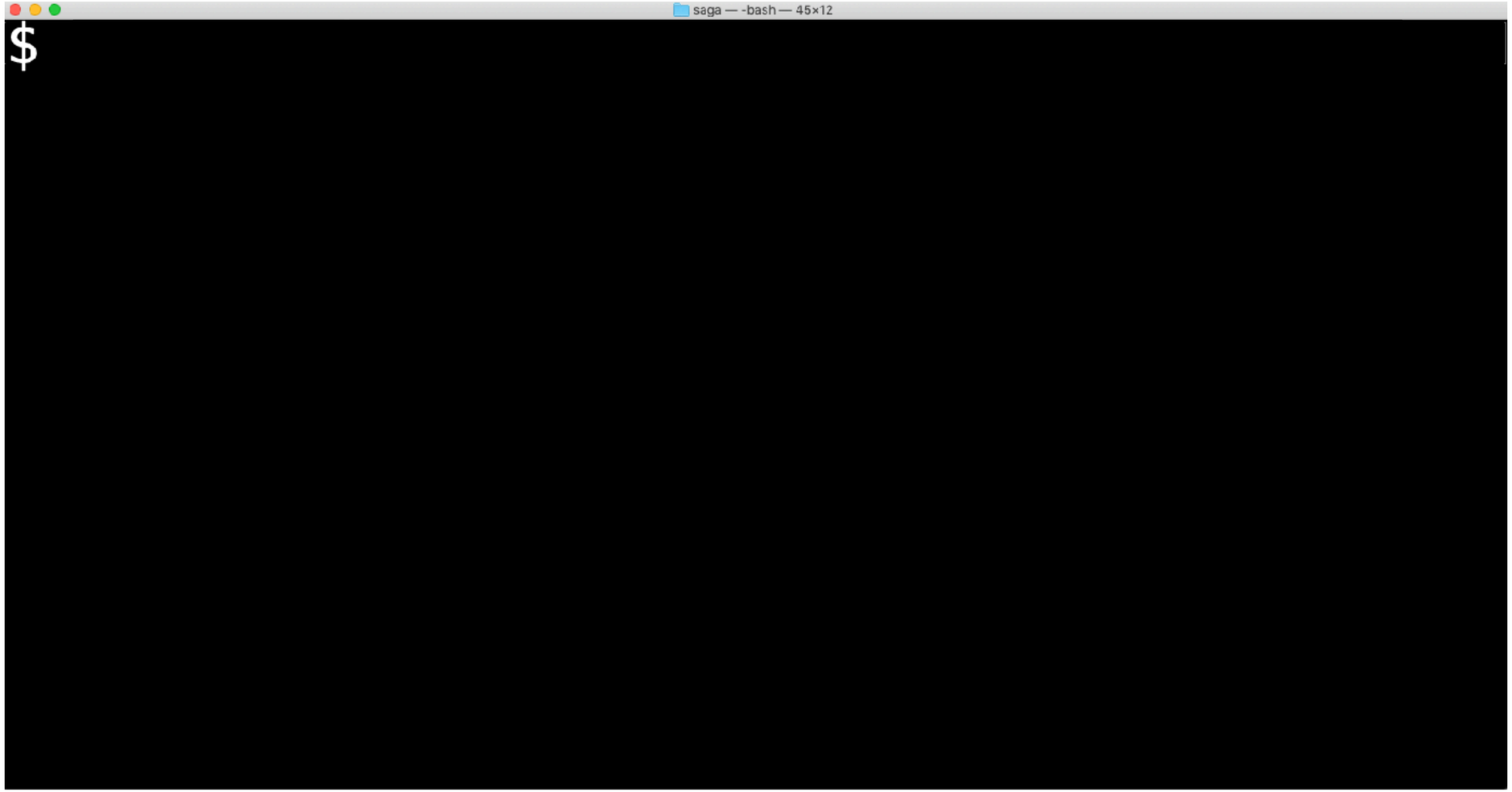
saga — -bash — 45×12

```
$ ./saga.sh -list
```

managing workflows

```
saga — -bash — 45×12
$ ./saga.sh -list

SAGAS:
   ORDER_PLACEMENT
   ORDER_CANCEL
   NEW_CUSTOMER

$
```

managing workflows



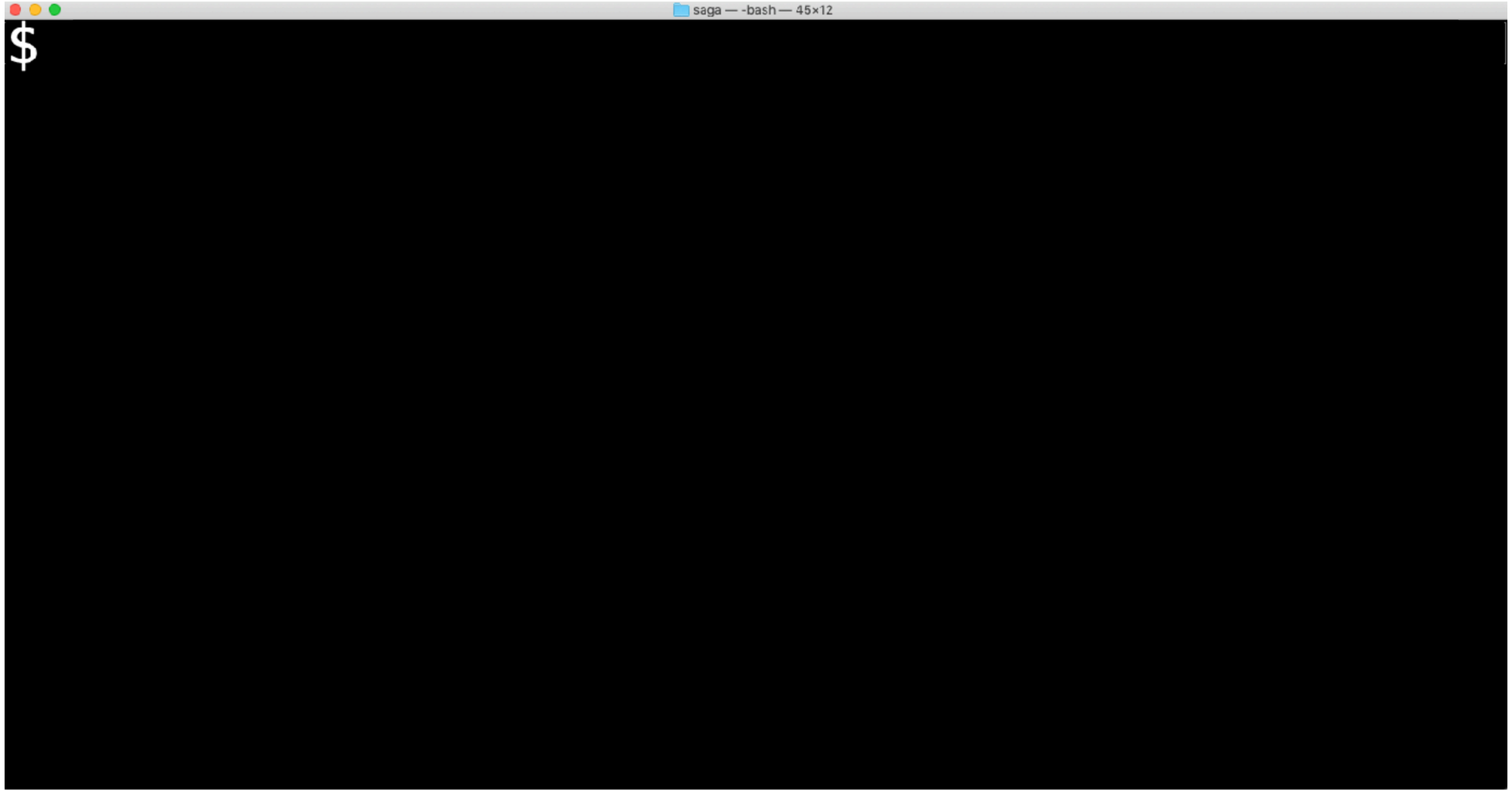
managing workflows

```
saga — -bash — 45×12
$ ./saga.sh ORDER_PLACEMENT -services
```

managing workflows

```
saga — -bash — 45×12
$ ./saga.sh ORDER_PLACEMENT -services

ORDER_PLACEMENT
  |
  |- OrderPlacementService
  |- PaymentService
  |- FulfillmentService
  |- NotificationService

$
```

managing workflows



managing workflows

```
saga — -bash — 45×12
$ ./saga.sh ORDER_PLACEMENT -statemachine
```

managing workflows

```
saga — -bash — 45×12
$ ./saga.sh ORDER_PLACEMENT -statemachine

ORDER_PLACEMENT
  |
  |- order_created
  |- payment_applied
  |- payment_rejected
  |- order_shipped
  |- order_delivered

$
```

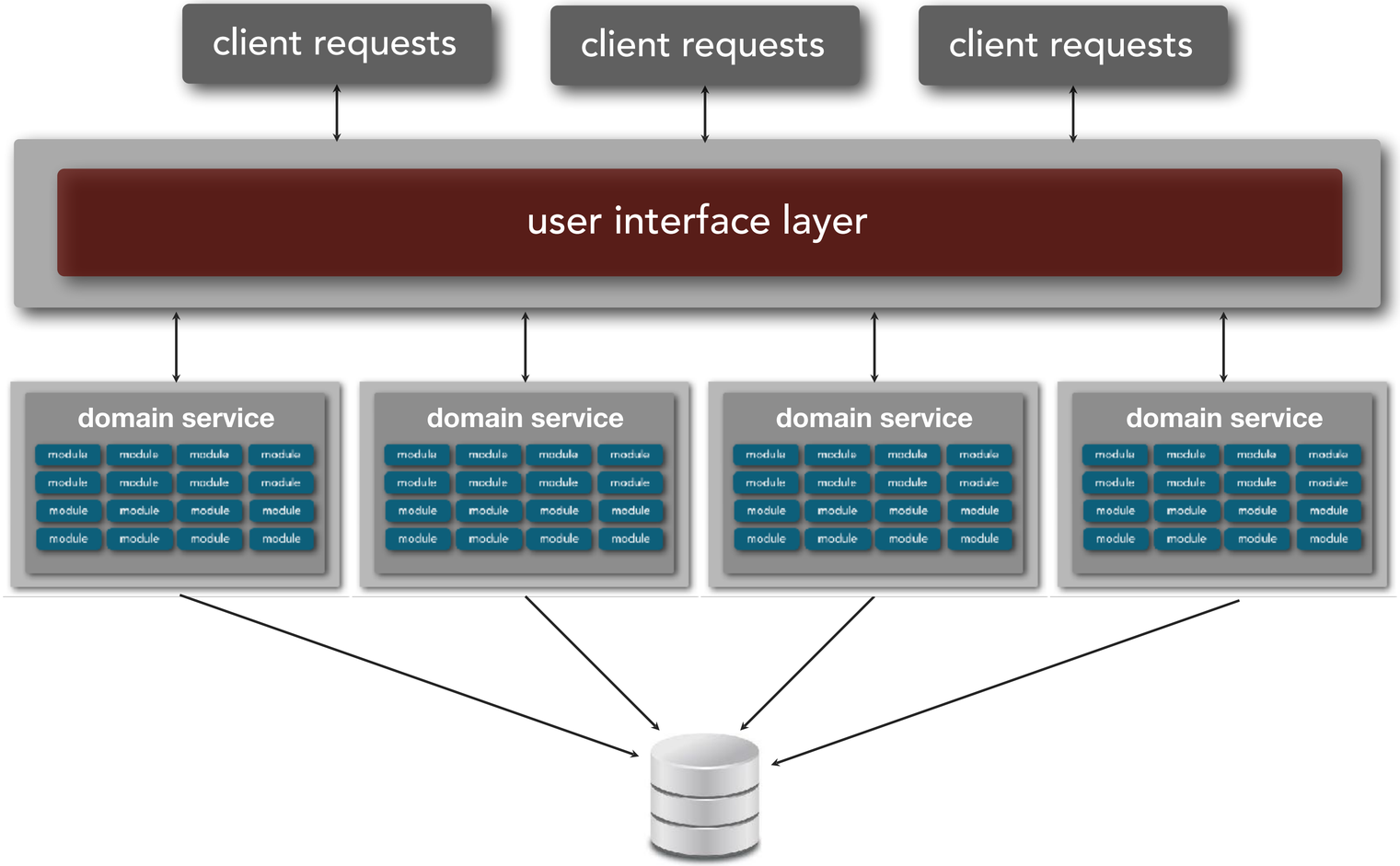
managing workflows

```
@ServiceEntrypoint
@Saga(Transaction.ORDER_PLACEMENT)
public class OrderPlacementServiceAPI {
    ...
}
```

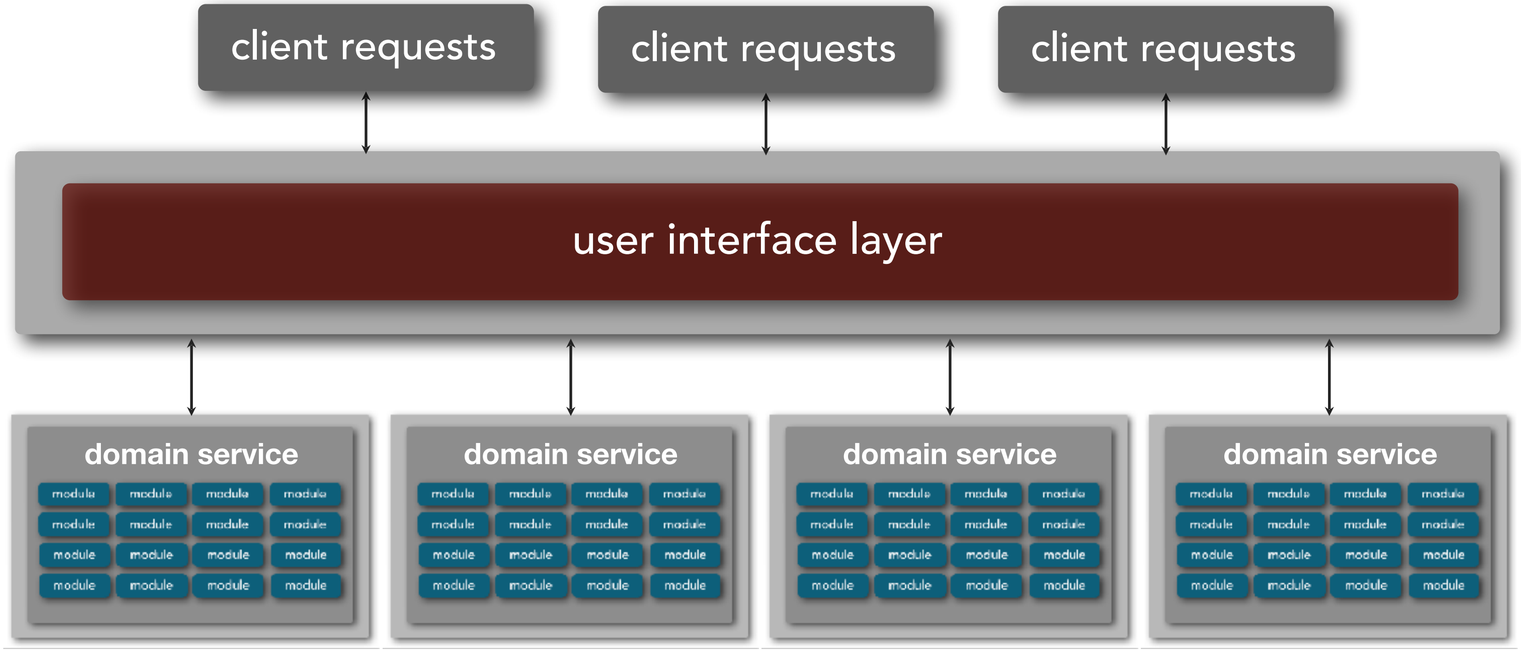
only those transactions that span multiple services are sagas!

Hybrid Architectures

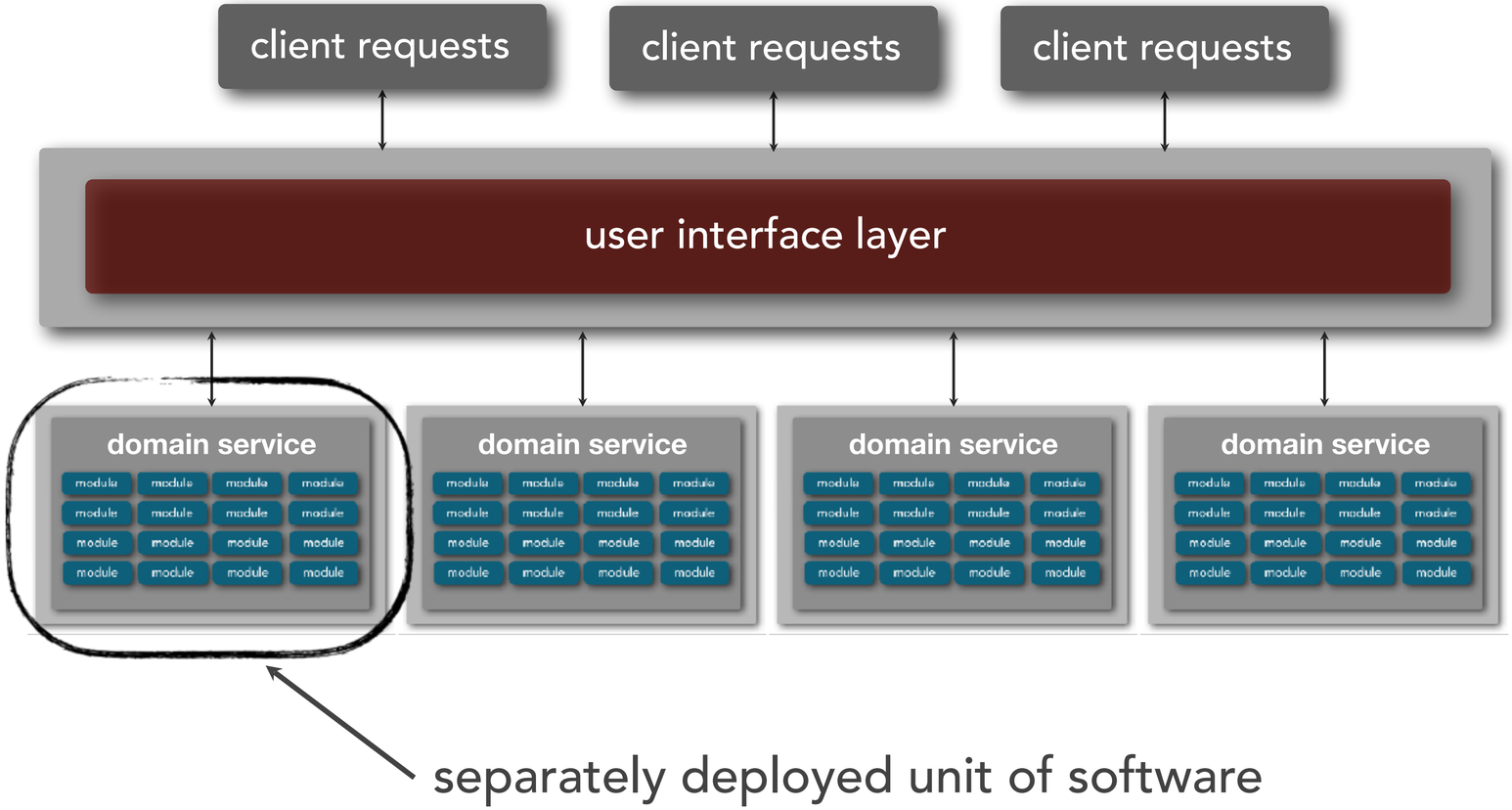
service-based architecture



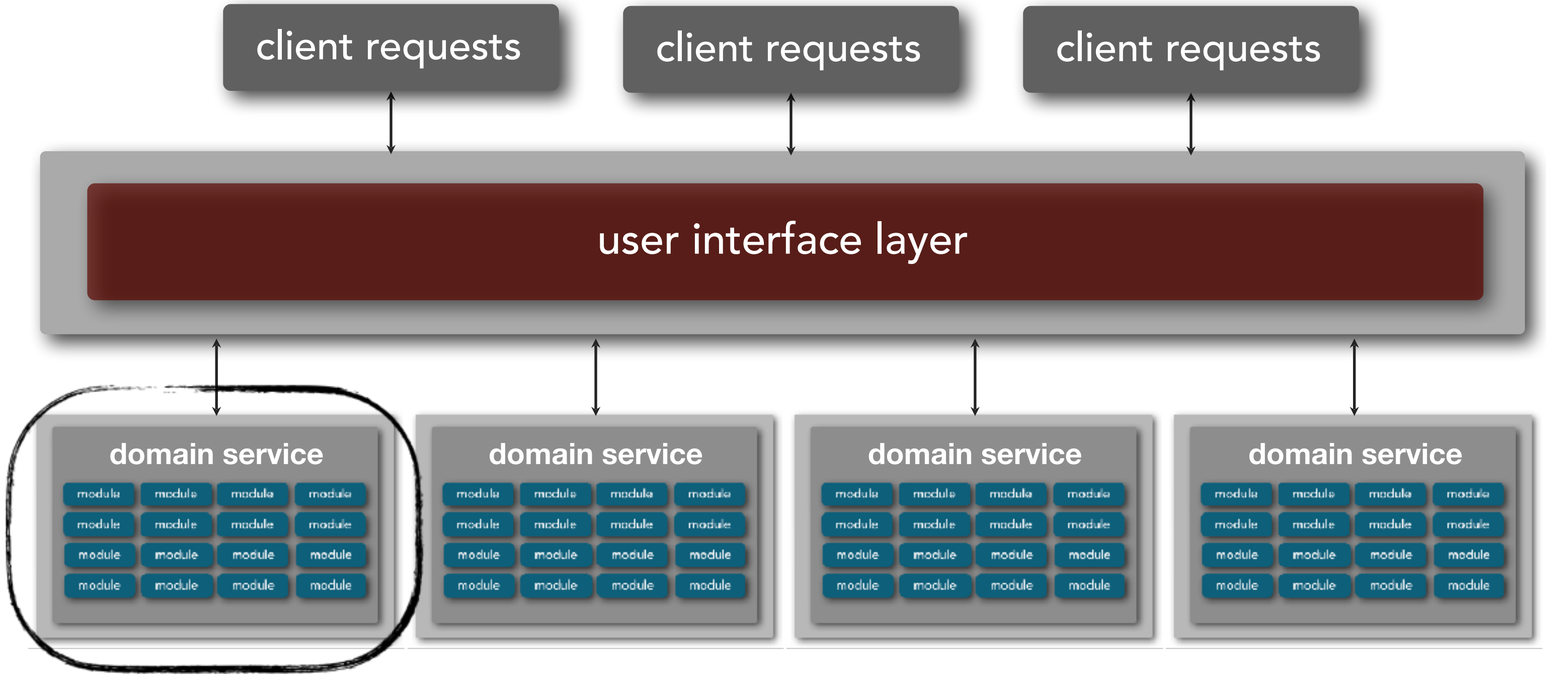
service-based architecture



service-based architecture

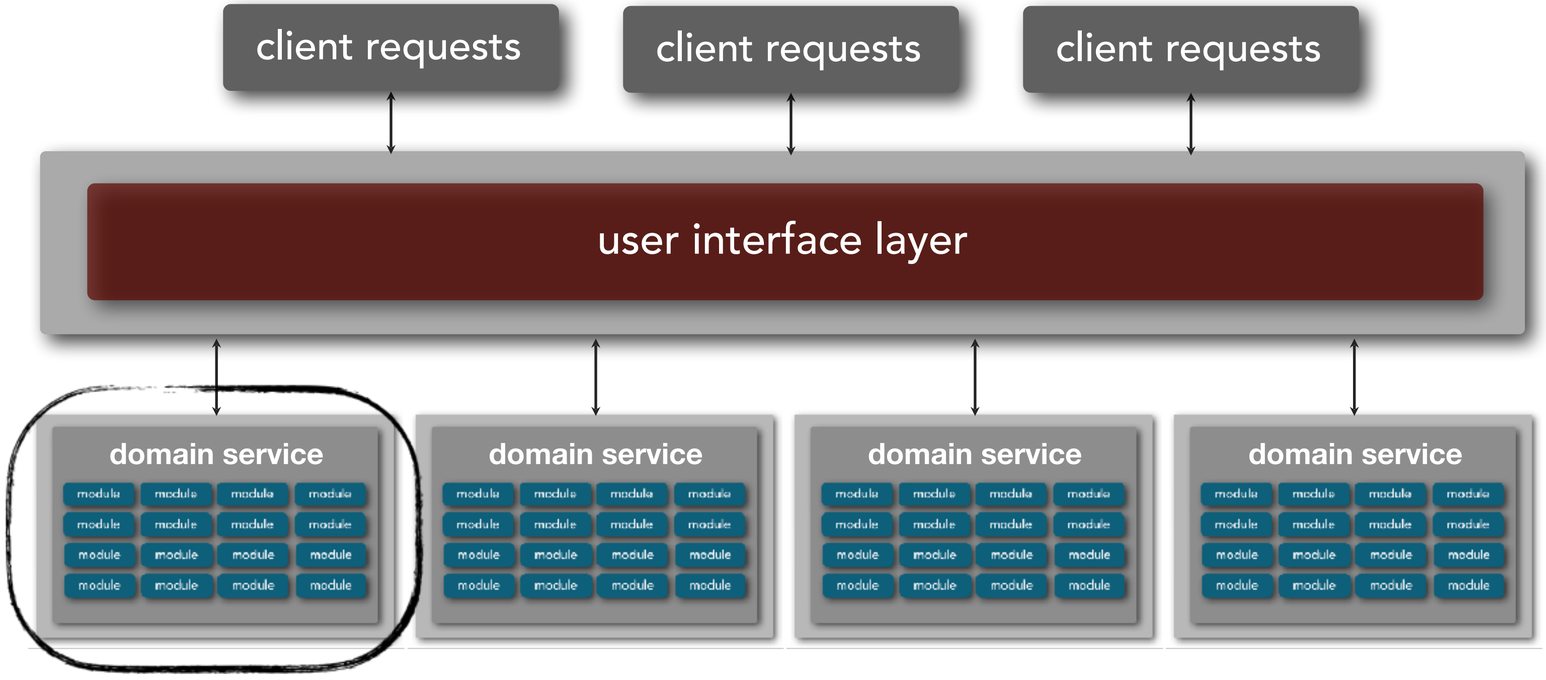


service-based architecture



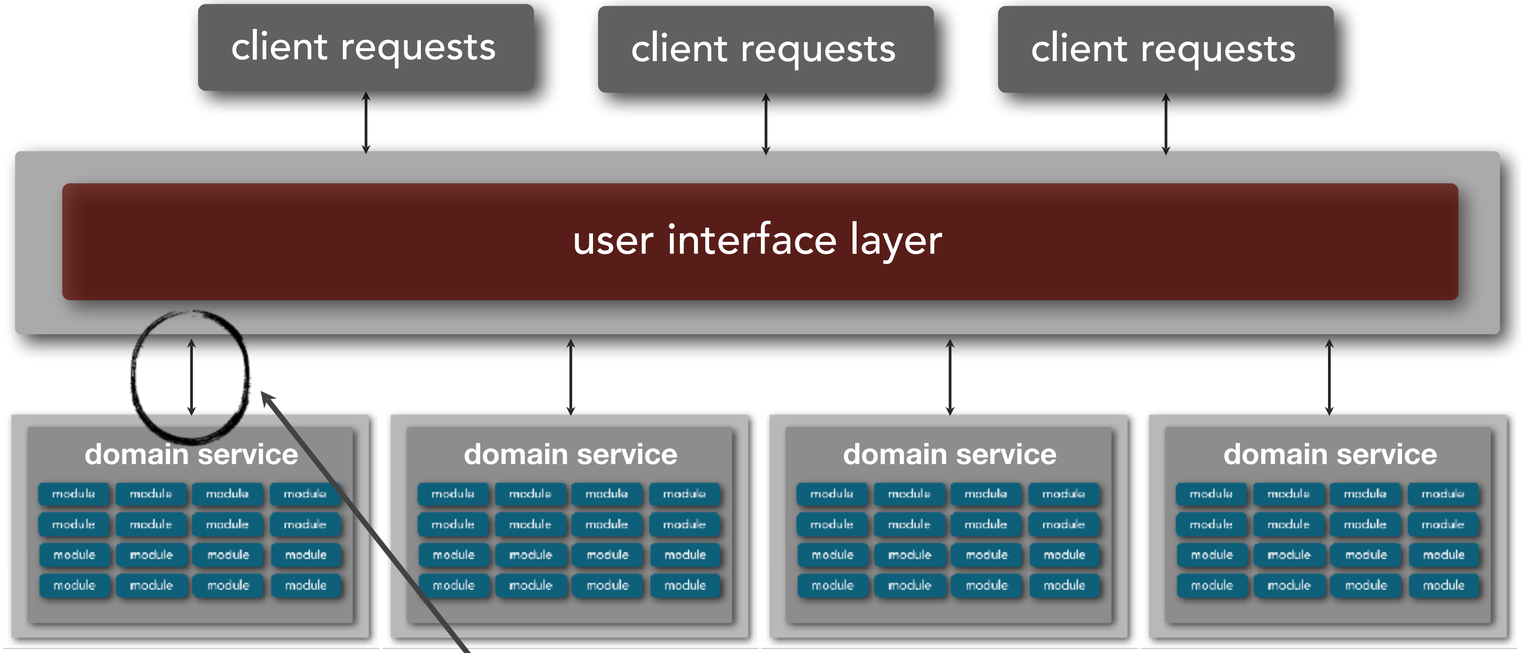
same deployment unit as original application
(ear, assembly, etc.)

service-based architecture



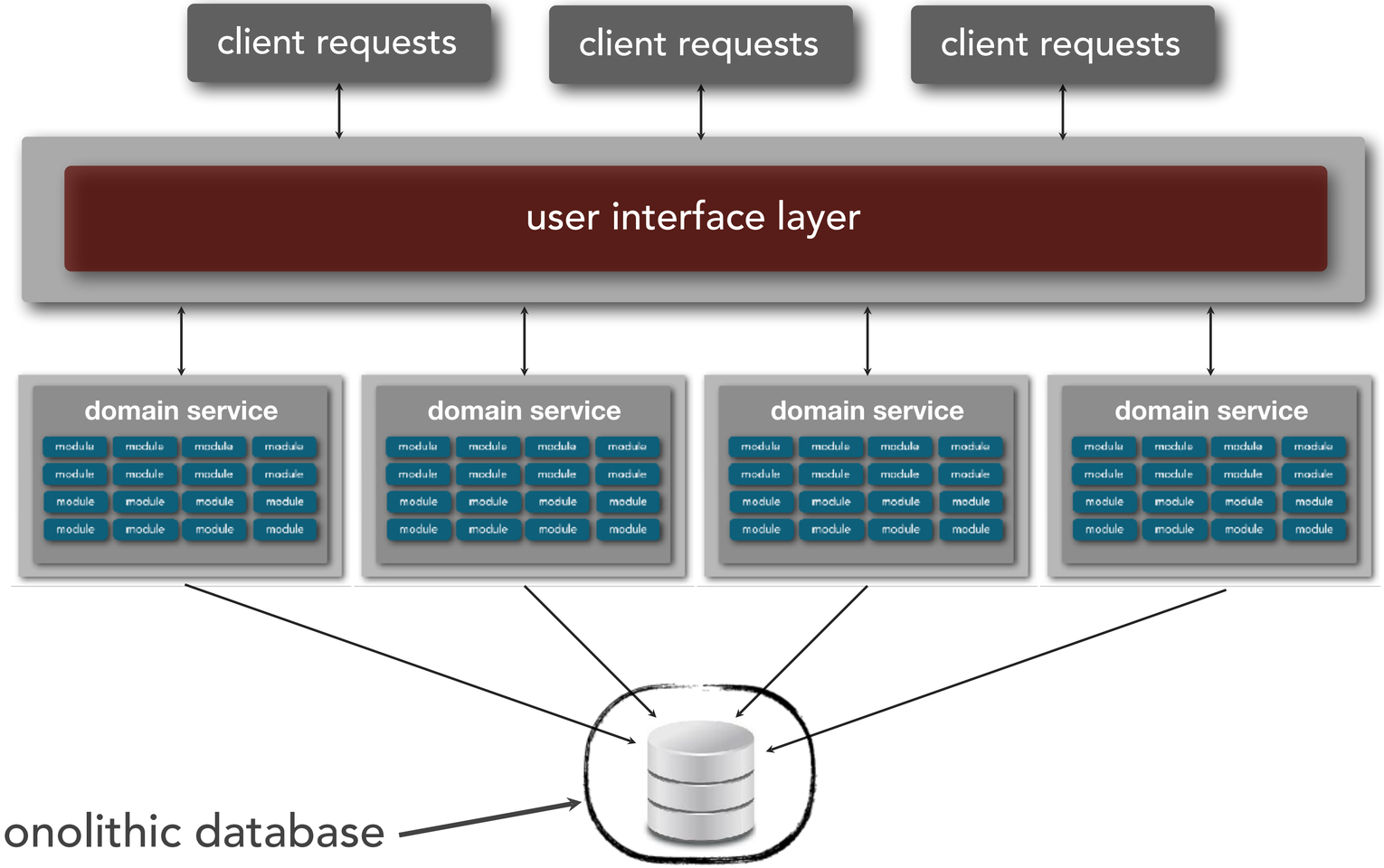
independent self-contained portion of the application (4 - 12 services)

service-based architecture



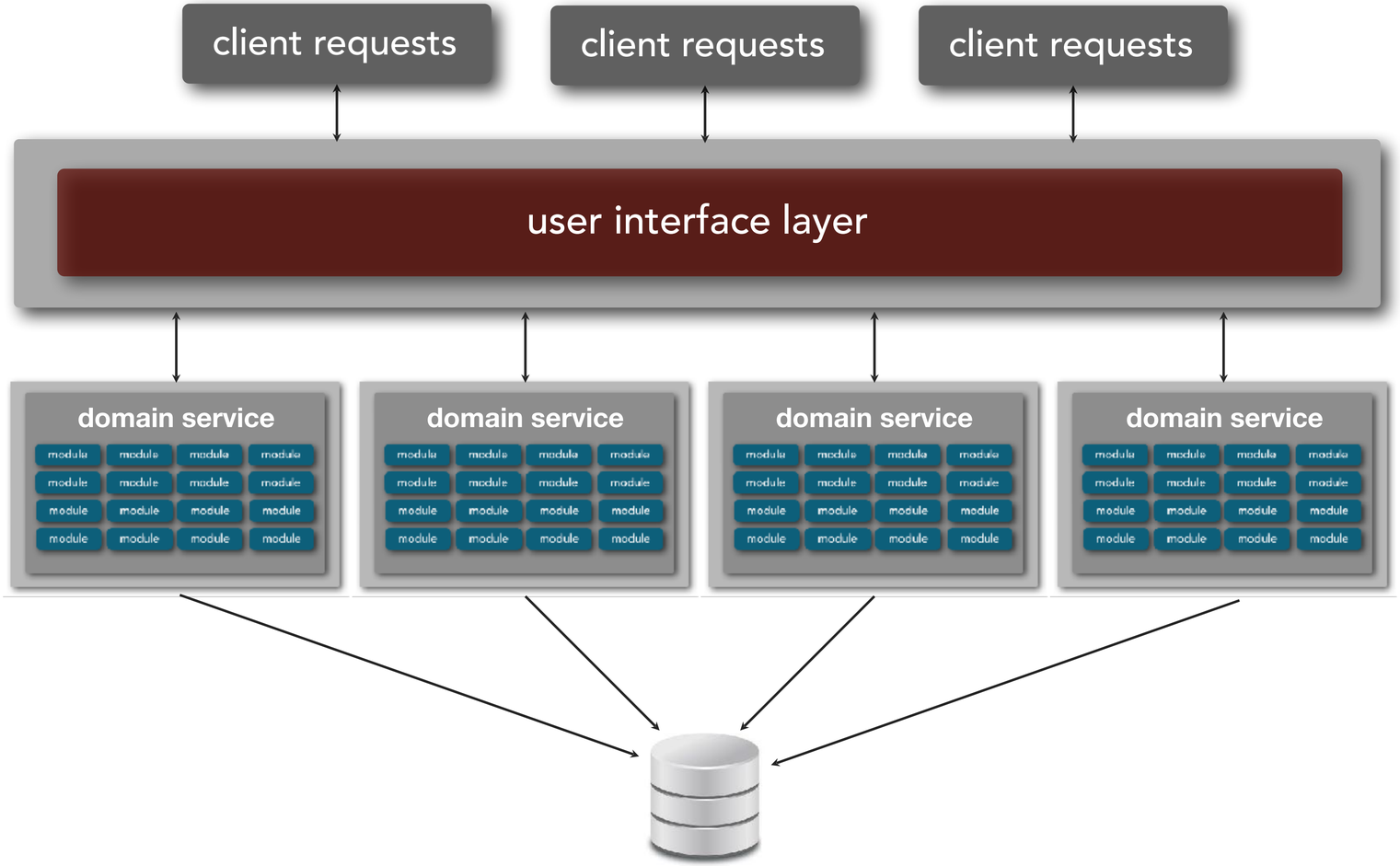
usually rest, soap, messaging, or remote procedure call

service-based architecture

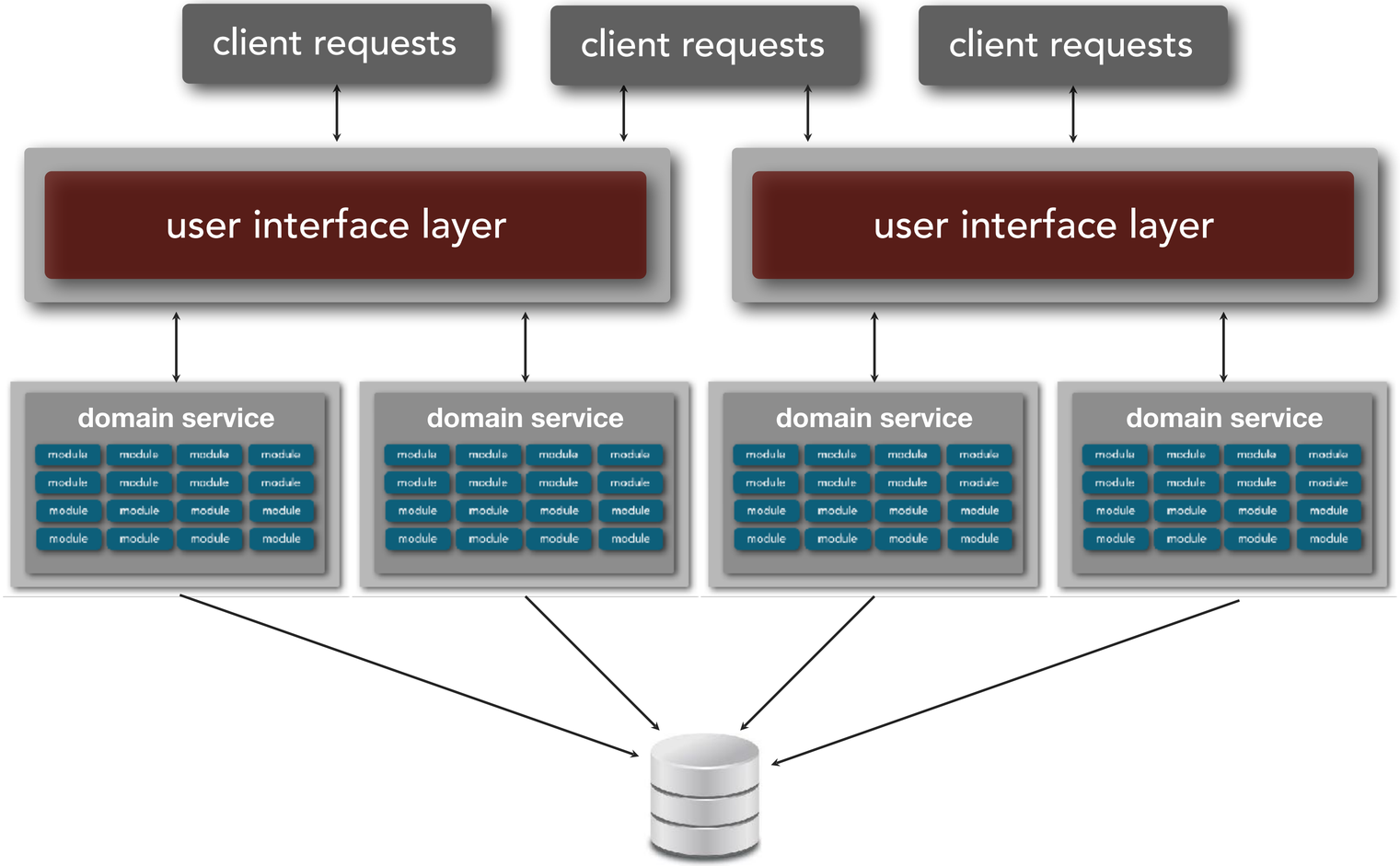


share a single monolithic database

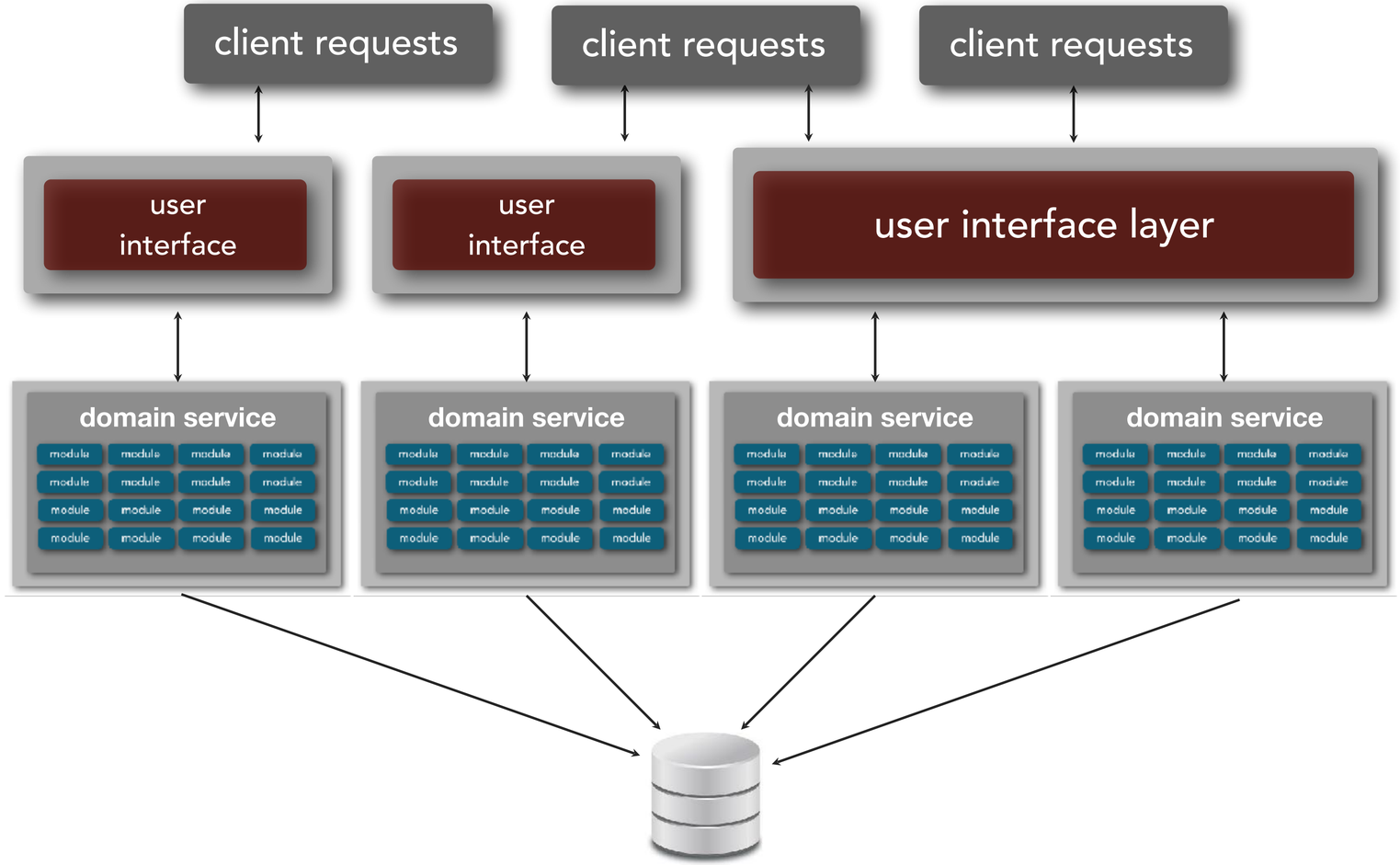
service-based architecture



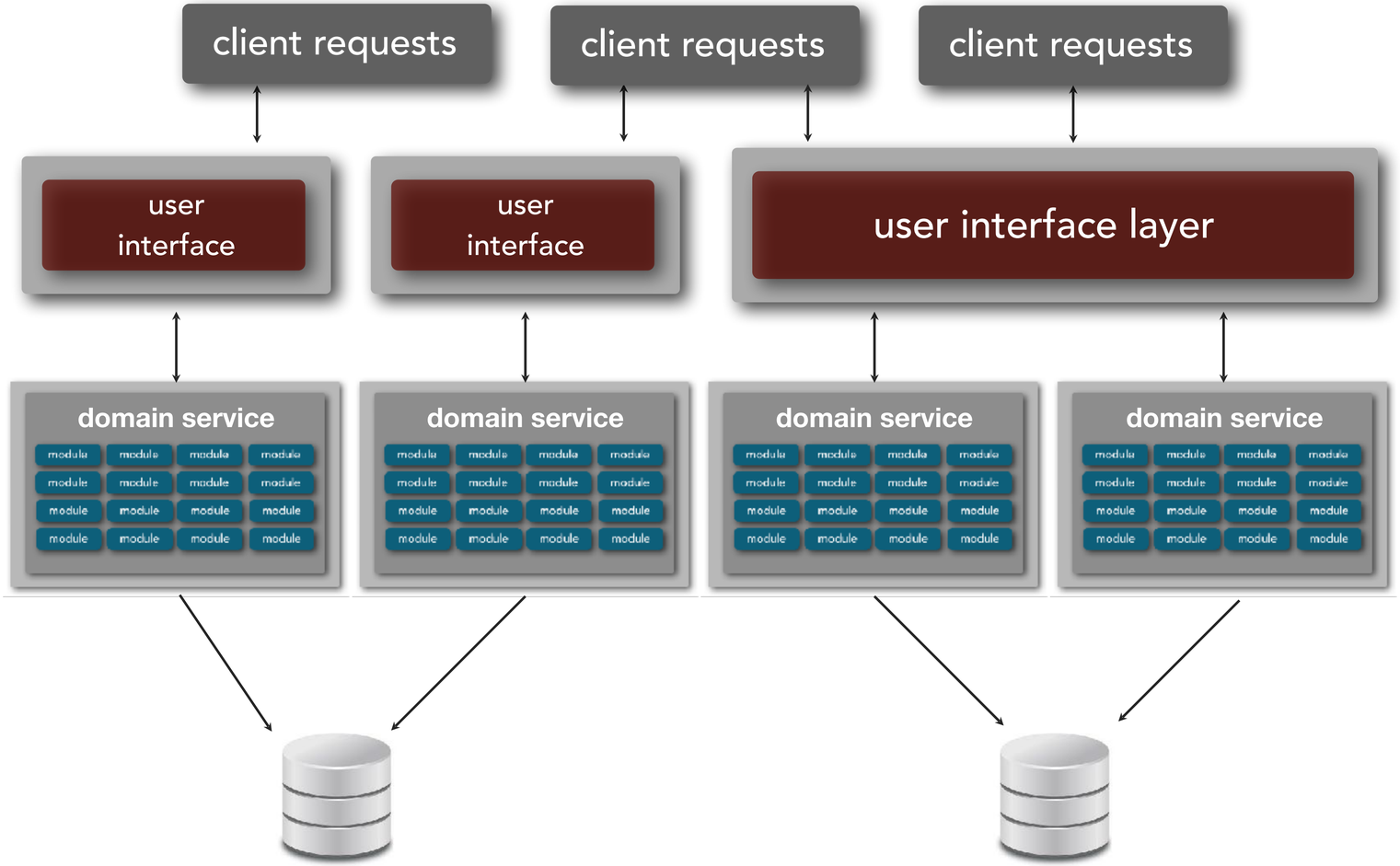
service-based architecture



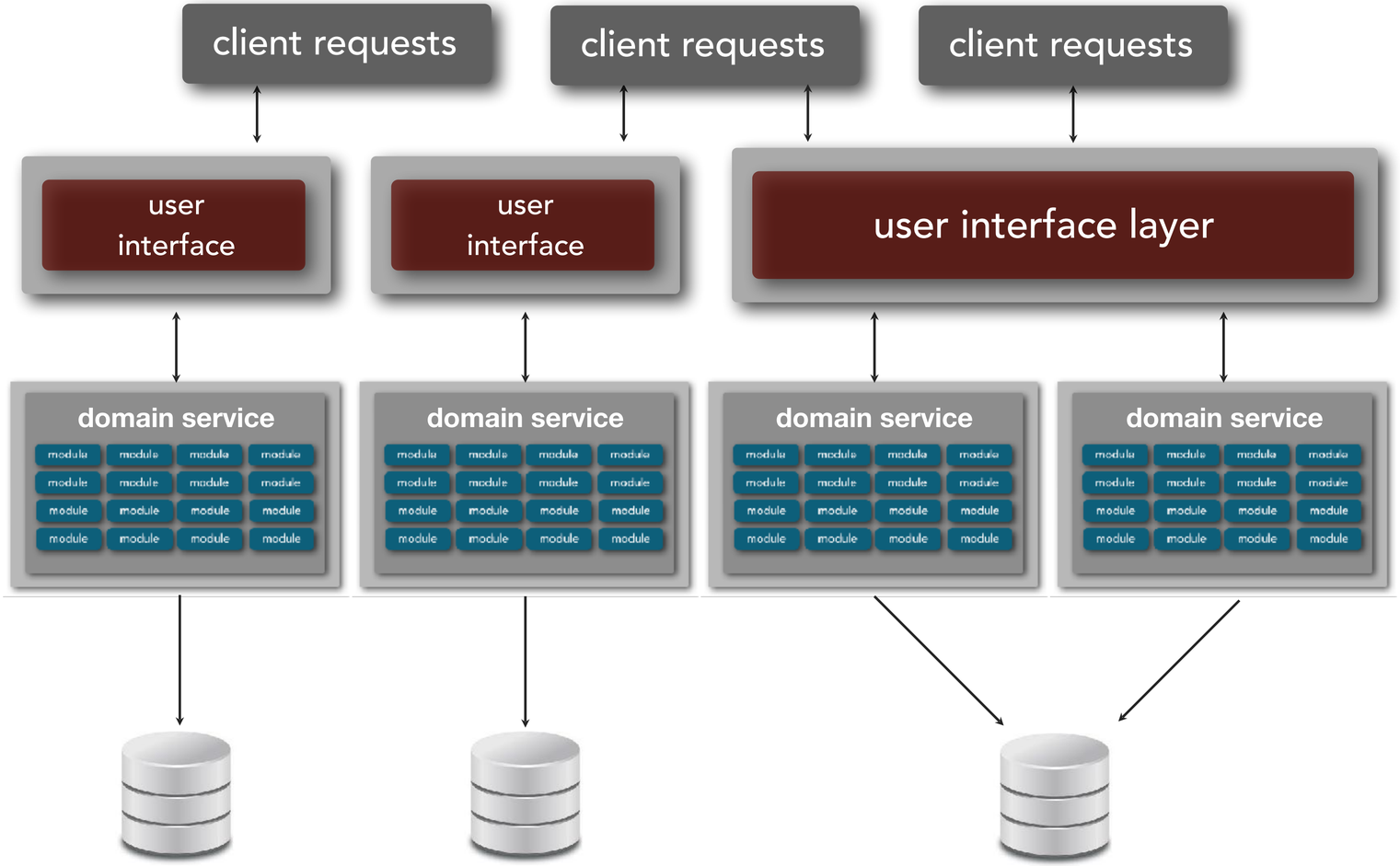
service-based architecture



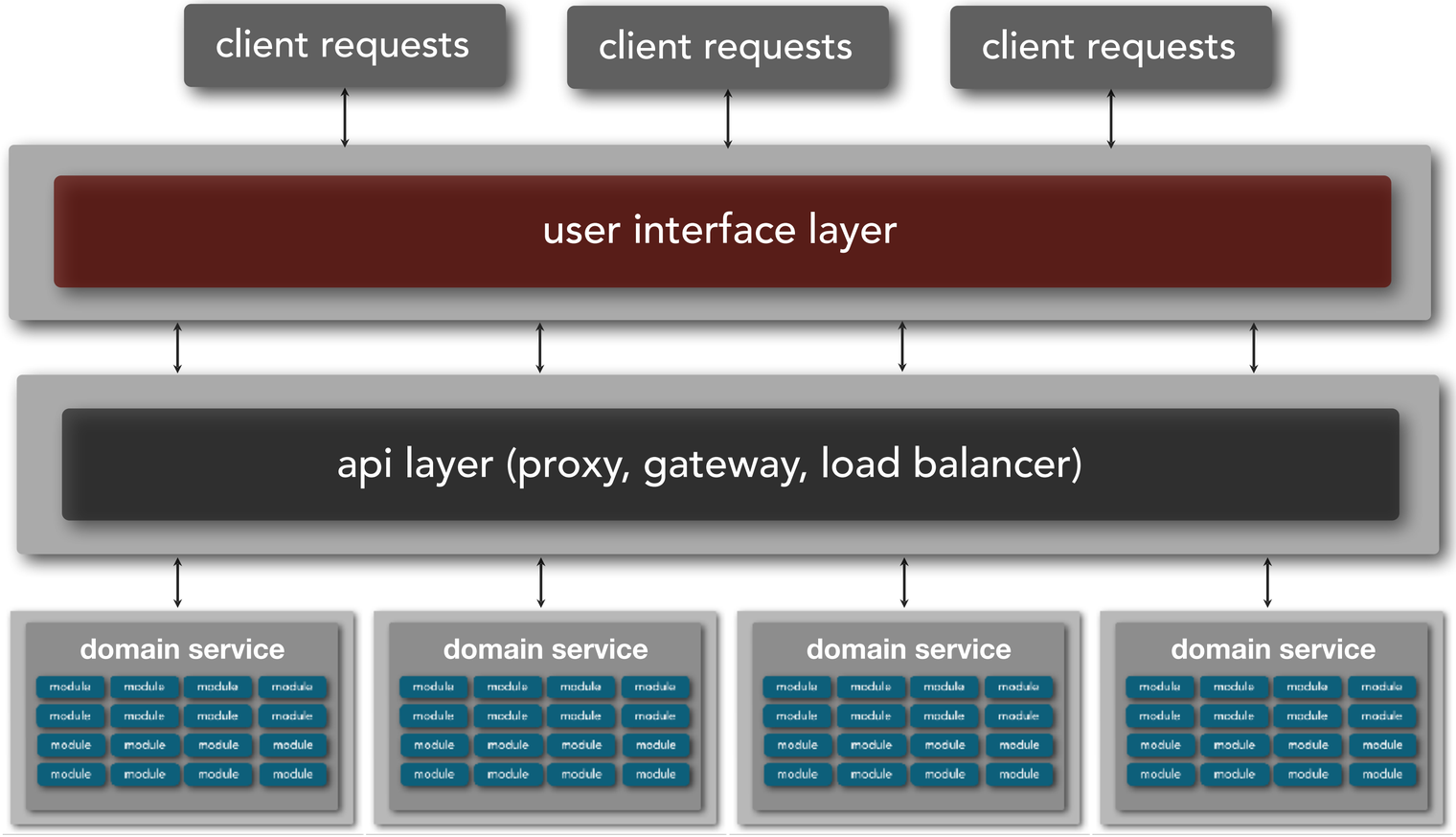
service-based architecture



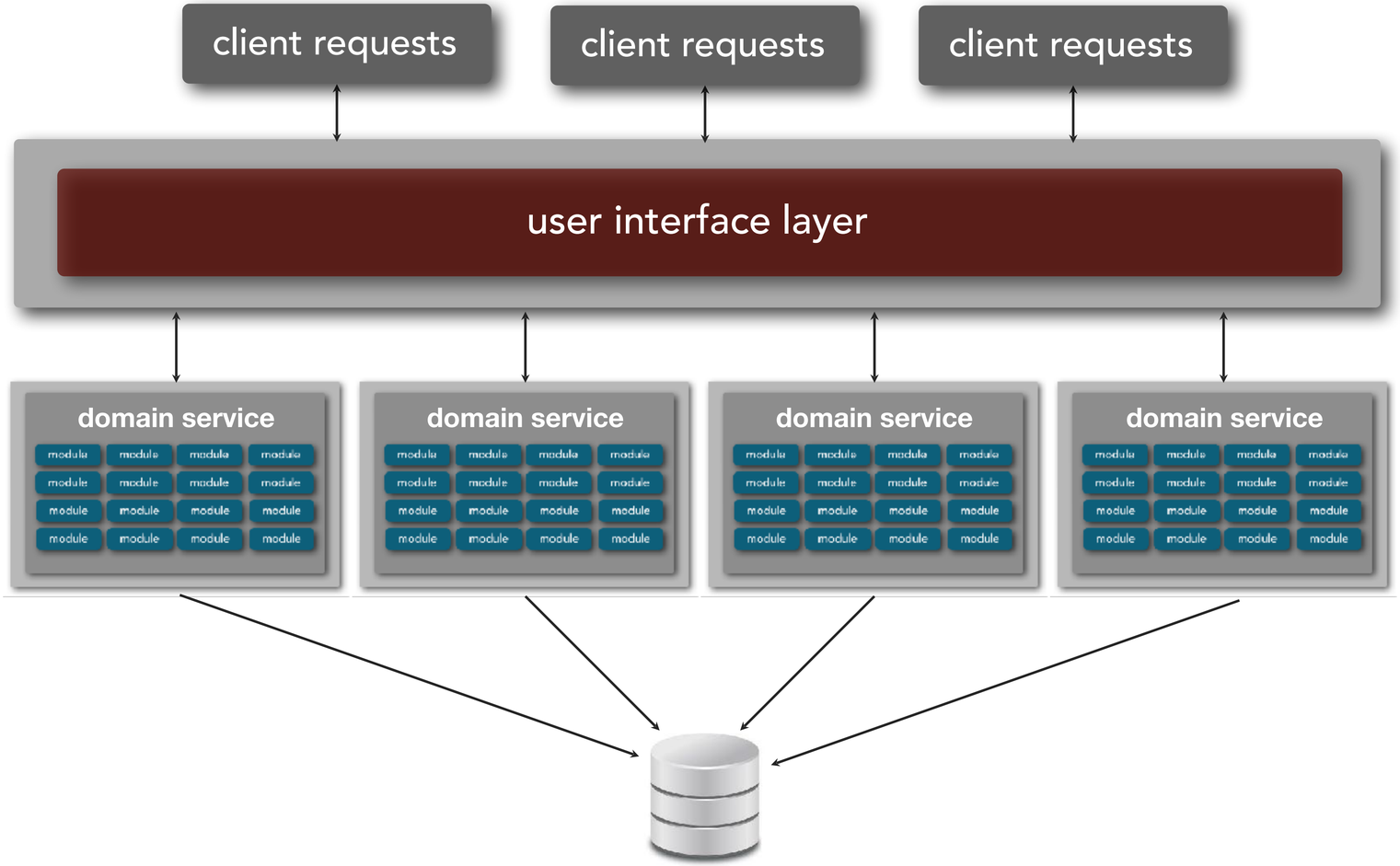
service-based architecture



service-based architecture

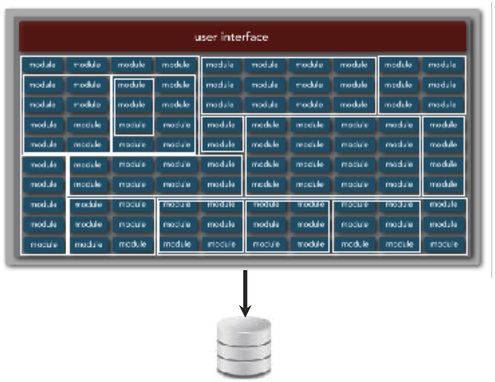


service-based architecture



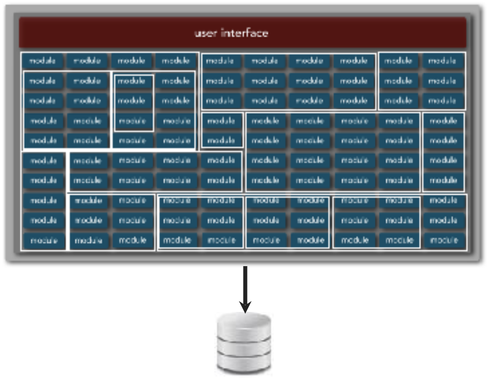
service-based architecture

service-based architecture

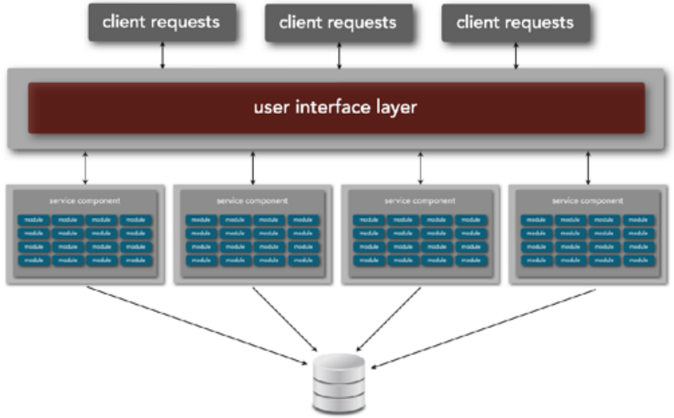


monolithic architecture

service-based architecture

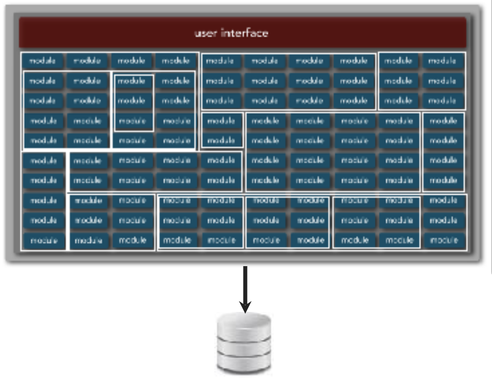


monolithic architecture

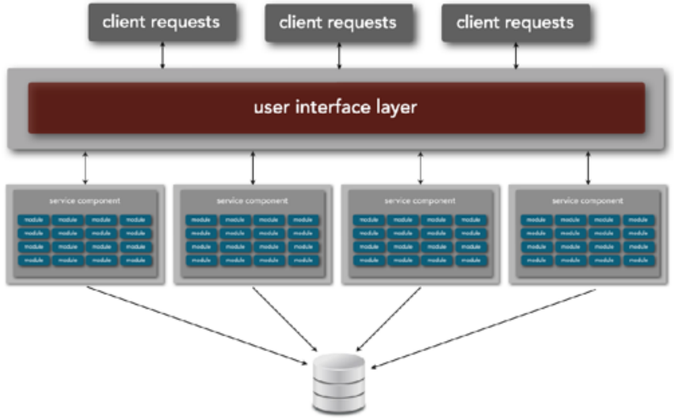


service-based architecture

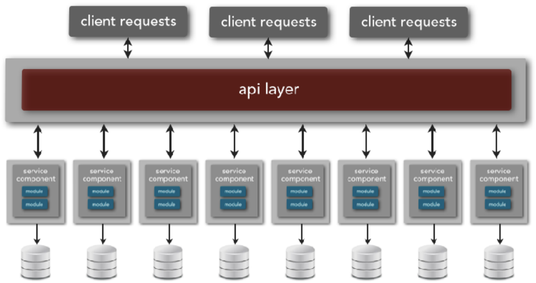
service-based architecture



monolithic architecture

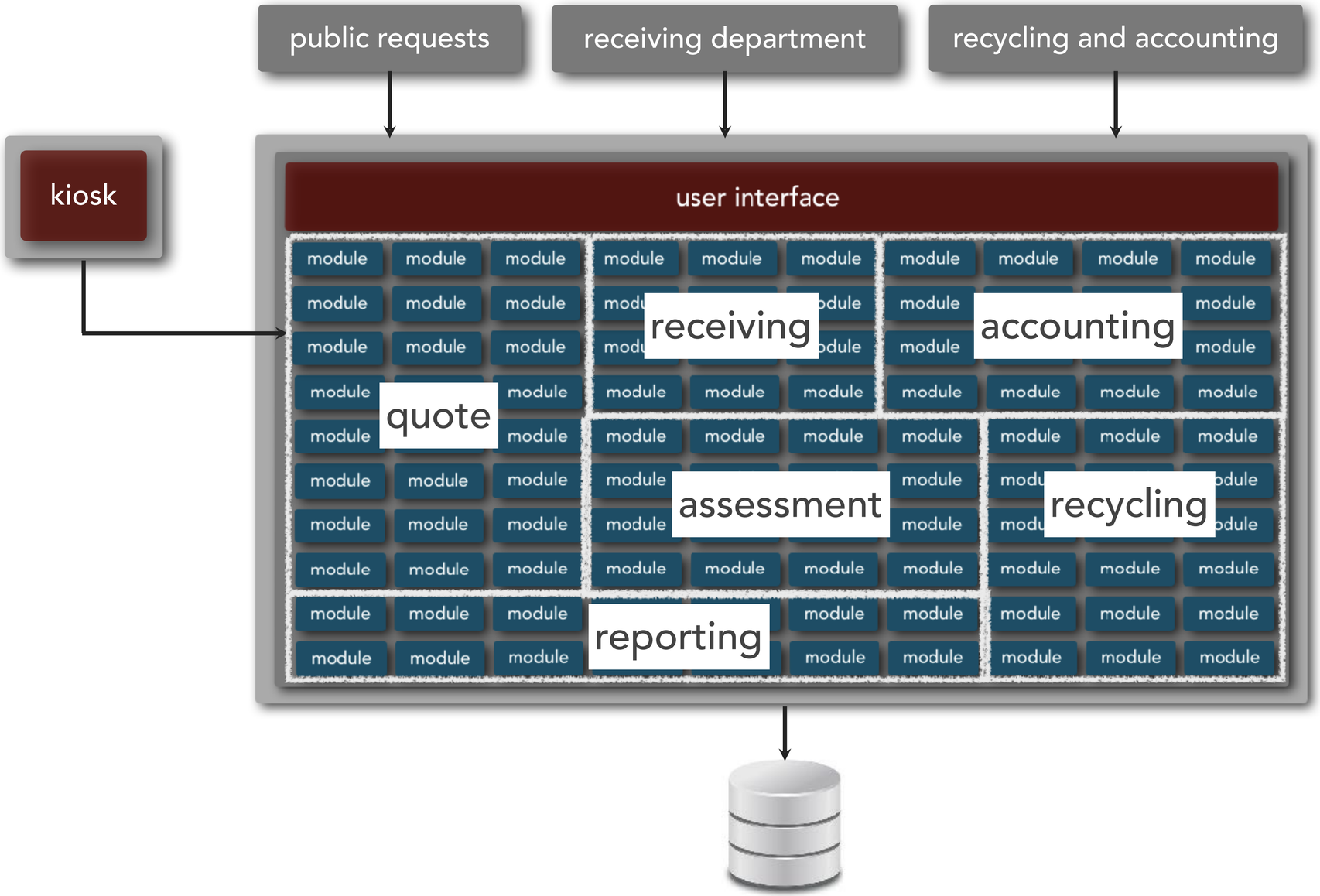


service-based architecture

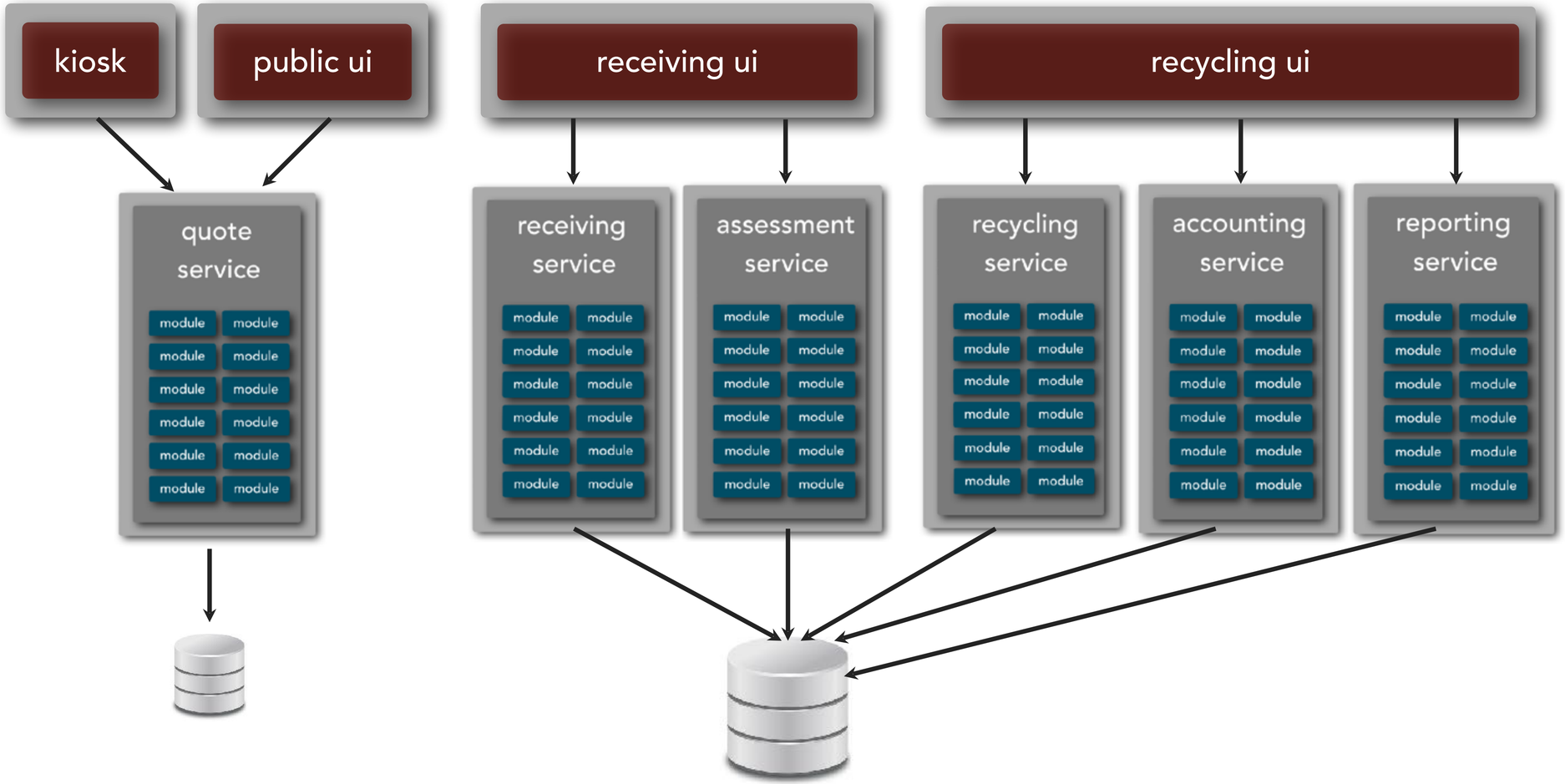


microservices architecture

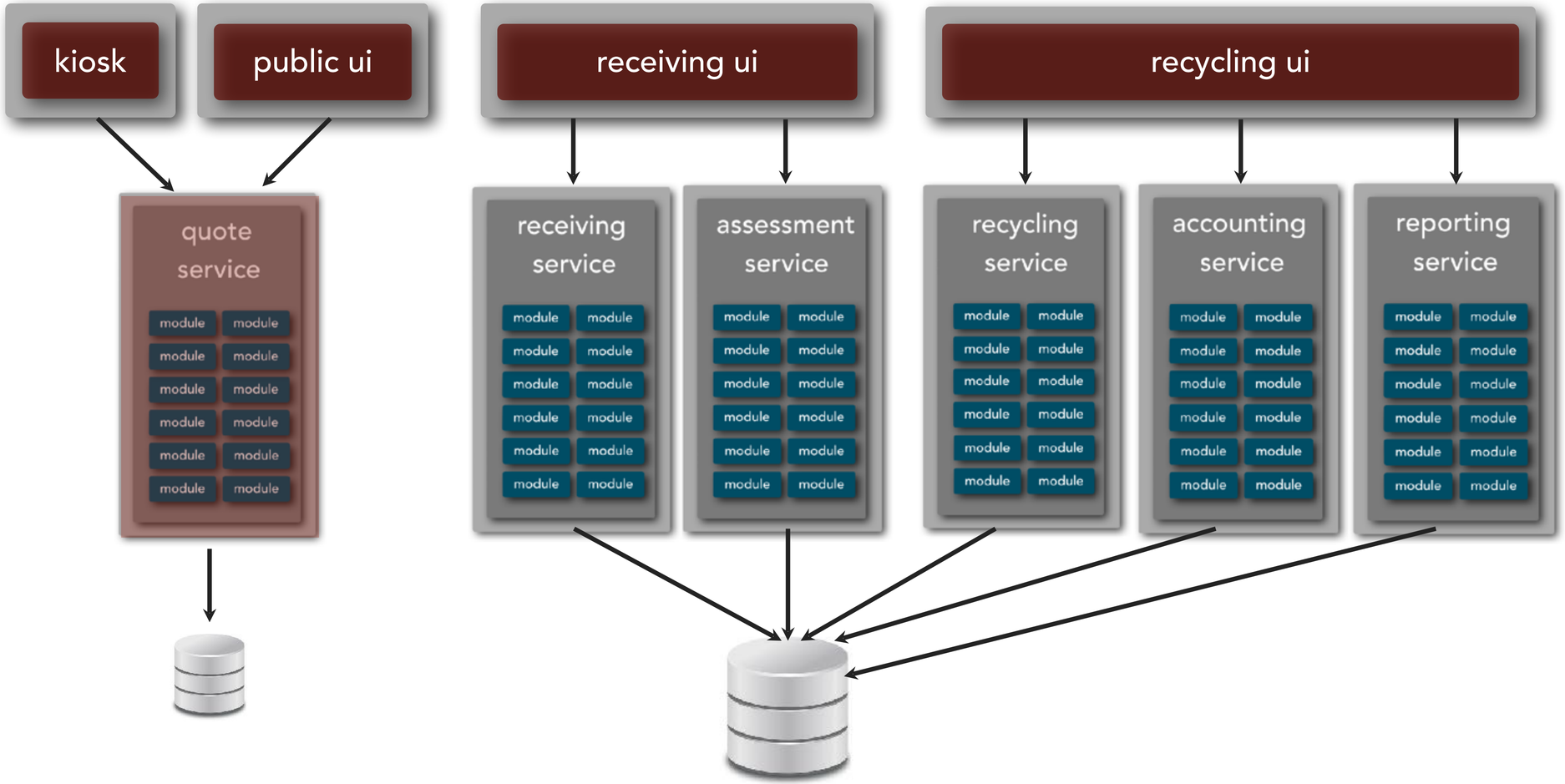
electronics recycling application



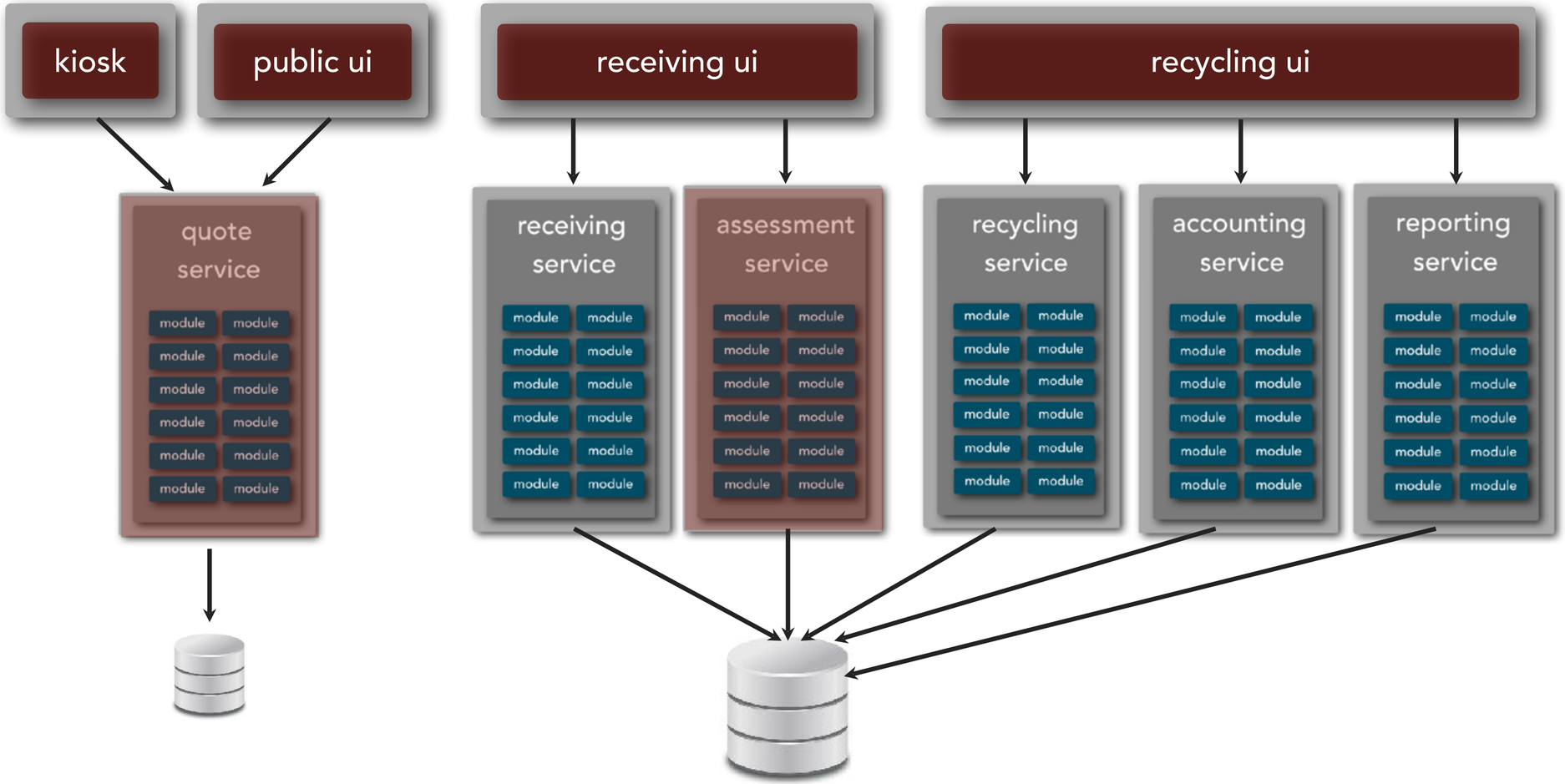
electronics recycling application



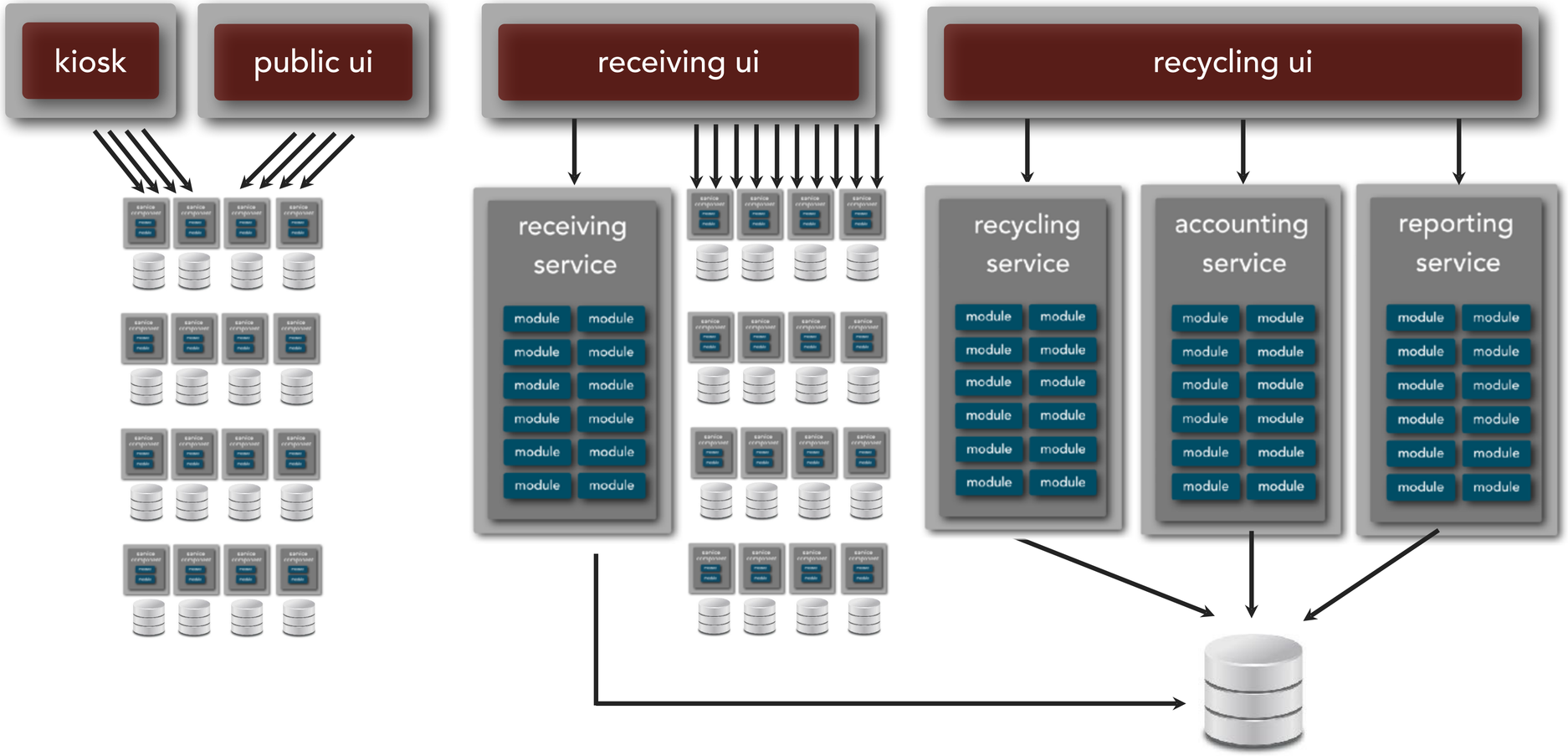
electronics recycling application



electronics recycling application



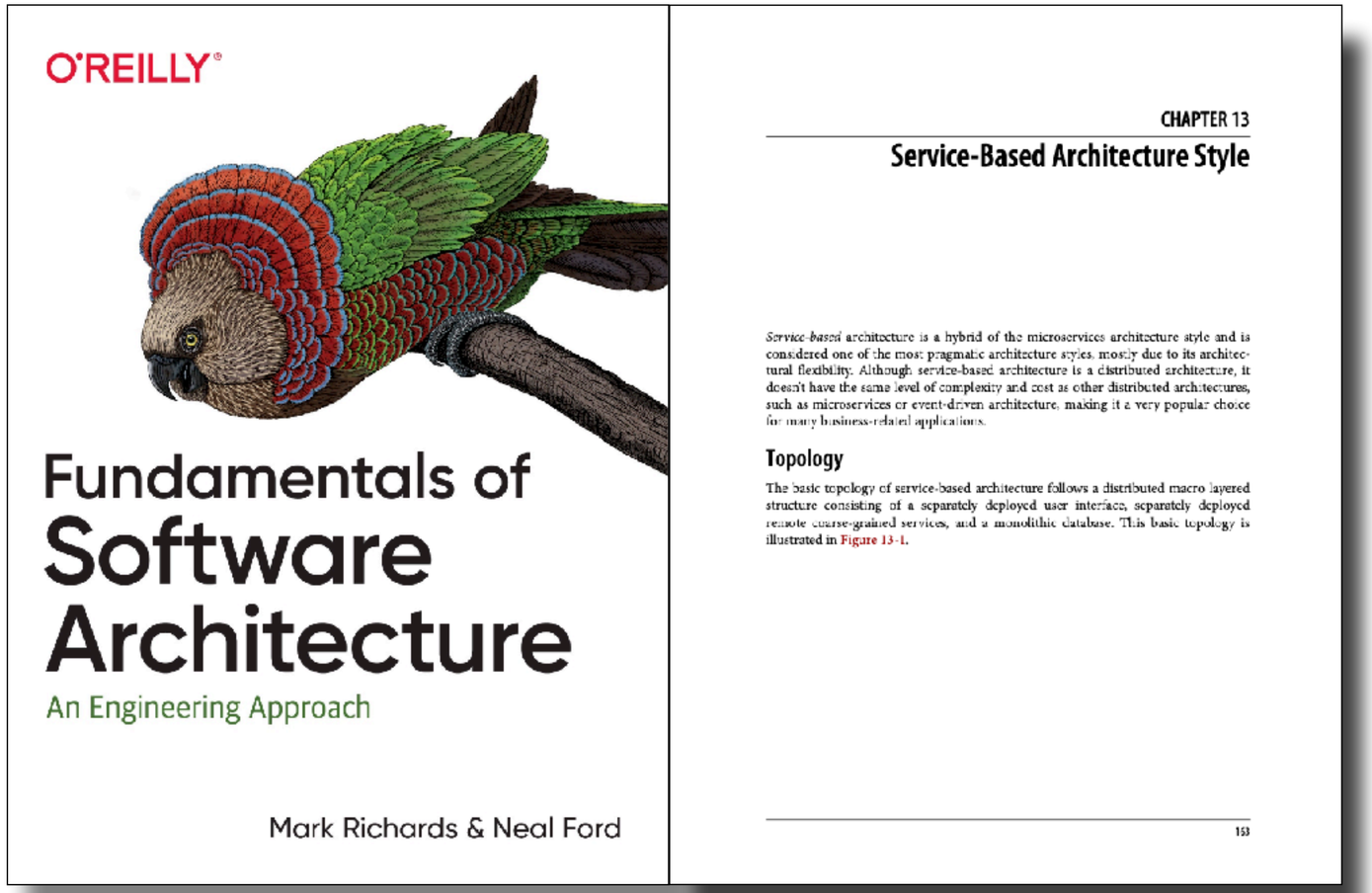
electronics recycling application



Fundamentals of Software Architecture

by Mark Richards and Neal Ford

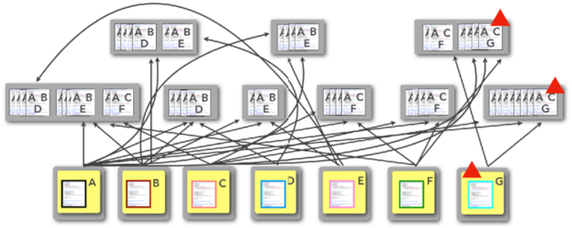
https://www.amazon.com/gp/product/1492043451

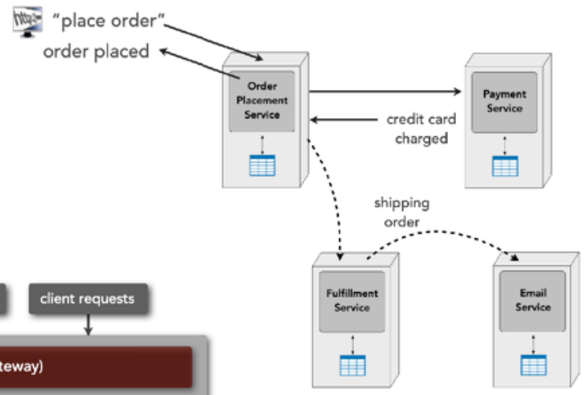


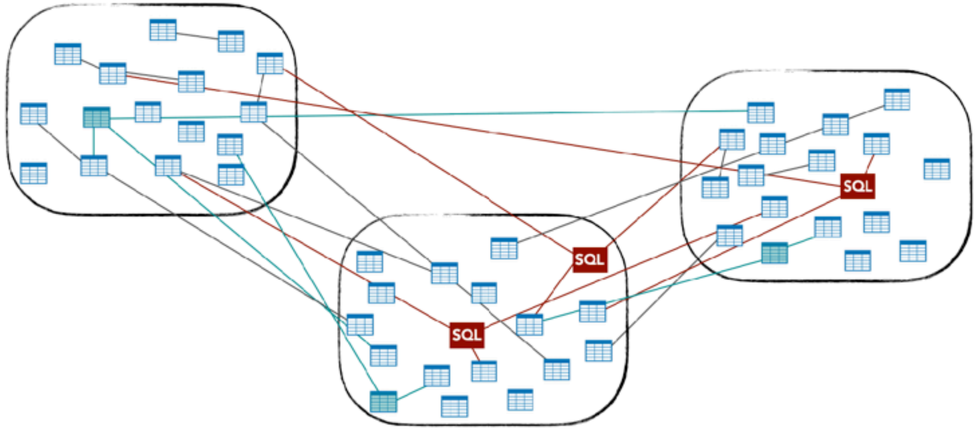
Summary

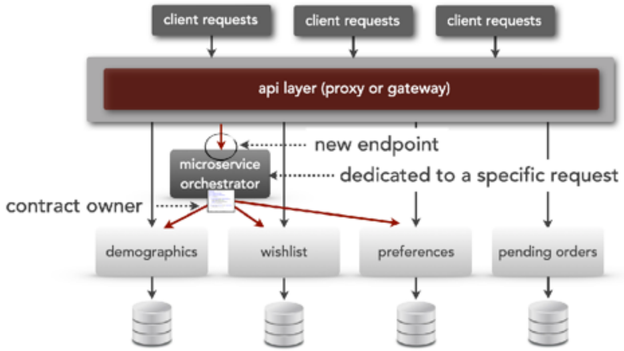
avoid jumping on the bandwagon; know when to use microservices and why

understand the challenges of microservices and determine whether it is the right architectural fit

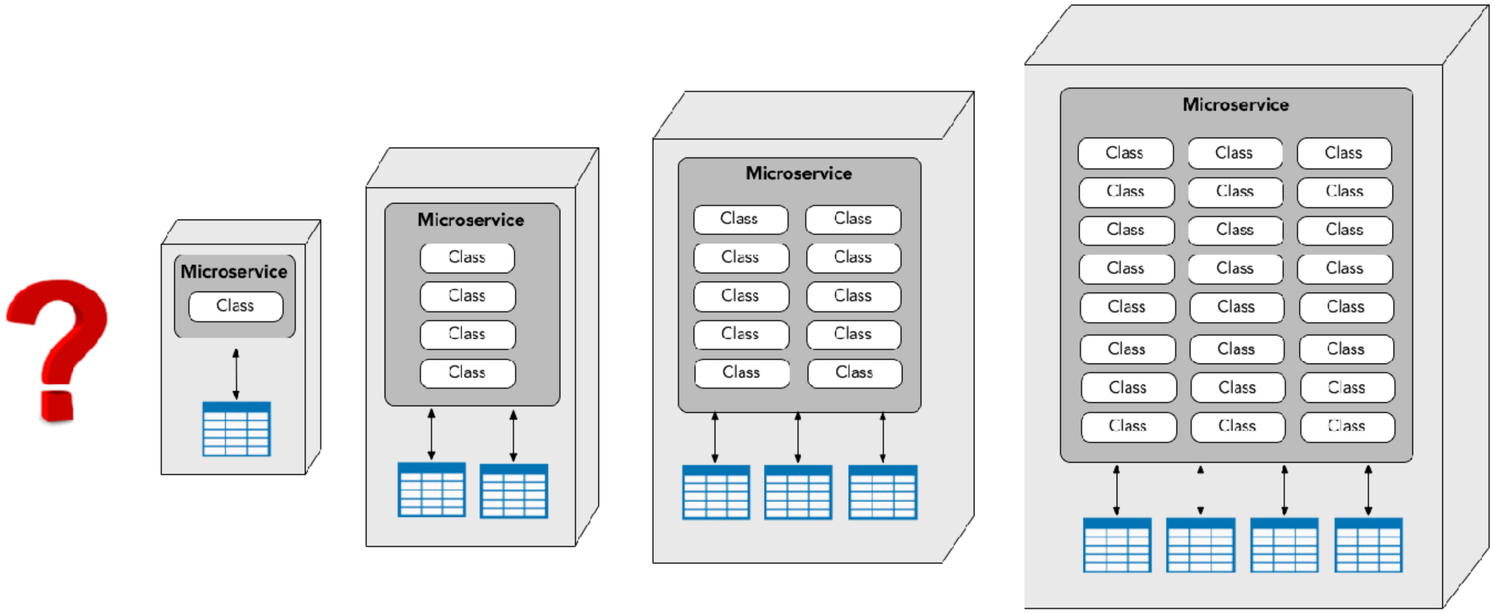








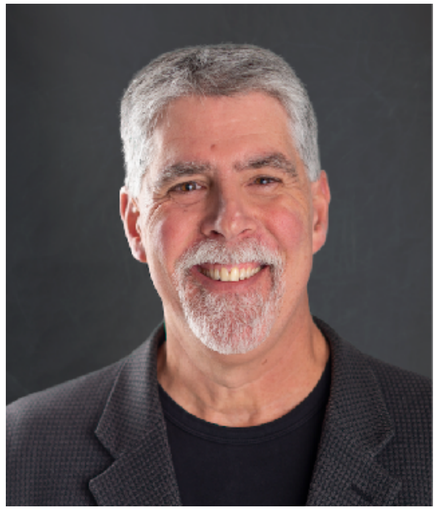
embrace modularity but be cautious of granularity



O'REILLY®

LIVE ONLINE TRAINING

Microservices Architecture and Design



Mark Richards

Independent Consultant

Hands-on Software Architect / Published Author

Founder, DeveloperToArchitect.com

https://www.linkedin.com/in/markrichards3

@markrichardssa