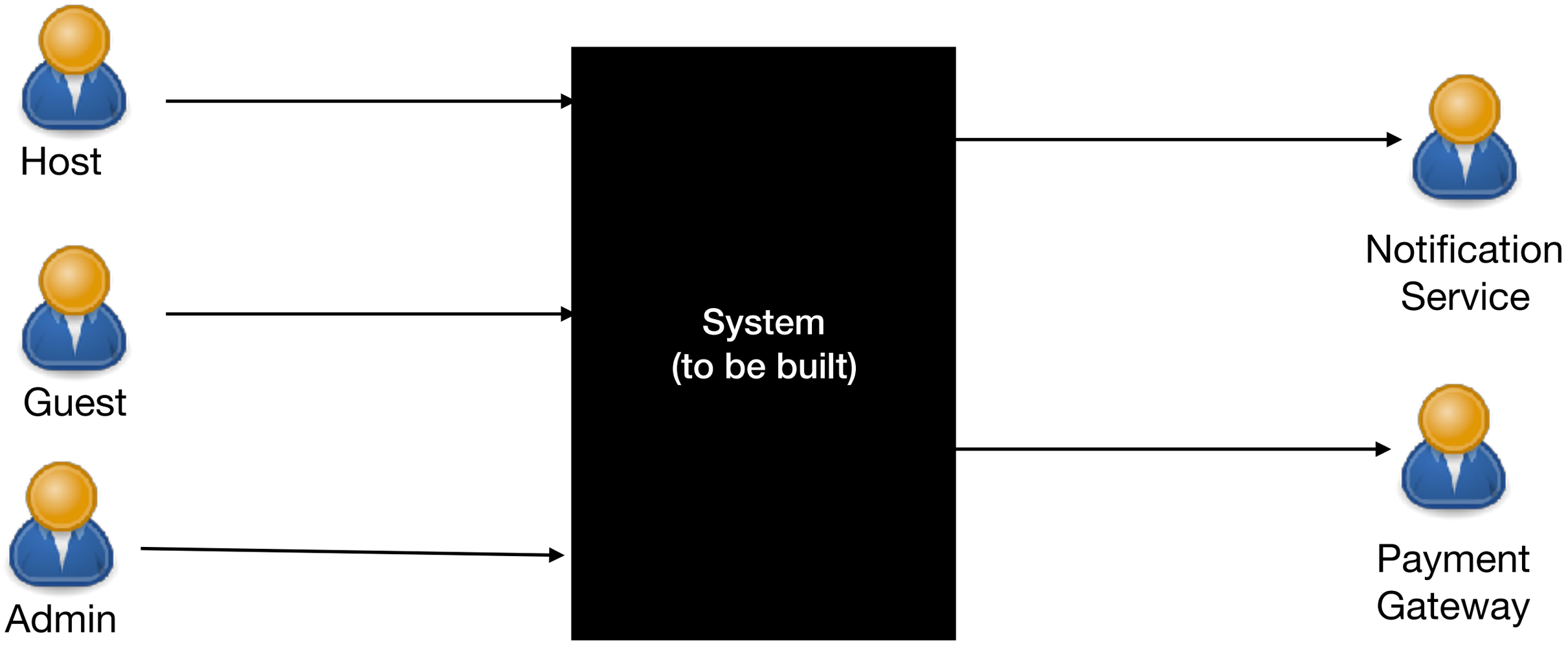
Airbnb Case Study

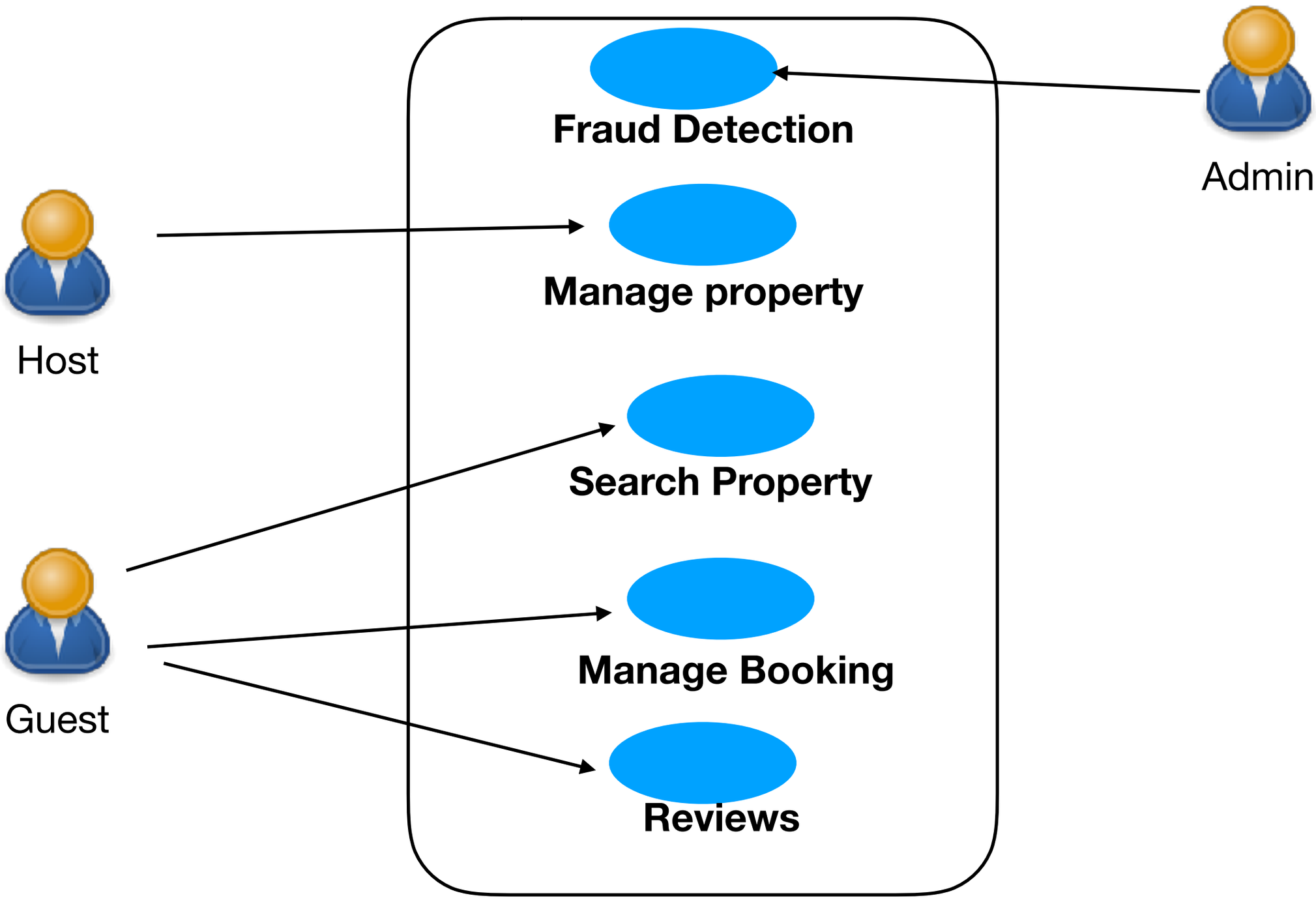
Architectural Requirements

 # Context View
 # Functional View
 # Constraints
 # Quality View *
 # Assumptions

Context View



Functional View



Constraints

- Should not have affinity to a cloud vendor. Should be able to deploy it in Azure, Aws, GCP or on prem.
- Data should reside in the country where it data was created.
- Should support SCIM for Identity management.
-

Quality Requirements

Source (who)	Stimulus (action)	Artifact (which)	Environment (context)	Response (output)	Measure (scale)
A guest	Searching for a property	On the web portal	during peak hours	Should get property listing	In < 2 seconds.
A unknown identity	requests to add a property	In the Web Portal	during normal hours	The response is to block access to the data and record the access attempts	with 100% probability of detecting the attack, 100% probability of denying access, and 50% probability of identifying the location of the
A guest	Submits a booking	On the portal	Duplicate submit	The guest is not doubly charged	With a 100% probability of detecting the duplicate request
Developer	Want to add a new payment gateway	On the portal	During maintenance	The payment gateway is added	In < 2 man days
The Database	Failed	In the Data Centre	During Operational Hours	Secondary is made Primary	In < 2 minutes

Assumptions

- During peak load there will be an maximum of 100000 active users connected to the portal
-

Architectural Design

Logical View
Deployment View
Security View

Logical View

System Decomposition
Persistence approach
Compute approach
Communication approach
cross cutting approach

Web Portal

View

Controller

Model

Listing Service

Current Booking Service

Booking History Service

Property Search Service

Cross Cutting Framework

Debug Log
Exception
Audit Log
Cache
Input Validation
...

Api

Domain

Data Layer

Api

Domain

Data Layer

Api

Domain

Data Layer

Api

Domain

Data Layer

File Storage

mongoDB

PostgreSQL

cassandra

elasticsearch

Web Portal

Create Booking

GetBooking

Booking Command Service

Booking Query Service

PostgreSQL

PostgreSQL

?

3NF
 # no duplicates
 # more joins
 # write friendly

DNF
 # duplicates
 # no joins
 # read friendly

Web Portal

Create Booking

GetBooking

Booking
Command Service

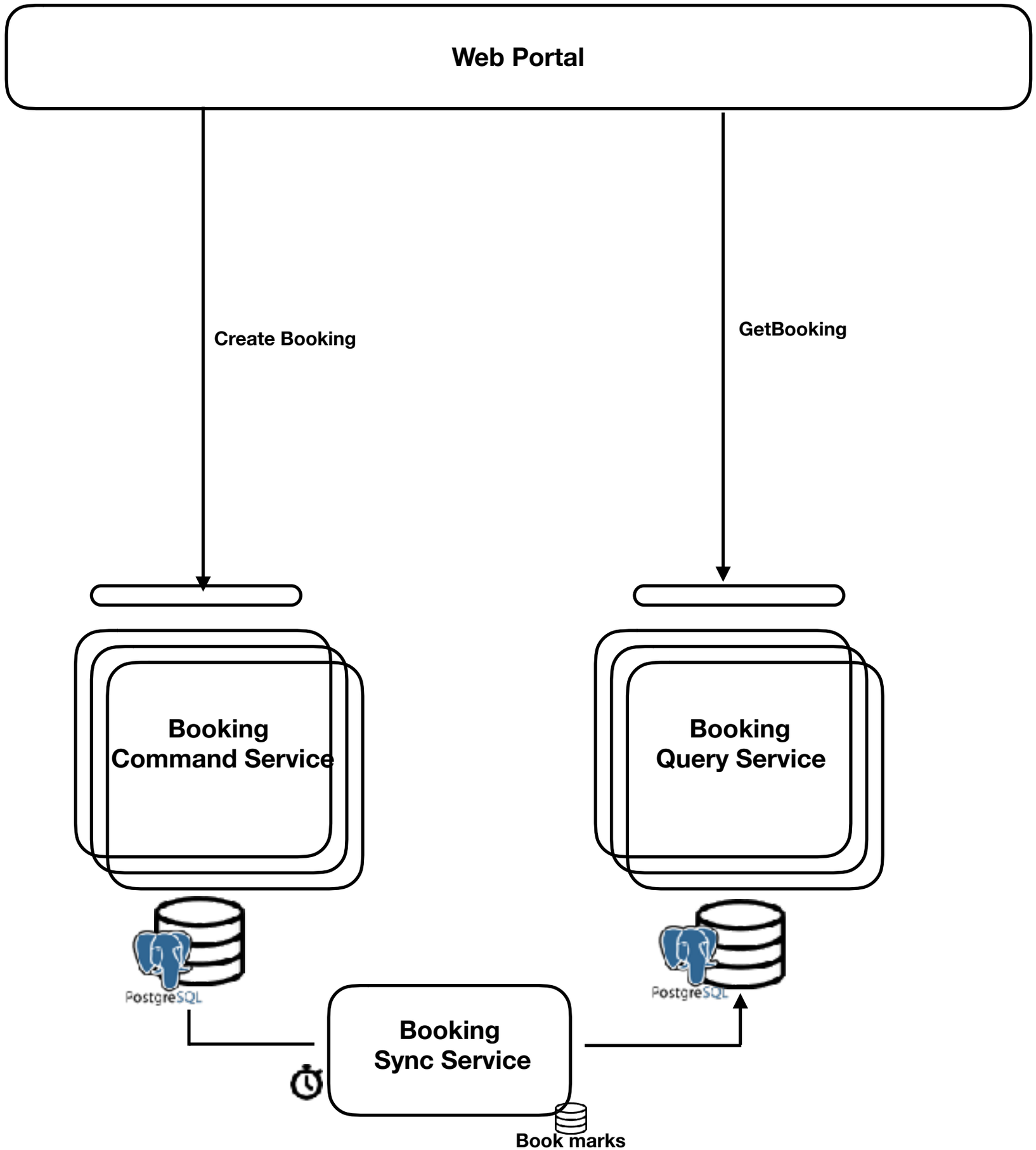
Booking
Query Service

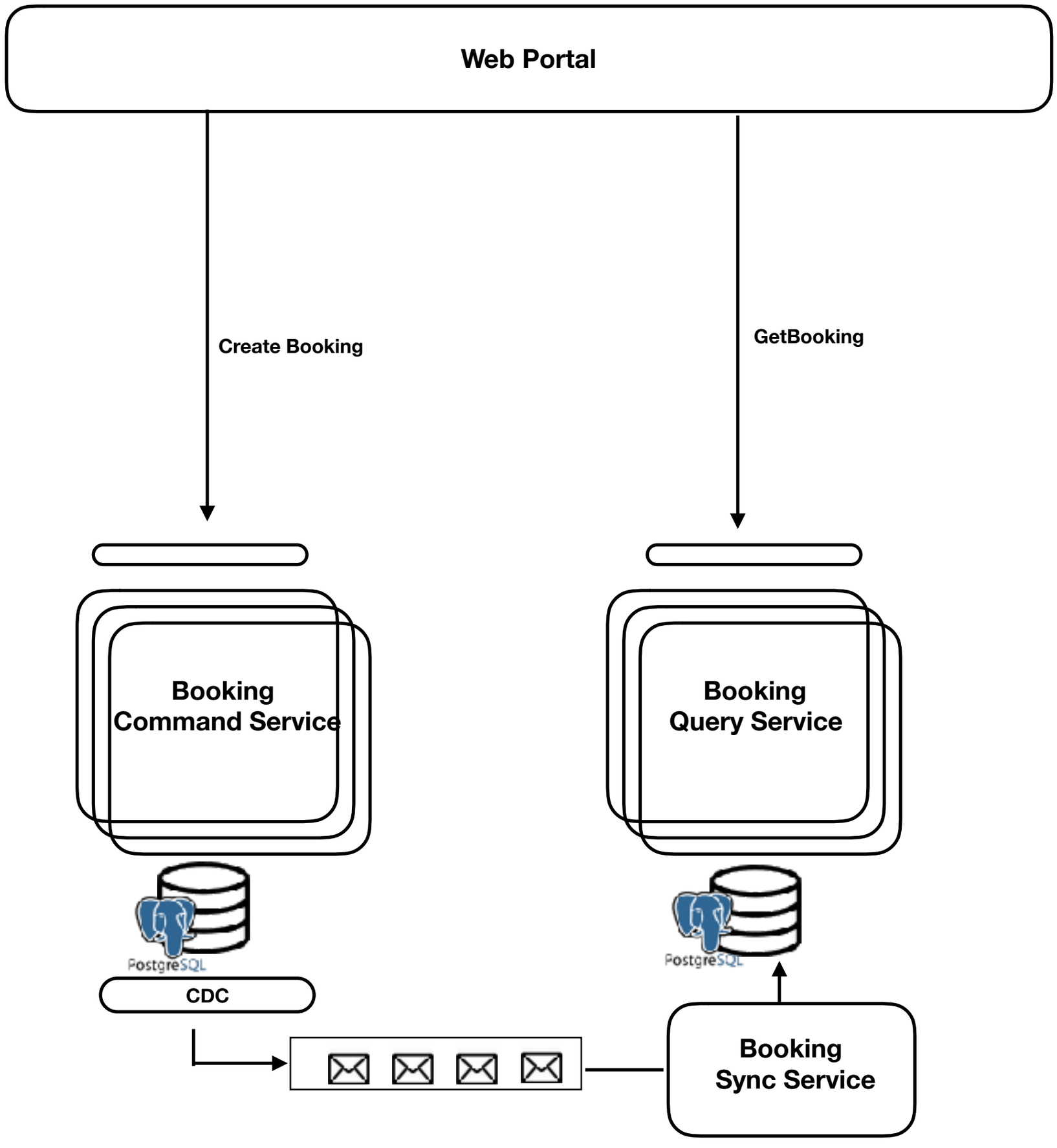
PostgreSQL

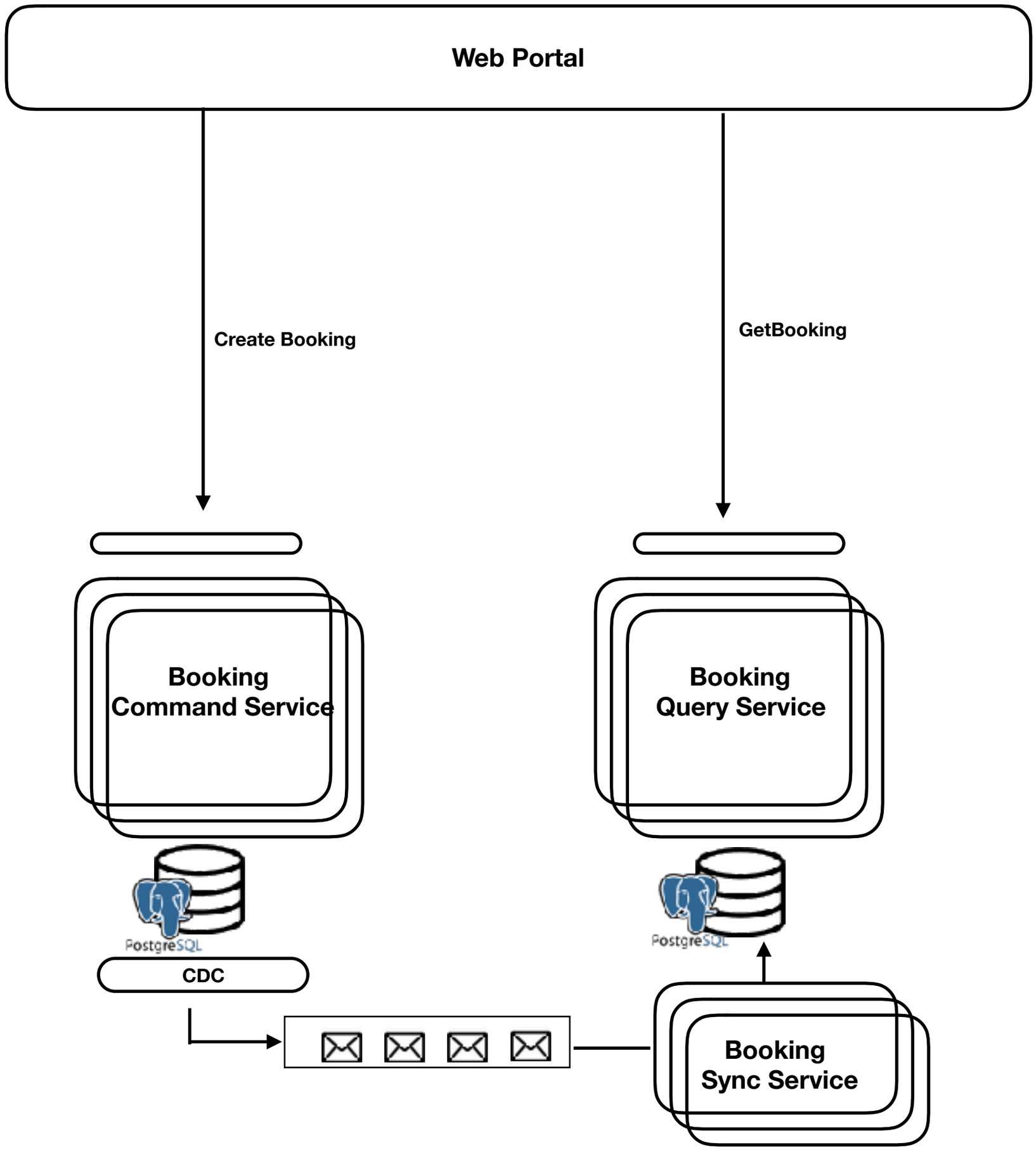
PostgreSQL

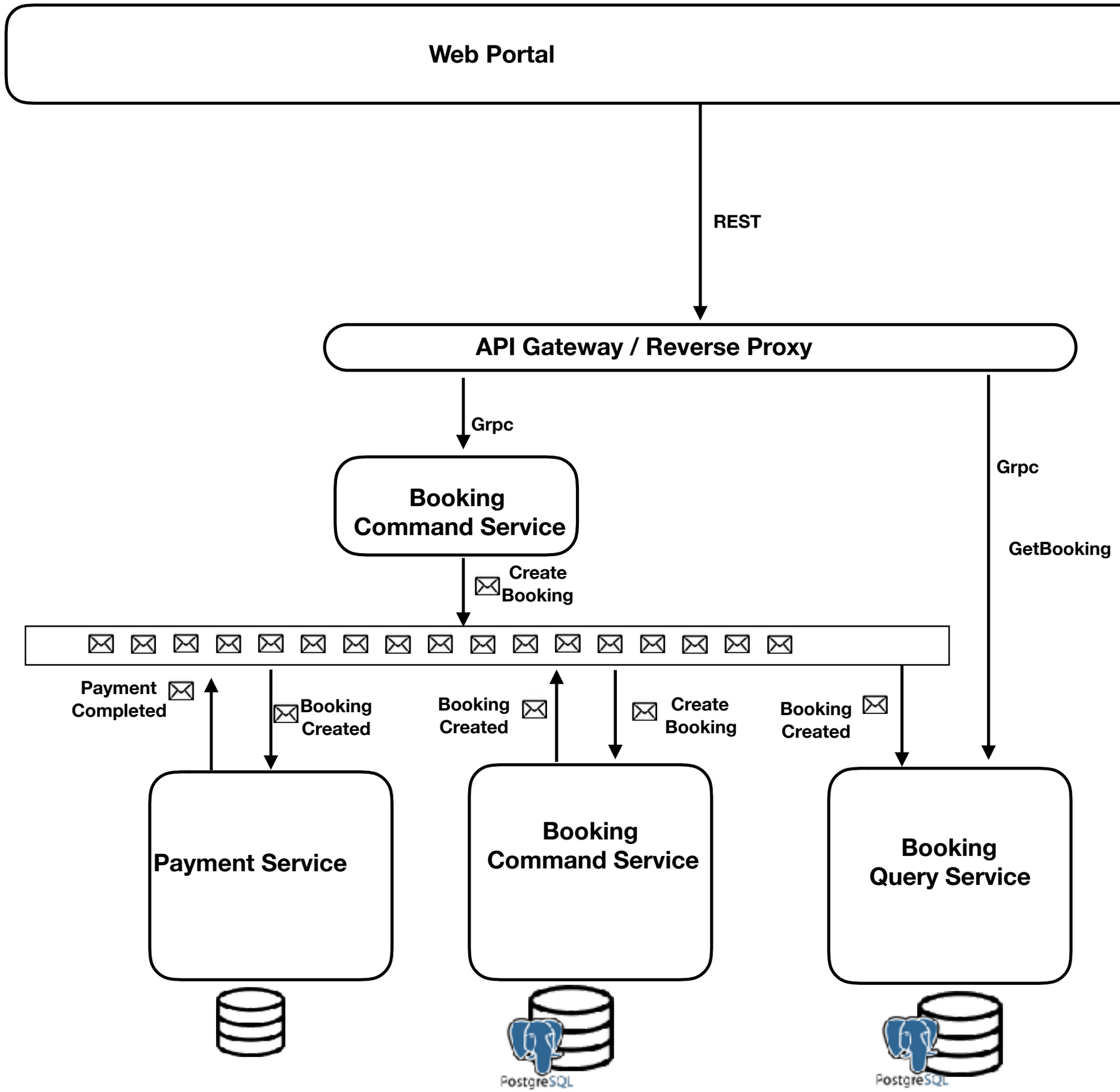
Booking
Sync Service

Book marks

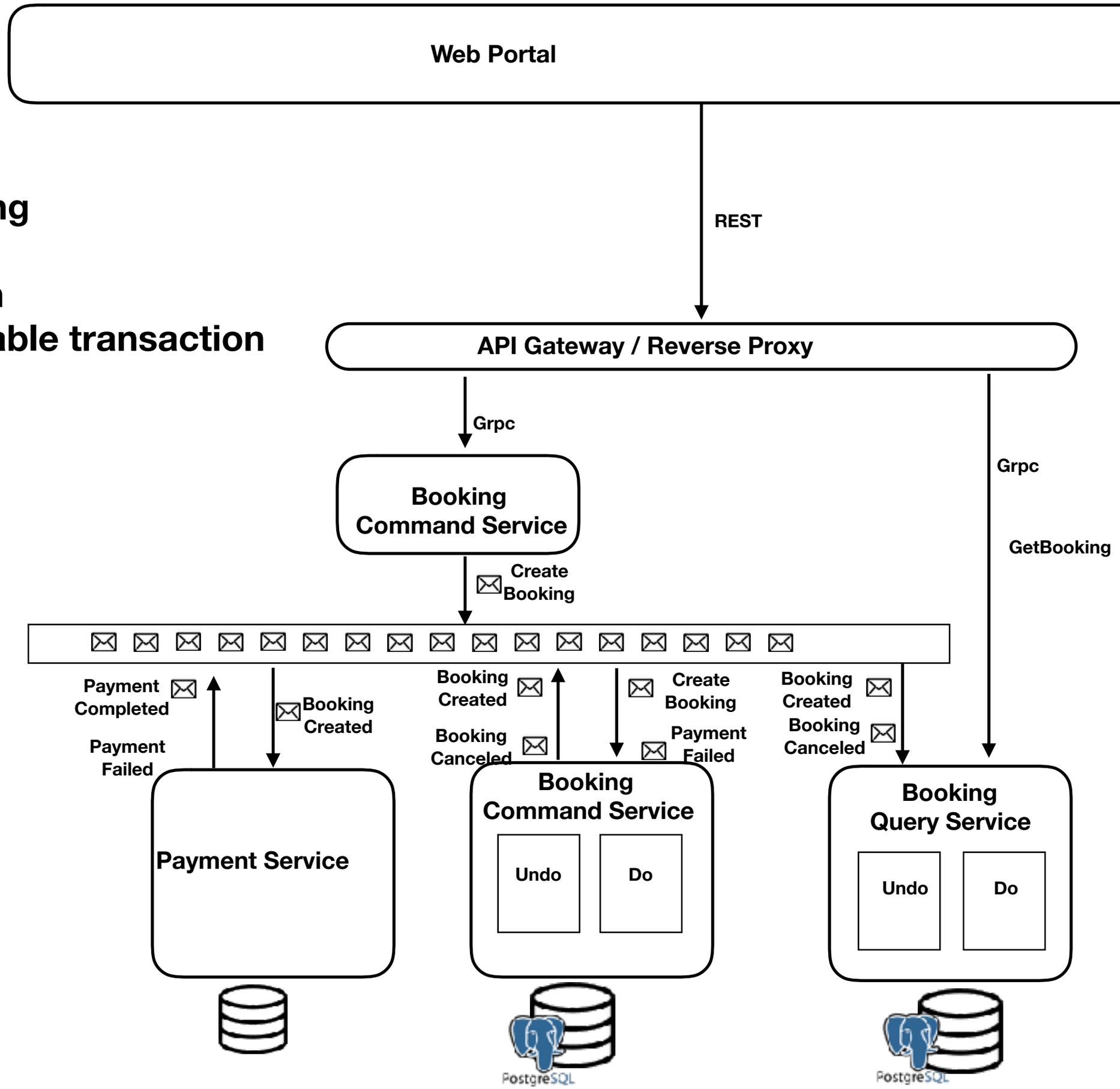

Web Portal
Create Booking
GetBooking
Booking
Command Service
Booking
Query Service
PostgreSQL
PostgreSQL
Booking
Sync Service
Book marks


Web Portal
Create Booking
GetBooking
Booking
Command Service
Booking
Query Service
PostgreSQL
PostgreSQL
CDC
Booking
Sync Service


Web Portal
Create Booking
GetBooking
Booking
Command Service
Booking
Query Service
PostgreSQL
PostgreSQL
CDC
Booking
Sync Service


Web Portal
REST
API Gateway / Reverse Proxy
Grpc
Booking
Command Service
Create
Booking
Grpc
GetBooking
Payment
Completed
Booking
Created
Booking
Created
Create
Booking
Booking
Created
Payment Service
Booking
Command Service
Booking
Query Service
PostgreSQL
PostgreSQL

EDA
Event Sourcing
CQRS
SAGA pattern
- compensatable transaction



Analytical Reports

Web Portal

REST

Read

API Gateway / Reverse Proxy

Grpc

Booking Command Service

Grpc

GetBooking

Create Booking

Data warehouse

Write

*

Payment Completed

Payment Failed

Booking Created

Booking Created

Booking Canceled

Create Booking

Payment Failed

Booking Created

Booking Canceled

Batch Analytics

Fraud Finding

...

Spark

Event Service

Object Store

Payment Service

PostgreSQL

Booking Command Service

Undo

Do

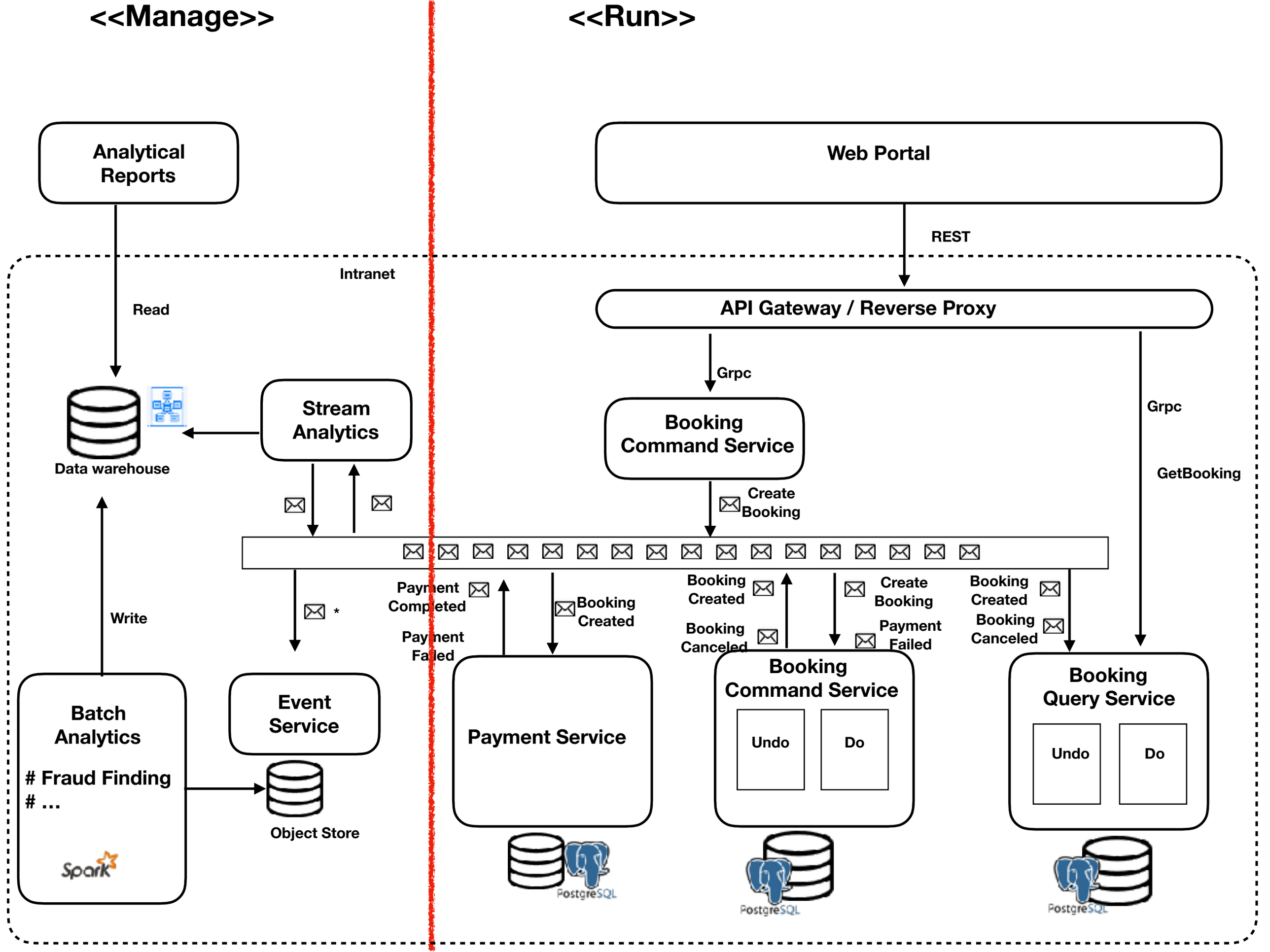
PostgreSQL

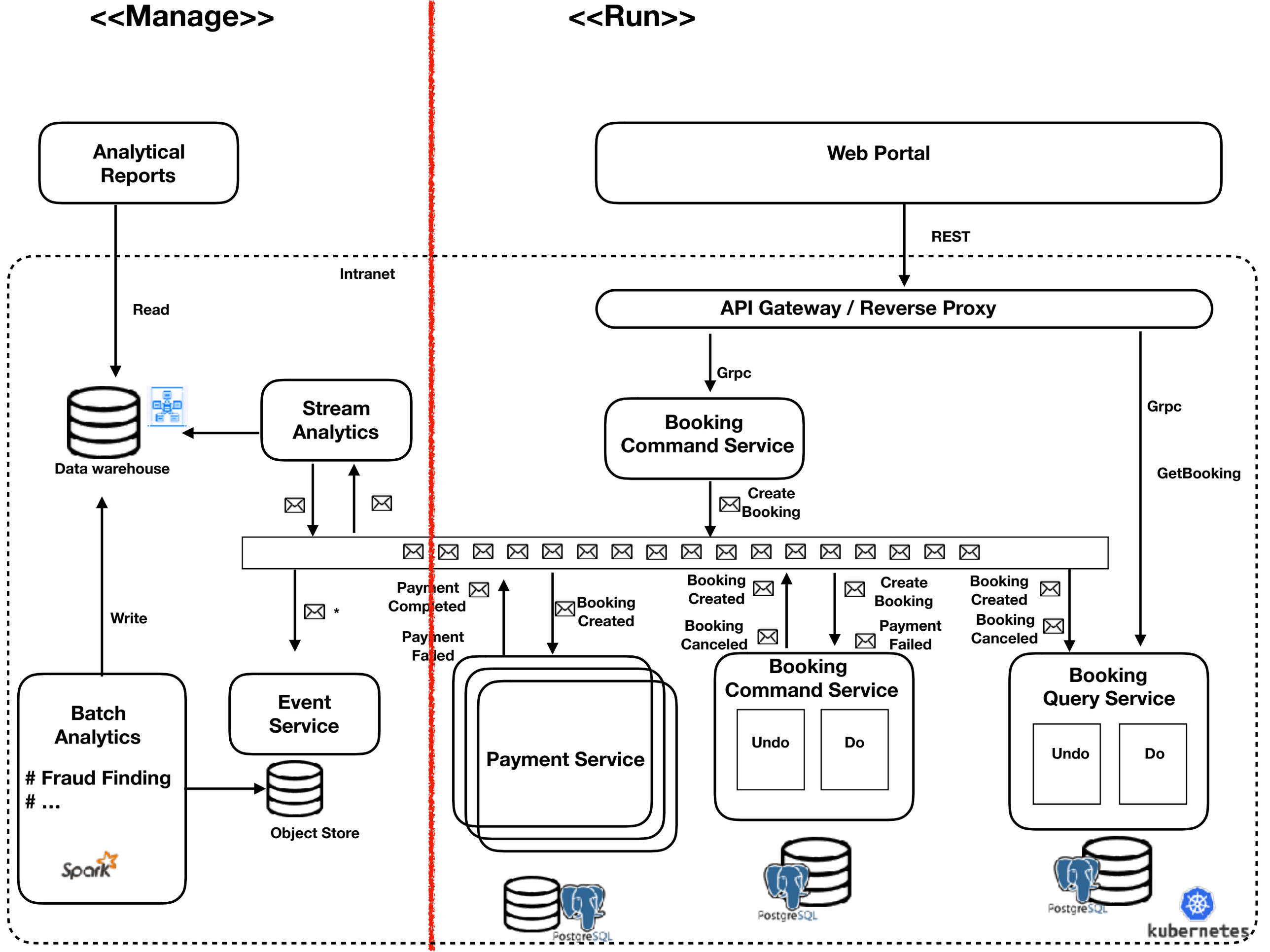
Booking Query Service

Undo

Do

PostgreSQL


<<Manage>>
<<Run>>
Analytical Reports
Web Portal
REST
Intranet
Read
API Gateway / Reverse Proxy
Grpc
Stream Analytics
Booking Command Service
Grpc
GetBooking
Data warehouse
Create Booking
Write
Payment Completed
Booking Created
Booking Created
Create Booking
Booking Created
Payment Failed
Booking Canceled
Booking Canceled
Payment Failed
*
Batch Analytics
Fraud Finding
...
Spark
Event Service
Object Store
Payment Service
PostgreSQL
Booking Command Service
Undo
Do
PostgreSQL
Booking Query Service
Undo
Do
PostgreSQL

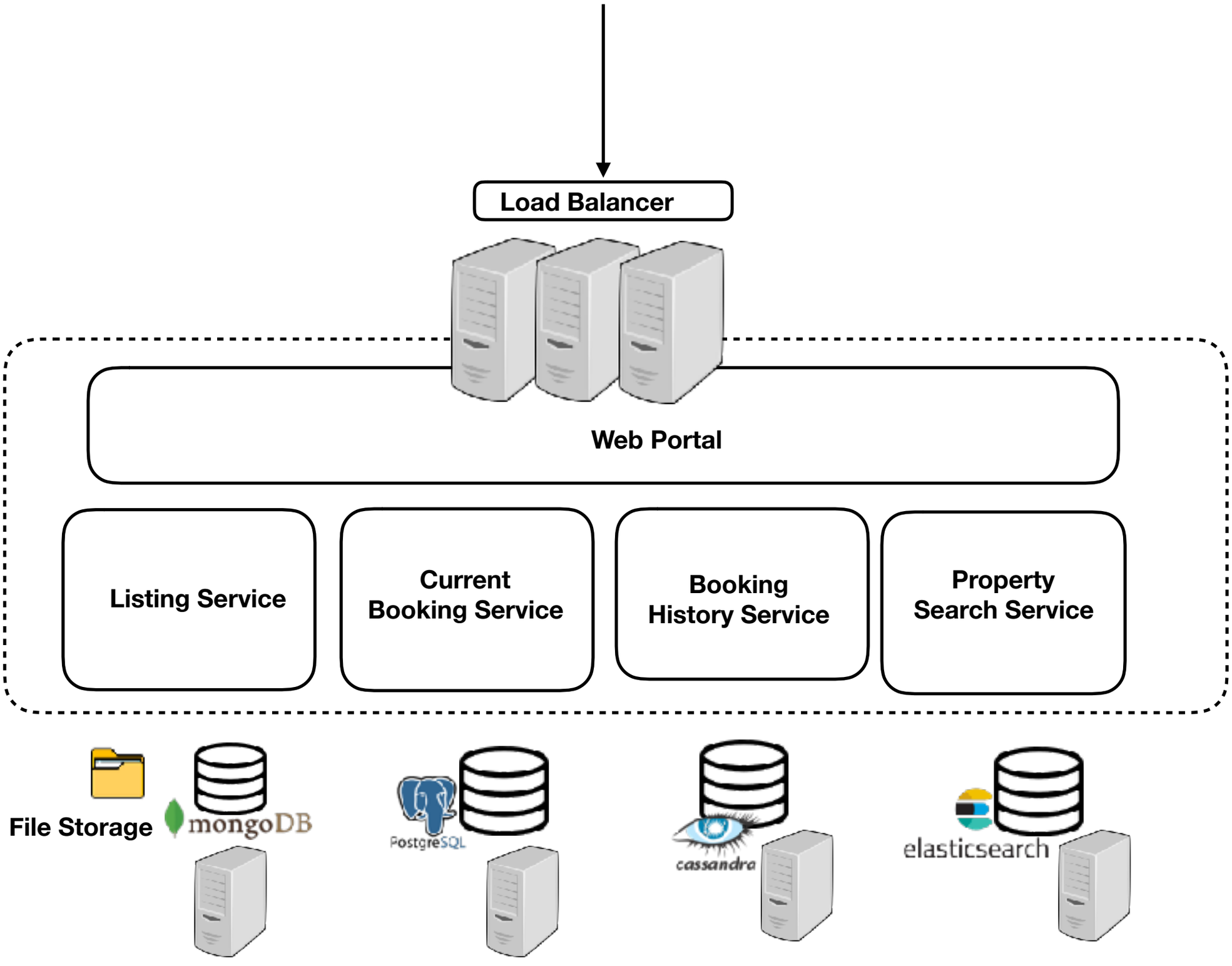

<<Manage>>
<<Run>>
Analytical Reports
Web Portal
REST
Intranet
Read
API Gateway / Reverse Proxy
Grpc
Stream Analytics
Booking Command Service
Grpc
GetBooking
Data warehouse
Create Booking
Write
Payment Completed
Booking Created
Booking Created
Create Booking
Booking Created
Payment Failed
Booking Canceled
Booking Canceled
Payment Failed
*
Batch Analytics
Fraud Finding
...
Spark
Event Service
Object Store
Payment Service
Booking Command Service
Undo
Do
Booking Query Service
Undo
Do
PostgreSQL
PostgreSQL
PostgreSQL
kubernetes

	Polling	Trigger	CDC	EDA
Scale	No	No	Yes	Yes
Causes extra load on db	Yes	Yes	No	No
Vendor locking	No	Yes	Yes	No
Low coupling	Yes	Yes	Yes	Yes

Deployment View

#

1


Load Balancer
Web Portal
Listing Service
Current
Booking Service
Booking
History Service
Property
Search Service
File Storage
mongoDB
PostgreSQL
cassandra
elasticsearch

2

Vnet 1

Firewall

External Load Balancer

Subnet 1

Static webserver
Apache/ nginx/...

Web Portal

Internal Load Balancer

Vnet 2

Subnet 2

Dynamic webserver
Tomcat/ JBoss/ Flask/
IIS/ ...

Listing Service

Current
Booking Service

Booking
History Service

Property
Search Service

Subnet 3

File Storage

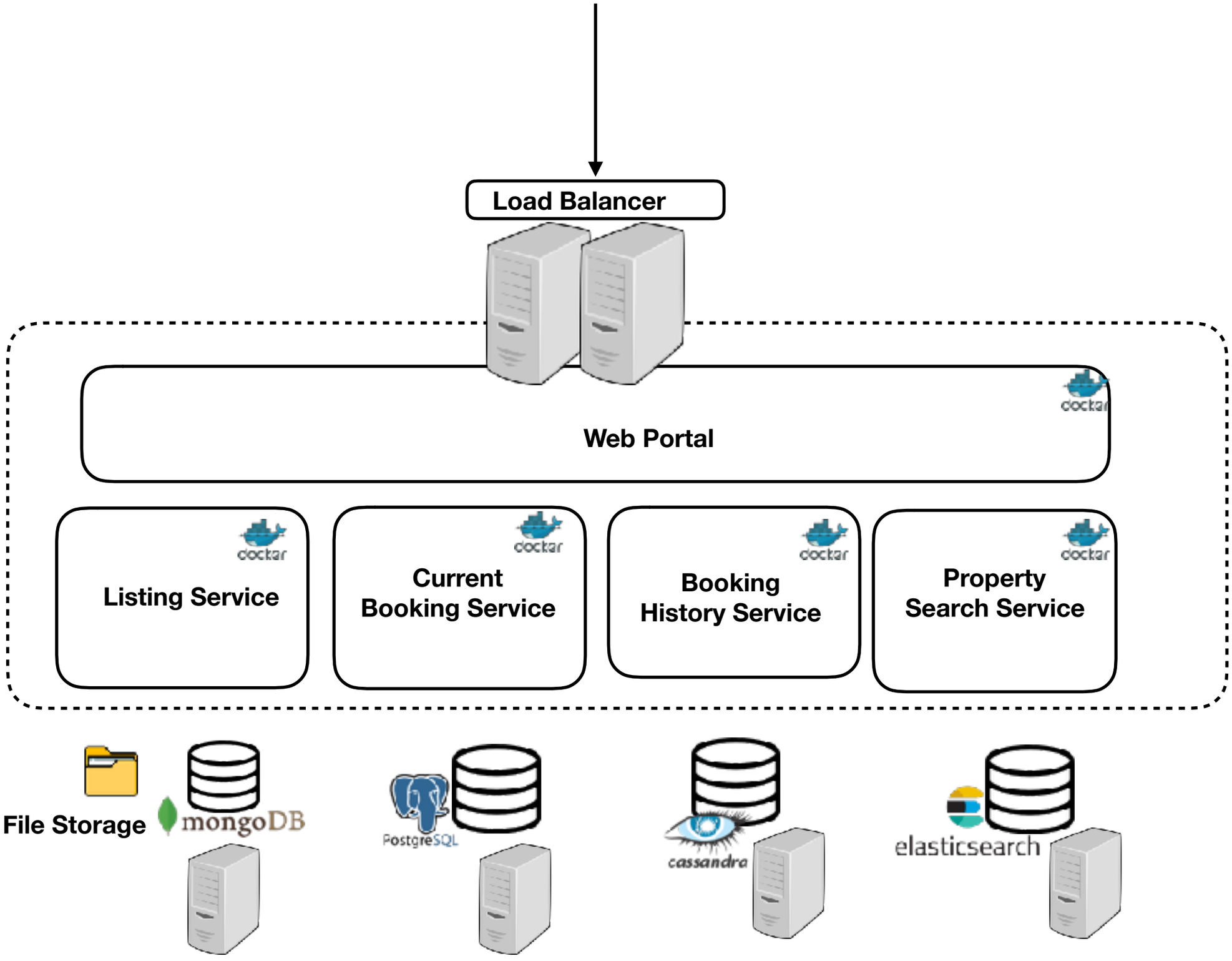
mongoDB

PostgreSQL

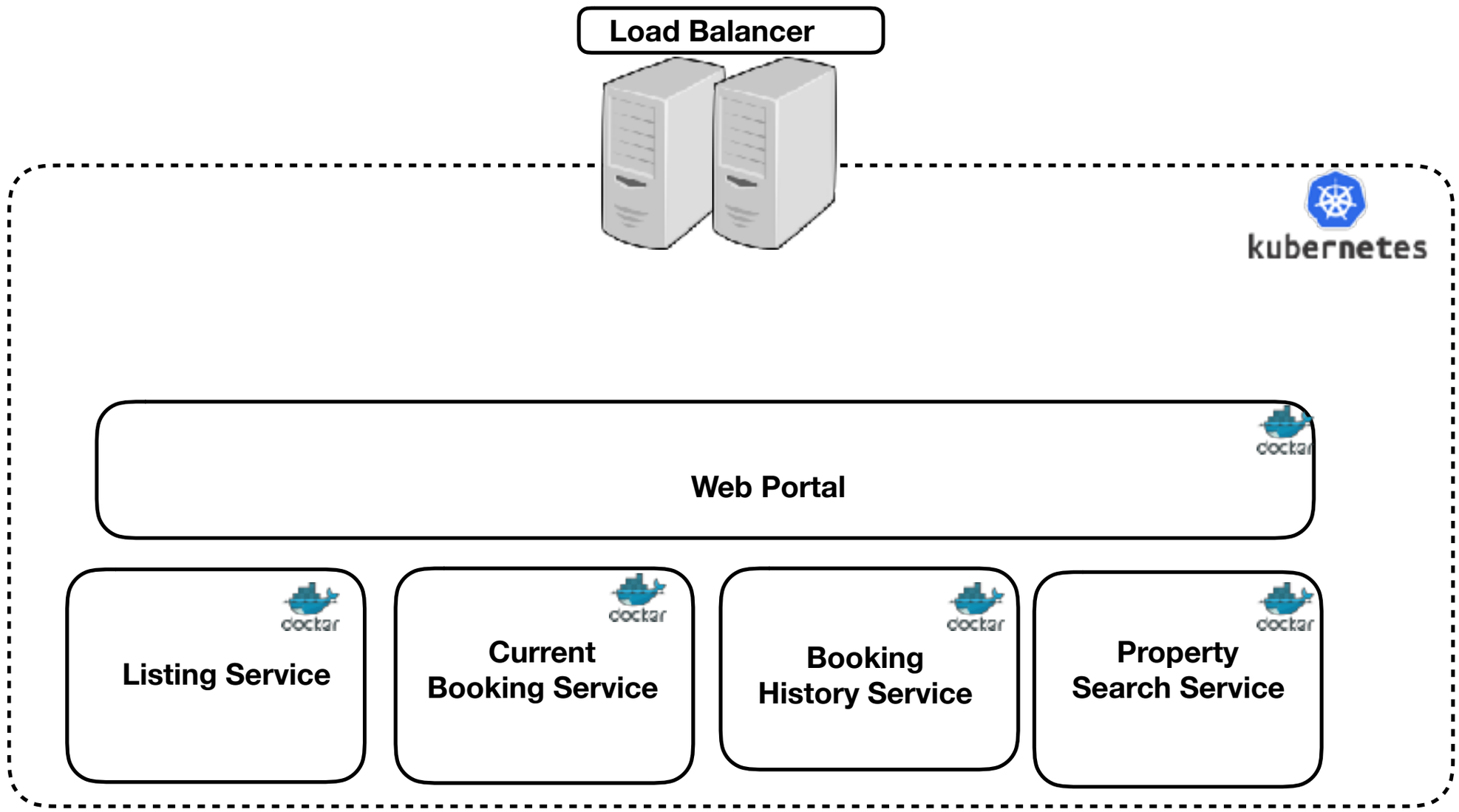
cassandra

elasticsearch

1


Load Balancer
Web Portal
Listing Service
Current
Booking Service
Booking
History Service
Property
Search Service
File Storage
mongoDB
PostgreSQL
cassandra
elasticsearch

1


Load Balancer
kubernetes
Web Portal
Listing Service
Current
Booking Service
Booking
History Service
Property
Search Service
File Storage
mongoDB
PostgreSQL
cassandra
elasticsearch

Architectural Justification

 # Evaluation (ATAM | ARID | SAAM| ….)

1. identify all Architectural approaches

- Ad1 : Use Cloud file Storage for image and videos
- Ad2 : use document db for listing
- Ad3 : use RDBMS for booking
- Ad4 : move booking history to a wide column data store and use rdbms only for current booking
- Ad5: use elastic search for property search
- Ad6 : decompose system by Domain (Poperty, booking, …)
- Ad7 : decompose portal using interaction pattern
- Ad 8: decompose domain services using layered pattern
- Ad 9: use Managed K8s to deploy Domain services and portal
- Ad 10: use managed services for database deployments

2. identify all quality requirements

qs1. A guest Searching for a property On the web portal during peak hours Should get property listing In < 2 seconds.

qs2. A unknown identity requests to add a property In the Web Portal during normal hours The response is to block access to the data and record the access attempts with 100% probability of detecting the attack, 100% probability of denying access, and 50% probability of identifying the location of the individual.

qs3. A guest Submits a booking On the portal Duplicate submit The guest is not doubly charged With a 100% probability of detecting the duplicate request.

qs4. Developer Want to add a new payment gateway On the portal During maintenance The payment gateway is added In < 2 man days.

qs5, The Database Failed In the Data Centre During Operational Hours Secondary is made Primary In < 2 minutes.

3. analyse Scenario -> Approach

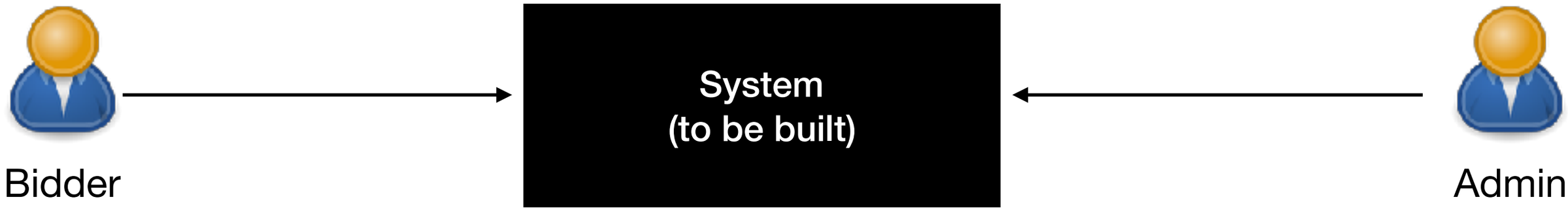
Quality Scenario	Approach	Risk ?	
Qs1	Ad1, Ad2, Ad5	Low	Sync design TBD
Qs2	?	High	TBD
Qs3	Ad3	Med	Booking service should be designed for idemptoency
QS4	Ad6, Ad8	Med	Design for pluggablility
QS5	Ad9, Ad10	Low	

Bidding Case Study

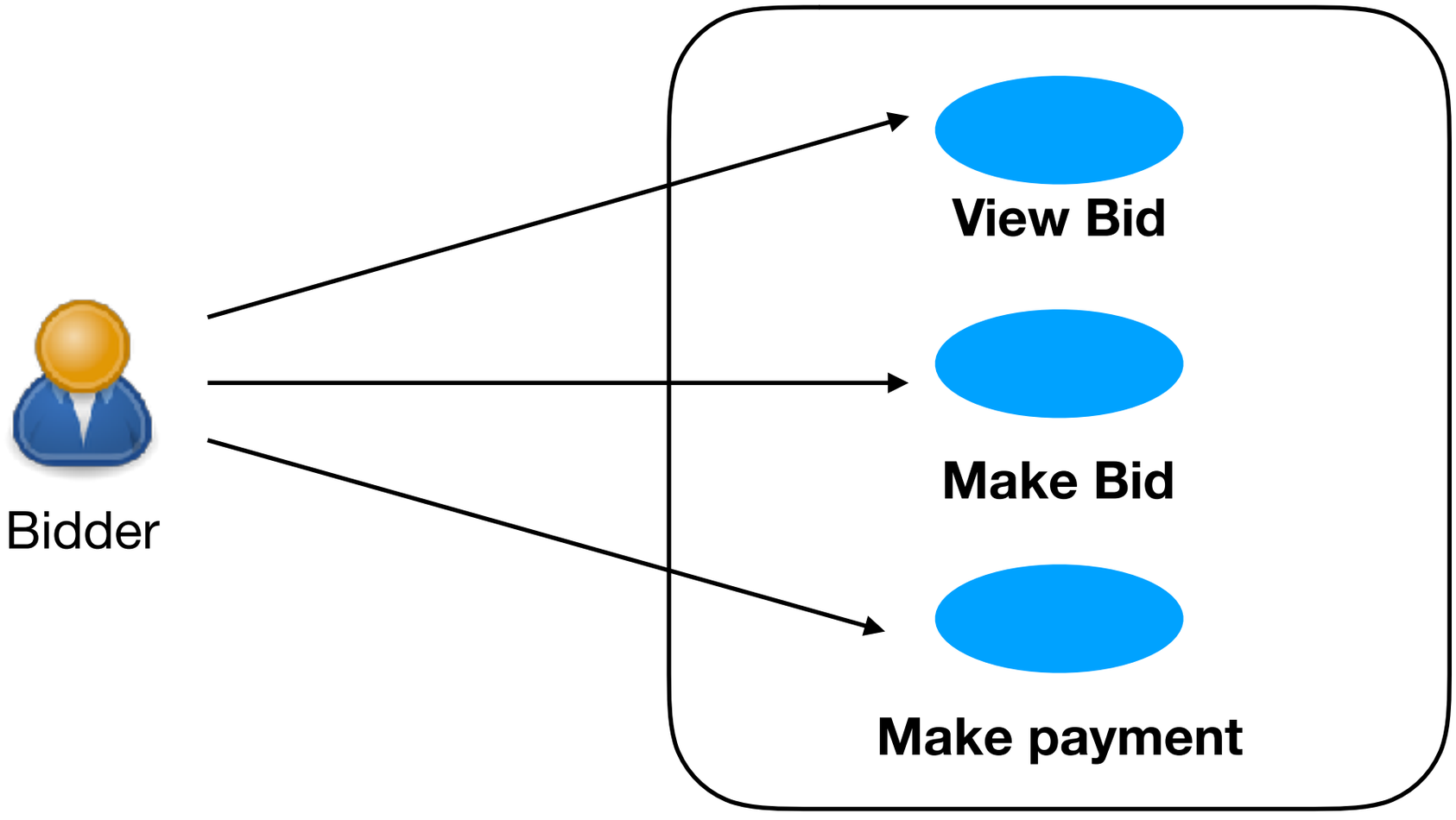
Architectural Requirements

 # Context View
 # Functional View
 # Constraints (removed)
 # Quality View *
 # Assumptions

Context View



Functional view



Quality View

Quality	Source (who)	Stimulus (action)	Artifact (which)	Environment (context)	Response (output)	Measure (scale)
Performance	As a Bidder	I should be able to see the Bids placed by other bidders	In the portal	When there are 100,000 bidders bidding	The Highest bid is shown to the bidder	In < 1 sec

Assumptions

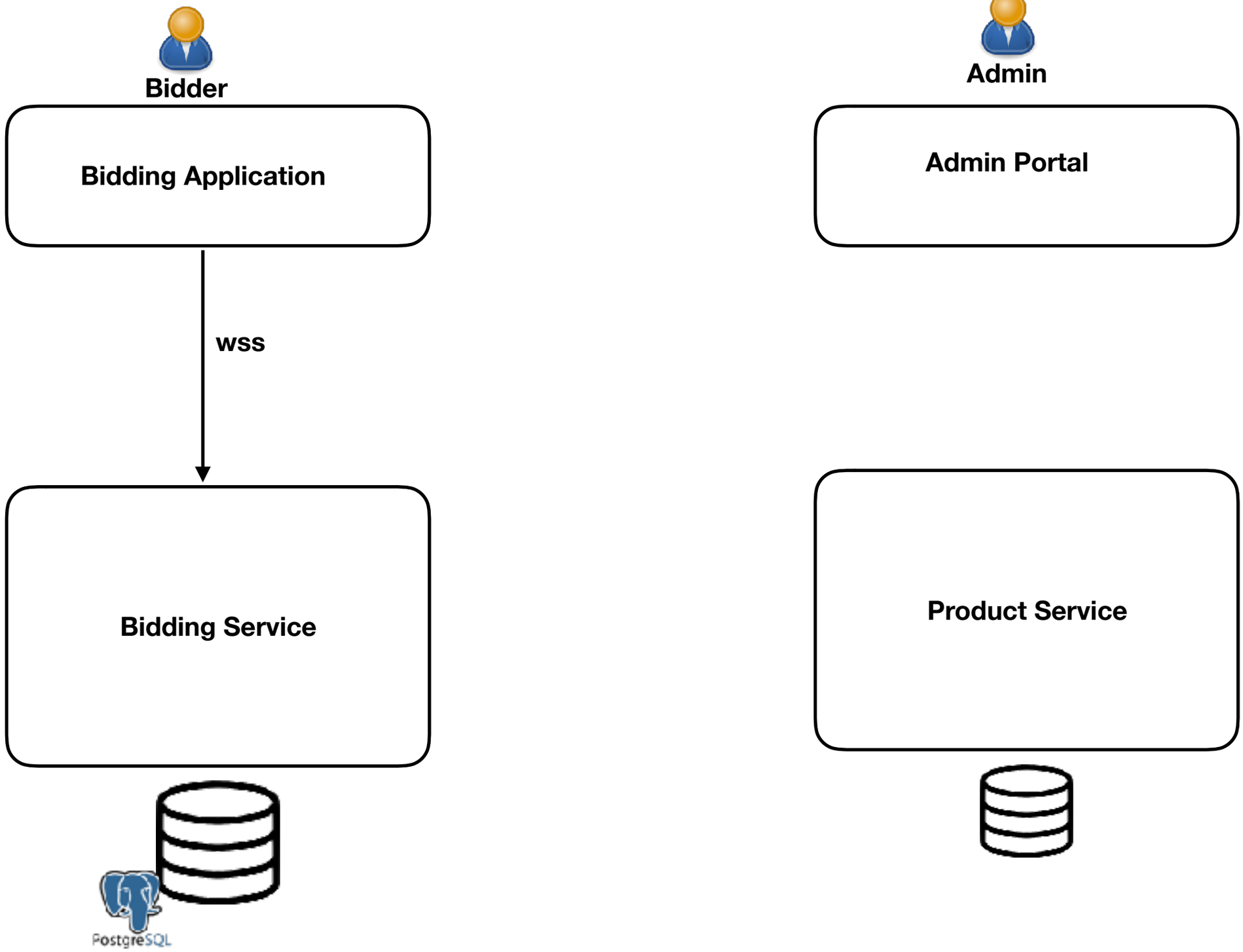
- User will only use Mobile Application. There is no web interface for the application.
- maximum bidding during peak load will be less then 1000 bids.
-

Architectural Design

 # Logical View
 # Deployment View
 # Security View (removed)

Logical View

System Decomposition
Persistence approach
Compute approach
Communication approach
cross cutting approach


Bidder
Admin
Bidding Application
Admin Portal
wss
Bidding Service
Product Service
PostgreSQL

Deployment View

#

Bidder 1

Bidding Application

Bidder 2

Bidding Application

Wss

Wss

Load Balancer

Bidder 1
Bidder 5

Bidding Service

Bidder 2
Bidder 3
Bidder 4

Bidding Service

Bidder 6
Bidder 7
Bidder 8

Bidding Service

PostgreSQL

- Pull
  - Short polling (time interval ?)
  - Long polling
- Push
  - SSE
  - Web socket

Bidder 1

Bidding Application

Bidder 2

Bidding Application

Wss

Wss

Load Balancer

Bidder 1
Bidder 5

Bidding Service

Bidder 2
Bidder 3
Bidder 4

Bidding Service

Bidder 6
Bidder 7
Bidder 8

Bidding Service

PostgreSQL

Bidder 1

Bidding Application

Bidder 2

Bidding Application

Wss

Wss

Load Balancer

Bidder 1
Bidder 5

Bidding Service

Bidder 2
Bidder 3
Bidder 4

Bidding Service

Bidder 6
Bidder 7
Bidder 8

Bidding Service

PostgreSQL

redis

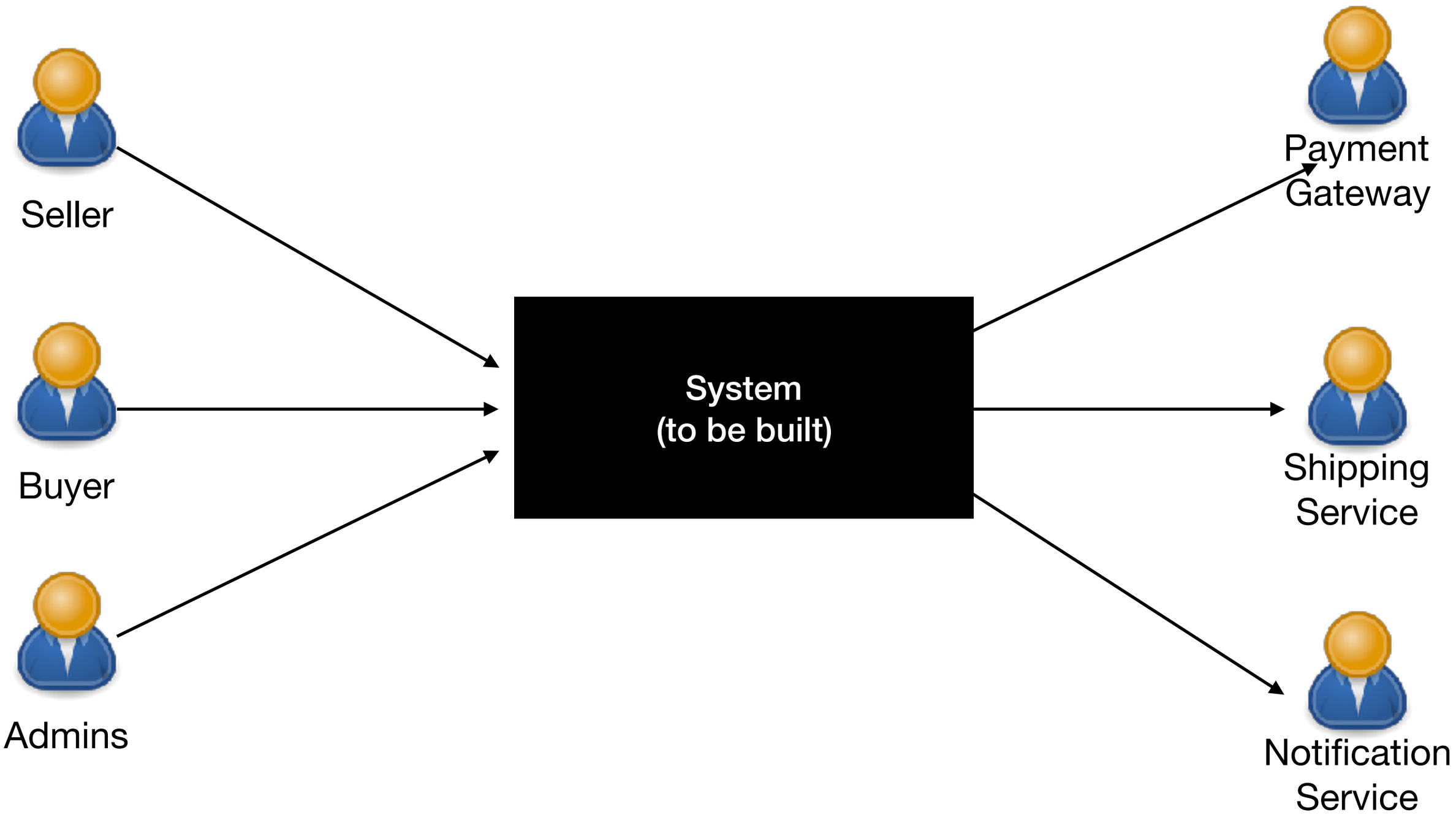
Bid Id -> Bid Details

Ecommerce Case Study

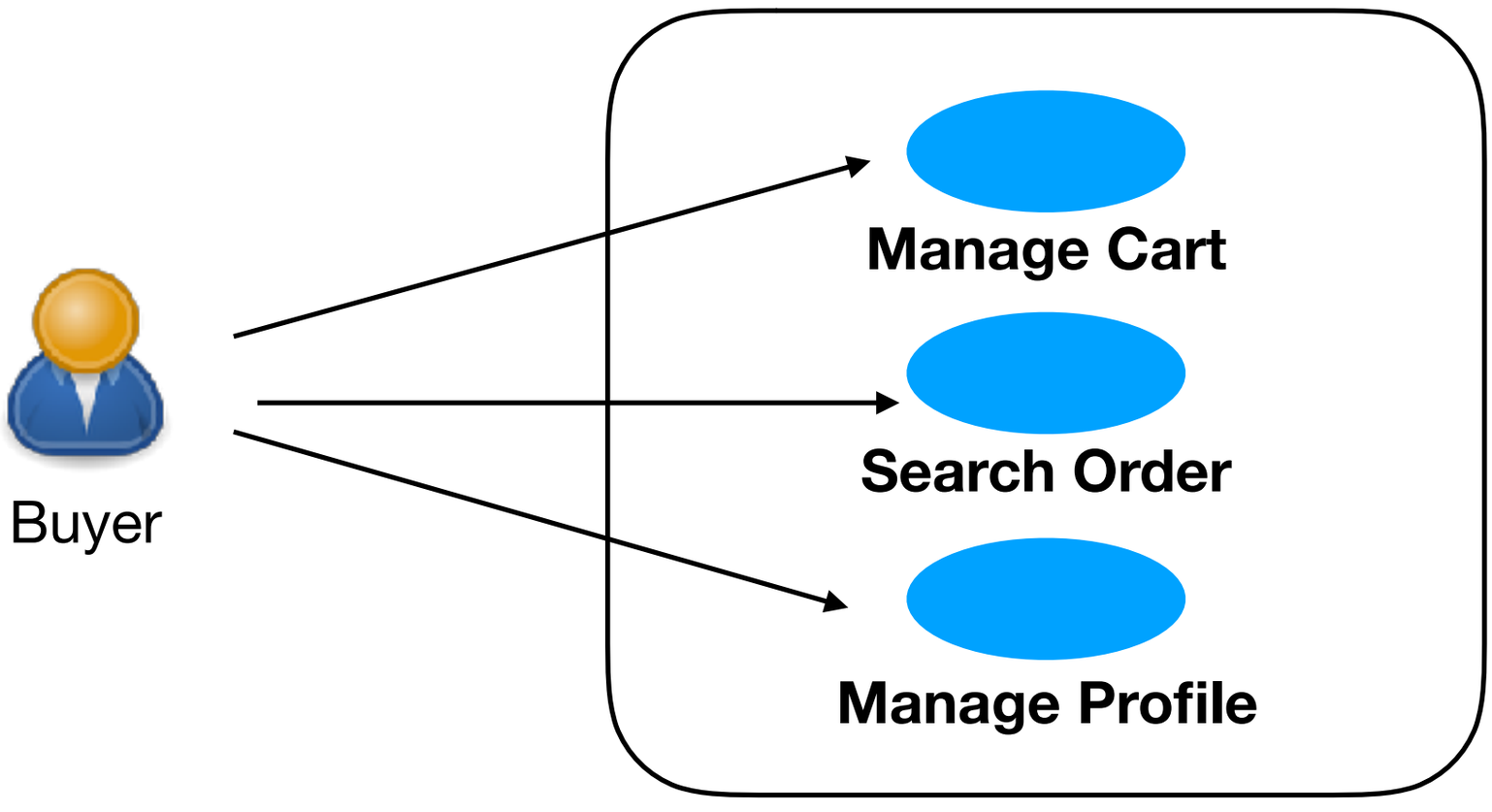
Architectural Requirements

 # Context View
 # Functional View
 # Constraints (removed)
 # Quality View
 # Assumptions (removed)

Context View



Functional view



Quality View

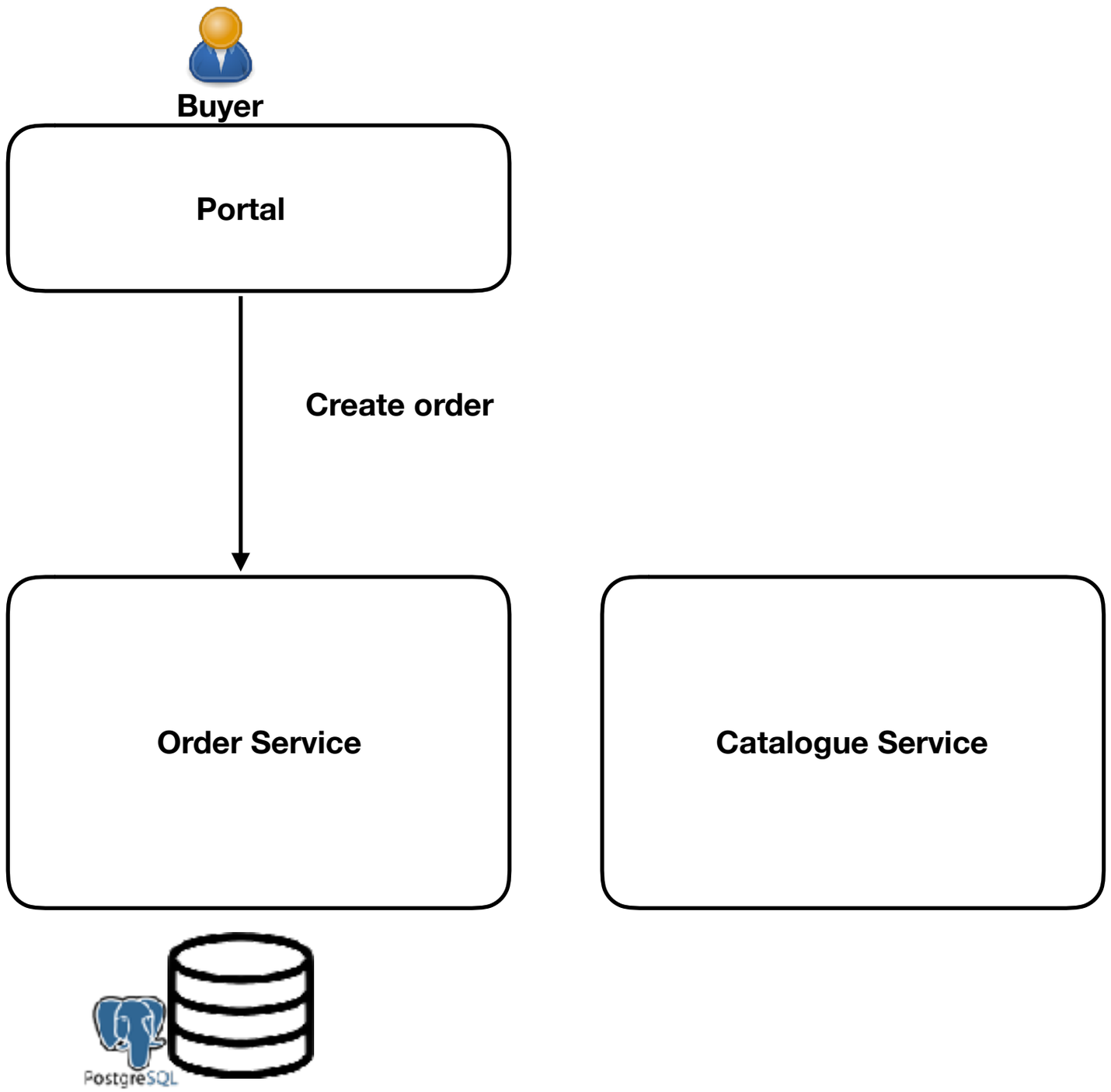
Quality	Source (who)	Stimulus (action)	Artifact (which)	Environment (context)	Response (output)	Measure (scale)
Scale	As a Buyer	I should be able to place an order	In the portal	During peak load.	The order is placed.	Should support 1 million users

Architectural Design

 # Logical View
 # Deployment View
 # Security View (removed)

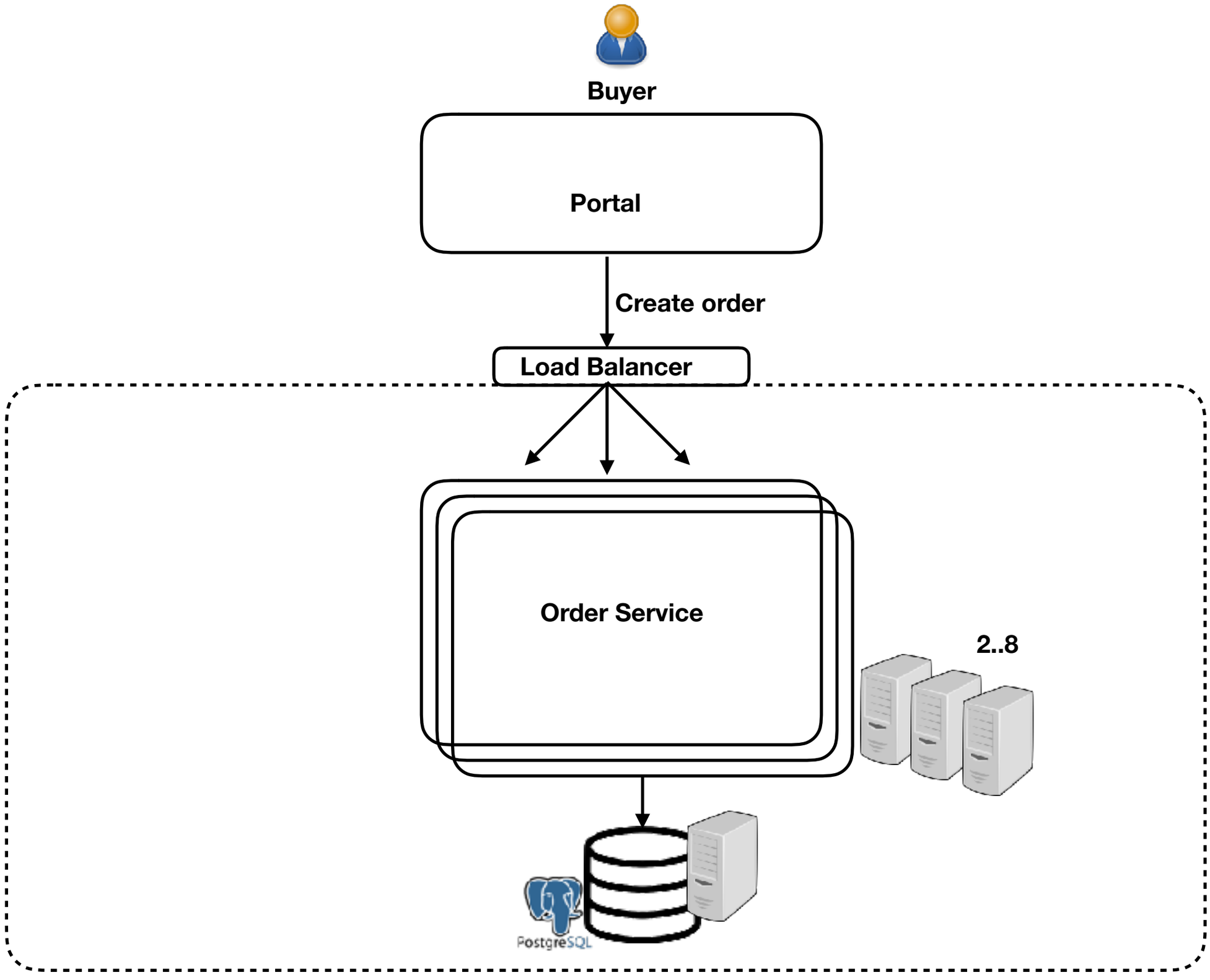
Logical View

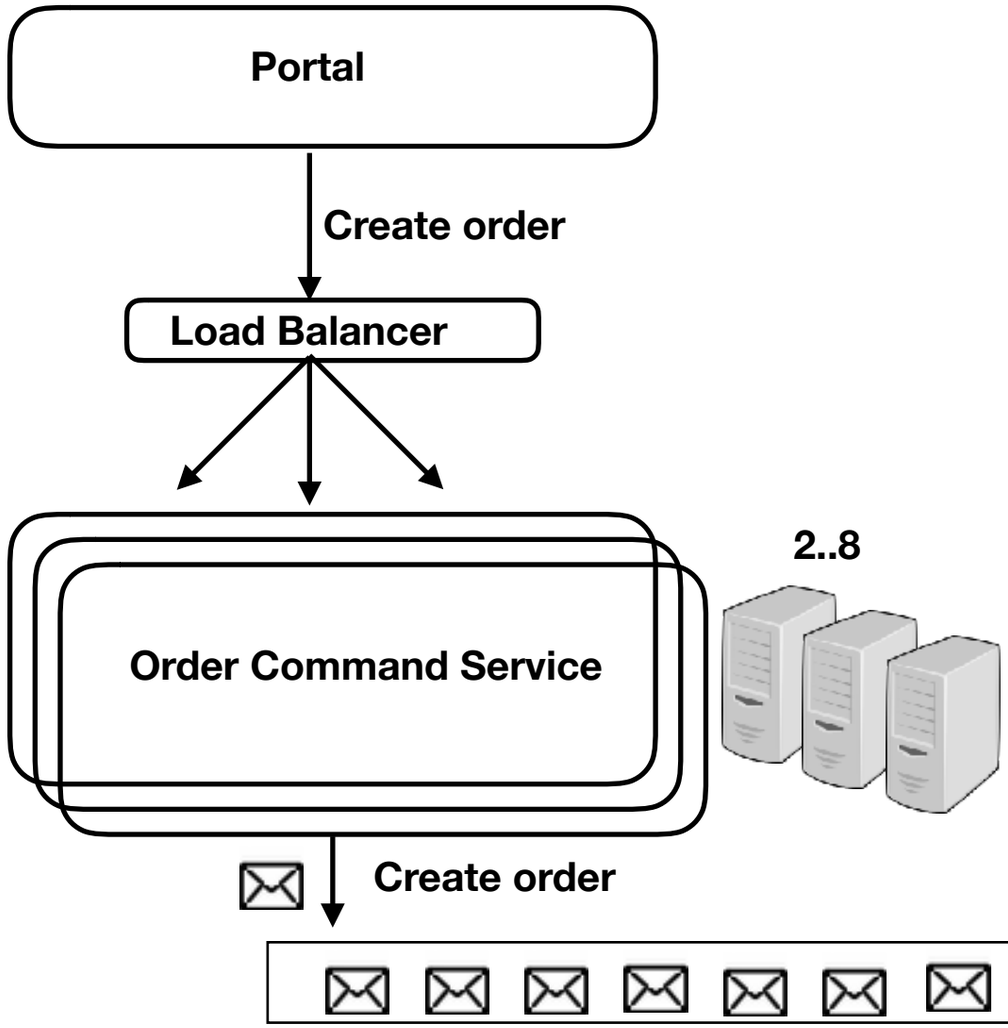
System Decomposition
Persistence approach
Compute approach
Communication approach
cross cutting approach

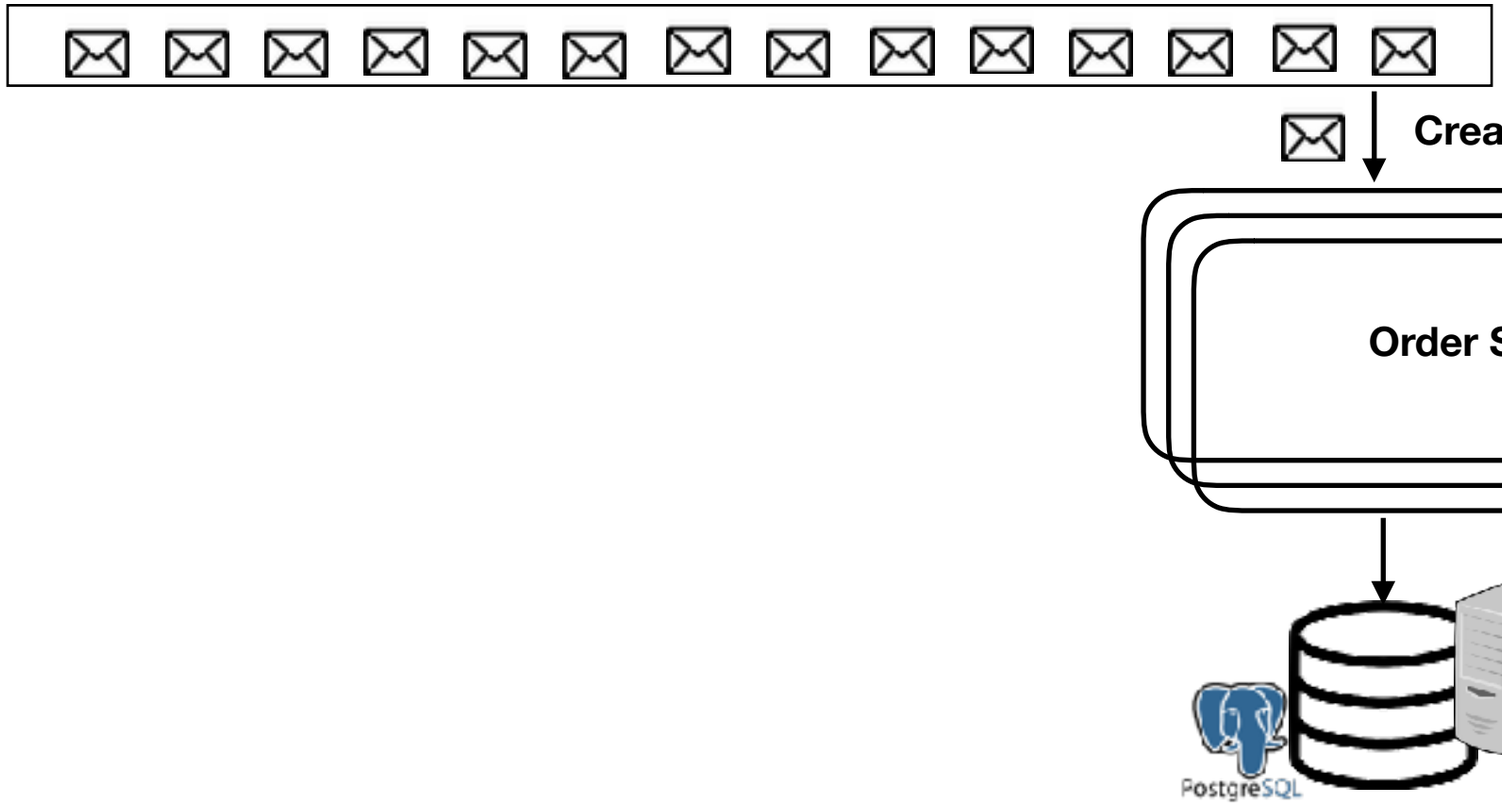

Buyer
Portal
Create order
Order Service
Catalogue Service
PostgreSQL

Deployment View

#


Buyer
Portal
Create order
Load Balancer
Order Service
2..8
PostgreSQL


Portal
Create order
Load Balancer
Order Command Service
2..8
Create order


Create order
Order Service
PostgreSQL

DropBox

Logical View

Boundary

View

View Model

Model

Meta data

Control

Sync Service

Domain

? God

Chunk Service

Conflict Mgmt Service ?

NetworkService

Upload Download Service

Local Watcher

Cloud Watcher

Metadata Service

Boundary

Share Service
Or
ACL Service

Rdbms

Notification Service

?

File Service

Object
Storage

Ledger Service

Wide Column db
(column-family)

Million of rows
Query on multiple columns

Document
Store

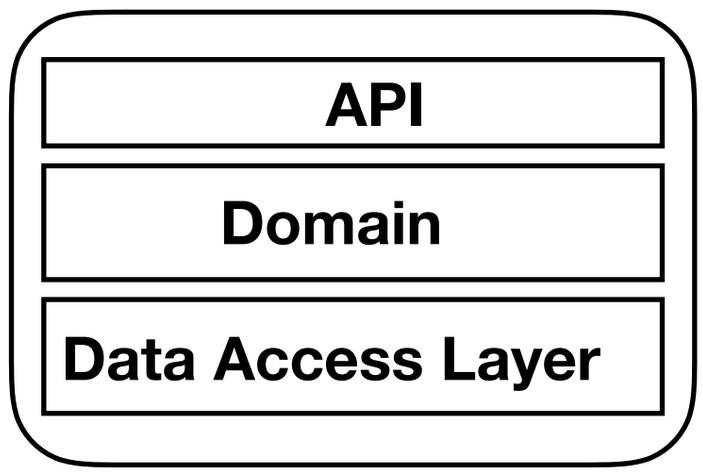
Quota/ metering

Metadata Service

Rdbms

User Management

Rdbms



Upload Download Service

Chunk Service

Cloud Watcher

Quota/ metering ?

Pull during startup (catch up)
Push while running

Grpc

File Authorization Service <<API>>

File Service <<API>>

Metadata Service <<API>>

User Management <<API>>

Notification Service

Rdbms

Object Storage

Rdbms

Rdbms

Grpc

Should we ?

Audit Service

Wide column

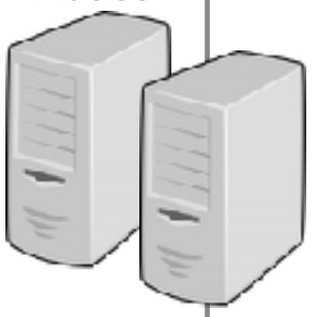
Firewall (TCP)

API Gateway /API Management / WAF (Http)

Load Balancer

Cluster

Nodes



Kubernetes ingress/ Istio Ingress / nginx reverse proxy

Notification Service

File Authorization Service <<API>>

Metadata Service <<API>>

User Management <<API>>

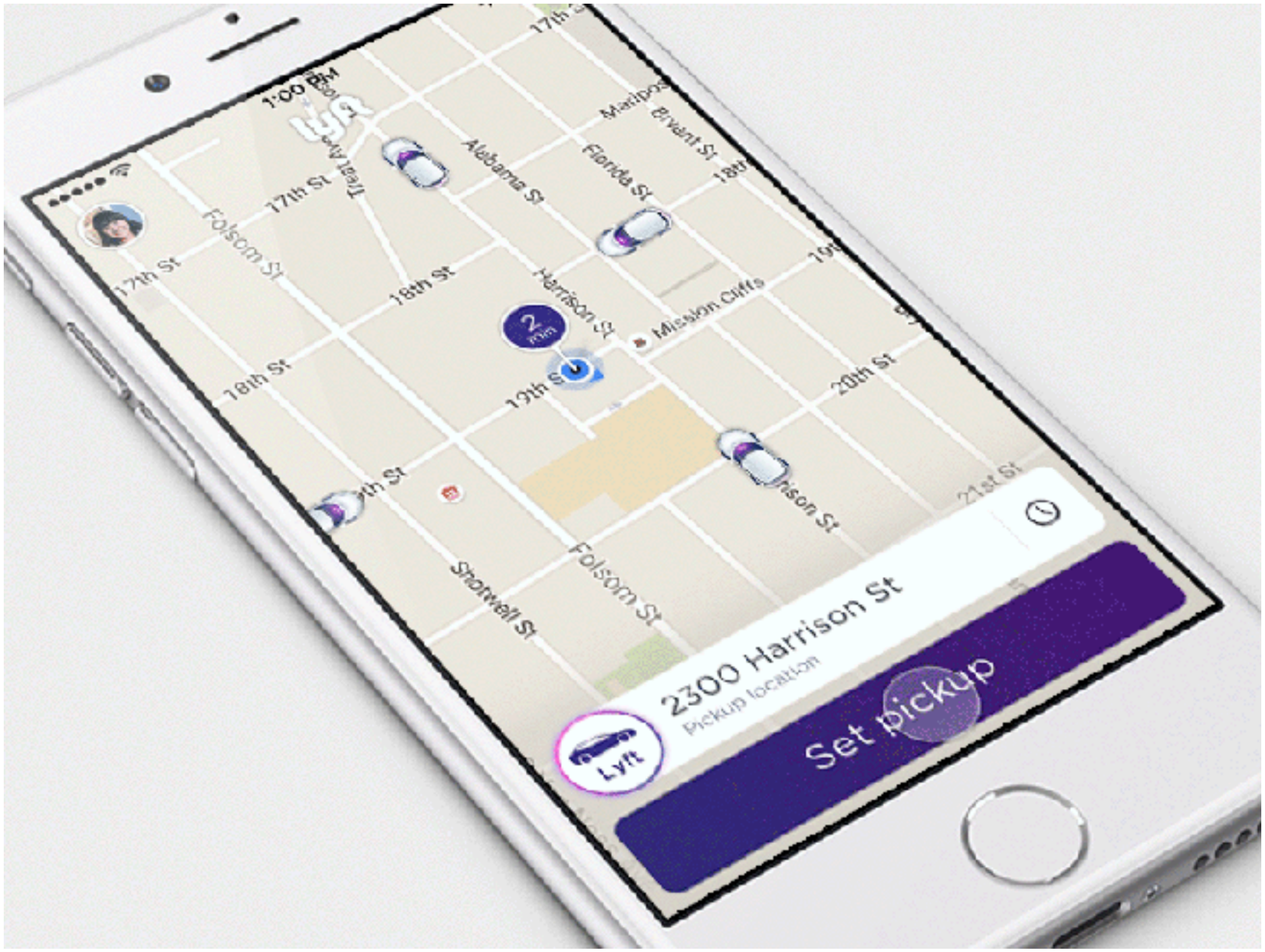
Object Storage

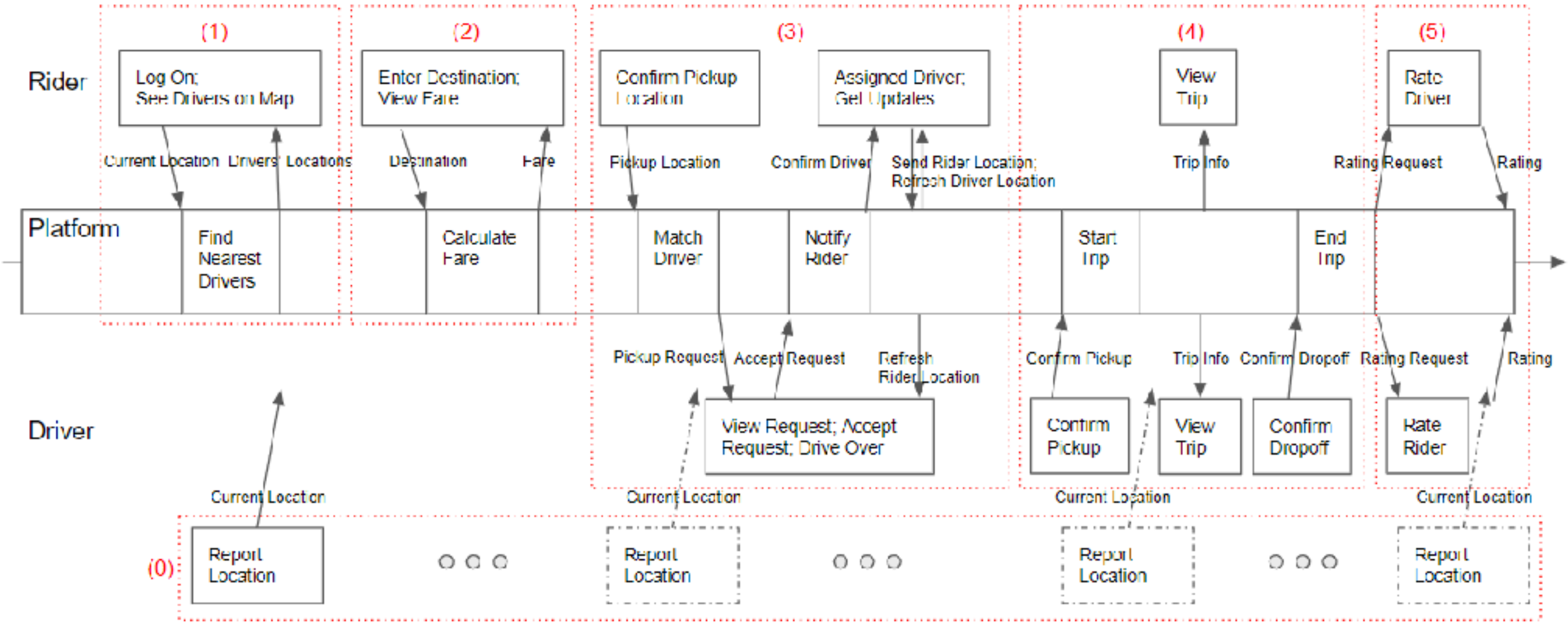
Wide column

Rdbms

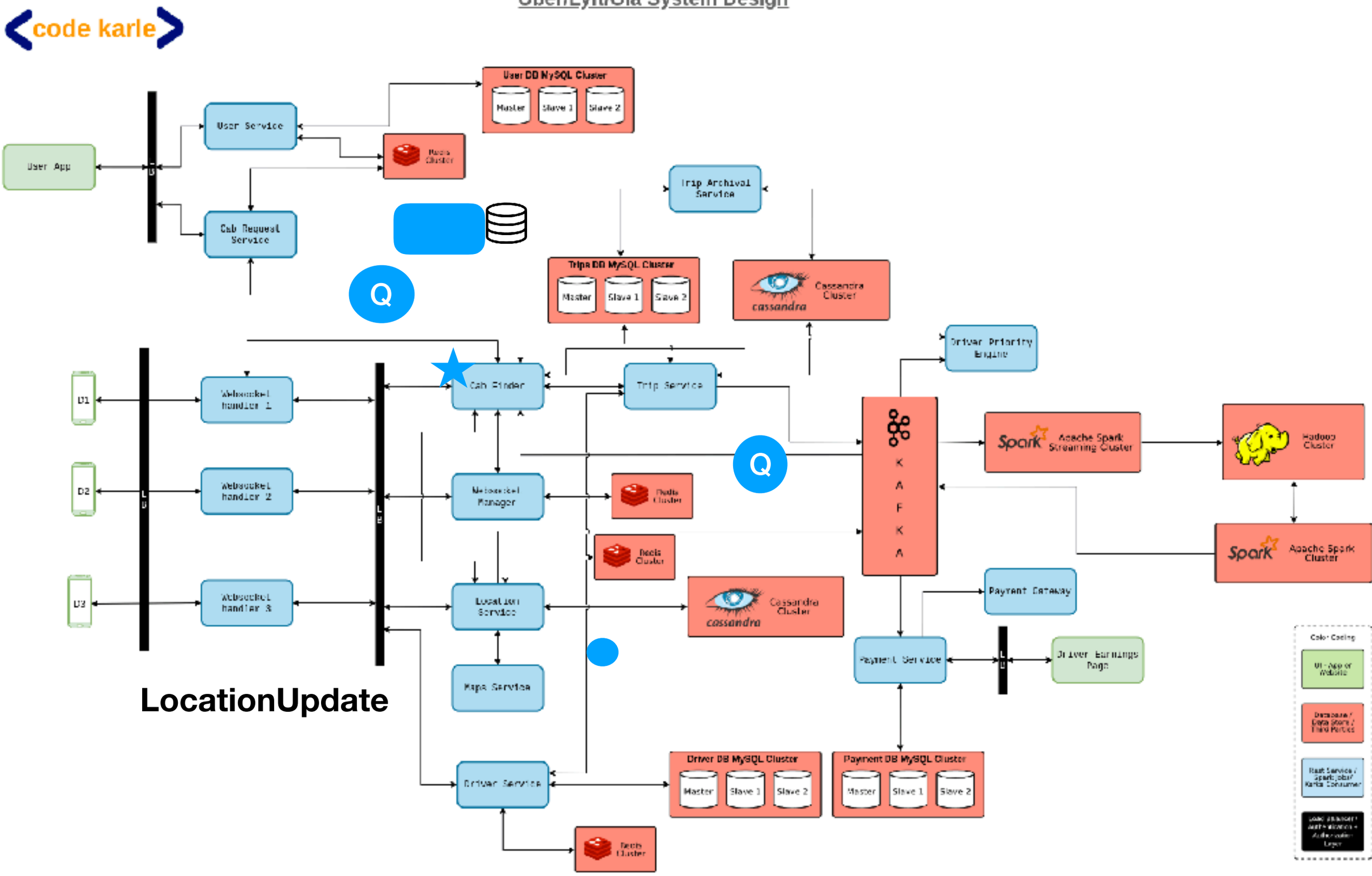
Message Queue

Uber reference architecture




(1)
(2)
(3)
(4)
(5)
Rider
Log On;
See Drivers on Map
Enter Destination;
View Fare
Confirm Pickup
Location
Assigned Driver;
Get Updates
View
Trip
Rate
Driver
Current Location
Drivers' Locations
Destination
Fare
Pickup Location
Confirm Driver
Send Rider Location;
Refresh Driver Location
Trip Info
Rating Request
Rating
Platform
Find
Nearest
Drivers
Calculate
Fare
Match
Driver
Notify
Rider
Start
Trip
End
Trip
Pickup Request
Accept Request
Refresh
Rider Location
Confirm Pickup
Trip Info
Confirm Dropoff
Rating Request
Rating
Driver
View Request; Accept
Request; Drive Over
Confirm
Pickup
View
Trip
Confirm
Dropoff
Rate
Rider
Current Location
Current Location
Current Location
Current Location
(0)
Report
Location
Report
Location
Report
Location
Report
Location

Uber/Lyft/Ola System Design



LocationUpdate

1.collect Telemetry

2.secure

3. Analyse

Customer

Device

Feature engineering
Masking
Anonymoization

Device

Gateway

Internet

wss

wss

mqtt

Message engine (7)

Stream Analytics

Analytics

Csv

clustering
sequence modelling
…

Batch Analytics

Object Storage

Device

APi

Wide column

Tableu
Looker
Power Bi

Reporting server

DWH

Device

Business User

Stock exchange Case Study