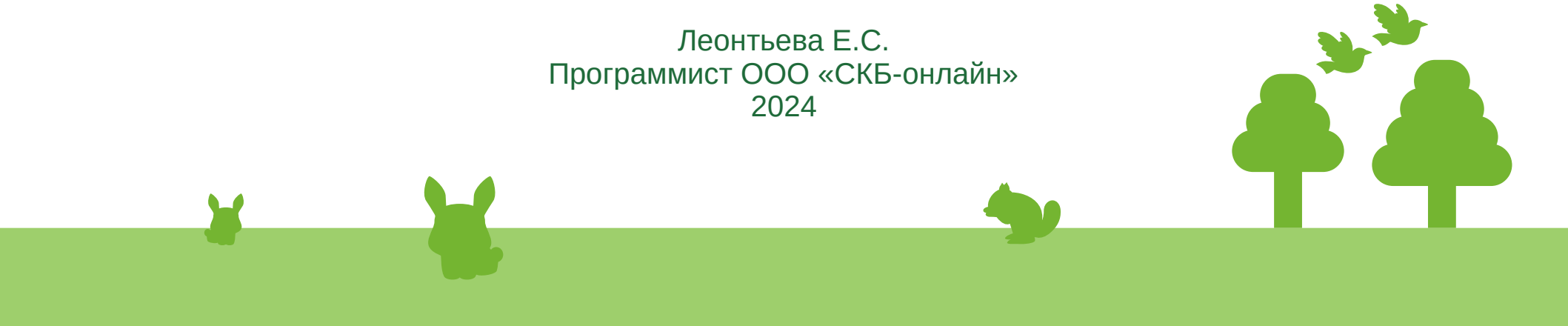
Курс: Микросервисная архитектура Otus

Итоговый проект

Интернет-магазин

Леонтьева Е.С.
Программист ООО «СКБ-онлайн»
2024



Цель работы

- Разработка учебного приложения с использованием паттернов микросервисной архитектуры

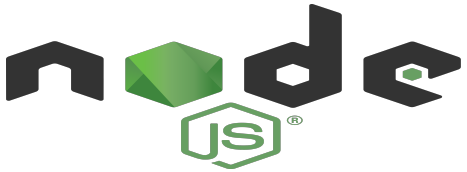
Цели обучения

Работаю в отделе интеграции, писала интеграционные приложения из одного микросервиса, которые взаимодействуют через WMQ, AMQA. Поверхностно знала технологии docker и kubernetes.

Цели:

- Научиться писать многосервисные приложения.
- Познакомиться ближе и лучше понимать работу docker, kubernetes.
- Познакомиться с разработкой API.
- Понимать смысл тех или иных архитектурных решений.

Технологический стек



Примененные паттерны

- Контейнеризация
- Server-Side Service Discovery
- Domain-Driven Design
- Database Per Service
- Сага (оркестрация)
- Health Check
- Идемпотентность

Что получилось

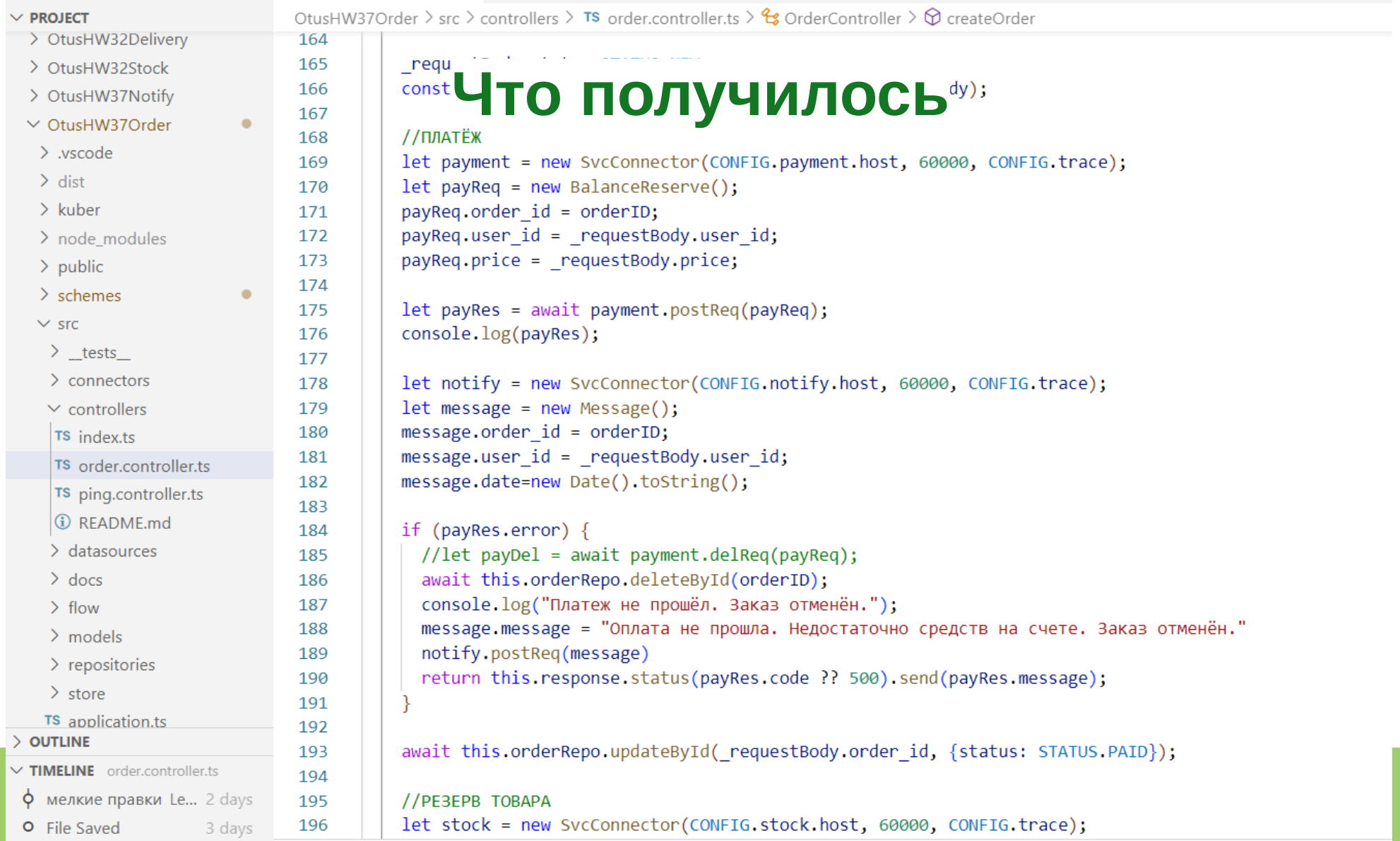
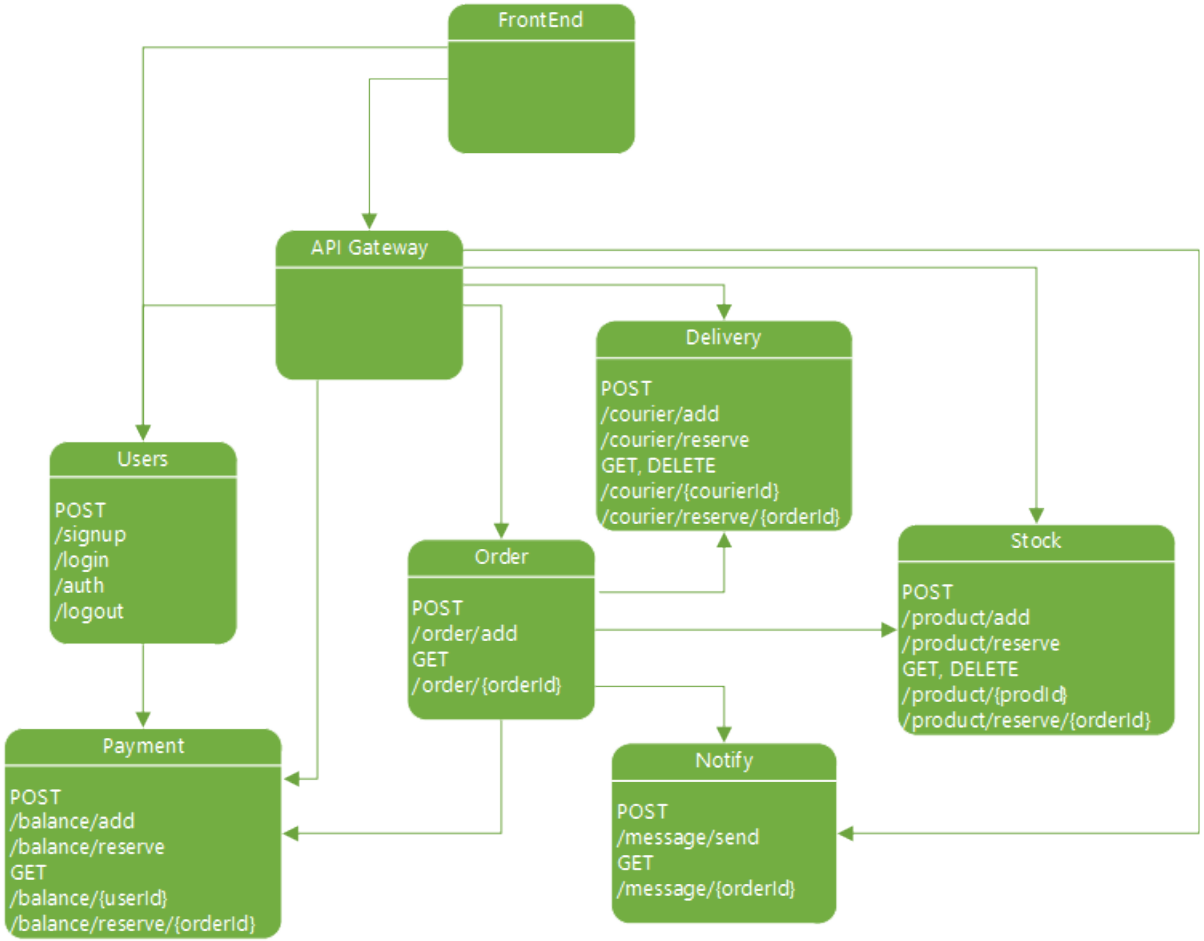


Схема взаимодействия сервисов



Исходники

1. ProjectHelm
2. User
3. Order
4. Payment
5. Stock
6. Delivery
7. Notify

Модели данных

Users

Users

PK	id
	username
	firstName
	lastName
	email
	phone
	isAdmin

UserCredentials

PK	id
	password
	userId

Payment

Balance

PK	user_id
	account
	balance

BalanceReserve

PK	id
	order_id
	user_id
	operation
	price
	completed

Order

Order

PK	order_id
	user_id
	product_id
	number
	price
	date
	status

Stock

Product

PK	product_id
	name
	number

ProductReserv

PK	order_id
	product_id
	number
	completed

Delivery

Courier

PK	courier_id
	name

CourierReserv

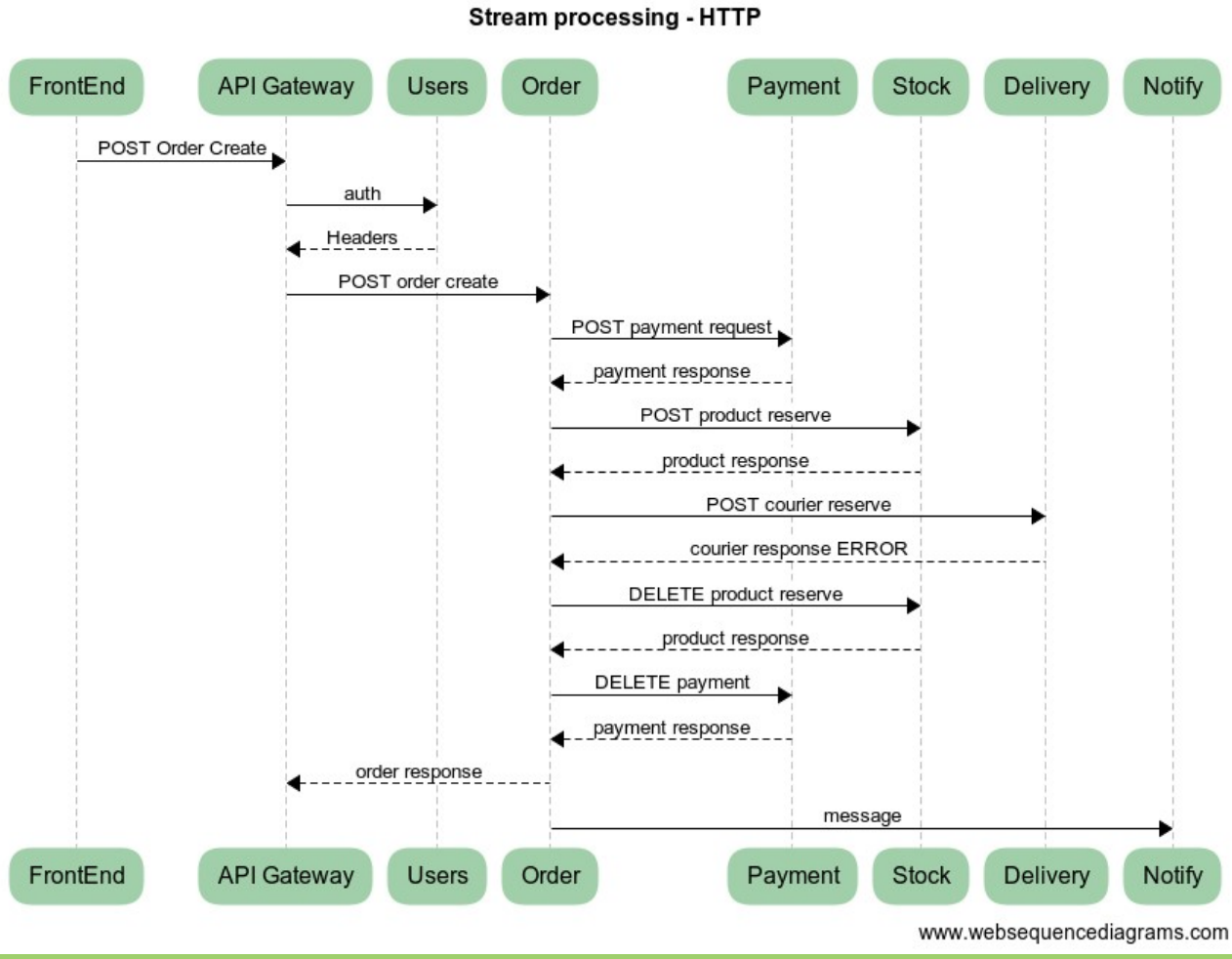
PK	order_id
	courier_id
	date

Notify

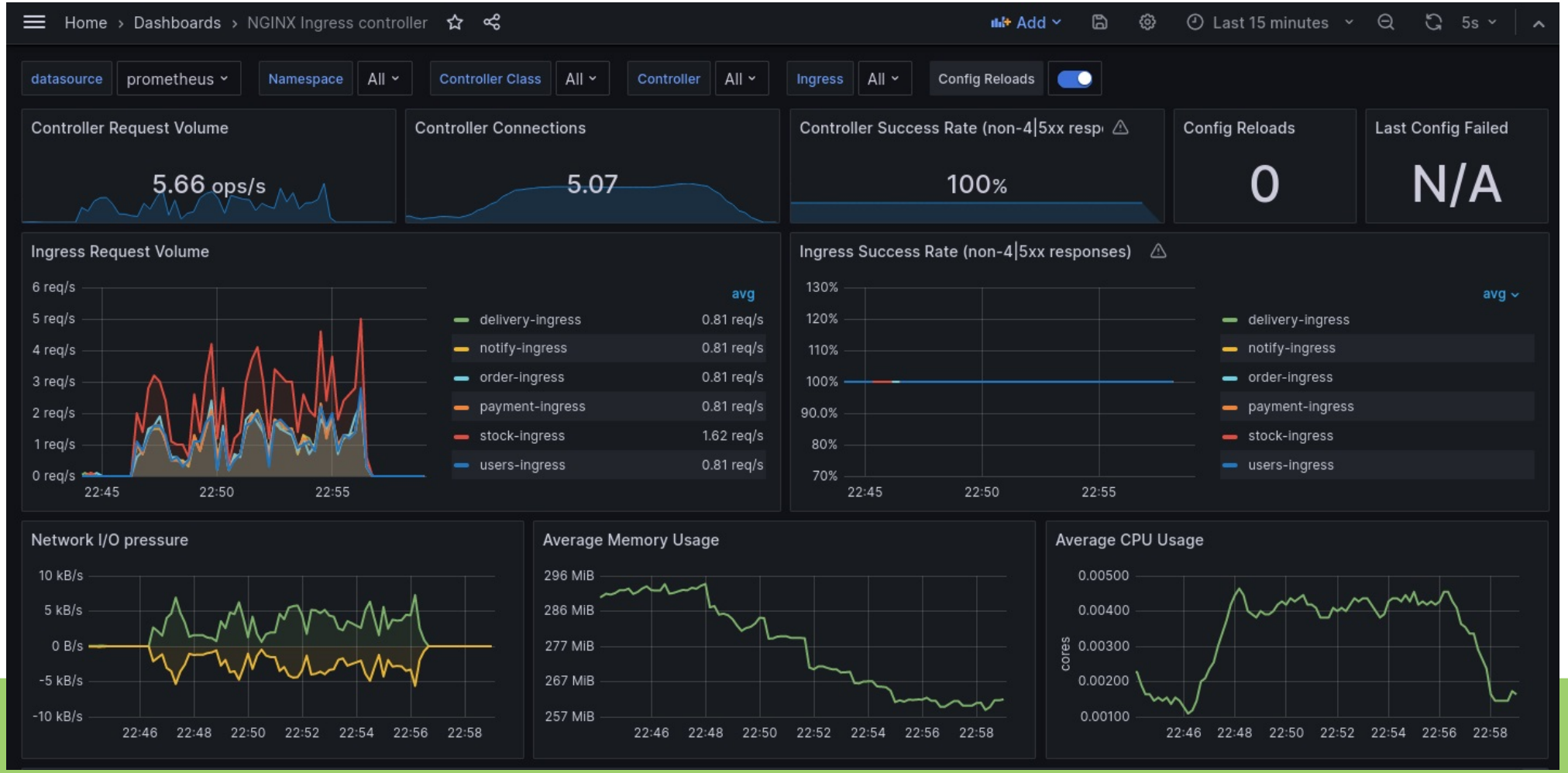
Message

PK	id
	order_id
	user_id
	date
	message

Cara


Stream processing - HTTP
FrontEnd
API Gateway
Users
Order
Payment
Stock
Delivery
Notify
POST Order Create
auth
Headers
POST order create
POST payment request
payment response
POST product reserve
product response
POST courier reserve
courier response ERROR
DELETE product reserve
product response
DELETE payment
payment response
order response
message
FrontEnd
API Gateway
Users
Order
Payment
Stock
Delivery
Notify
www.websequencediagrams.com

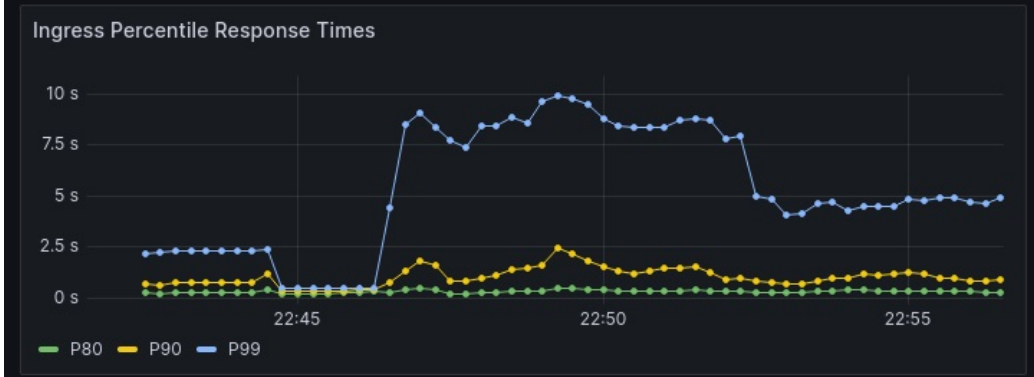
Grafana

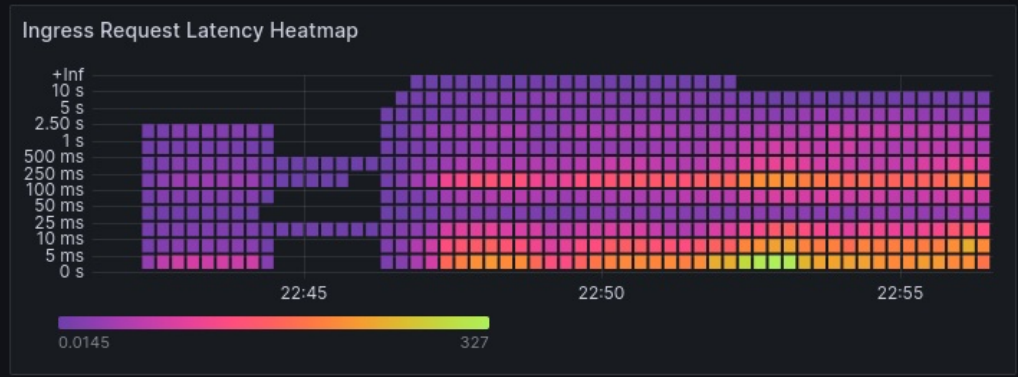

Home › Dashboards › NGINX Ingress controller
Add
Last 15 minutes
5s
datasource
prometheus
Namespace
All
Controller Class
All
Controller
All
Ingress
All
Config Reloads
Controller Request Volume
5.66 ops/s
Controller Connections
5.07
Controller Success Rate (non-4|5xx resp
100%
Config Reloads
0
Last Config Failed
N/A
Ingress Request Volume
6 req/s
5 req/s
4 req/s
3 req/s
2 req/s
1 req/s
0 req/s
22:45
22:50
22:55
avg
delivery-ingress 0.81 req/s
notify-ingress 0.81 req/s
order-ingress 0.81 req/s
payment-ingress 0.81 req/s
stock-ingress 1.62 req/s
users-ingress 0.81 req/s
Ingress Success Rate (non-4|5xx responses)
130%
120%
110%
100%
90.0%
80%
70%
22:45
22:50
22:55
avg
delivery-ingress
notify-ingress
order-ingress
payment-ingress
stock-ingress
users-ingress
Network I/O pressure
10 kB/s
5 kB/s
0 B/s
-5 kB/s
-10 kB/s
22:46
22:48
22:50
22:52
22:54
22:56
22:58
Average Memory Usage
296 MiB
286 MiB
277 MiB
267 MiB
257 MiB
22:46
22:48
22:50
22:52
22:54
22:56
22:58
Average CPU Usage
cores
0.00500
0.00400
0.00300
0.00200
0.00100
22:46
22:48
22:50
22:52
22:54
22:56
22:58

Grafana

Ingress Percentile Response Times and Transfer Rates

Ingress	P50 Latency	P90 Latency	P99 Latency	IN	OUT
users-ingress	156 milliseconds	312 milliseconds	775 milliseconds	120.30 B/s	96.30 B/s
stock-ingress	6 milliseconds	133 milliseconds	446 milliseconds	226.80 B/s	167.10 B/s
payment-ingress	232 milliseconds	1 second	7 seconds	135.60 B/s	84.00 B/s
order-ingress	907 milliseconds	4 seconds	8 seconds	191.10 B/s	104.10 B/s
notify-ingress	7 milliseconds	18 milliseconds	24 milliseconds	89.10 B/s	141.30 B/s
delivery-ingress	11 milliseconds	163 milliseconds	652 milliseconds	112.80 B/s	78.90 B/s





Выводы и планы по развитию

- При увеличении нагрузки перейти на асинхронное взаимодействие (Kafka/MQ).
- Логирование в ELK.
-