Order Processing System

Категория	Технология
Язык	Go 1.22+
API	gRPC + REST (Gin)
База данных	MySQL 8.0 + GORM
Очереди	Apache Kafka 3.x (KRaft)
Кэш	Redis
Observability	Prometheus + Grafana + Jaeger
CI/CD	GitHub Actions
Контейнеризация	Docker + Kubernetes

1. Технологический стек

Категория	Технология	Назначение
Язык	Go 1.22+	Все сервисы
API (внутренний)	gRPC + Protobuf	Межсервисное взаимодействие
API (внешний)	Gin	REST API Gateway
ORM	GORM	Работа с БД
База данных	MySQL 8.0	Основное хранилище
Кэш	Redis	Rate limiting, JWT blacklist, кэш
Очереди	Kafka 3.x (KRaft)	Асинхронные события
Метрики	Prometheus	Сбор метрик
Визуализация	Grafana	Дашборды
Трейсинг	Jaeger	Distributed tracing
CI/CD	GitHub Actions	Автоматизация
Контейнеры	Docker + K8s	Деплой
Безопасность	mTLS	Шифрование между сервисами
Логирование	Zerolog	Структурированные логи (RU)

2. Архитектура системы

Микросервисная архитектура с Saga Orchestration. Внешние клиенты → REST API Gateway → gRPC сервисы. Асинхронное взаимодействие через Kafka с Outbox Pattern.

Компоненты:

Уровень	Компонент	Порт	Описание
Gateway	API Gateway (Gin)	:8080	REST → gRPC, JWT, Rate Limit
Сервисы	User Service	:50051	Регистрация, авторизация, JWT
Сервисы	Order Service	:50052	CRUD заказов, Saga Orchestrator
Сервисы	Payment Service	:50053	Обработка платежей
Инфра	MySQL	:3306	Основная БД
Инфра	Kafka (KRaft)	:9092	Брокер сообщений
Инфра	Redis	:6379	Кэш, Rate limiting, JWT blacklist
Observability	Prometheus	:9090	Метрики
Observability	Grafana	:3000	Дашборды
Observability	Jaeger	:16686	Трейсинг

3. Описание сервисов

3.1 API Gateway

- Маршрутизация HTTP → gRPC
- JWT аутентификация + blacklist в Redis
- Rate limiting через Redis
- Circuit Breaker
- Correlation ID + Trace ID

3.2 User Service

- Регистрация/авторизация
- Генерация JWT с jti (уникальный ID токена)
- Logout → добавление jti в blacklist

3.3 Order Service (Saga Orchestrator)

- CRUD заказов
- Координатор Saga транзакций
- Outbox Pattern для гарантии доставки
- Компенсирующие транзакции при откате

3.4 Payment Service

- Приём Saga команд из Kafka
- Идемпотентность операций
- Retry с exponential backoff

4. Saga Orchestration

Пример: Создание заказа

Шаг	Сервис	Действие	Компенсация
1	Order Service	Создать заказ (PENDING)	Удалить заказ
2	Payment Service	Списать средства	Вернуть средства
3	Order Service	Подтвердить (CONFIRMED)	—

Состояния Saga:

Состояние	Описание
STARTED	Saga инициирована
PAYMENT_PENDING	Ожидание ответа от Payment
COMPLETED	Успешно завершена
COMPENSATING	Выполняется откат
FAILED	Завершена с ошибкой

5. Kafka 3.x (KRaft)

Kafka работает в режиме KRaft (без Zookeeper). Контроллеры встроены в брокеры.

Топики:

Топик	Producer	Consumer	Payload
saga.commands	Order Service	Payment Service	saga_id, command, order_id, amount
saga.replies	Payment Service	Order Service	saga_id, status, payment_id, error
dlq.saga	Kafka	—	Failed messages

Outbox Pattern:

1. В одной транзакции: сохранить заказ + записать в outbox
2. Worker читает outbox → отправляет в Kafka
3. После успешной отправки — помечает как обработанную

6. JWT с механизмом инвалидации

Проблема: отозванный токен валиден до истечения TTL. Решение: blacklist в Redis.

Алгоритм:

1. При генерации JWT добавляем claim jti (уникальный ID)
2. При logout: SET jwt:blacklist:{jti} 1 EX {remaining_ttl}
3. При валидации: если ключ существует → токен невалиден

Redis TTL:

Ключ	TTL	Описание
jwt:blacklist:{jti}	remaining TTL токена	Отозванные токены
rate:{ip}	1 минута	Rate limiting
order:{id}	10 минут	Кэш заказа
idempotency:{key}	24 часа	Идемпотентность

7. Тестирование

Покрытие только критичных путей. Без тестов на геттеры/сеттеры.

Тип	Что покрываем	Инструмент
Unit	Saga-координатор (переходы состояний)	go test + testify
Unit	Валидация бизнес-правил	go test + testify
Integration	Repository → MySQL	testcontainers-go
Integration	Kafka producer/consumer	testcontainers-go
E2E	Happy-path: создание → оплата → подтверждение	go test + httptest

8. CI/CD (GitHub Actions)

CI Pipeline (на каждый push/PR):

1. Checkout кода
2. Setup Go 1.22
3. golangci-lint run
4. go test ./... -v -race -cover
5. docker build (multi-stage)
6. Push образов в Container Registry

CD Pipeline (на merge в main):

1. Deploy на staging (kubectl apply)
2. Smoke tests
3. Deploy на production (manual approve)

9. Паттерны надёжности

Circuit Breaker:

Состояние	Поведение
CLOSED	Нормальная работа
OPEN	Все запросы отклоняются
HALF-OPEN	Один тестовый запрос

Порог: 5 ошибок подряд → OPEN. Таймаут: 30 сек.

Retry с Exponential Backoff:

- Максимум попыток: 3
- Задержка: 100ms → 200ms → 400ms
- Jitter: ±10%

10. Структура проекта

Директория	Назначение
proto/	gRPC контракты (*.proto)
pkg/	Общие пакеты (kafka, logger, config, middleware)
services/	Микросервисы
deployments/	Docker + Kubernetes
.github/workflows/	CI/CD пайплайны
Makefile	Команды сборки

Структура сервиса:

Директория	Назначение
cmd/main.go	Точка входа
internal/domain/	Бизнес-сущности
internal/repository/	Работа с БД
internal/service/	Бизнес-логика
internal/grpc/	gRPC handlers
internal/saga/	Saga координатор
migrations/	SQL миграции

11. Makefile команды

Команда	Описание
make proto	Генерация Go кода из proto
make build	Сборка Docker образов
make up	Запуск системы
make down	Остановка
make test	go test ./... -v -cover
make lint	golangci-lint run
make migrate	Применение миграций

12. Ключевые паттерны

- Clean Architecture — domain/repository/service/handler
- Saga Orchestration — распределённые транзакции
- Outbox Pattern — гарантия доставки в Kafka
- Circuit Breaker — защита от каскадных падений
- Idempotency — безопасная повторная обработка
- Graceful Shutdown — корректное завершение
- Distributed Tracing — сквозной трейсинг
- JWT Blacklist — инвалидация токенов

13. План реализации

Последовательность реализации проекта. Тесты пишутся внутри каждой фазы, сразу после реализации компонента.

Фаза 1: Инфраструктура

Шаг	Компонент	Описание
1.1	Docker Compose	MySQL, Redis, Kafka (KRaft), Jaeger
1.2	Makefile	proto, build, up, down, test, lint, migrate
1.3	Структура папок	proto/, pkg/, services/, deployments/

Фаза 2: Общие пакеты (pkg/)

Шаг	Пакет	Описание
2.1	pkg/logger	Zerolog, структурированные логи на RU
2.2	pkg/config	Чтение конфигов из ENV/YAML
2.3	pkg/kafka	Producer/Consumer обёртки
2.4	pkg/middleware	gRPC interceptors (tracing, logging)

Фаза 3: User Service

Шаг	Компонент	Тесты
3.1	Proto-файлы + генерация	—
3.2	Domain (User, Claims)	—
3.3	Repository	Unit: mock DB
3.4	Service (Register, Login, Logout)	Unit: бизнес-логика
3.5	JWT генерация с jti	Unit: валидация токенов

3.6	Blacklist в Redis	Integration: testcontainers
3.7	gRPC handlers	Unit: mock service

Фаза 4: API Gateway

Шаг	Компонент	Тесты
4.1	Gin роутинг	—
4.2	JWT middleware + blacklist check	Unit: валидный/невалидный токен
4.3	Rate limiting (Redis)	Unit: превышение лимита
4.4	gRPC клиент к User Service	Integration: mock gRPC server
4.5	Correlation ID + Trace ID	—

Фаза 5: Order Service (без Saga)

Шаг	Компонент	Тесты
5.1	Proto-файлы + генерация	—
5.2	Domain (Order, OrderItem)	—
5.3	Repository	Integration: testcontainers MySQL
5.4	Service (Create, Get, List)	Unit: бизнес-правила
5.5	gRPC handlers	Unit: mock service
5.6	Подключение к Gateway	E2E: создание заказа через REST

Фаза 6: Saga + Outbox

Шаг	Компонент	Тесты
6.1	Таблица outbox + модель	—
6.2	Saga координатор (состояния)	Unit: все переходы состояний
6.3	Outbox worker → Kafka	Integration: testcontainers Kafka
6.4	Компенсирующие транзакции	Unit: откат при ошибке

Фаза 7: Payment Service

Шаг	Компонент	Тесты
7.1	Domain (Payment)	—
7.2	Consumer saga.commands	Integration: testcontainers Kafka
7.3	Идемпотентность (Redis)	Unit: повторный запрос
7.4	Producer saga.replies	Integration: проверка сообщений
7.5	Retry с exponential backoff	Unit: количество попыток

После Фазы 7: E2E тест полного flow — создание заказа → оплата → подтверждение.

Фаза 8: Observability

Шаг	Компонент	Описание
8.1	Prometheus метрики	HTTP/gRPC latency, error rate, Kafka lag
8.2	Grafana дашборды	Сервисы, Kafka, Redis, MySQL
8.3	Jaeger tracing	Propagation через gRPC и Kafka headers

Фаза 9: Надёжность + CI/CD

Шаг	Компонент	Описание
9.1	Circuit Breaker в Gateway	Порог 5 ошибок, таймаут 30 сек
9.2	Graceful Shutdown	Завершение goroutines, закрытие коннектов
9.3	K8s манифесты	Deployments, Services, ConfigMaps
9.4	Health checks	Liveness + Readiness probes
9.5	GitHub Actions CI	lint → test → build → push
9.6	GitHub Actions CD	staging → smoke → production

Сводка по тестам

Фаза	Unit тесты	Integration тесты	E2E тесты
3. User Service	Repository, Service, JWT	Redis blacklist	—
4. API Gateway	JWT middleware, Rate limit	gRPC client	—
5. Order Service	Service, Handlers	MySQL repository	REST → Order
6. Saga + Outbox	Координатор, Компенсации	Kafka producer	—
7. Payment Service	Идемпотентность, Retry	Kafka consumer	Full flow

Принцип

Написал компонент → сразу тест к нему. Не накапливать технический долг.