Микросервисная архитектура приложения для онлайн-покупок

1. Фронтенд (React/Vue.js + TypeScript)

Фронтенд отвечает за взаимодействие с пользователем и отображение интернет-магазина.

Основные функции:
- каталог товаров с фильтрами и поиском;
- корзина покупок и оформление заказа;
- личный кабинет пользователя (история заказов, избранное);
- отзывы и рейтинги товаров;
- админ-панель для управления товарами и заказами (если требуется).

Фронтенд общается с бэкендом через API Gateway, используя REST/GraphQL и JWT для авторизации.

2. API Gateway (Node.js + Express / Apollo Server)

Единая точка входа для всех запросов от клиентов.

Функции:
- маршрутизация запросов к нужным микросервисам;
- балансировка нагрузки между сервисами;
- агрегация данных (например, товар + отзывы + наличие на складе);
- кеширование частых запросов (Redis);
- защита от DDoS и валидация JWT.

3. Сервис аутентификации (Node.js + JWT + PostgreSQL)

Отвечает за:
- регистрацию и вход (email/пароль, OAuth через Google/Facebook);
- управление ролями (покупатель, продавец, админ);
- восстановление пароля;
- JWT-авторизацию.

Хранит данные пользователей в PostgreSQL, сессии – в Redis.

4. Сервис товаров (Node.js + PostgreSQL/Elasticsearch)

Функционал:
- управление каталогом (CRUD товаров, категории, теги);
- поиск и фильтрация (Elasticsearch для быстрого поиска);

- остатки на складах;
- рекомендации товаров (коллаборативная фильтрация).

5. Сервис заказов (Node.js + PostgreSQL)

Основные задачи:
- создание и обработка заказов;
- управление статусами (оформлен, оплачен, доставлен, отменен);
- интеграция с платежными системами (Stripe, PayPal);
- история заказов пользователя.

6. Сервис корзины (Node.js + Redis)

- Временное хранение корзины (Redis для скорости).
- Синхронизация между устройствами.
- Применение промокодов и скидок.

7. Сервис платежей (Node.js + Stripe API / PayPal API)

- Обработка платежей (карты, СПБ, SberPay и т.д.).
- Возвраты средств.
- Подписки и рекуррентные платежи.

8. Сервис доставки (Node.js + внешние API)

- Расчет стоимости и сроков доставки (интеграция с Boxberry, СДЭК, DHL).
- Отслеживание статуса доставки.
- Уведомления клиента о статусе.

9. Сервис уведомлений (Node.js + RabbitMQ / Kafka)

- Email/SMS-уведомления (о заказе, доставке, акциях).
- Push-уведомления в мобильное приложение.
- Интеграция с Telegram/WhatsApp.

Работает асинхронно через RabbitMQ/Kafka.

10. Сервис отзывов и рейтингов (Node.js + MongoDB)

– Хранение отзывов и оценок товаров.
– Модерация отзывов (автоматическая и ручная).
– Анализ sentiment-анализа (NLP для выявления негатива).

MongoDB подходит из-за гибкости структуры отзывов.

11. Брокер сообщений (RabbitMQ / Kafka)

– Асинхронная обработка событий (новый заказ, отмена, оплата).
– Очереди для уведомлений и логирования.
– Event Sourcing для отслеживания изменений.

12. Мониторинг (Prometheus + Grafana + ELK)

– Метрики сервисов (загрузка CPU, память, время ответа).
– Логирование ошибок (ELK Stack)
– Бизнес-метрики (конверсия, средний чек, популярные товары)

Взаимодействие сервисов на примере типичных сценариев

Сценарий 1. Оформление заказа

1. Пользователь добавляет товары в корзину (Redis).

2. При оформлении заказа API Gateway отправляет запрос в сервис заказов.

3. Сервис заказов:

   – проверяет наличие товаров (сервис товаров);
   – создает заказ в PostgreSQL;
   – отправляет событие в RabbitMQ (уведомления, платежи).

4. Сервис платежей обрабатывает оплату.

5. Сервис уведомлений отправляет письмо с подтверждением.

6. Сервис доставки рассчитывает сроки и отправляет трек-номер.

Сценарий 2. Поиск товара

1. Пользователь вводит запрос в поиск.

2. API Gateway отправляет запрос в сервис товаров.

3. Elasticsearch быстро находит совпадения.

4. Дополнительно подгружаются отзывы и наличие на складе.

Заключение

Такая архитектура обеспечивает:

- масштабируемость (можно увеличивать мощности отдельных сервисов);
- отказоустойчивость (падение одного сервиса не ломает всю систему);
- гибкость (легко добавлять новые функции, например, подписки или чат с поддержкой);
- производительность (кеширование, асинхронная обработка).