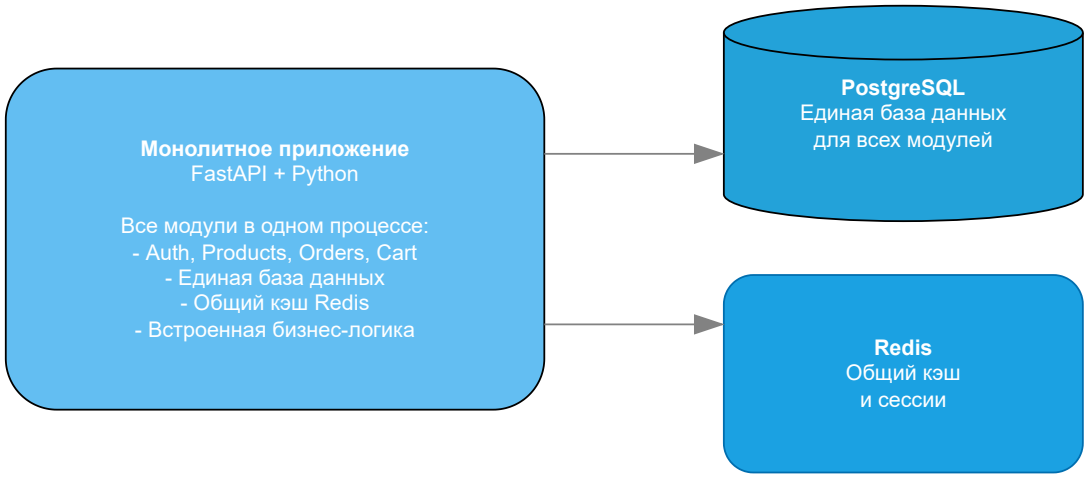
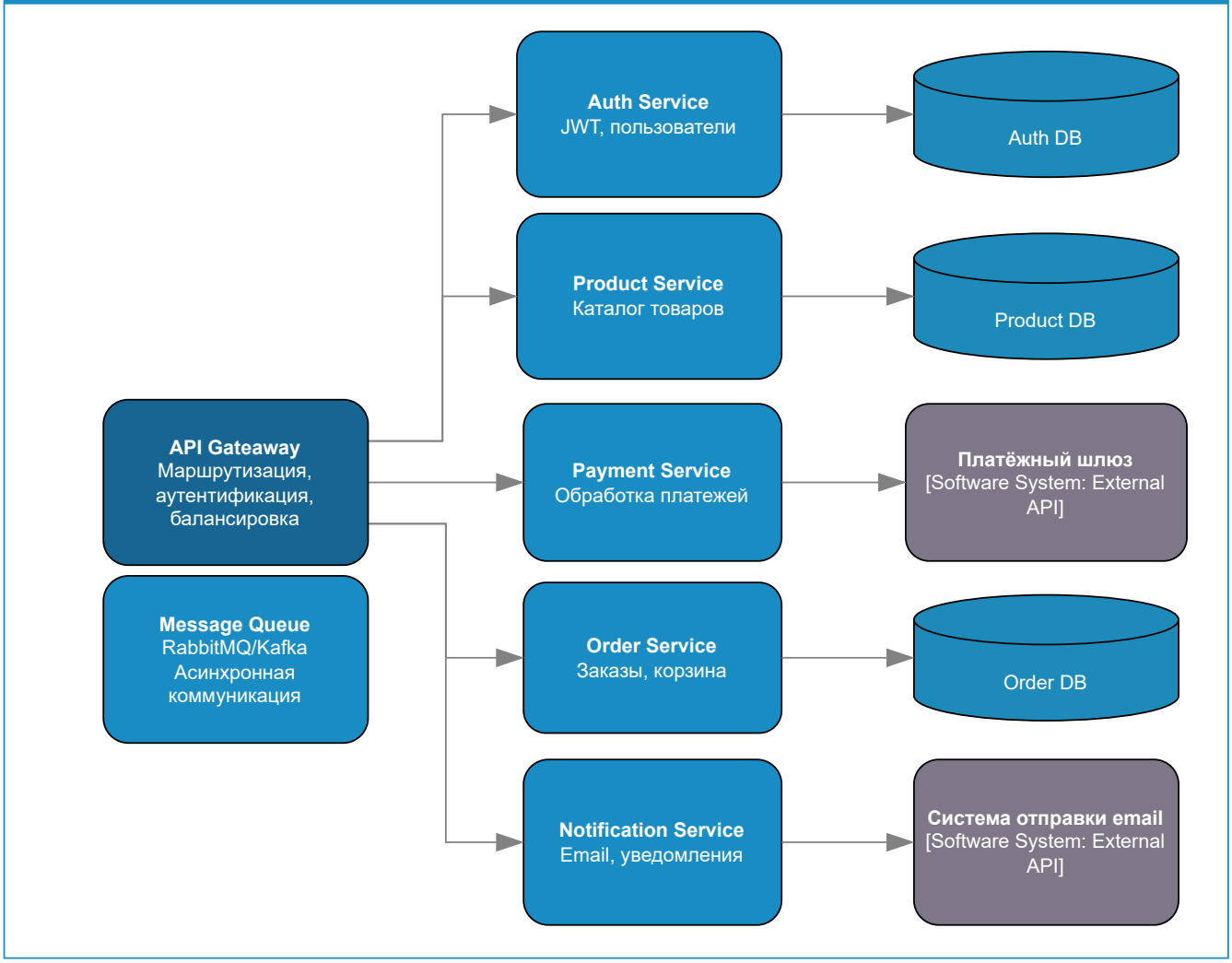
Вариант 1: Монолитная архитектура



✅ Простая отладка
✅ Единая кодовая база
✅ Простота разработки и деплоя
✅ Минимальные накладные расходы

⛔ Зависимость технологического стека
⛔ Сложность внесения изменений
⛔ Сложность масштабирования
⛔ Единая точка отказа

Вариант 2: Микросервисная архитектура



✅ Разные технологии
✅ Гибкость разработки
✅ Устойчивость к отказам
✅ Независимое масштабирование

⛔ Проблема согласованности данных
⛔ Накладные расходы сети
⛔ Сложность оркестрации
⛔ Сложность отладки

Требования к ресурсам для разных масштабов

Масштаб	CPU	RAM	Storage	Монолит	Микросервисы
10^3 пользователей	2-4 ядра	4-8 GB	50-100 GB	✓ Экономия	✗ Избыточно
10^5 пользователей	8-16 ядер	16-32 GB	500 GB - 1 TB	~ Окей	✓ Оптимально
10^6 пользователей	32-64+ ядер	64-128+ GB	2-5+TB	✗ Сложно	✓ Идеально

Рассуждения

Монолит

Идеально для: Стартапов, MVP, небольших проектов

Когда выбирать: Команда < 10 человек, ограниченные ресурсы, быстрый вывод на рынок

Критически важно: Чистая архитектура с четким разделением слоев

Микросервисы

Идеально для: Крупных проектов, enterprise-решений

Когда выбирать: Команда > 20 человек, необходимость независимых циклов разработки

Критически важно: Мониторинг, оркестрация (Kubernetes), DevOps культура

Гибридный подход

Учитывая текущую структуру проекта, можно рассмотреть постепенную миграцию:

1) Начать с монолита с четким модульным разделением
2) Выделять микросервисы по мере роста и выявления узких мест
3) Использовать Domain-Driven Design для определения границ сервисов

Этот подход позволяет получить преимущества обеих архитектур, минимизируя риски.