

Skillbox

Распределенная архитектура

Андрей Гордиенков


Solution Architect
ABAX

Обо мне

Solution Architect @ ABAX

- 15 лет в IT
- IT-тренер:
 - Архитектура
 - Разработка
 - Базы данных
- Докладчик на конференциях
- Основной язык C#



 Twitter: @violettape

 LinkedIn: <https://www.linkedin.com/in/violettape/>

В прошлом модуле

- Рассмотрели историю становления архитектуры ПО
- Узнали, что такое монолит
- Рассмотрели заблуждения о распределенных вычислениях

В этом модуле

- «Первое правило» распределенных систем
- Архитектурные стили для распределенных систем
- Как выбрать стиль?

В ЭТОМ ВИДЕО

- Рассмотрите «первое правило» распределенных систем
- Значение правила на практике
- Базовые стили для реализации бизнес-логики

Распределенная система

Вольное определение:

«Набор независимых компьютеров, которые для пользователя выглядят как один»

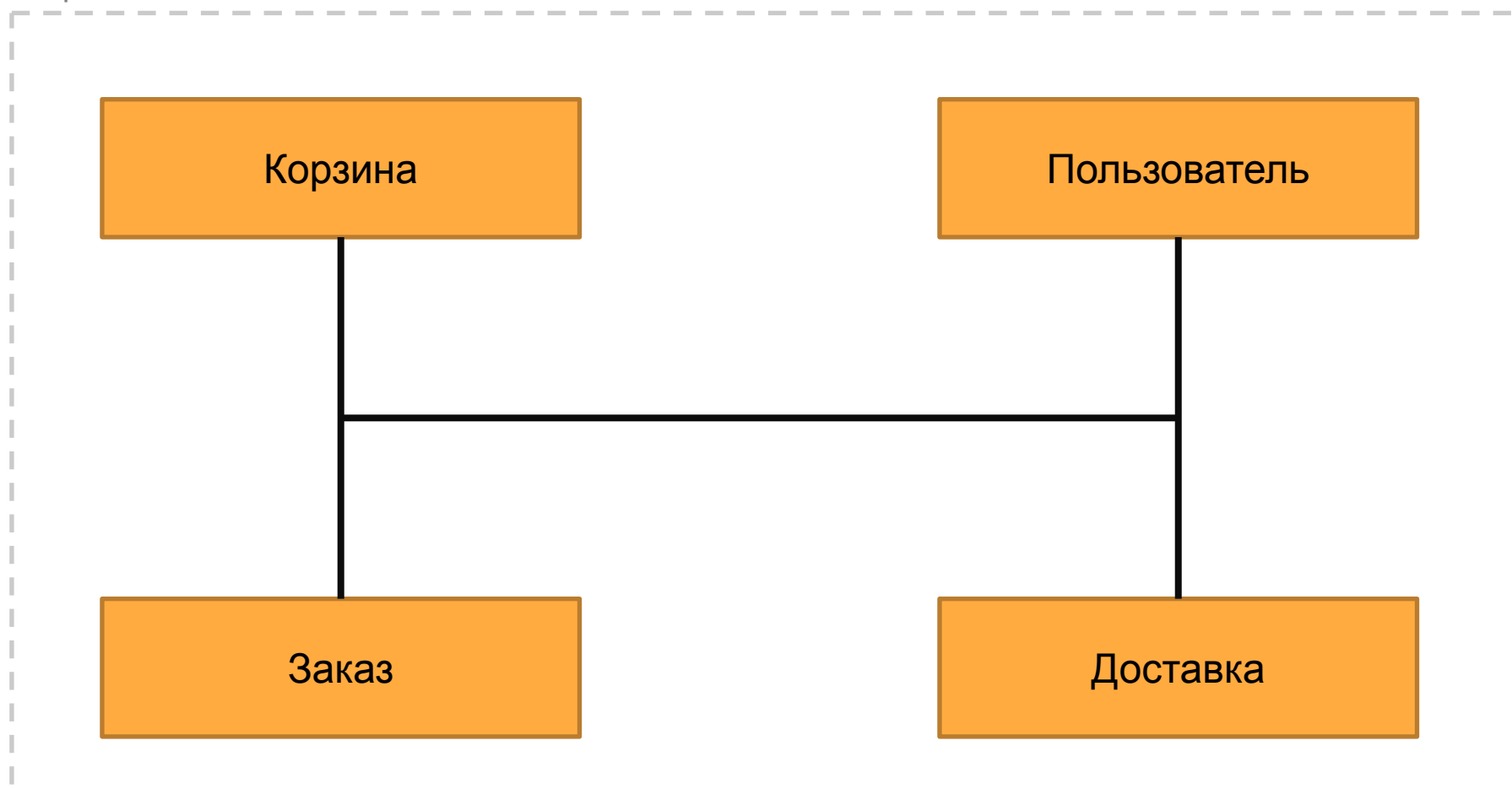
Andrew Tannenbaum

Характеристики:

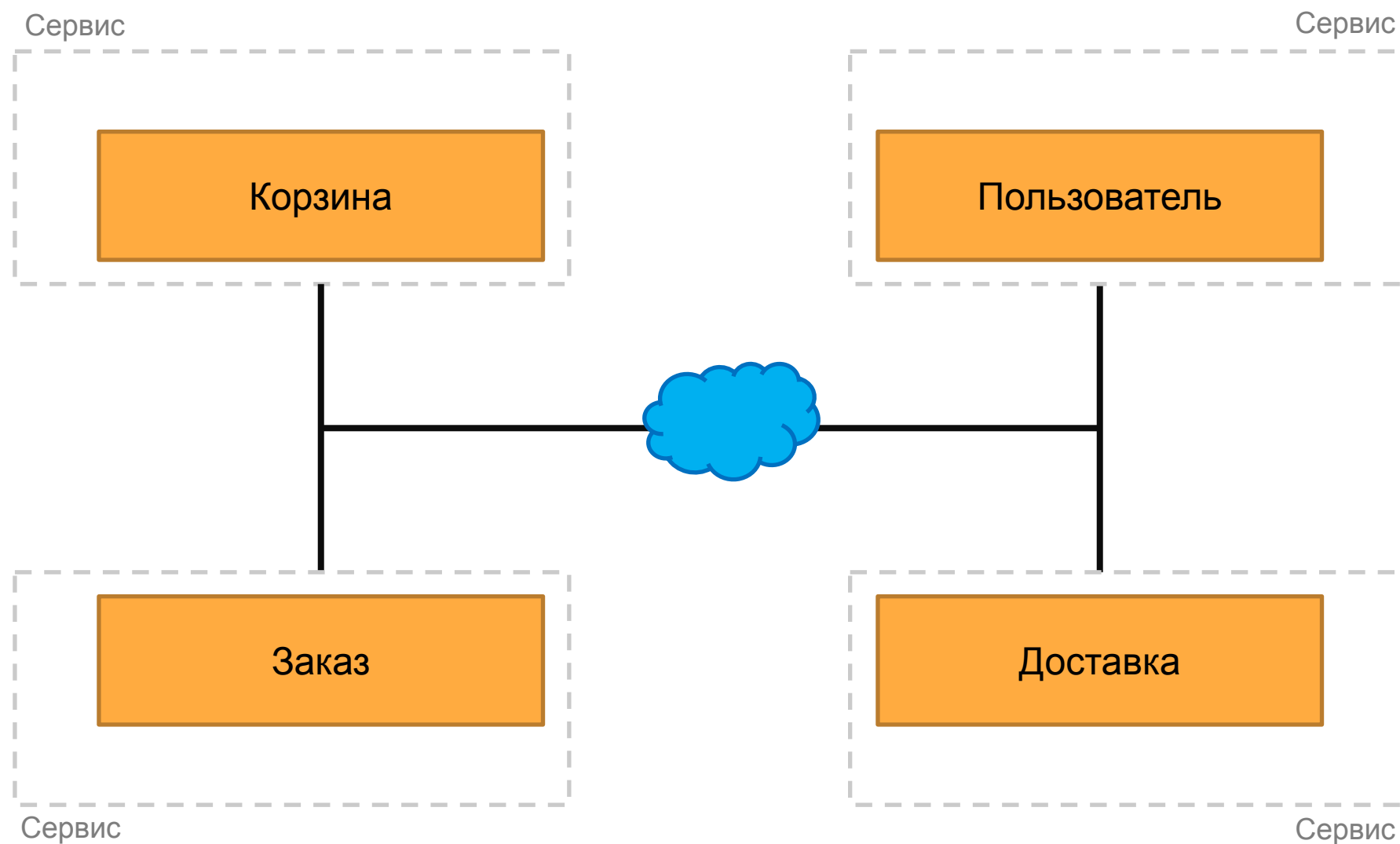
- Компьютеры работают одновременно
- Сбой компьютеров происходит независимо
- Компьютеры не имеют общей синхронизации

«Первое правило»

Сервис

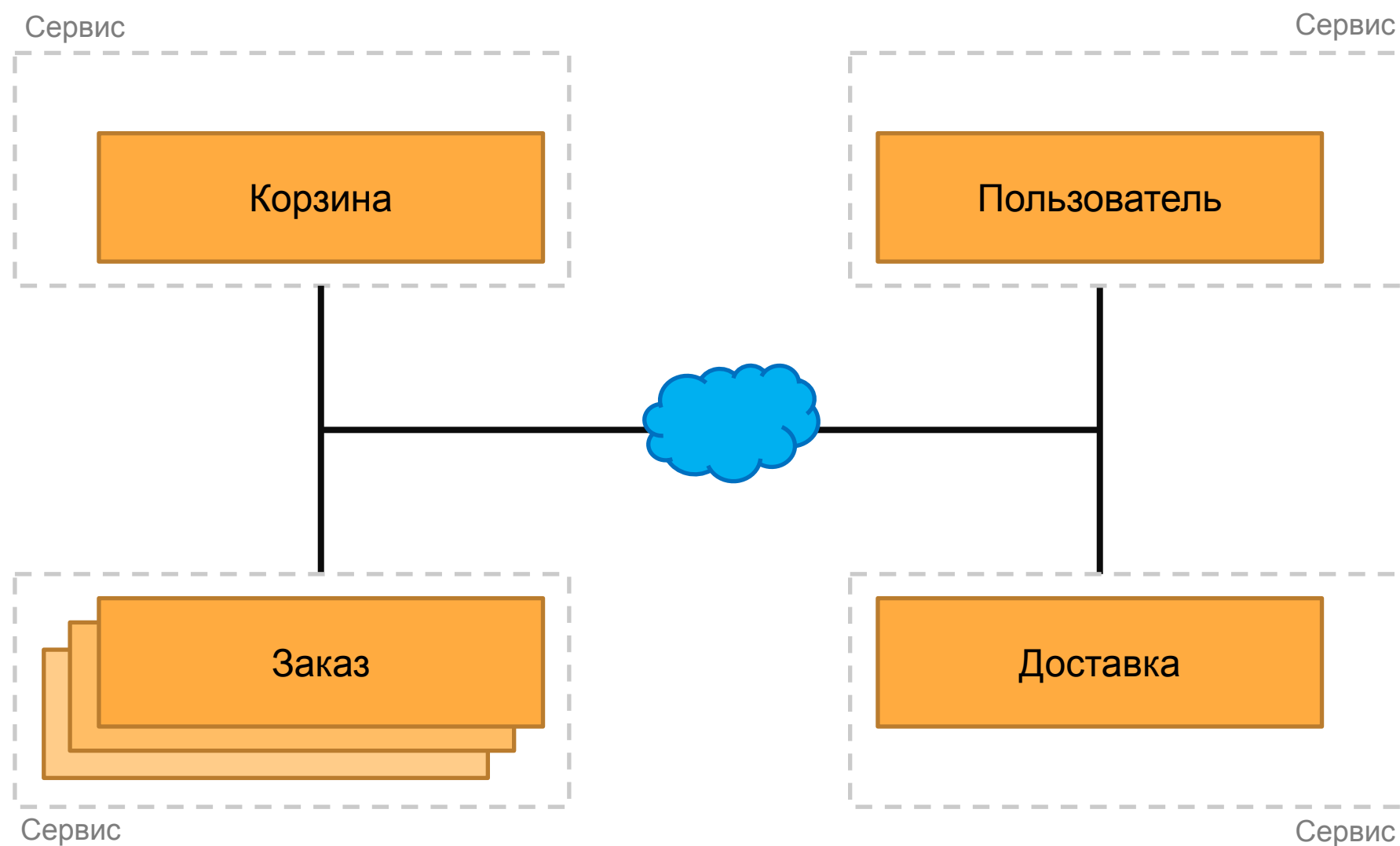


«Первое правило»



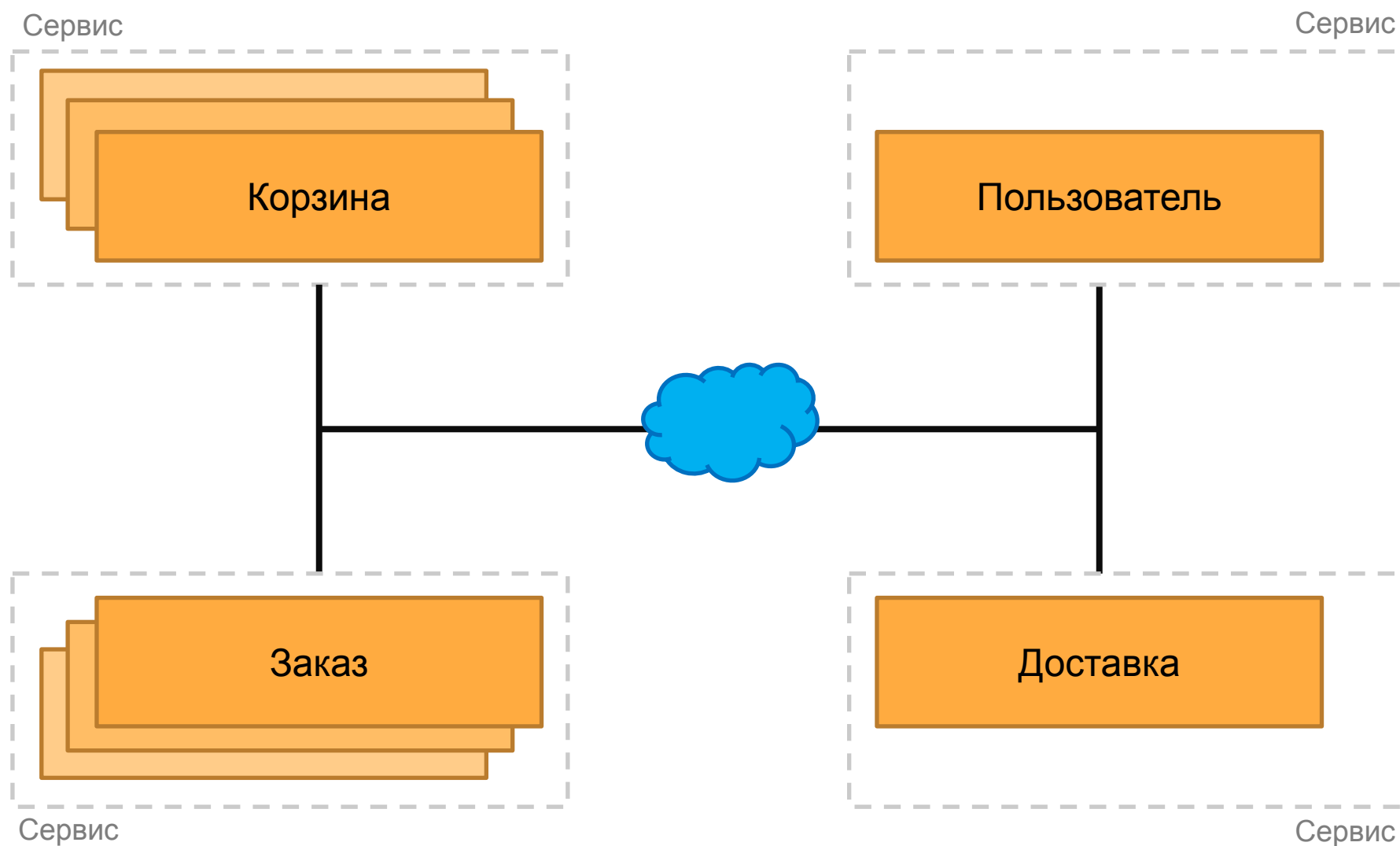
- Независимость разработки

«Первое правило»



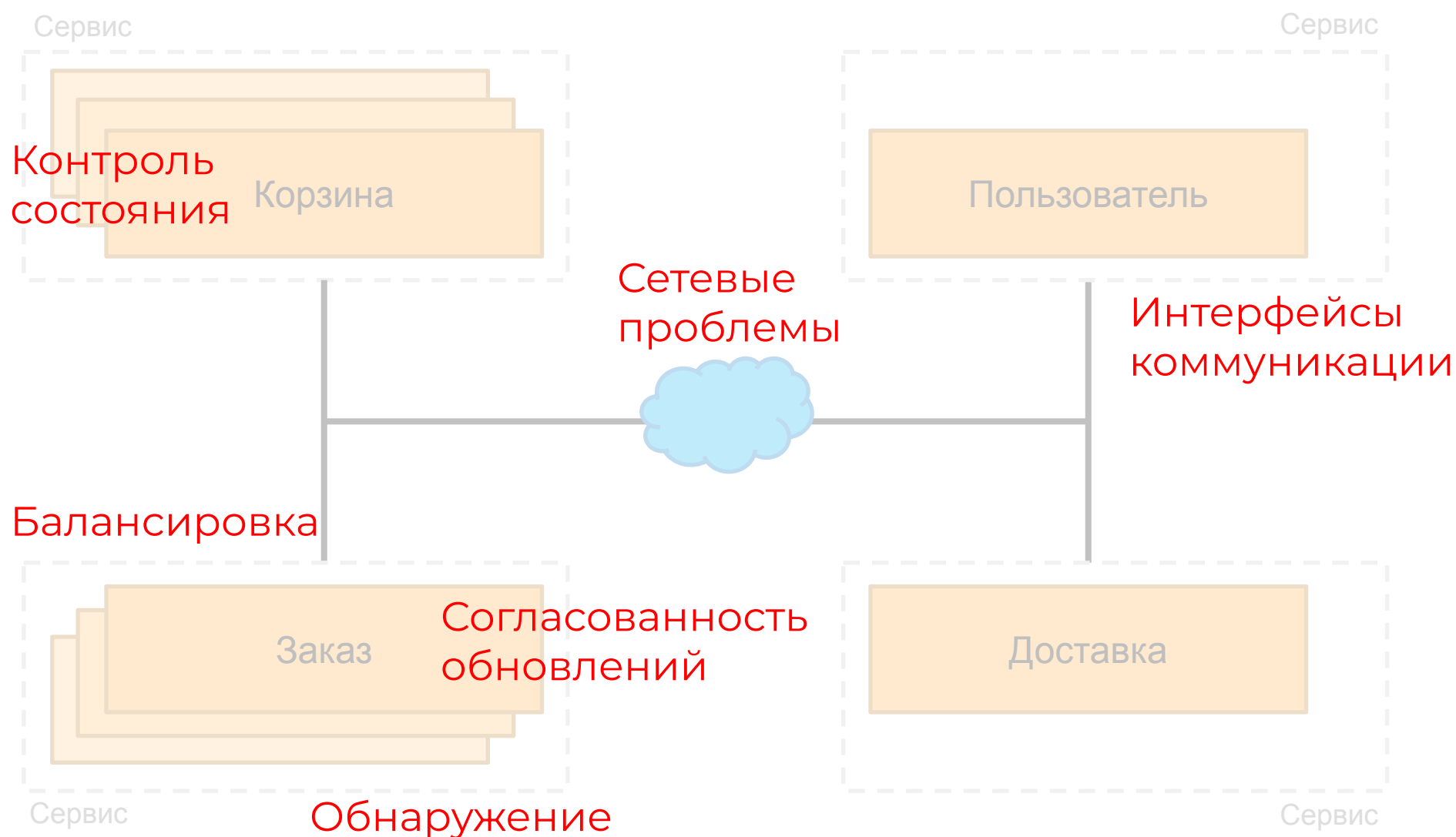
- Независимость разработки
- Производительность сервисов

«Первое правило»



- Независимость разработки
- Производительность сервисов
- Независимость поставки
- Легкость интеграции

«Первое правило»



- Независимость разработки
- Производительность сервисов
- Независимость поставки
- Легкость интеграции

«Первое правило» распределенных систем



**Не распределяйте сущности системы, пока
этого можно избежать**

Как узнать, когда можно?

Когда будут известны основные домены и под-домены,
и их зоны ответственности.

Как узнать, когда можно?

Когда будут известны основные домены и под-домены,
и их зоны ответственности.

Опять DDD?

- Transactional Script
- Table Module
- Domain-Driven Design (DDD)

Transactional Script

метод **Заказ**(корзина, пользователь)

 ПроверитьКорзину(корзина)

 ПроверитьНаличиеТоваров(корзина)

 сумма = ПодсчитатьСуммуЗаказа(корзина)

 сумма =+ ПодсчитатьЦенуДоставки(корзина, пользователь)

 фактура = Оплата(пользователь, сумма)

 ВыслатьУведомление(пользователь, фактура)

 ОтправитьЗаказНаСклад(фактура)

 ОчиститьКорзину(корзина)

метод **ПроверитьКорзину**(корзина)

метод **ПроверитьНаличиеТоваров**(корзина)

...

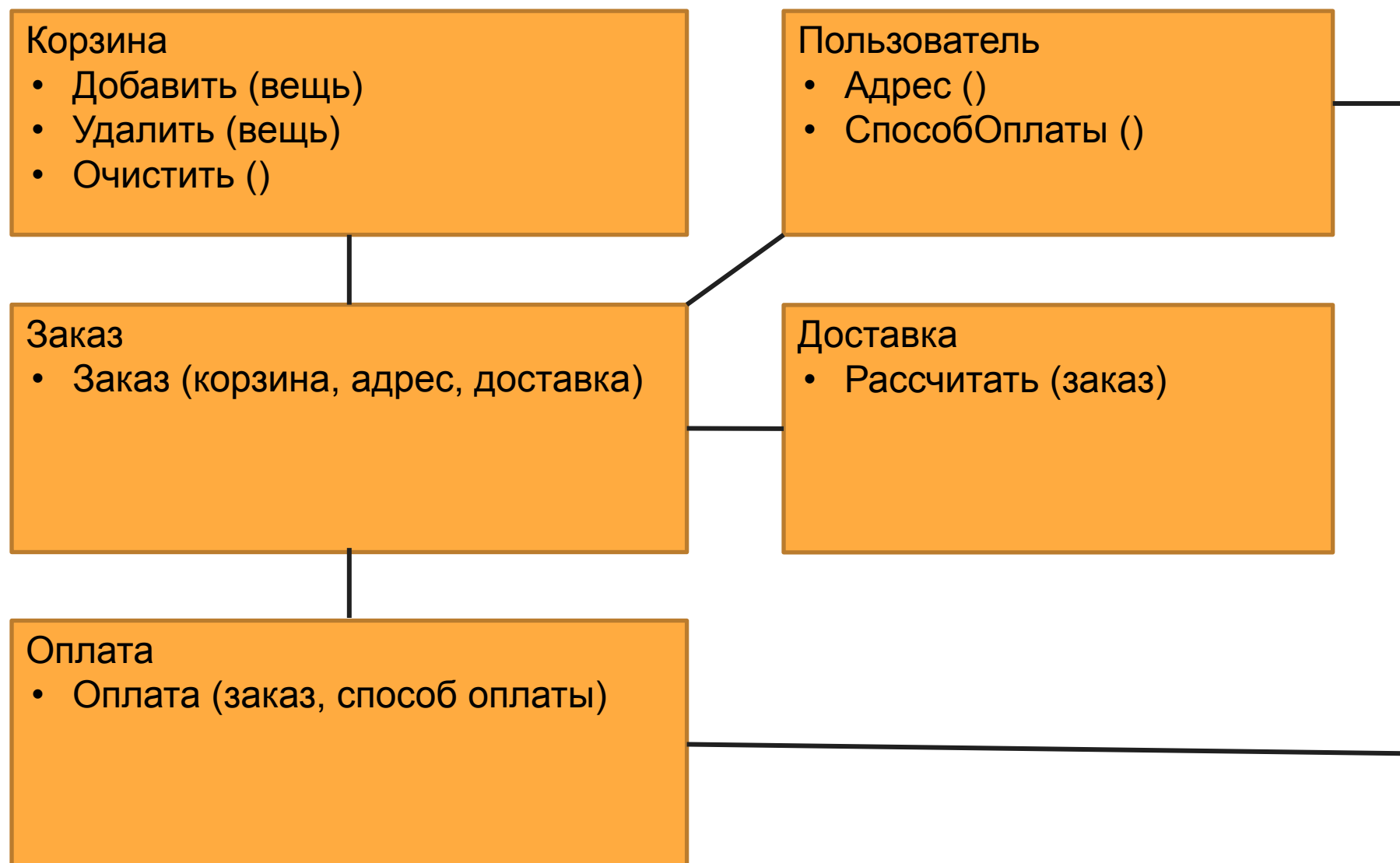
Table based

Вещь	Цена	Заказ	Кому	Корзина	Статус	Доставка	Куда
AAA	10	R-0101	John	X			
BBB	51	R-0101	John	X			
CCC	15	R-0101	John	X			
AAA	10	S-2232	Iren		Оплачено	Да	Address X
CCC	15	R-0433	John		Ожидает оплаты	Самовывоз	

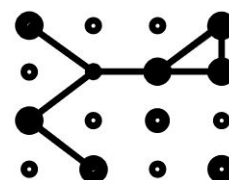
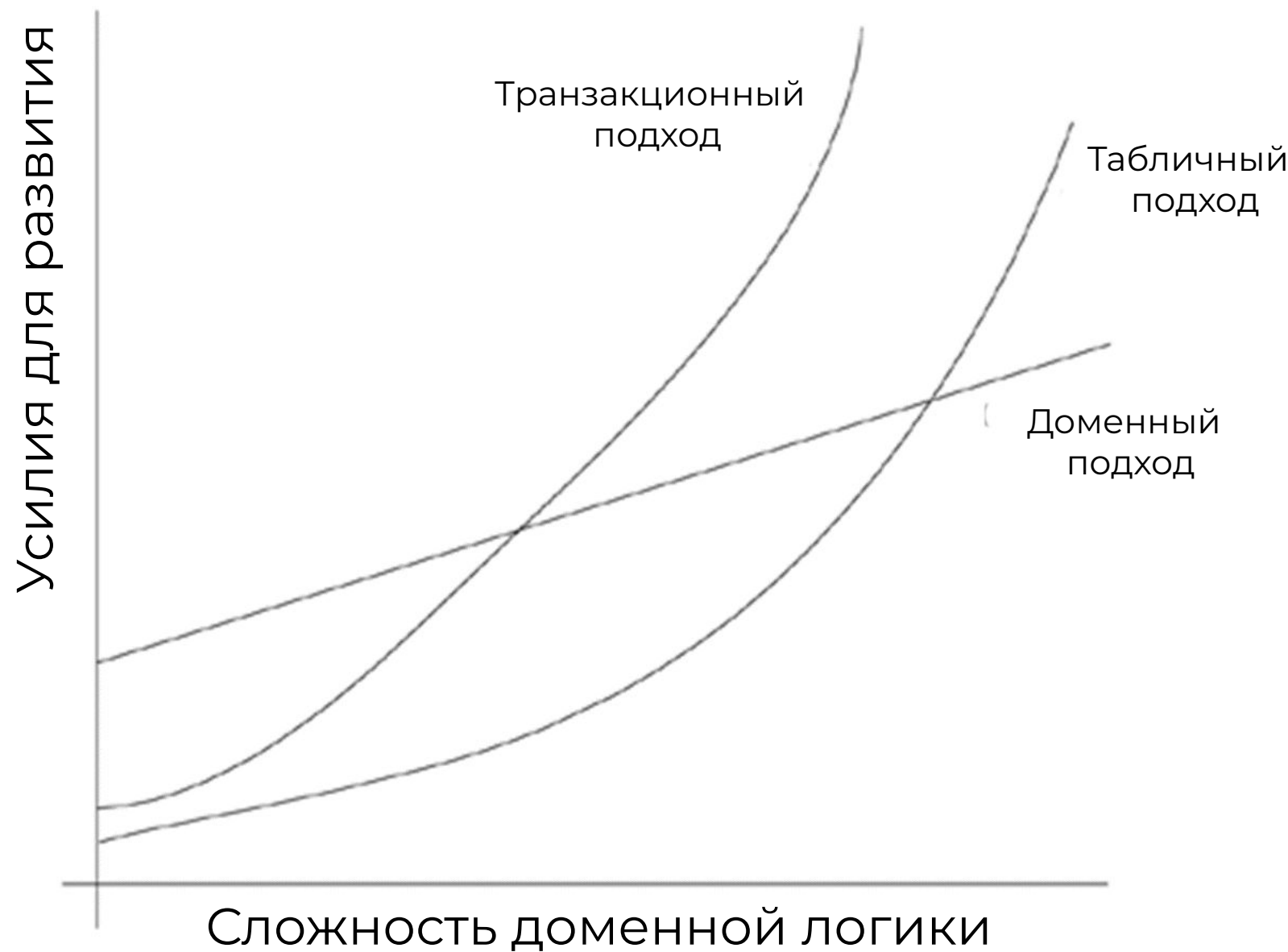
Несколько ключевых таблиц над которыми производятся CRUD операции и поиск:

- **C**reate
- **R**ead
- **U**ppdate
- **D**eleate
- НайтиКорзинуДля()
- НайтиОплаченныеЗаказыДля()
- ...

Domain-Driven Design



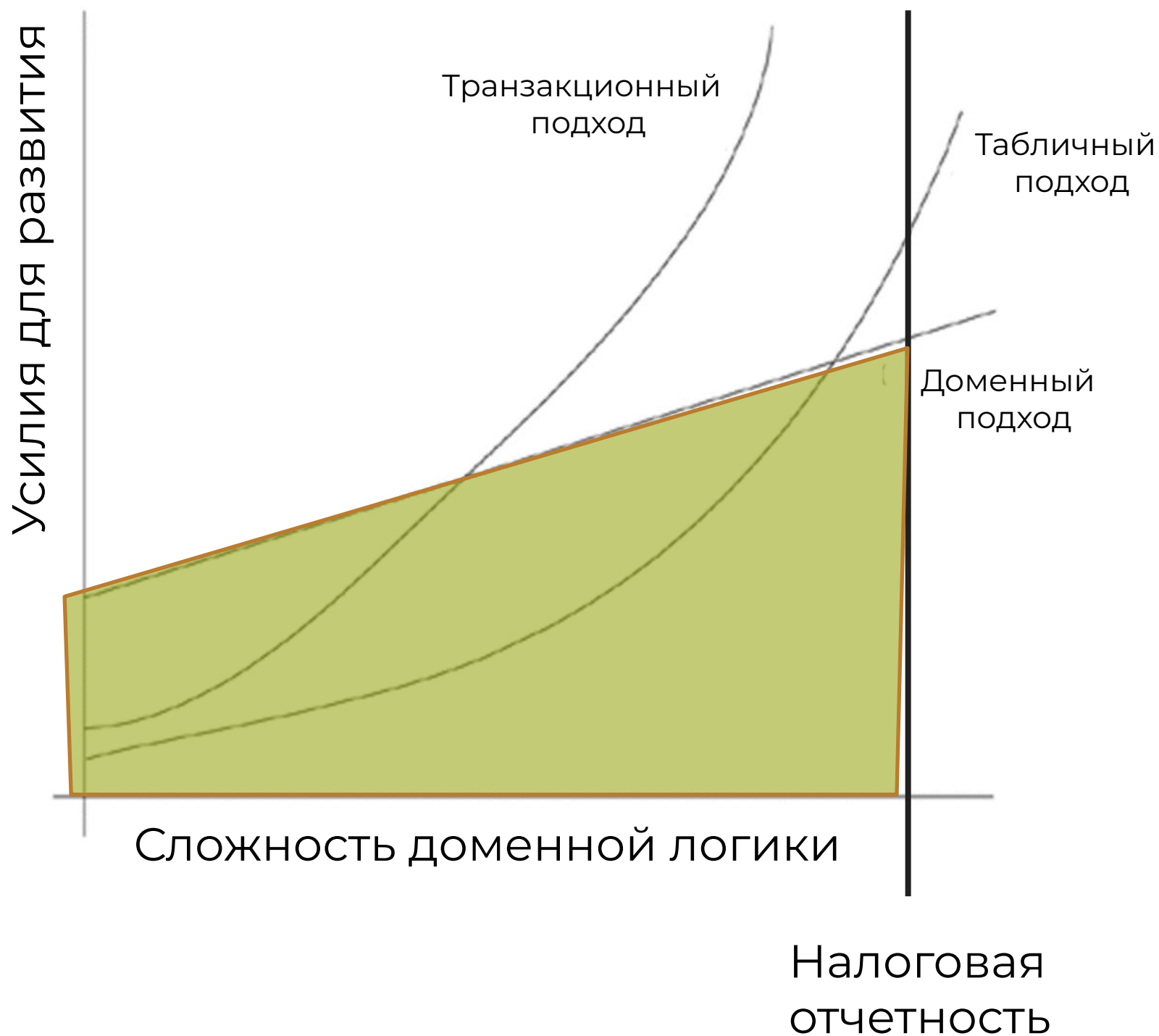
Сравнение подходов



Сравнение подходов



Сравнение подходов



«Первое правило»

Сервис заказов

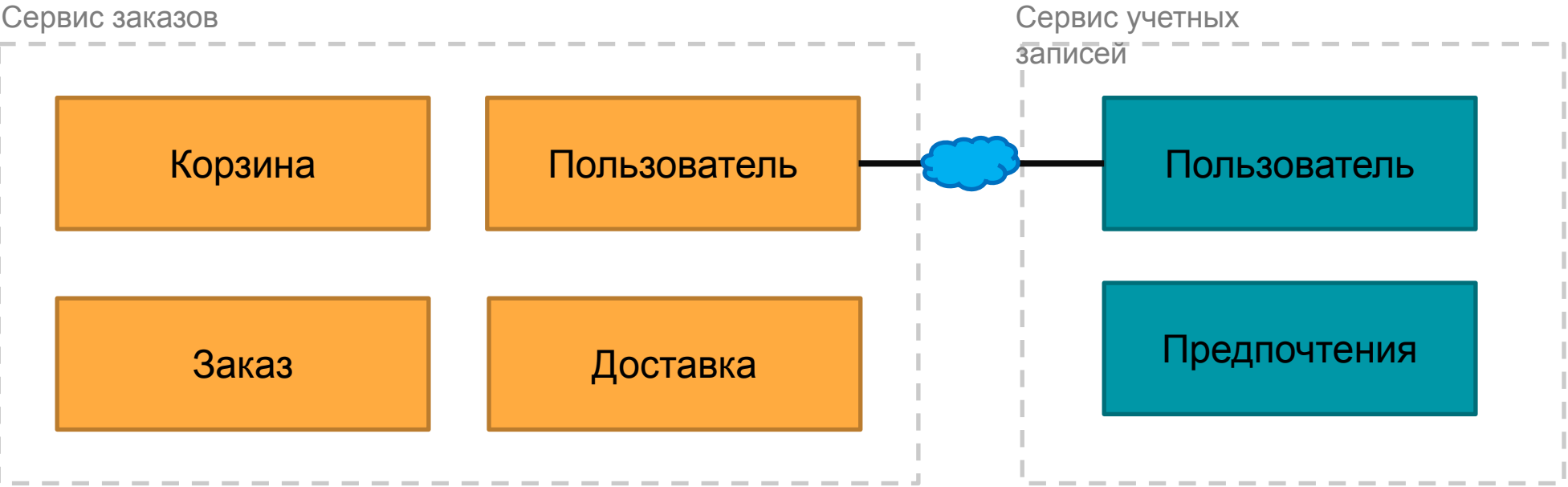
Корзина

Пользователь

Заказ

Доставка

«Первое правило»

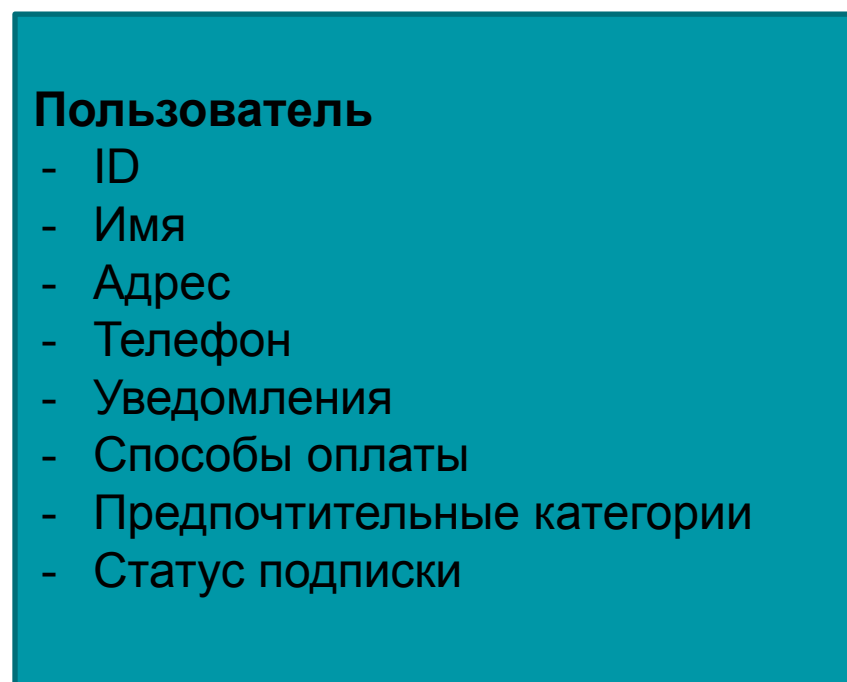
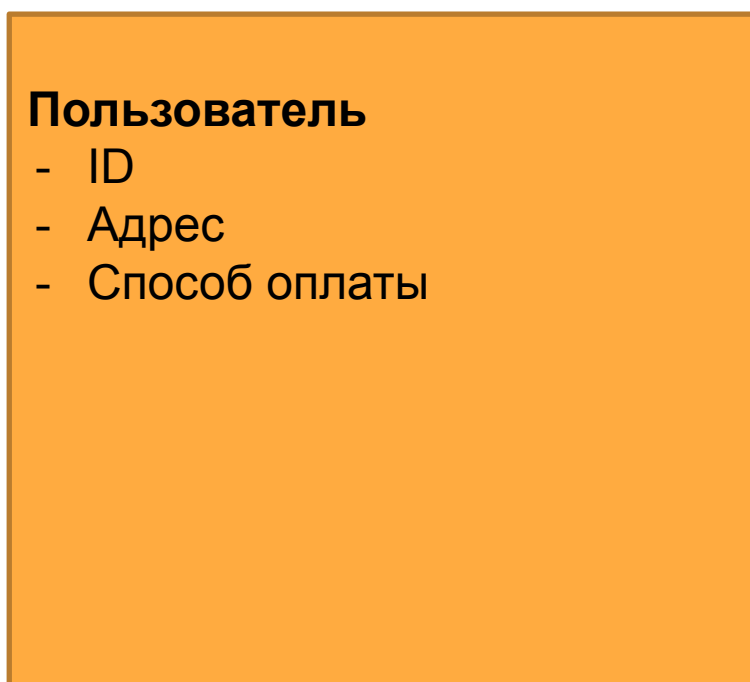
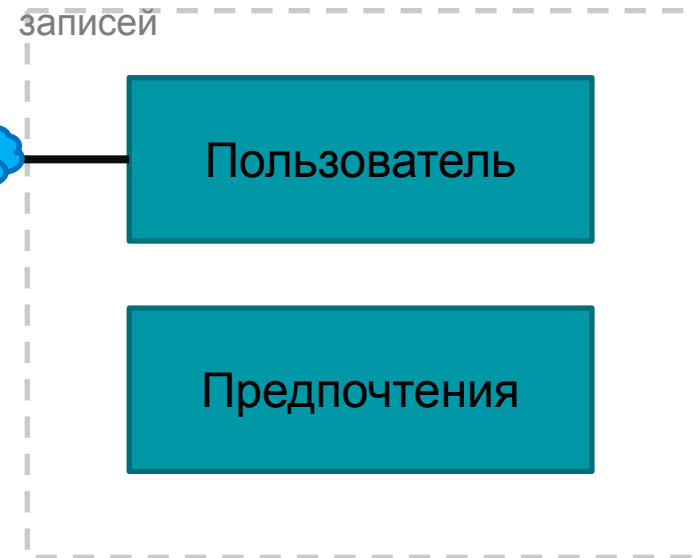


«Первое правило»

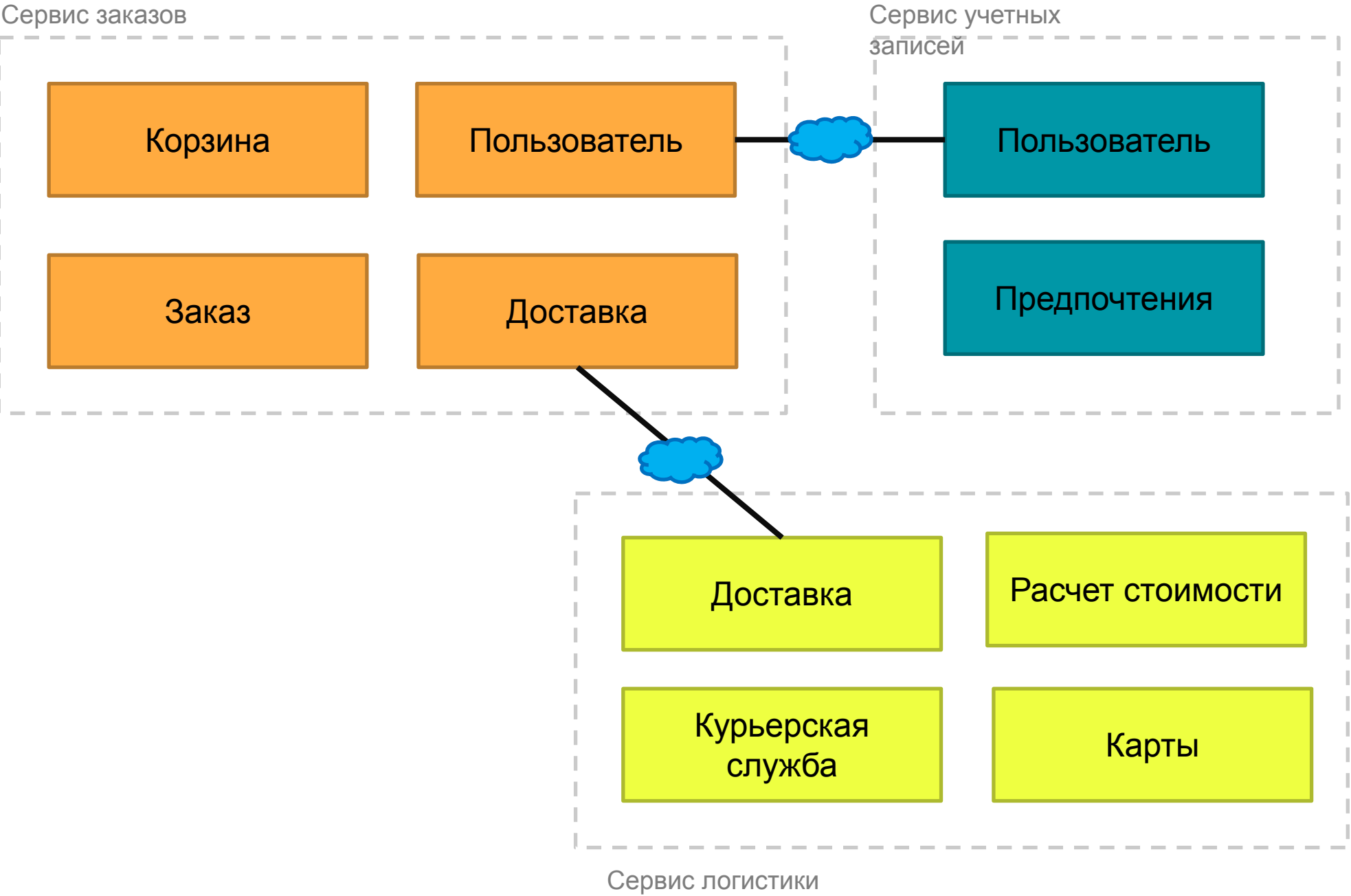
Сервис заказов



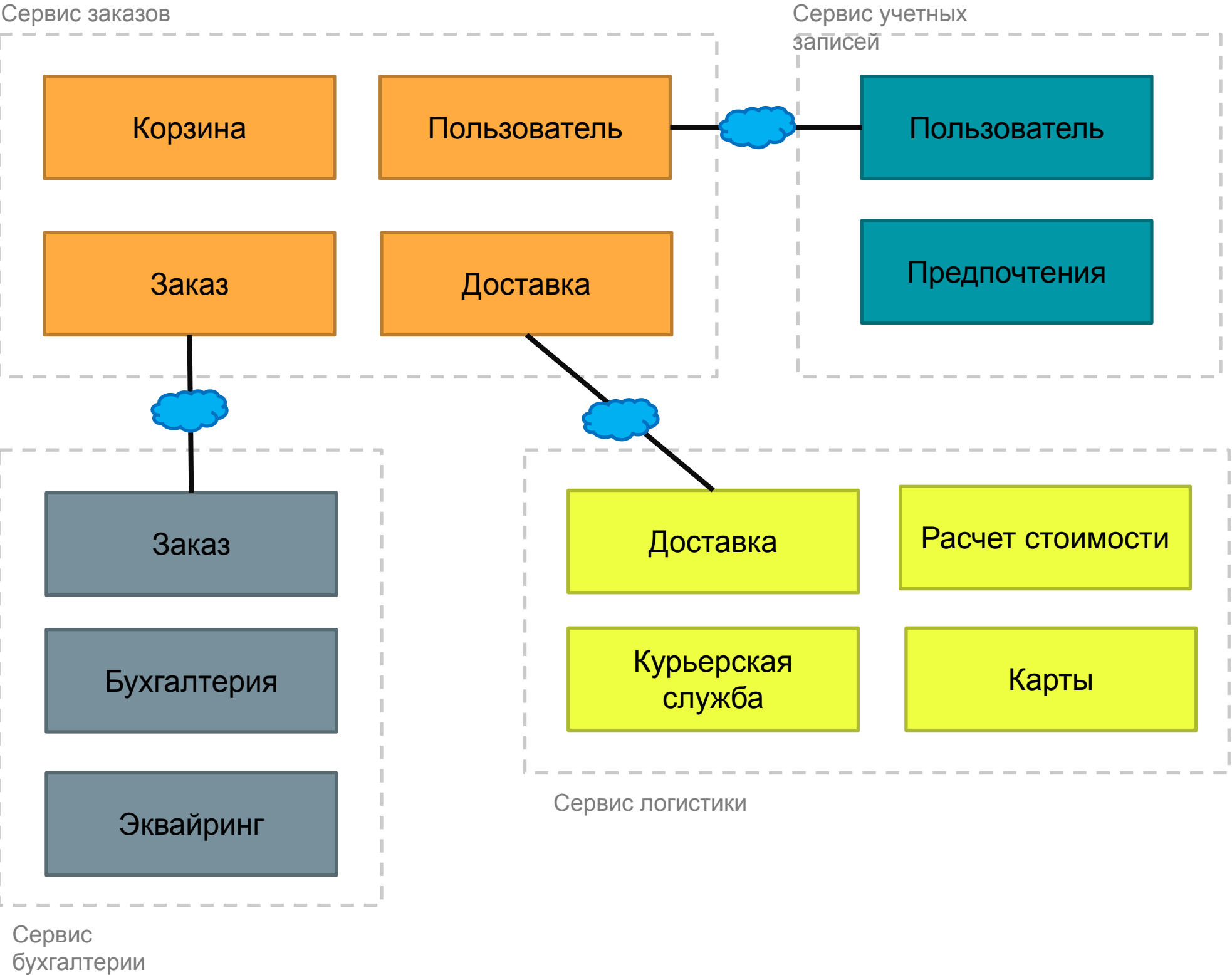
Сервис учетных записей



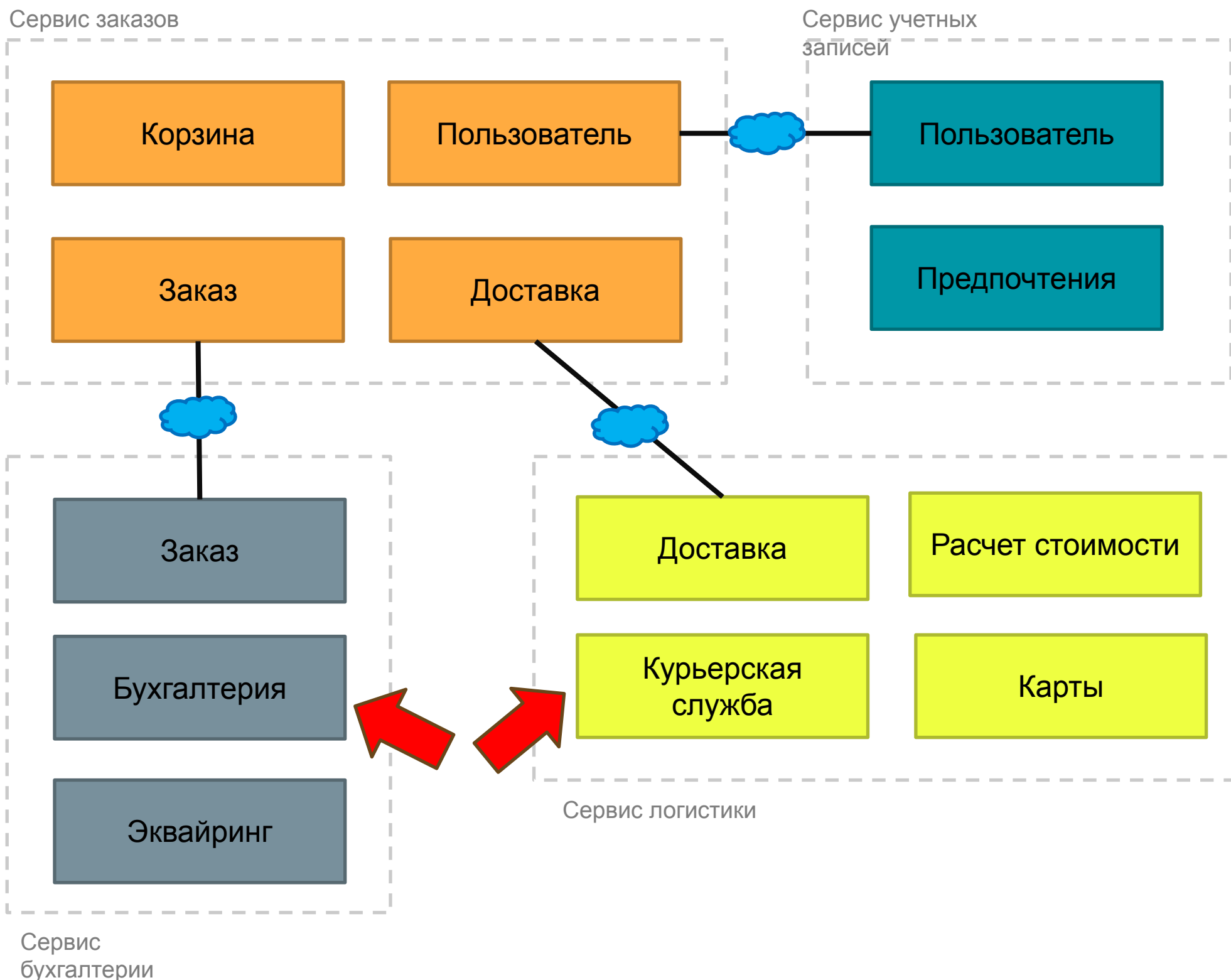
«Первое правило»



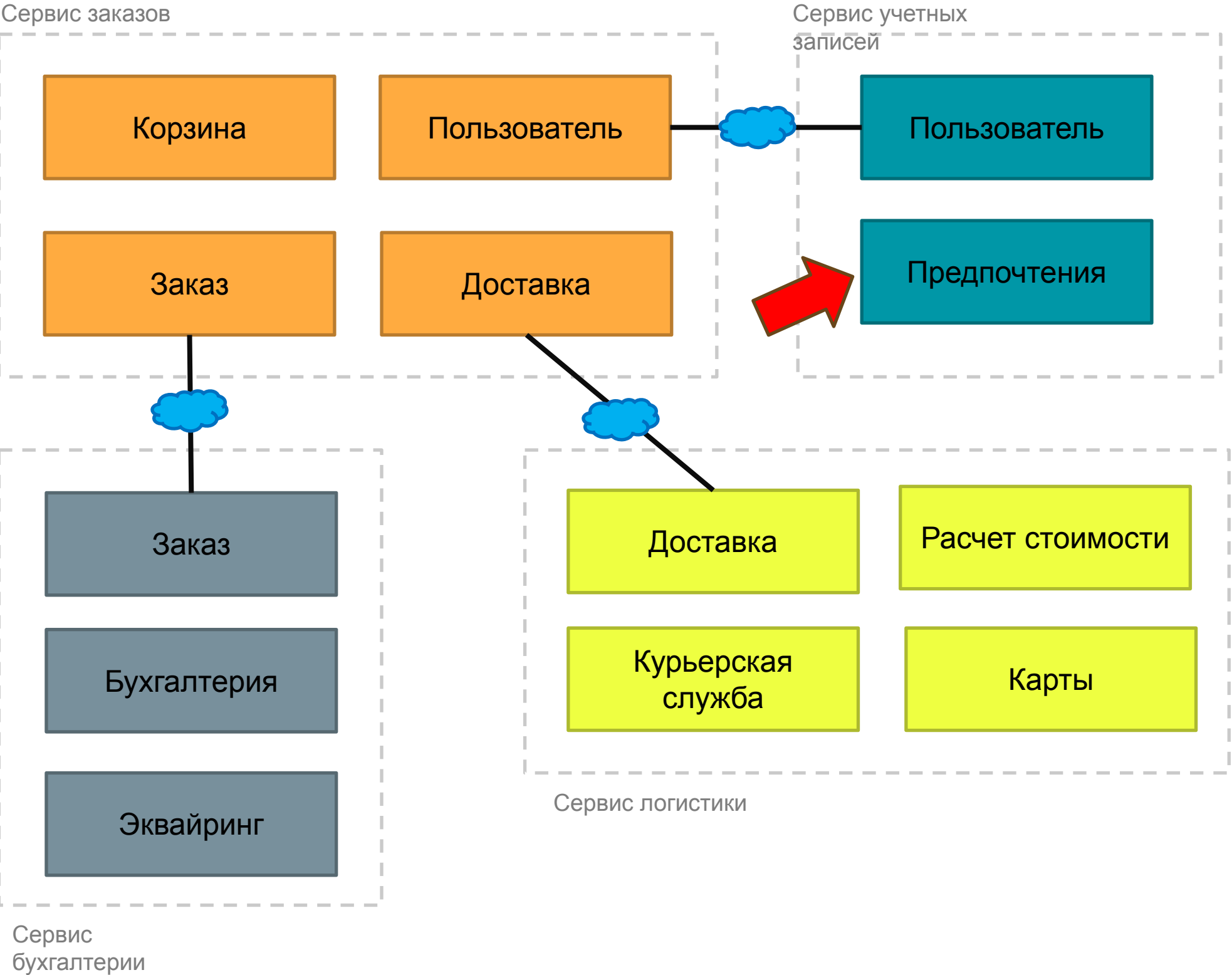
«Первое правило»



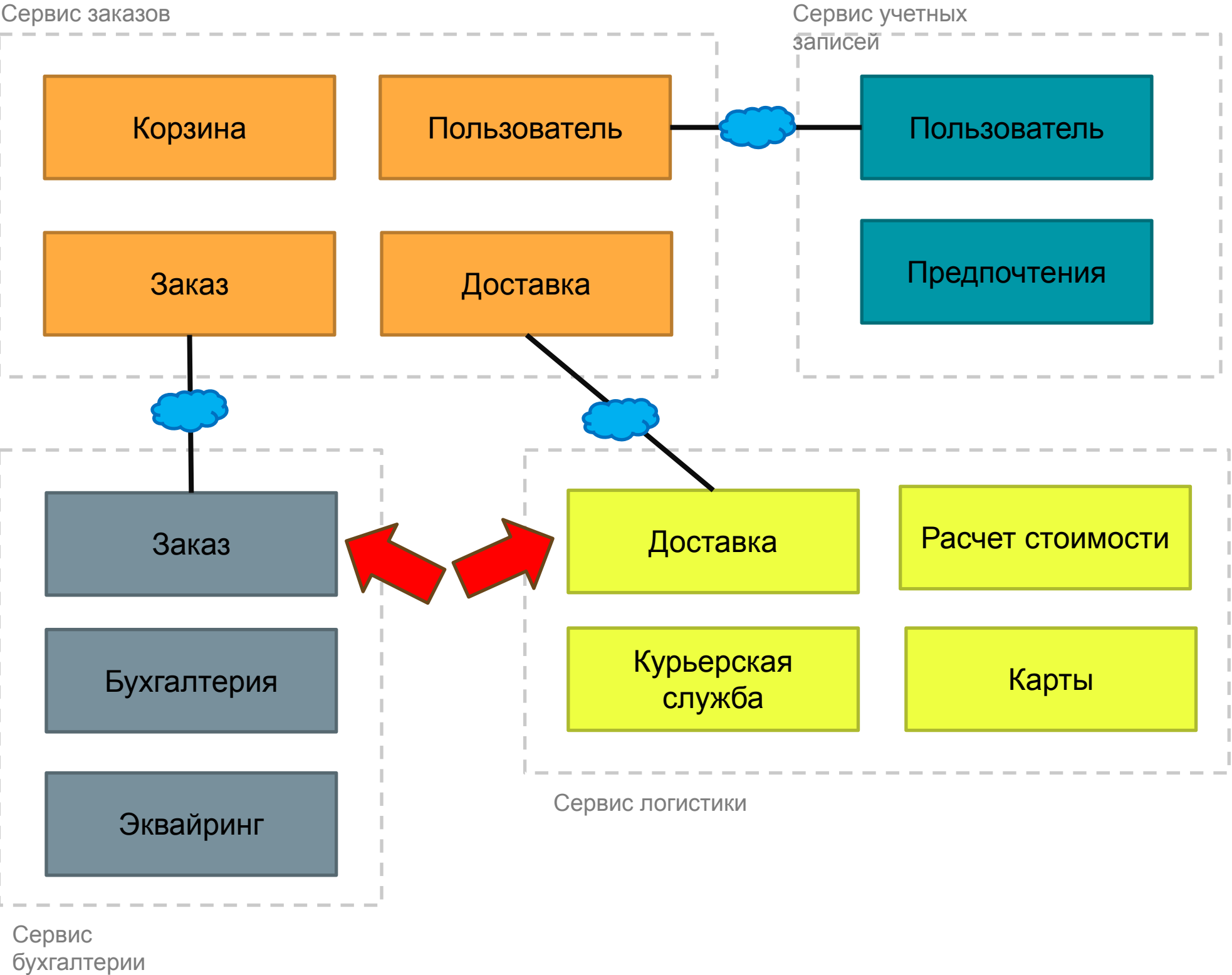
«Первое правило»



«Первое правило»



«Первое правило»



Выводы

- Не распределяйте сущности, которые участвуют в одном бизнес-процессе
- Это нормально иметь *разные представления* одной сущности в зависимости от задач сервиса
- Для реализации бизнес-логики в сервисах существуют и другие подходы, кроме DDD.

Что дальше?

- Рассмотрение шаблонов распределенной архитектуры.
- Начнем с понятия и принципов сервис-ориентированной архитектуры (SOA).