

# Технологии программирования

## Лекция 11 Архитектурные паттерны

Старичков Н.Ю.

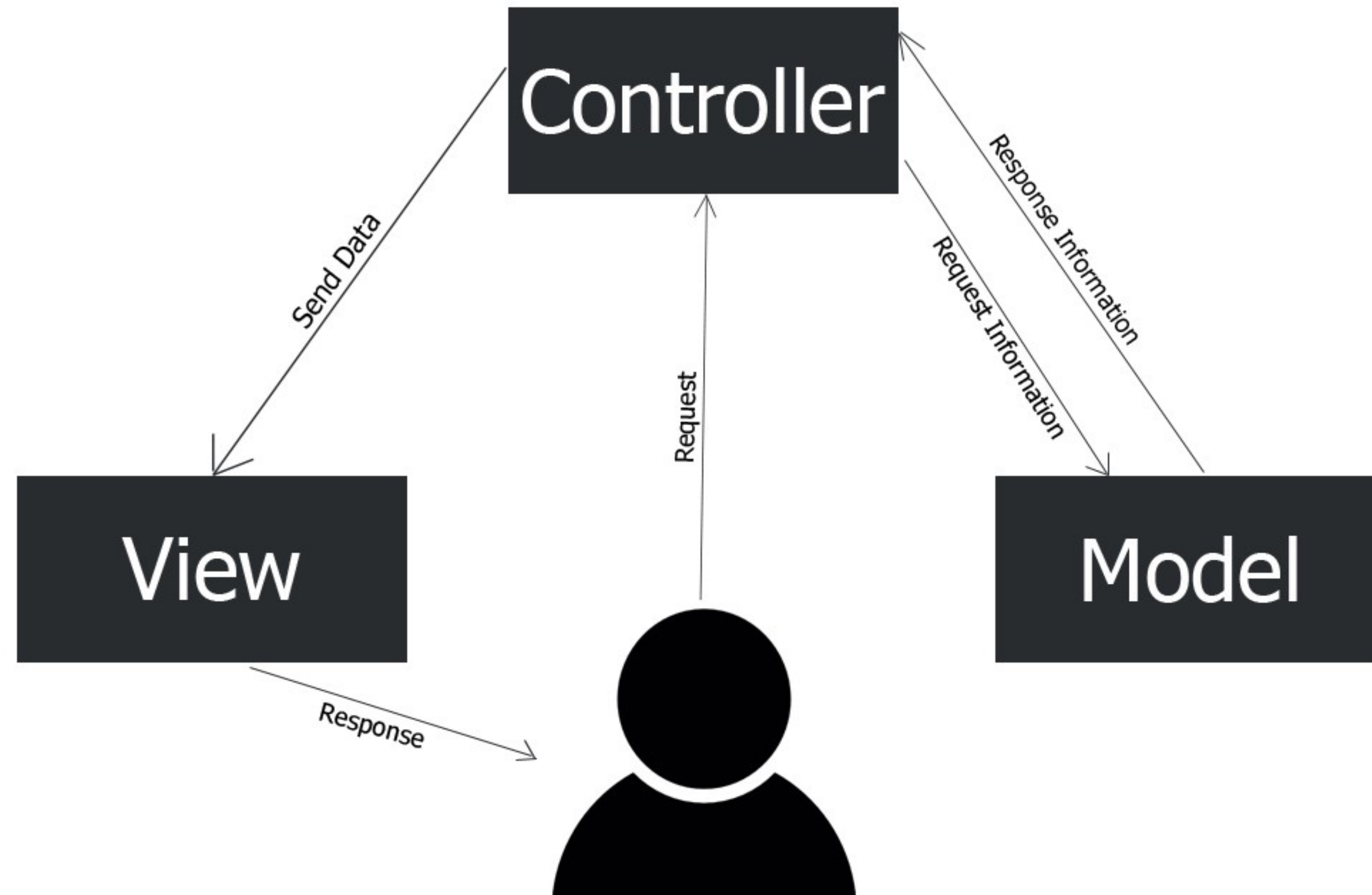
# Классификация архитектуры ПО

- Локальные
- Распределенные
  - Файл-серверные
  - Клиент-серверные
    - Двухзвенные
    - Трехзвенные
    - Многозвенные

# Локальные приложения

Общеизвестный архитектурный паттерн - MVC

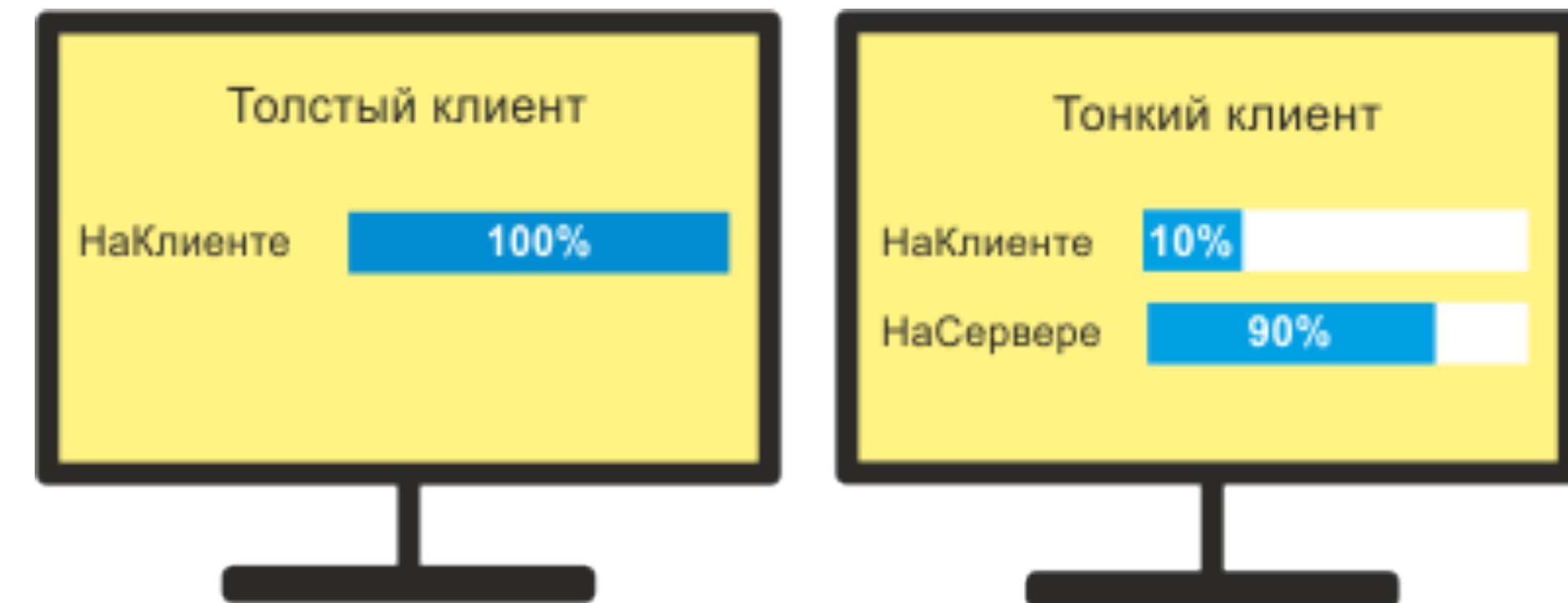
## Model-View-Controller



# Клиент-серверная архитектура

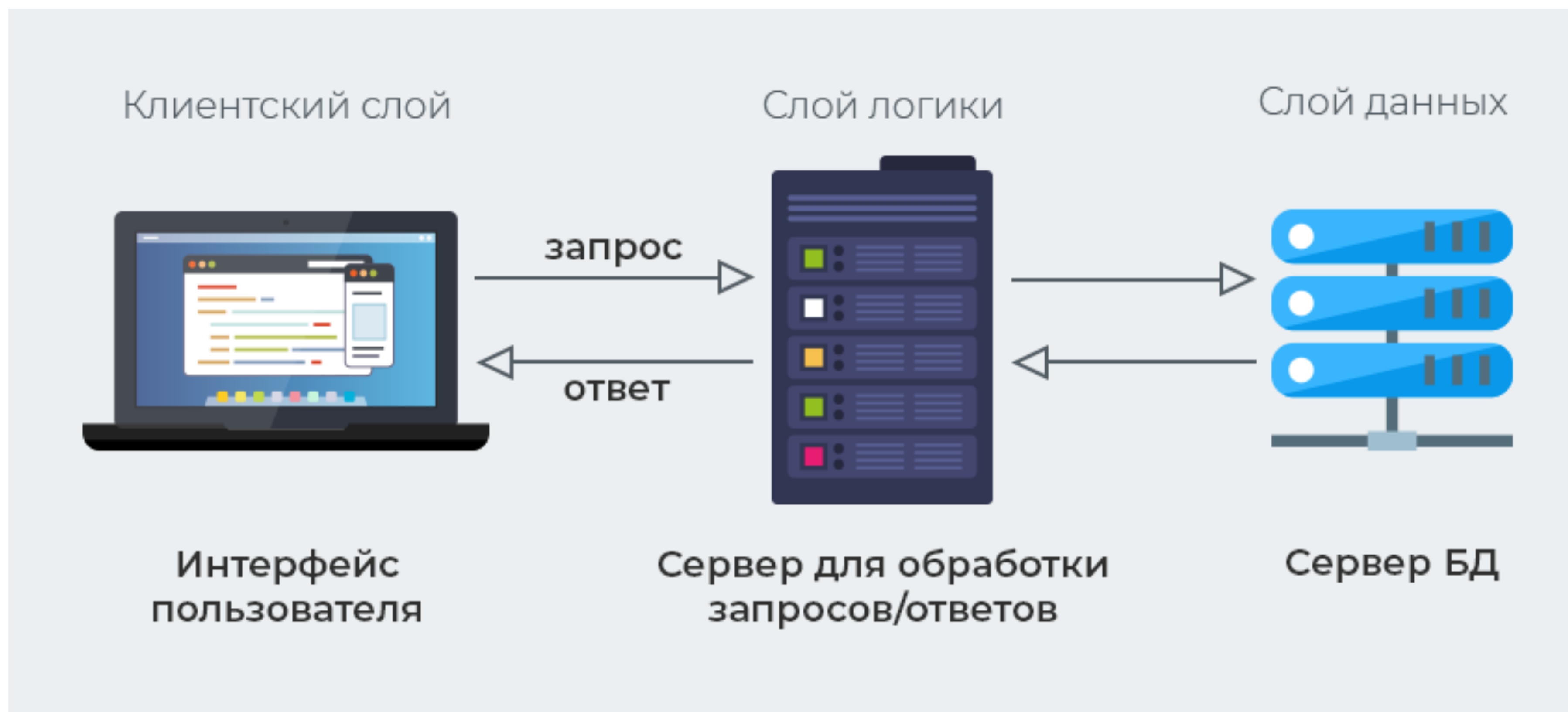
## Тонкий и толстый клиент

- **Тонкий клиент**
  - Обработка действий пользователя
  - Получение и представление информации с сервера
- **Толстый клиент**
  - Бизнес-логика
  - Хранение данных



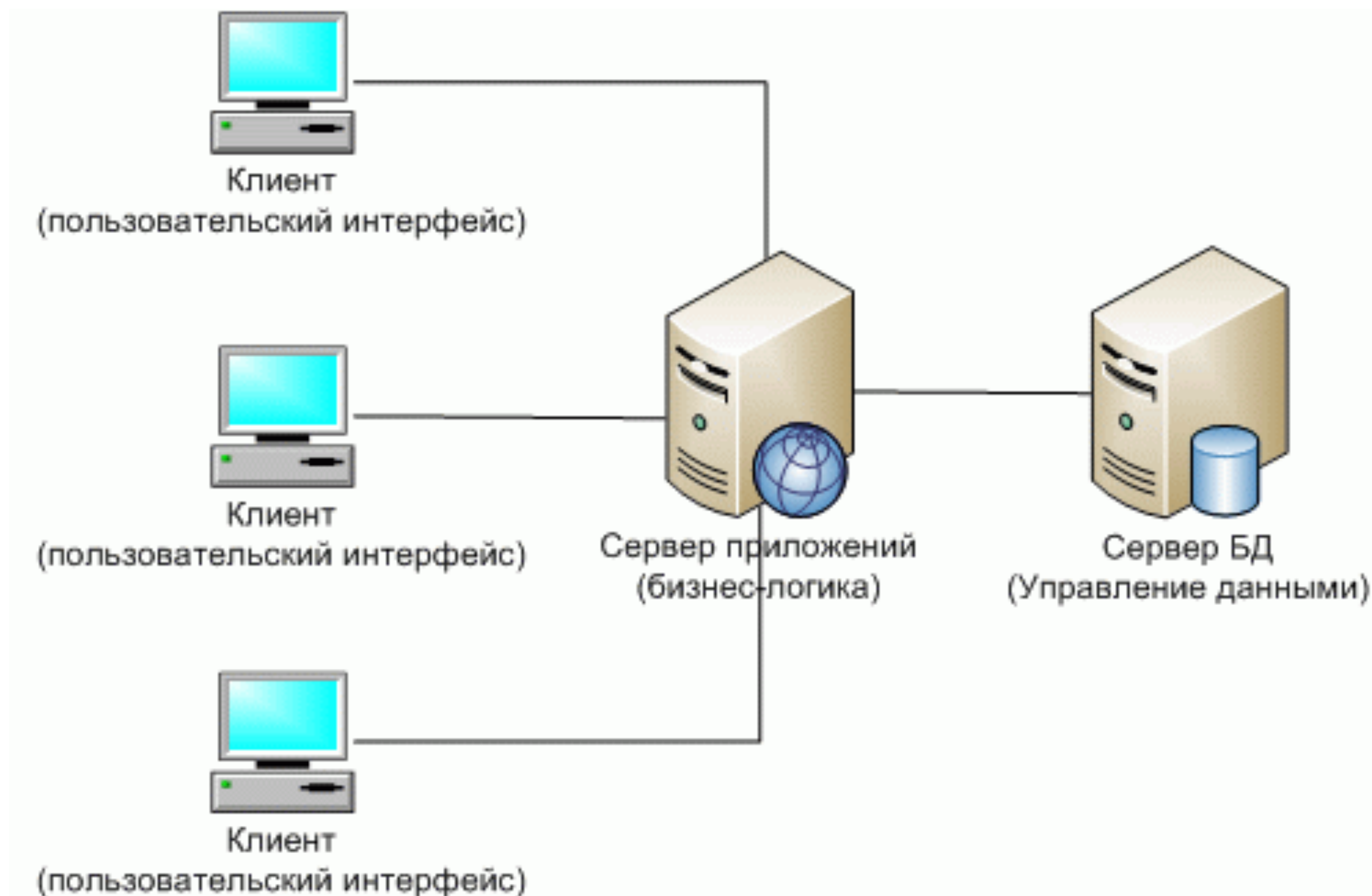
# Клиент-серверная архитектура

## Трехзвенная архитектура



# Клиент-серверная архитектура

## Трехзвенная архитектура



# Highload

# Как считать нагрузку?

- Много данных
- Много пользователей
- Много хитов / запросов

**Реально лучше мерить в запросах + запросы/данные**



# Тип проекта и влияние на нагрузку

**От типа проекта зависит:**

- Количество данных для хранения
- Сложность выполнения запроса
- Количество данных, требуемых для выполнения запроса
- Количество пользователей
- Количество действий и скорость действий каждого пользователя
- Ожидаемое время реакции системы

# Ресурсы для обеспечения производительности

- Ресурсы для вычислений
- Ресурсы для хранения данных
  - Вопрос не только в объеме, но и в скорости доступа
  - Разные сценарии доступа к данным
    - Ограниченное / последовательное / случайное
    - Read-only / запись-чтение-поиск
- Ресурсы для передачи данных
  - сервер  $\longleftrightarrow$  пользователь
  - сервер  $\longleftrightarrow$  сервер

# Масштабирование

- Горизонтальное
- Вертикальное
- Функциональное разбиение
- Шардинг

# Горизонтальное масштабирование

- Увеличение производительности системы за счет подключения дополнительного оборудования (серверов)

# Горизонтальное масштабирование

- Увеличение производительности системы за счет подключения дополнительного оборудования (серверов)
- Как быть с данными?
- Как быть с синхронизацией данных?

# Вертикальное масштабирование

- Увеличение производительности системы за счет увеличения мощности существующего оборудования (серверов)

# Вертикальное масштабирование

- Увеличение производительности системы за счет увеличения мощности существующего оборудования (серверов)
- Проблема в том, что до бесконечности увеличивать мощность все равно не получится

# Функциональное разбиение

- Разные функциональные модули развернуты и работают на разных серверах



# Функциональное разбиение

- Разные функциональные модули развернуты и работают на разных серверах
- Все равно рано или поздно упруемся в ограничение мощности сервера

# Шардинг

- Разбиение ИС на «области», где каждая область / группа областей размещается на своем оборудовании (сервере)

# Шардинг

- Разбиение ИС на «области», где каждая область / группа областей размещается на своем оборудовании (сервере)
- Как правильно разбить на области? А если есть общие данные?
- Как обеспечить синхронизацию данных в областях, если нужно?
- А если есть интеграции с внешними системами, какие именно данные?

# Типичная архитектура веб-сервиса

- **Фронтенд** - легкое решение, «тонкий клиент»
- **Бекенд** - собственно, тяжелый сервер (горизонтальное/вертикальное/функц.разбиение/шардинг)
- **База данных** - горизонтальное (репликации), вертикальное, шардинг

# Типичная архитектура нагруженной ИС

- **АРМ** - «тонкий»(в том числе, порталные решения)/«толстый» клиент
- **Бекенд** - сервер / кластер серверов. Горизонтальное, вертикальное, шардинг(по областям), функц.разбиение (в том числе, микросервисы) + комбинация подходов
- **База данных** - разделение баз, распределенные базы + горизонтальное/вертикальное масштабирование

>>: tbc...