Распределенные вычислительные системы

Модель «Клиент-Сервер»

Модель взаимодействия клиент-сервер

Проблема: централизация vs децентрализация

- Это одна из старейших проблем в IT
- Пример:
  - *King J.L. Centralized versus decentralized computing: organizational considerations and management options // ACM Computing Surveys. Vol. 15, Issue 4. 1983. P. 319-349.*

До 70-х годов: централизованная модель

- До середины 70-х годов прошлого века доминировала централизованная модель:
  - Высокая стоимость телекоммуникационного оборудования
  - Слабая мощность вычислительных систем

80-е – 90-е: мейнфреймы

- Появление систем разделения времени и удаленных терминалов - предпосылка возникновения клиент-серверной архитектуры.
- Ресурсы мейнфреймов предоставлялись конечным пользователям посредством удаленного соединения.
- Дальнейшее развитие телекоммуникационных систем и появление персональных компьютеров дало толчок развитию клиент-серверной парадигме обработки данных

Клиент-серверная архитектура

- Согласно парадигме клиент-серверной архитектуры:
  - один или несколько клиентов и один или несколько серверов
  - совместно с базовой операционной системой
  - и средой взаимодействия
  - образуют единую систему, обеспечивающую распределенные вычисления, анализ и представление данных

Применение клент-серверной модели

- Процесс разработки распределенных приложений достаточно сложен и одной из наиболее важных задач является решение того, как
  - *функциональность приложения должна быть распределена между клиентской и серверной частью.*

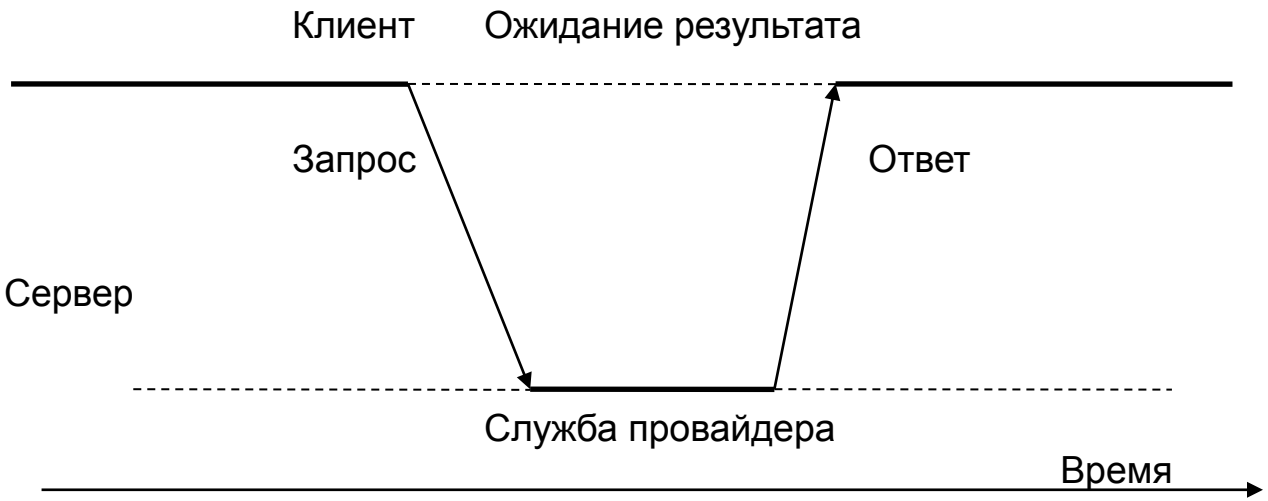
Алгоритм работы клиент-серверной архитектуры

В классическом случае данная схема функционирует следующим образом:

- клиент формирует и посылает запрос на сервер;
- сервер производит необходимые манипуляции с данными, формирует результат и передаёт его клиенту;
- клиент получает результат, отображает его на устройстве вывода и ждет дальнейших действий пользователя.

Цикл повторяется, пока пользователь не закончит работу с сервером.

Обобщение взаимодействия «Клиент-Сервер»



Надежность сети

- Если базовая сеть надежна как локальная сеть, взаимодействие может быть реализовано простым протоколом, без установления соединения (выигрыш эффективности)
- Что, если сообщения пропадают? Если теряется ответ?
  - Отправлять повторно?
  - Надеяться на лучшее?
- А если это операция перевода 10 000$ с одного счета на другой?

Надежные протоколы соединения

- В прикладных протоколах используется TCP/IP:
  - До посылки запроса клиент должен установить соединение
  - Сервер использует то же соединение для посылки ответа

Уровни клиент-серверной архитектуры

Уровни клиент-серверной архитектуры

- Сегодня выделяют 3 основных уровня клиент-серверной архитектуры:
  - Уровень представления (пользовательского интерфейса)
  - Уровень бизнес-логики (обработки)
  - Уровень данных

Уровень представления

- Обычно реализуется на клиентах
- Организует методы взаимодействия с приложением
- Простейший вариант:
  - символьный дисплей (терминал) к мейнфрейму

Уровень бизнес-логики

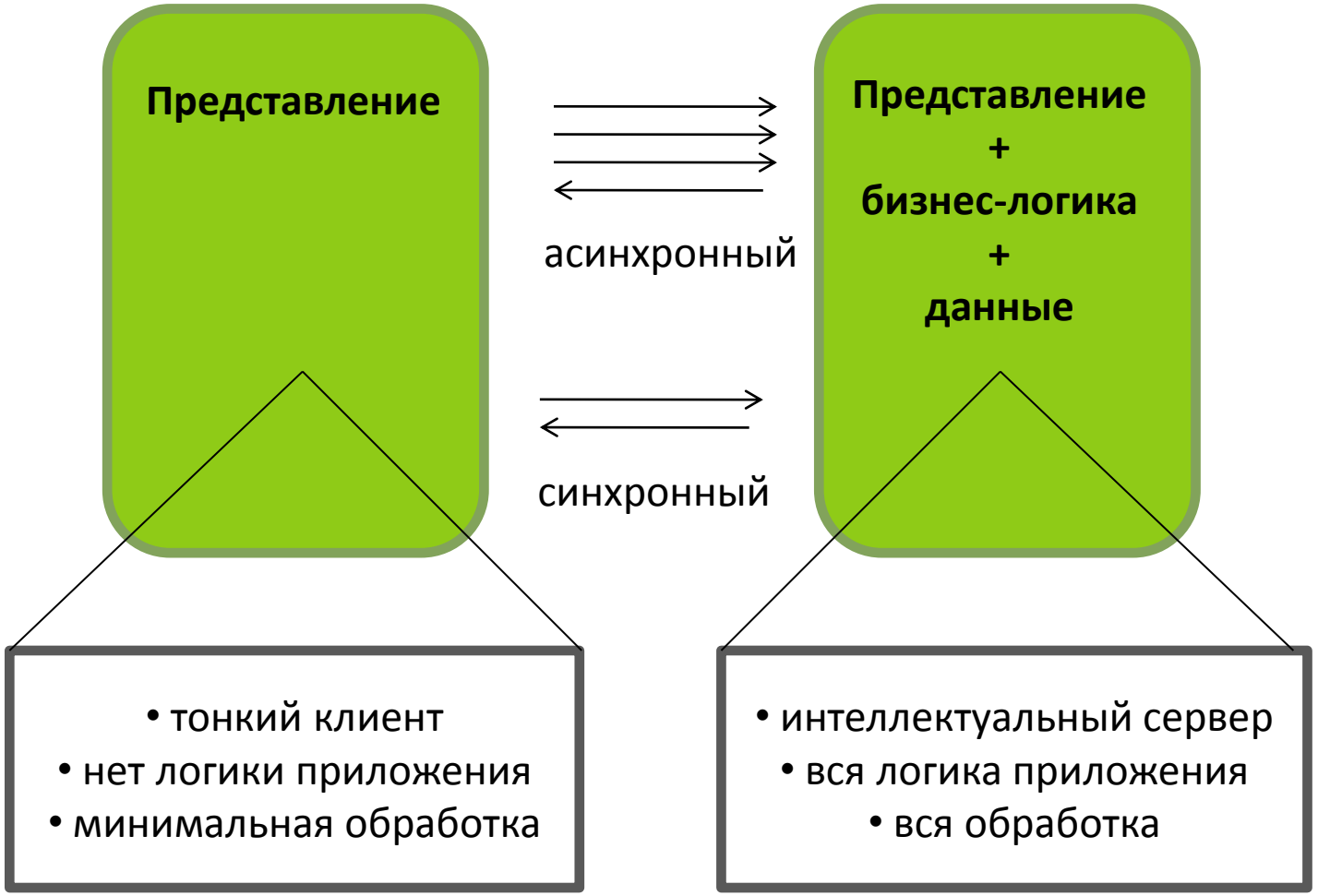
- Бизнес-логика – это совокупность правил, принципов и зависимостей поведения объектов предметной области системы.
- Синоним: логика предметной области (Domain Logic).
- Пример:
  - формула расчета зарплаты + налоги;
  - оценка качества обучения на основе оценок студента;
  - отказ от отеля при отмене рейса авиакомпанией.

Уровень данных

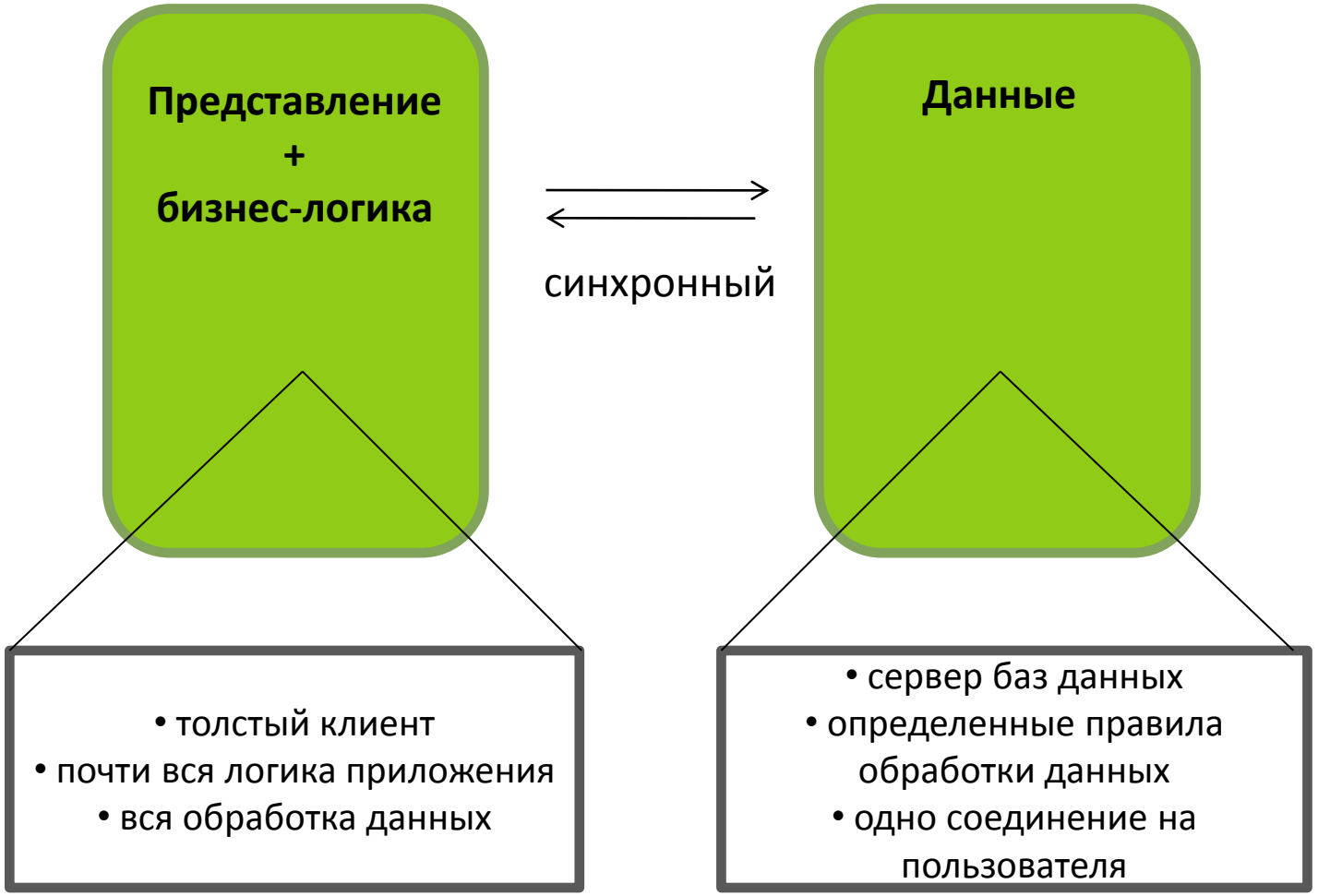
- Программы, которые предоставляют данные обрабатывающим приложениям
- требование СОХРАННОСТИ: когда приложение не работает, данные должны сохранятся в определенном месте;
- требование ЦЕЛОСТНОСТИ: метаданные (описание таблиц, ограничения и т.п.) должны исполнятся и проверяться на этом уровне
- Обычно реализуется реляционной БД

История и типы клиент-серверной архитектуры

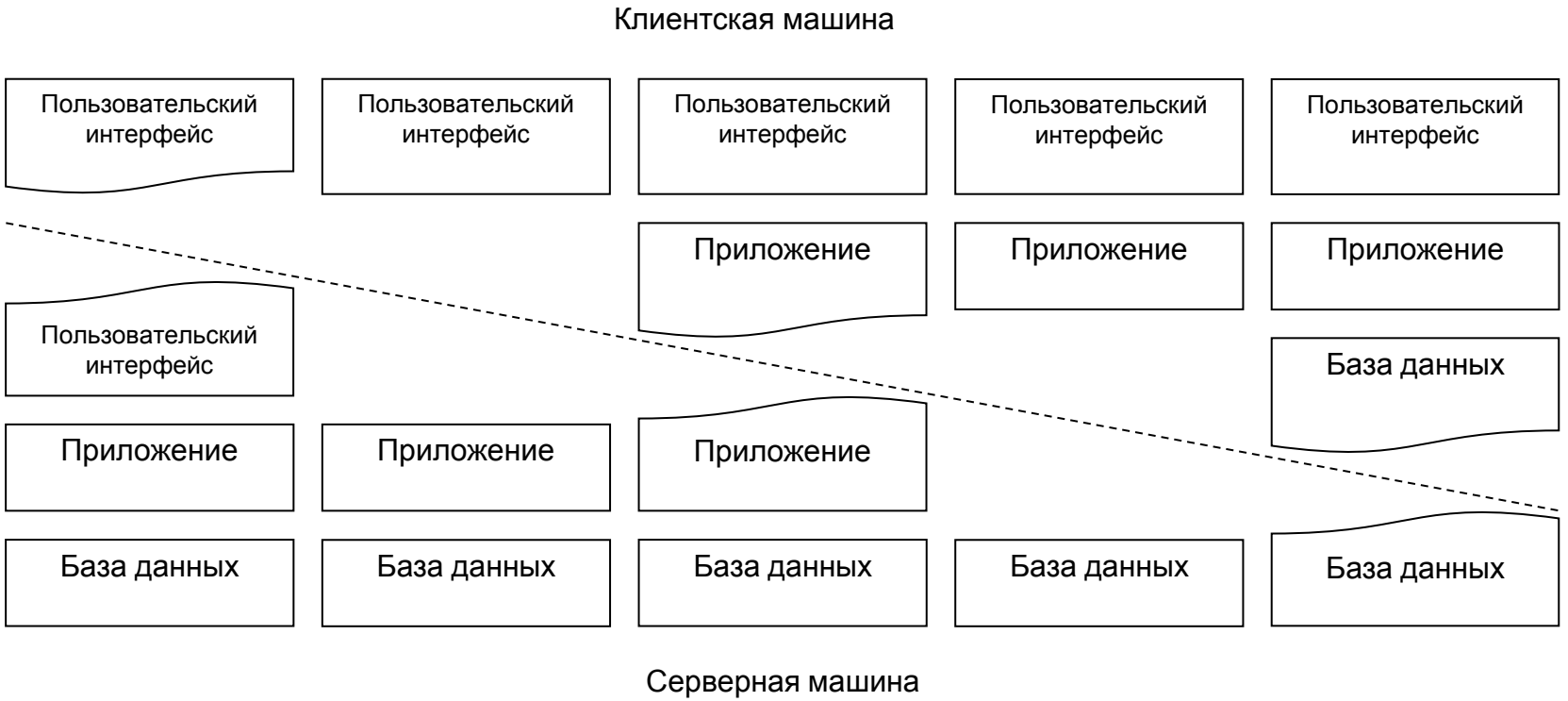
Однозвенная клиент-серверная архиетктура



Двузвенная архитектура



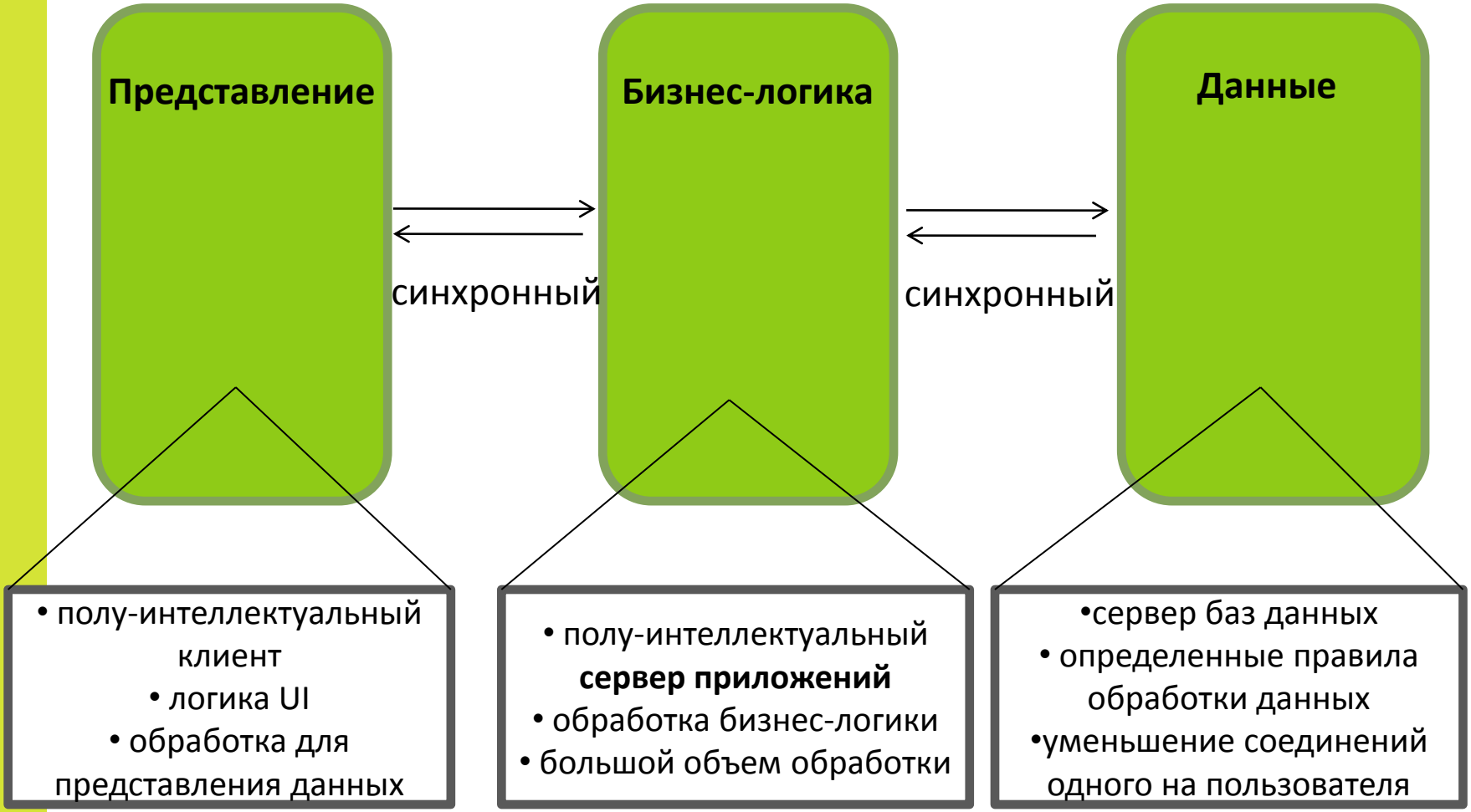
Альтернативные варианты двузвенной архитектуры



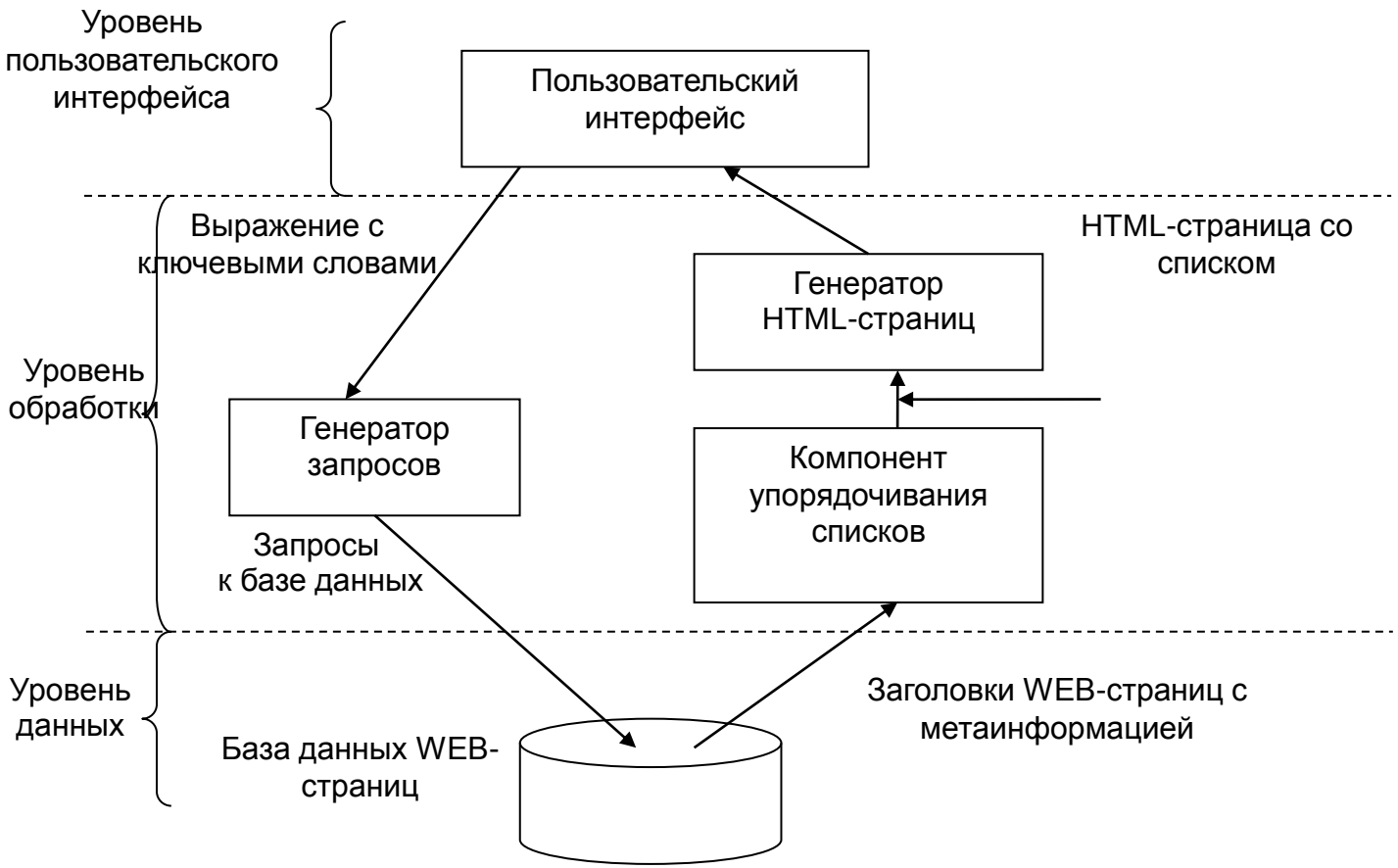
Минусы двузвенной архитектуры

- Чрезвычайные затраты на поддержание рабочих станций, которые должны обрабатывать бизнес-логику
- Чрезвычайная сложность обновления приложения при незначительном изменении бизнес-логики (необходимо переустановить все клиенты)
- Каждая рабочая станция – уникальный набор ПО, который может конфликтовать с клиентом и влиять на его работу

Трехзвенная архитектура



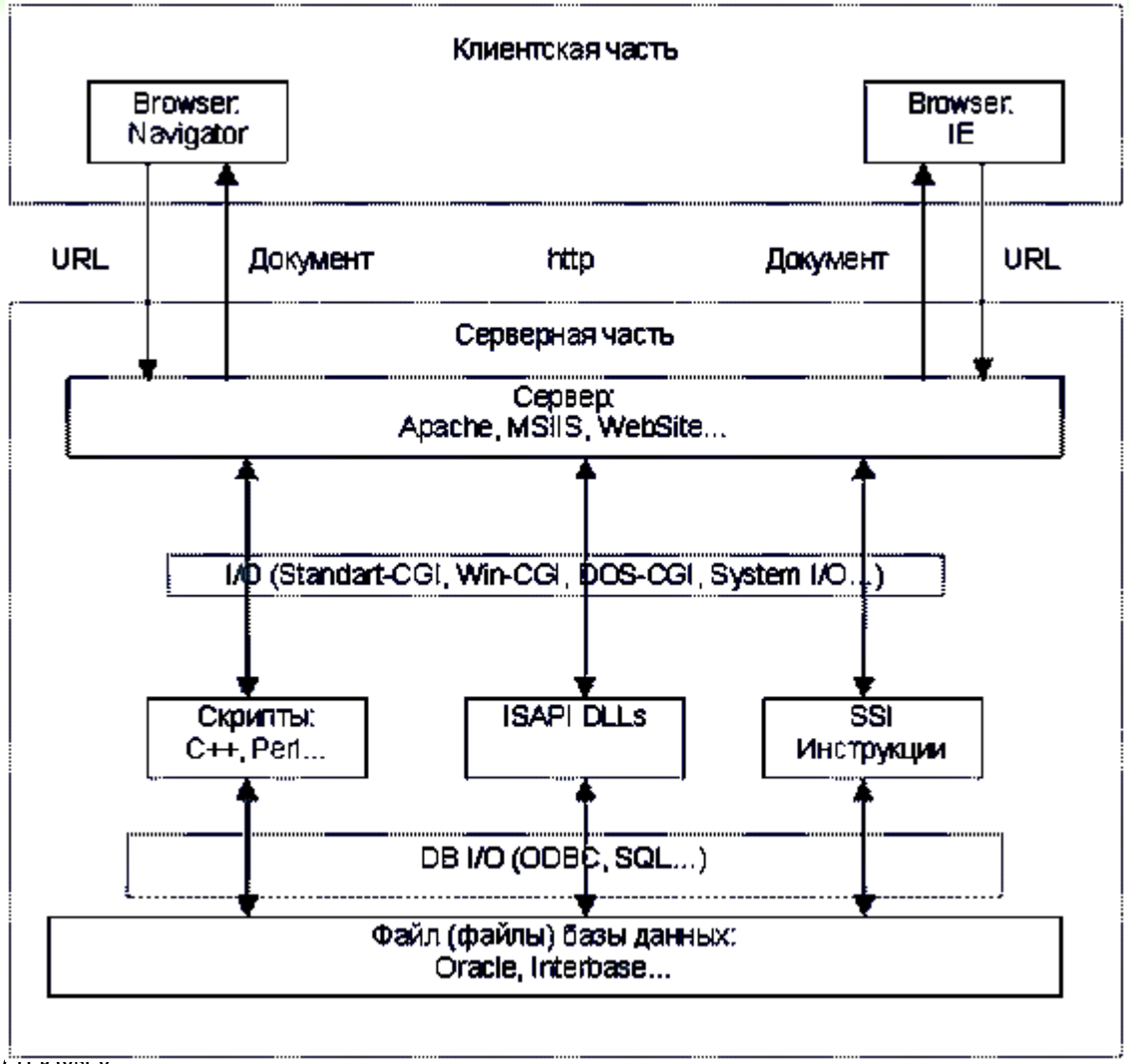
Пример трехзвенной архитектуры – поисковая машина



Современный пример многозвенной архитектуры

- 1. Браузер клиента->
  - 2. Сервер IIS->
    - 3. Исполняющая среда ASP.NET 2.0->
      - 4. Провайдер данных ADO.NET 2.0 ->
        - 5. Сервер MySQL ->
      - 6. Провайдер данных ADO.NET 2.0 ->
    - 7. Исполняющая среда ASP.NET 2.0->
  - 8. Сервер IIS ->
- 9. Браузер клиента

Клиент-сервер в HTTP



Горизонтальное распределение

Методы разбиения на уровни

- Вертикальное деление:
  - UI
  - бизнес-логика
  - данные
- Горизонтальное деление:
  - Клиент или сервер может содержать физически разделенные части логически однородного модуля

Пример горизонтального распределения

Встреча и обработка входящих запросов

Реплицированные WEB-серверы, содержащие одни и те же WEB-страницы