Многоуровневая Архитектура

Подготовил: Казаков Александр

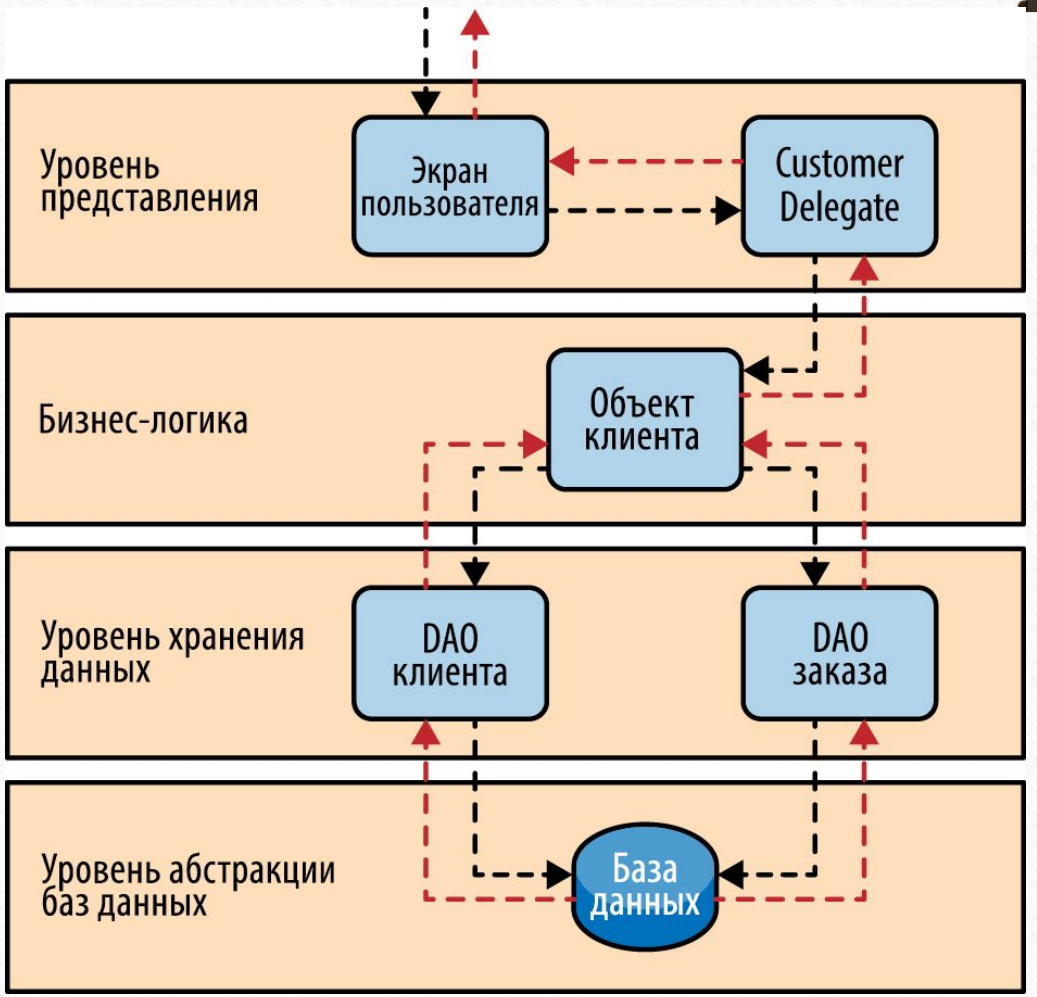
Группа 732

многоуровневая архитектура

- В программной инженерии **многоуровневая архитектура** или **многослойная архитектура** — клиент-серверная архитектура, в которой разделяются функции представления, обработки и хранения данных. Наиболее распространённой разновидностью многоуровневой архитектуры является **трёхуровневая архитектура**.

Распространённые слои

- логически разделённых на слои архитектурах информационных систем наиболее часто встречаются следующие четыре слоя:
- **Слой представления** (слой UI, UIL, пользовательский интерфе́йс, уровень представления в многоуровневой архитектуре)
- **Слой приложения** (сервисный слой, сервисный уровень или GRASP уровень управления)
- **Слой бизнес-логики** (слой предметной области, BLL, доменный слой)
- **Слой доступа к данным** (слой хранения данных, DAL, слой инфраструктуры; логирование, сетевые взаимодействия и другие сервисы, требующиеся для поддержания конкретного слоя бизнес-логики)


Уровень представления
Экран пользователя
Customer Delegate
Бизнес-логика
Объект клиента
Уровень хранения данных
DAO клиента
DAO заказа
Уровень абстракции баз данных
База данных

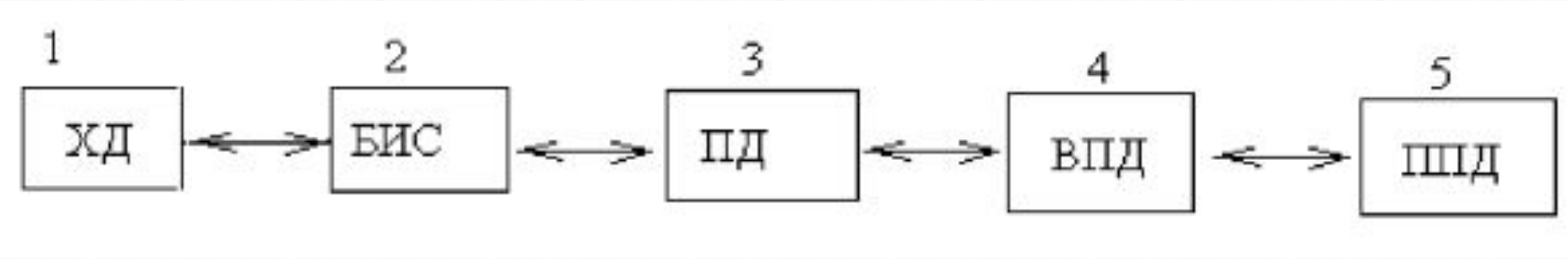

Запрос
Уровень представления
Уровень логики
Уровень данных
БД

Основные задачи, для решения которых применяется многоуровневый подход, обычно сводятся к следующим:

- повышение производительности системы за счет переноса части функциональности на аппаратно выделенные сервера приложений;
- повышение структурированности программных систем за счет реализации компонентов в виде независимых модулей (объектов), допускающих повторное использование; обеспечение возможности комплектации готовой системы из таких модулей;
- потребность в интеграции различных приложений в едином интерфейсе -- формирование доступа ко всем ресурсам и программным средствам через единую точку входа (портал).

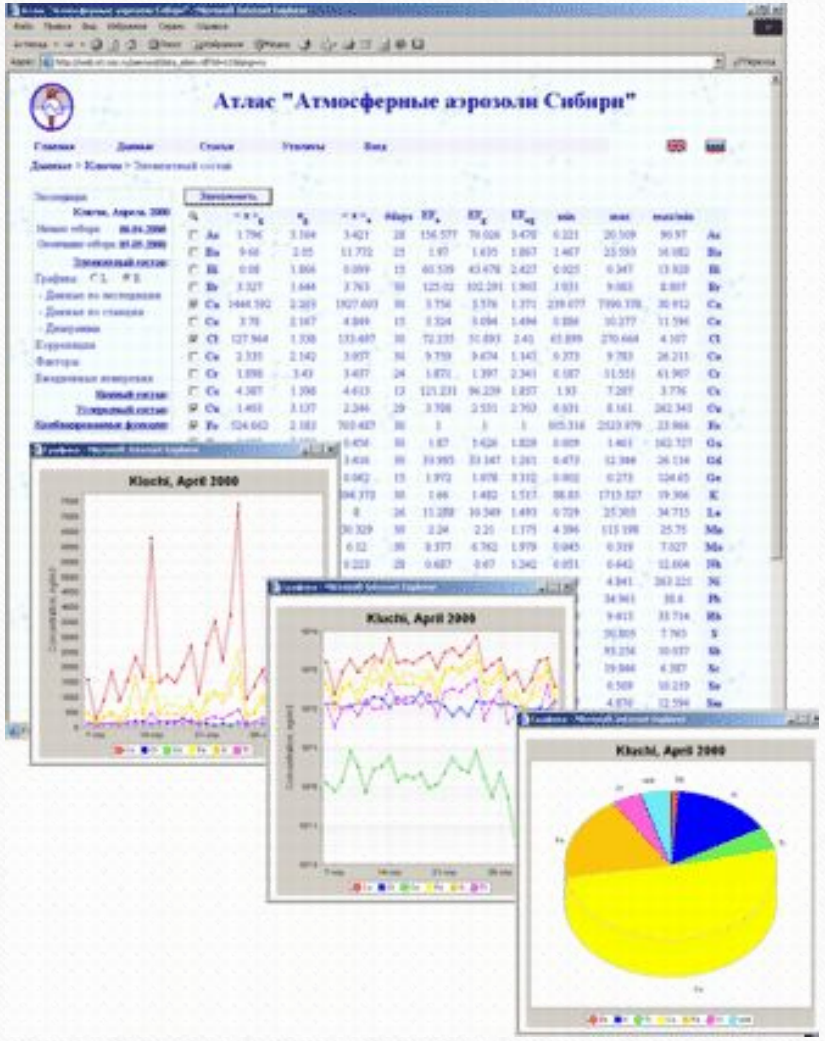
На каждом уровне реализовывается отдельный вид обработки данных:

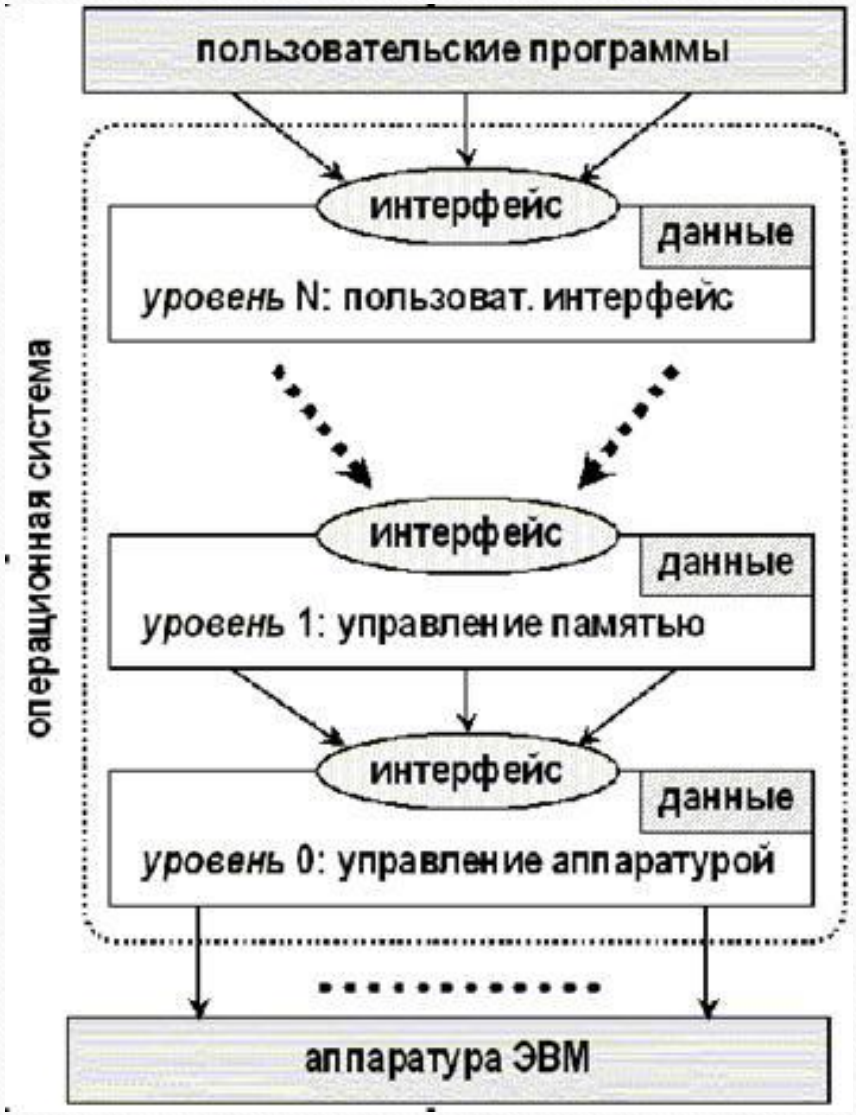
- хранилище данных (ХД) -- набор зарегистрированных баз данных, структура которых задана в системе регистрации данных;
- базовые информационные структуры (БИС), объединение которых составляет содержание коллекций;
- провайдер данных (ПД) -- приложение, обеспечивающее обработку унифицированных именованных запросов к коллекция и формирование ``внутреннего представления документа" (ВПД);
- обработчик ВПД -- формирует унифицированные именованные запросы к коллекции и отбор информации в ВПД;
- формирование ``презентационного представления документа" (ППД) в соответствии с выбранным стилем -- приложение, которое осуществляет визуализацию документа в удобном для пользователя виде, а также пользовательский интерфейс, с которого вводятся параметры запроса.


1
ХД
2
БИС
3
ПД
4
ВПД
5
ШПД

Данная архитектура обеспечивает взаимодействие служб:

- публикации данных, поддержка и их аутентичности и качества;
- поиска и представления информации;
- анализа распределенных данных.
- поддержку интероперабельности в глобальной программно-аппаратной инфраструктуре;
- поддержку диспетчеризации, включая идентификацию доступных ресурсов, статистика использования и загрузки ресурсов и пр.;
- поддержку системы безопасности и контроля доступа, в т.ч. гибкое регулирование объема прав и привилегий пользователей;
- поддержку использования данных в удаленных архивах (включая протоколы, которые необходимо использовать для работы с гетерогенными источниками данных, и библиотеки программных комплексов) и др.


Атлас "Атмосферные аэрозоли Сибири"
Kluchi, April 2000
Kluchi, April 2000
Kluchi, April 2000


пользовательские программы
интерфейс
данные
уровень N: пользоват. интерфейс
интерфейс
данные
уровень 1: управление памятью
интерфейс
данные
уровень 0: управление аппаратурой
операционная система
аппаратура ЭВМ

Плюсы:

- Каждый уровень этой архитектуры выполняет строго ограниченный набор функций (которые не повторяются от слоя к слою) и не знает о том, как устроены остальные уровни. Поэтому «содержимое» уровней можно изменять без риска глобальных конфликтов между слоями.

  В целом многоуровневые приложения настолько распространены, что для их разработки создаются специальные генераторы шаблонов. Например, LASG для Visual Studio предлагает несколько методов генерации кода, которые автоматизируют рутинные задачи и помогают выстраивать уровни приложения.

Недостатки:

- В программировании есть присказка, что любую проблему можно решить добавлением еще одного уровня абстракции. Однако такой подход в конечном счете может привести к плохой организации кода и запутать разработчиков.

  Отсюда вытекает еще одна проблема — низкая скорость работы. Очень много информации начинает бесполезно проходить от слоя к слою, не используя бизнес-логику. Иногда эту проблему называют sinkhole anti-pattern — шаблон проектирования, когда количество бесполезных операций начинает преобладать над полезными.

  Поиск багов в многоуровневых системах также может быть затруднен. Прежде чем попасть в базу данных, информация проходит через все уровни (так как БД является конечным компонентом). Если по какой-то причине эта информация повреждается (или теряется по пути), то для поиска ошибки приходится анализировать каждый уровень по отдельности.

СПАСИБО ЗА ПРОСМОТР!