Архитектура клиент-серверных приложений. Тонкий и толстый клиент

Многослойная организация приложения как основа клиент-серверной архитектуры

Термин «клиент – серверная архитектура», если не оговорены подробности, сообщает нам не так много об организации системы: имеется один или несколько типов клиентских приложений, которые обслуживаются серверной компонентой. Логическая структура такой системы практически эквивалентна структуре обычной грамотно спроектированной локальной программы: она содержит несколько функционально-ориентированных слоев. Каждый слой реализуется системой взаимодействующих классов и связан с соседними программными интерфейсами. Вот так выгладит типовая структура приложения в документе «Руководство MicroSoft по проектированию архитектуры приложений» [14].

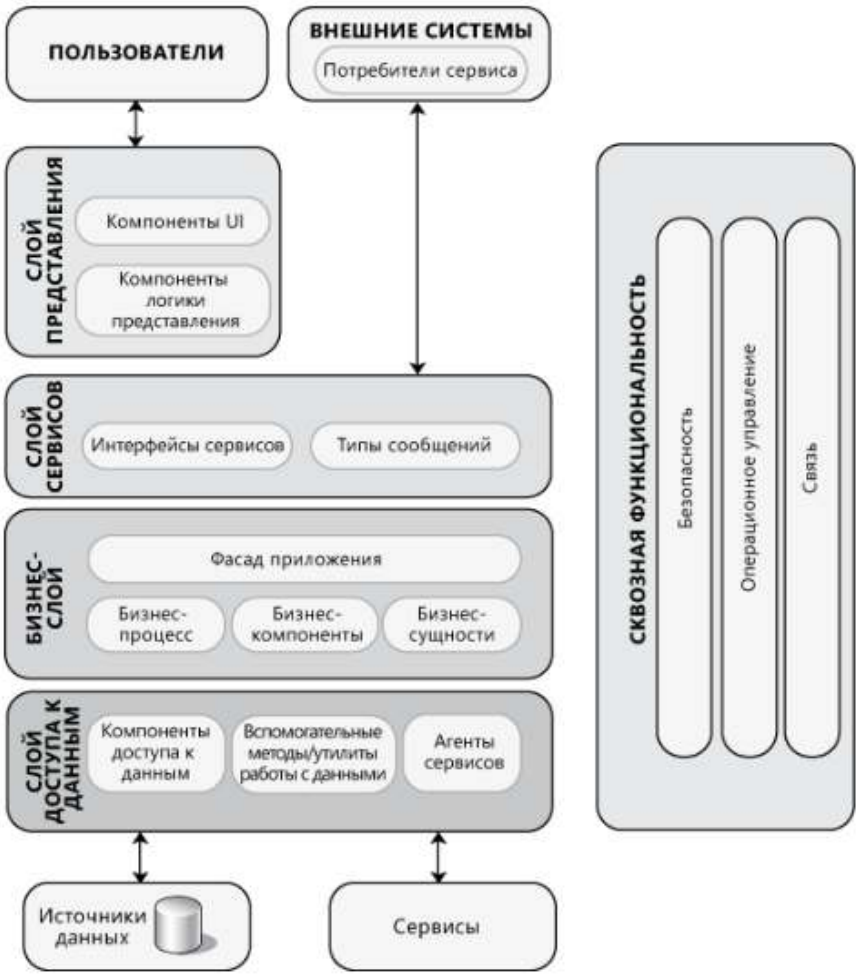

рис.1.2 Клиент-серверная архитектура от MicroSoft.

В приведенной схеме выделены следующие функционально-ориентированные слои:

- **представление (вид, View)** - компоненты и процессы отображения и взаимодействия с пользователем или внешней средой. Первое правило «хорошего тона» - код взаимодействия с пользователем (отображение, ввод/вывод) не должен находиться в компонентах, отвечающих за представление и обработку данных в программе;

- **бизнес-модель –** любая информационная система должна содержать адекватное представление тех внешних физических объектов и их связей, с которыми она взаимодействует или управляет. Второе правило «хорошего тона» - это представление должно быть выполнено в отдельном слое в соответствии с технологией ООП в виде бизнес-объектов (бизнес-сущностей) и связей между ними, а также классов, реализующих их поведение (бизнес-процессов);
- **доступ к данным –** объекты бизнес-модели (бизнес-сущности и бизнес-компоненты, а также связи между ними) хранятся в таблицах БД, из которых производится их сборка/разборка. Возможно также хранение (сериализация) бизнес-объектов в двоичных или текстовых (XML,JSON) форматах;
- если серверная компонента является универсальной, то она обрамляется слоем **сервиса,** который обеспечивает множественный доступ клиентов любого типа, как правило, к функционалу бизнес-слоя. Т.к. такой сервис является открытым, то он предусматривает собственную систему авторизации и защиты;
- в системе также реализуется **сквозная функциональность,** т.е. средства, необходимые для поддержки функционирования на всех уровнях приложения, например, сбои и восстановление, логирование исключений, кэширование данных, синхронизация и т.п..

В клиент-серверной архитектуре разделение компонент между клиентом и сервером делается, как правило, на границе слоев, а программный интерфейс заменяется протоколом.

Приведенная структура соответствует текущим реалиям типовых клиент-серверных приложений. Но, если рассмотреть структуру приложения в более широких границах, то появляются дополнительные слои и подслои, на основе которых можно описать дополнительные варианты организации клиент-серверных приложений:

- **слой 0**: примитивные элементы интерфейса пользователя и устройства, его обеспечивающие: графический экран, отдельные окна приложений, пискели, графические примитивы, события, связанные с нажатием/отпусканием клавиш, кнопок мыши, различных датчиков, движков, сенсоров и т.п.;
- **слой 1**: графические объекты и управляющие элементы оконного приложения. Сюда входят такие компоненты, как меню, кнопки, списки, текстовые поля, а также события, в них происходящие (выбор пунктов меню, нажатие кнопок, выбор элемента списка, потеря и получение фокуса, клик мышью по элементу управления. Этот слой реализуется библиотекой оконных классов;
- **слой 2 –** бизнес-логика приложения. В свою очередь может состоять из подслоев, реализующих следующие функции:
  - **2.**1. – представление (View), в которое включаются все элементы реализации интерфейса пользователя;
  - **2.**2. – логика бизнес-модели - описание «долгосрочного» ее поведения (Controller);
  - **2.**3. – объекты бизнес-модели и методы их преобразования, не связанные с долгосрочным поведением (Model);
  - **2.**4. – средства сборки объектов бизнес-модели из табличных объектов БД;
- **слой 3 –** база данных (СУБД). В свою очередь, может состоять из 3 подслоев:
  - **3.1.** табличные объекты БД, DAO (data access objects) – объекты доступа к данным;

- **3.**2. библиотека программного доступа к БД;
- **3.**3. сама программная компонента БД (СУБД).

- **слой 4 –** данные на физическом носителе (файлы).

При реализации программы в виде распределенной системы «клиент-сервер» часть слоев остается в клиентском приложении, часть переносится на сервер. В зависимости от их количества в клиенте говорят о «**тонком**» и «**толстом**» клиенте (в терминах Microsort – **насыщенное приложение**). На границе разделения слоев клиента и сервера создается протокол, специфика которого зависит от выбора этой границы:

- набор и структура сообщений протокола соответствует видам и содержимому элементов интерфейса на границе слоев – процедур, параметров, объектов. Этим определяется также и объемы передаваемых данных и интенсивность трафика;
- синхронный или асинхронный характер взаимодействия в протоколе – в зависимости от того, возникают ли в слоях сервера **асинхронные события,** о которых требуется уведомлять клиента.

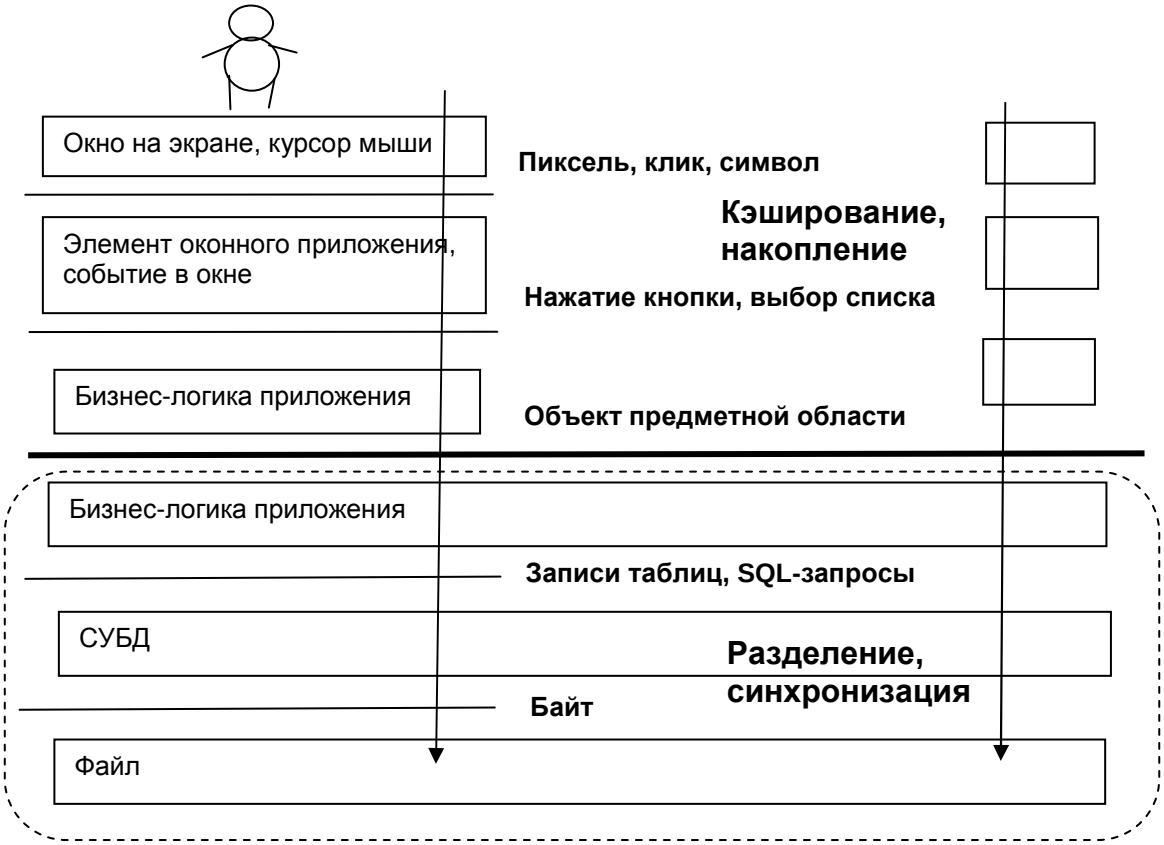

рис.1.3. Слои клиент-серверного приложения

Граница разделения клиентской и серверной части имеет значение еще и с точки зрения взаимодействия клиентов и синхронизации их работы. **Синхронизация и взаимодействие** проще производить в серверной компоненте, поэтому те слои, в которых ее легче реализовать, переносятся в серверное приложение (в противном случае на сервере необходимы примитивы синхронизации, например, транзакции для неделимой последовательности операций в БД). С другой стороны, слои в клиентском приложении могут осуществлять **накопление и кэширование** данных, что позволяет реализовать режим автономной работы клиента при отсутствии связи с сервером, а также увеличить его производительность. Но тогда возникает проблема **синхронизации данных клиента**

с данными сервера. Такие же проблемы возникают, если клиент добавляет собственные данные к существующим бизнес-объектам или создает новые.

Рассмотрим варианты создания клиента в зависимости от выбора точки «разрезания» слоев:

Слои 0-1. Обычно интерфейс между этими слоями скрыт в операционной системе клиента. Фактически он представляет собой сопряжение аппаратных модулей с соответствующими драйверами. Поэтому дилетантская реализация такого «тончайшего» клиента в универсальных системах не всегда возможна, поскольку требует перехвата потока событий на этом уровне. К тому же объем передаваемых данных в таком случае максимален. **Пример**: удаленный рабочий стол.

Слои 1-2. Типичный тонкий клиент, который управляется сервером на уровне отображения отдельных графических примитивов, изменения содержимого оконных объектов (тестовых полей, списков и т.п..), а также передачи серверу всех событий, происходящих в них (нажатие кнопок, выбор в списках, получение/потеря фокуса). Требует передачи большого объема данных, особенно при использовании графики. **Пример**: язык разметки HTML – браузер исполняет роль тонкого клиента, отображая сгенерированное сервером описание графического интерфейса (web-страницы) и уведомляя его о происходящих в нем событиях. **В мобильных приложениях** тонкие клиенты такого рода не особенно распространены, но его идеи могут использоваться, если структура графического интерфейса программируется сервером.

Слой 2. Поскольку этот слой связан с бизнес-моделью (программной моделью предметной области), то и объектами протокола являются элементы бизнес модели – команды (действия над бизнес-объектами) и сами бизнес-объекты. Внутри слоя возможны несколько вариантов границы клиент-сервер:

- если граница располагается между подслоями **2.1-2.2,** то получается **максимально возможный тонкий клиент,** сами бизнес-объекты и их поведение контролируются сервером, а клиент отображает бизнес-объекты и передает серверу сообщения о действиях, производимых пользователем над ними;
- если граница лежит между подслоями **2.2-2.3,** или **2.3-2.4,** то клиент уже является **толстым** («в меру упитанным») – управляющая логика приложения находится в клиенте, а на сервер выносятся алгоритмы автономной работы с бизнес-объектами, а также средства их сборки/разборки из объектов БД.

Слой 3. Стандартным вариантом разделения внутри этого слоя является БД с сетевым (удаленным) доступом, т.е. на подслое **3.2.** Многие СУБД используют соединения сети TCP/IP и свой внутренний протокол для доступа внешних клиентов к БД, в том числе и для администрирования, поэтому принципиальной разницы между локальным и дистанционным доступом к БД нет. Фактически граница разделения клиентской и серверной частей находится в библиотеке программного доступа к БД (ODBC, для Java - JDBC). Получившийся клиент является «толстым». Объем передаваемых данных по сети может быть относительно небольшим, поскольку передаются результаты SQL-запросов, содержащих необходимые выборки данных.

Пример клиент-серверной архитектуры. Система учета рейтинга успеваемости

При соблюдении Современные средства разработки позволя

Система учета рейтинга включает в себя 6 основных видов компонент.

Сервер базы данных MySQL со стандартными средствами сетевого доступа удаленных Java-клиентов с использованием библиотек JDBC. Сервер обеспечивает непосредственный доступ клиентских программ к таблицам БД с использованием SQL-запросов. Клиентские приложения, взаимодействуя с сервером MySQL, работают в режиме «толстого» клиента, т.е. реализуют в себе весь функционал, кроме работы с таблицами.

Web-сервер и серверная компонента (сервлет) реализует в себе большую часть бизнес-логики системы и использует http-протокол для передачи клиентской программе готовых объектов бизнес-модели. Данные запроса передаются в виде обычной строки запроса к web-серверу. Клиентские приложения в этом режиме работают как «тонкие» клиенты.

4 вида клиентов: Java–клиент администратора, Java–клиент преподавателя, Android-клиент преподавателя, Java-клиент студента.

Для реализации такого многообразия с минимальным дублированием кода используется разделение многослойная архитектура. Программный код некоторых слоев может использоваться в различных клиентах, а также либо в клиентском, либо в серверном приложении:

1. уровень внешнего представления (View) – оконный интерфейс клиента;
2. контроллер бизнес-логики клиентских приложений преподавателя – реализует общую бизнес-логику на различных платформах (Java-DeskTop и Android);
3. бизнес-объекты, а также базовый класс уровня, содержащий общую часть алгоритмов работы с бизнес-объектами;
4. производный класс для сохранения бизнес-объектов в двоичных файлах;
5. производный класс для сборки бизнес-объектов из табличных объектов БД;
6. уровень табличных объектов БД;
7. интерфейс для доступа к табличной БД и его реализации для MySQL и SQLite
8. библиотеки для программного доступа к БД: JDBC для MySQL и внутренняя библиотека для БД SQLite в Android;
9. сами СУБД: MySQL – на сервере, SQLite – в Android-клиенте;
10. производный класс для передачи объектов бизнес-модели в XML-формате серверному приложению и получения ответных (WebAPI) (для базового класса п.3);
11. серверная компонента (сервлет), взаимодействующая с предыдущей компонентой.

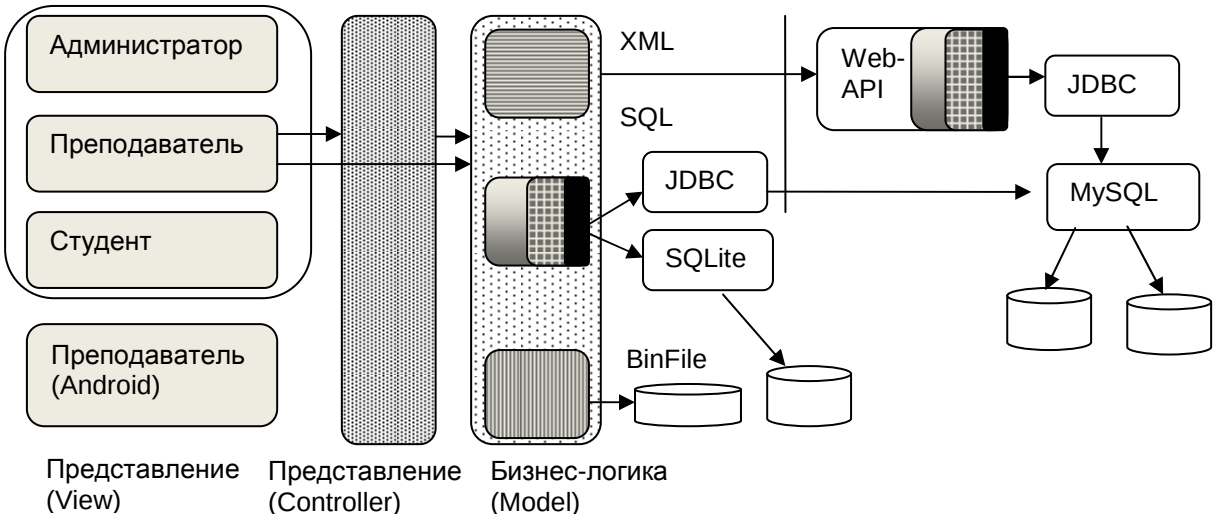

рис.1.4. Архитектура системы учета рейтинга успеваемости

MySQL — MySQL-сервер

JDBC — библиотека доступа к серверу для java (JDBC);

SQLite — внутренняя библиотека Android для работы с БД SQLite

интерфейс для доступа к табличной БД и его реализации для MySQL и SQLite

табличные объекты БД

сборка объектов бизнес-модели из табличных объектов

передача объектов бизнес-модели в XML-формате серверному приложению и получение ответных (WebAPI)

сохранение объектов бизнес-модели в двоичный файл (локальная копия рейтинга)

бизнес-логика, алгоритмы работы с объектами бизнес-модели

контроллер представления – алгоритмы поведения внешних представлений

внешнее представление – формы, диалоги, меню (GUI).

Различные варианты реализации приложений, а также режимы их работы получаются простым соединением слоев, например:

- 1-2-3-4 – автономный клиент с локальной копией данных;
- 1-2-3-5-6-7-8-сеть-9 – толстый клиент на основе стандартной библиотеки доступа к БД;
- 1-2-3-10-сеть-11-5-6-7-8-9 – тонкий клиент с использованием web-сервиса;