Архитектура программного обеспечения инфокоммуникационных систем

Раздел 3

Архитектура основных типов приложений

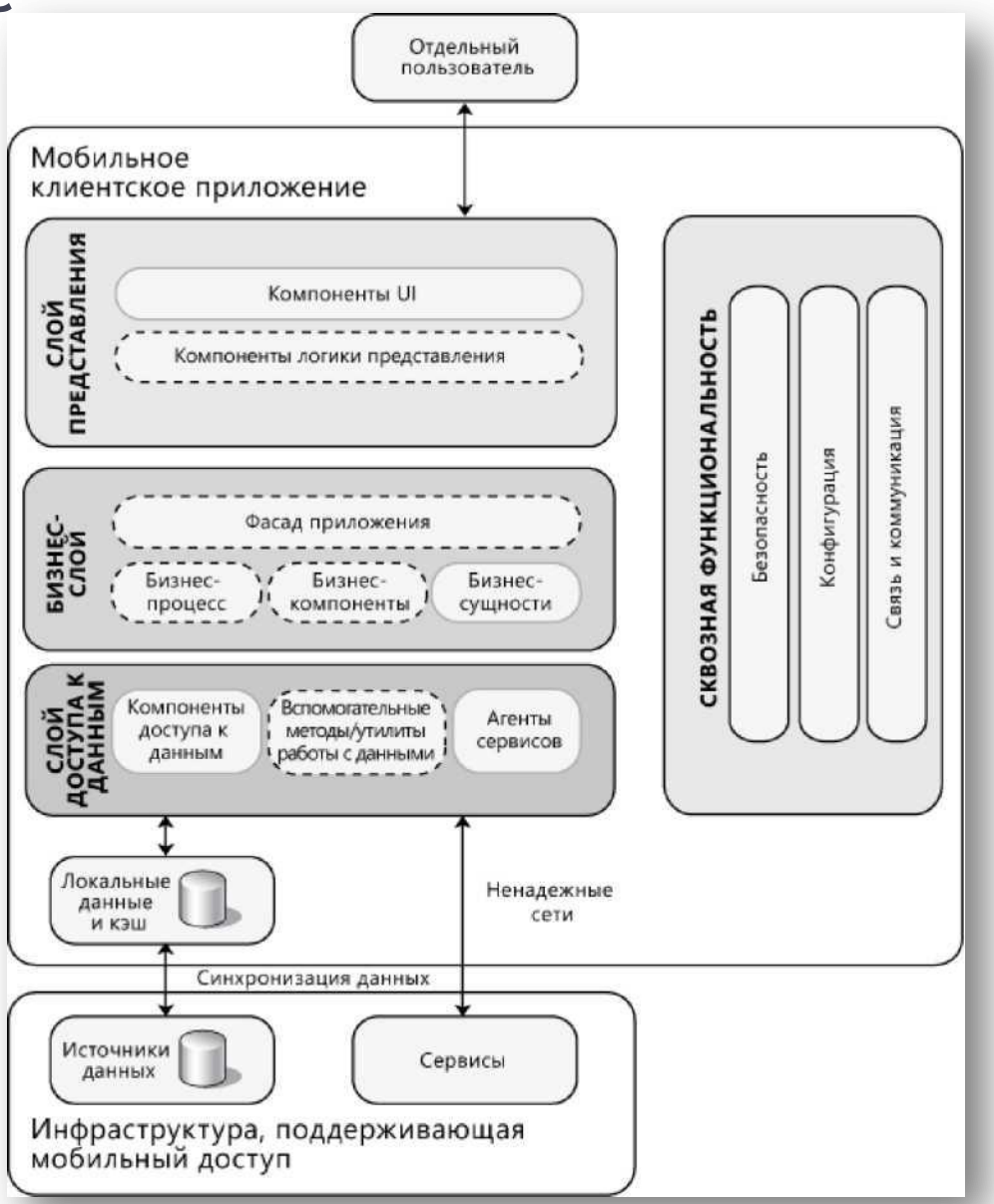
Типы приложений

- Мобильные приложения
  - приложения этого типа могут разрабатываться как тонкий клиент или насыщенное клиентское мобильные приложение
- Насыщенные клиентские приложения
  - приложения этого типа обычно разрабатываются как самодостаточные приложения с графическим пользовательским интерфейсом
- Насыщенные Интернет-приложения
  - насыщенные Интернет-приложения выполняются в изолированной программной среде браузера, которая ограничивает доступ к некоторым возможностям клиента
- Сервисные приложения
  - сервисы предоставляют бизнес-функциональность для совместного использования и позволяют клиентам доступ к ней из локальной или удаленной системы
- Веб-приложения
  - приложения этого типа, как правило, поддерживают сценарии с постоянным подключением

Мобильное приложение

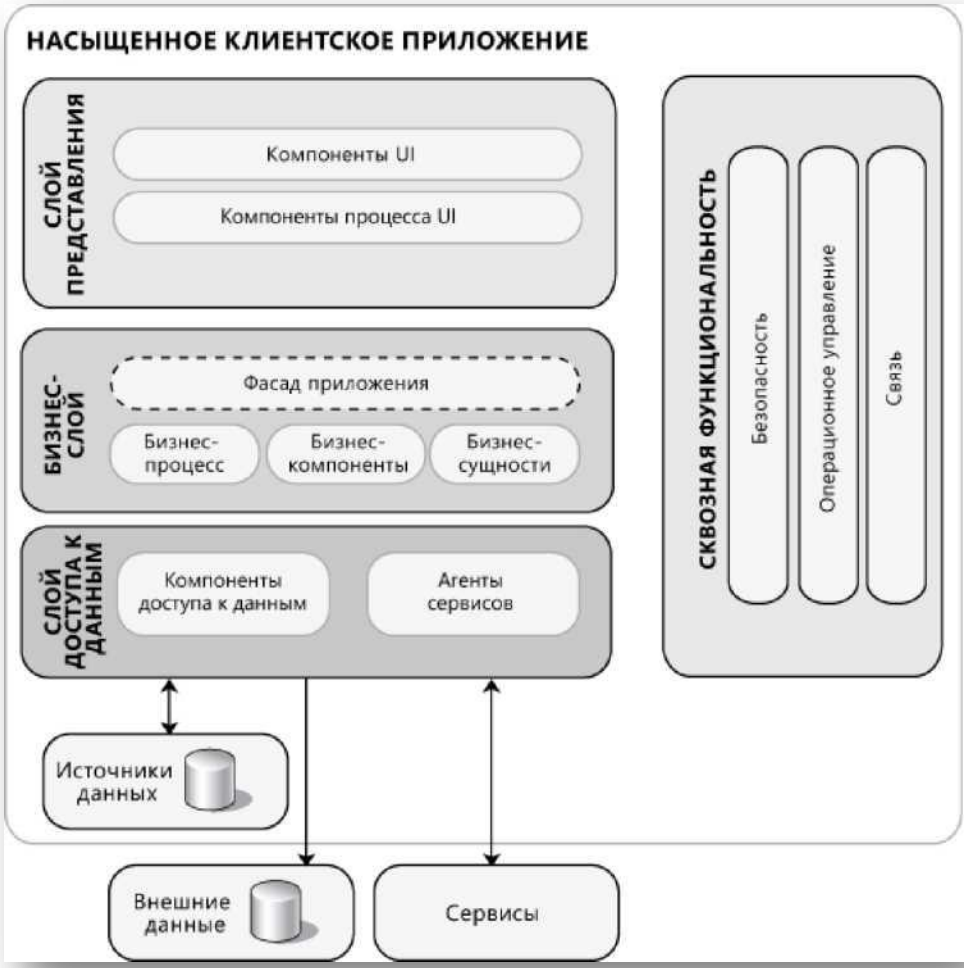
Мобильное приложение

- структурируется как многослойное приложение, включающее слой пользовательского интерфейса (представления), бизнес-слой и слой доступа к данным
- обычно реализует поддержку сценариев без подключения через использование локально кэшированных данных, синхронизация которых выполняется при установлении подключения



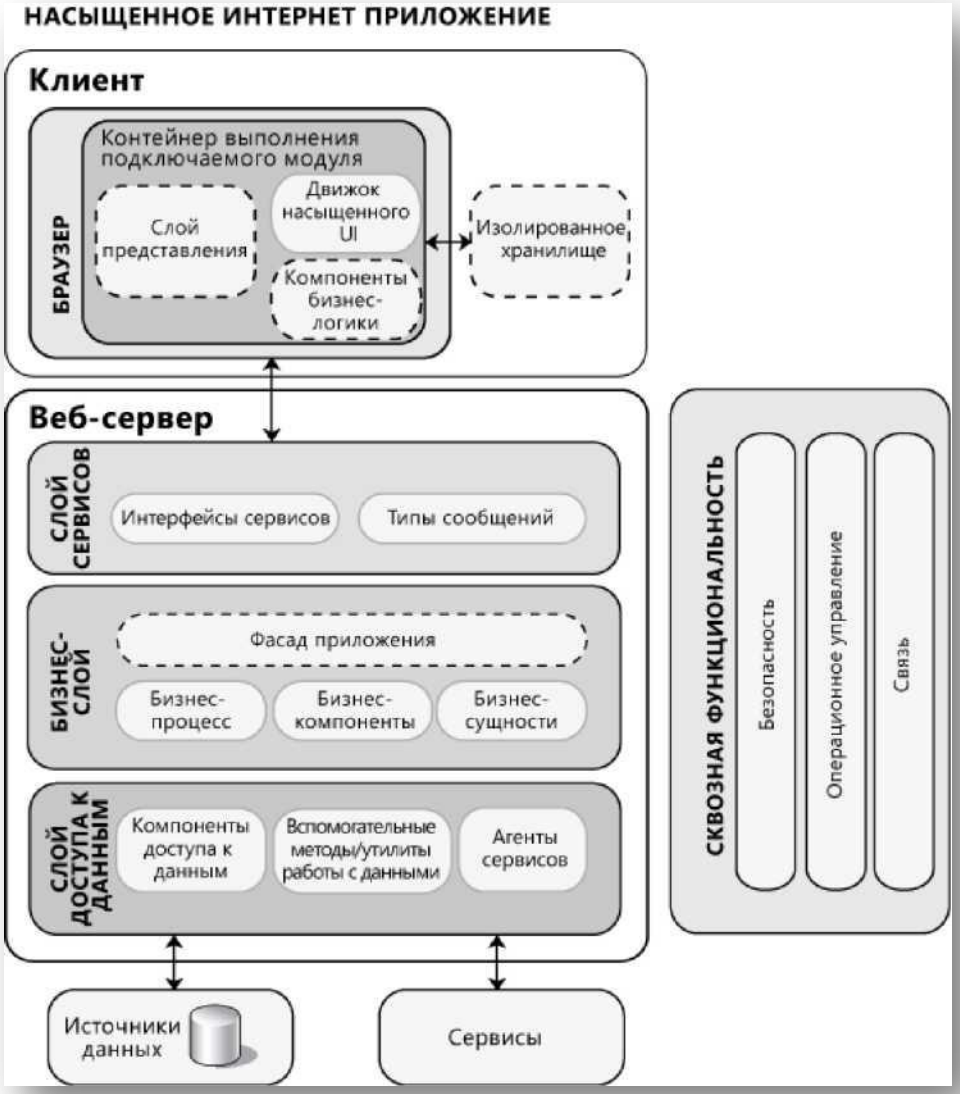
Насыщенное клиентское приложение

- Насыщенные клиентские пользовательские интерфейсы могут обеспечить интерактивное, насыщенное взаимодействие с пользователем с минимальным временем отклика для приложений, которые должны выполняться как самодостаточное приложение, в сценариях с подключением, без постоянного подключения и без подключения



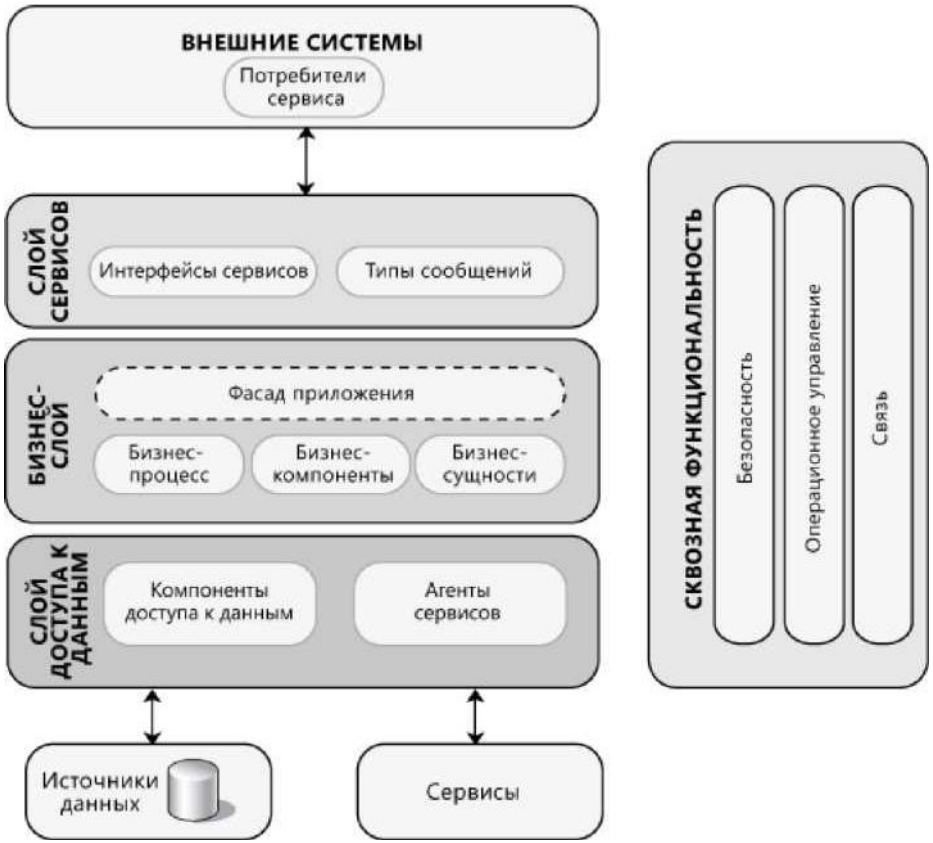
Насыщенное Интернет-приложение

- Насыщенное Интернет-приложение (RIA) выполняется в браузере в изолированной программной среде.
- К преимуществам RIA, по сравнению с традиционными Веб-приложениями, относятся более насыщенный пользовательский интерфейс, улучшенное время отклика приложения и эффективность работы с сетью



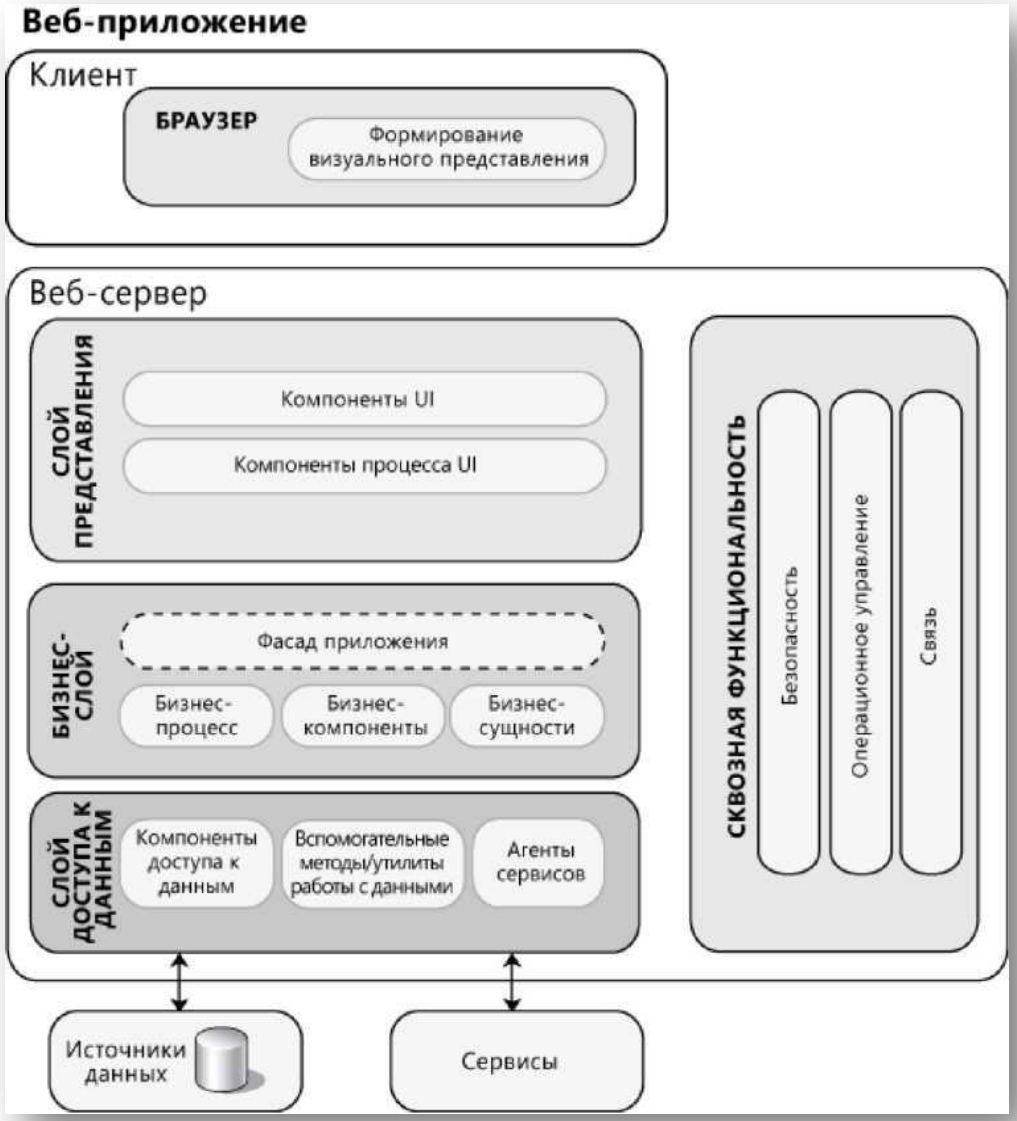
Сервис

- сервис - это открытый интерфейс, обеспечивающий доступ к единице функциональности.
- сервис, фактически, предоставляет программный сервис вызывающей стороне, которая потребляет этот сервис



Веб-приложение

- В Веб-приложениях логика размещается на стороне сервера. Эта логика может состоять из множества отдельных слоев. Типовым примером является трехслойная архитектура, включающая слой представления, бизнес-слой и слой доступа к данным



Принципы проектирования Веб-приложения

Основной целью архитектора ПО при проектировании Веб-приложения является максимальное упрощение дизайна через разделение задач на функциональные области, обеспечивая при этом безопасность и высокую производительность

- Выполните логическое разделение функциональности приложения
- Используйте абстракцию для реализации слабого связывания между слоями
- Определитесь с тем, как будет реализовано взаимодействие компонентов друг с другом
- Используйте кэширование для сокращения количества сетевых вызовов и обращений к базе данных
- Используйте протоколирование и инструментирование
- Продумайте аспекты аутентификации пользователей на границах доверия
- Не передавайте конфиденциальные данные по сети в виде открытого текста
- Проектируйте Веб-приложение для выполнения под менее привилегированной учетной записью

Проектирование насыщенных клиентских приложений

- Выбирайте соответствующую технологию, исходя из требований, предъявляемых к приложению.
  - Windows Forms, WPF, XAML Browser Applications (XBAP) и OBA.
- Разделяйте логику представления и реализацию интерфейса.
- Используйте такие шаблоны проектирования, как Presentation Model и Supervising Presenter (или Supervising Controller),
- Выясните задачи представления и потоки представления
- Спроектируйте подходящий и удобный интерфейс
- Применяйте разделение функциональных областей во всех слоях
- Используйте существующую общую логику представления
- Обеспечивайте слабое связывание клиента с удаленными сервисами, которые он использует
- Избегайте тесного связывания с объектами других слоев
- Сократите количество обращений к сети при доступе к удаленным уровням

Проектирование насыщенных Веб-приложений

Насыщенные Веб-приложения (Rich Internet Applications, RIA) поддерживают насыщенные графические элементы и сценарии с применением потокового мультимедиа, обеспечивая при этом преимущества развертывания и удобства обслуживания, присущие Веб-приложению

- Выбирайте RIA, исходя из предполагаемой аудитории, насыщенного интерфейса и простоты развертывания
- Используйте Веб-инфраструктуру посредством сервисов
- Используйте вычислительные мощности клиента
- Обеспечивайте выполнение в безопасной программной среде браузера
- Определитесь с тем, насколько сложным будет UI
- Используйте сценарии для повышения производительности или сокращения времени отклика приложения
- Предусмотрите ситуацию отсутствия установленного подключаемого модуля

Проектирование мобильных приложений

При разработке мобильного приложения можно создавать тонкий Веб-клиент или насыщенный клиент

- Определитесь, создается ли насыщенный клиент, тонкий Веб-клиент или насыщенное Интернет-приложение
- Определите, какие типы устройств будут поддерживаться
- В случае необходимости учтите сценарии без постоянного подключения и с ограниченной полосой пропускания
- Проектируйте UI, подходящий для мобильных устройств, учитывая ограничения платформы
- Создавайте многослойную архитектуру, подходящую для мобильных устройств, которая повышает возможности повторного использования и удобство обслуживания
- Учитывайте ограничения, налагаемые ресурсами устройства, такие как время работы батареи, объем памяти и частота процессора

Проектирование сервисных приложений

- Используйте многослойный подход и избегайте тесного связывания слоев
- Проектируйте слабо детализированные операции
- При проектировании контрактов данных обеспечивайте возможность расширения и повторного использования
- При проектировании строго придерживайтесь контракта сервиса
- Проектируйте автономные сервисы
- При проектировании должна быть учтена возможность поступления недействительных запросов
- Сервисы должны управляться на основе политик и иметь явные границы
- Разделяйте функциональность сервиса и операции инфраструктуры
- Размещайте основные компоненты слоя сервисов в отдельных сборках
- Избегайте использования сервисов данных для предоставления отдельных таблиц базы данных
- Обеспечьте при проектировании, чтобы сервисы рабочего процесса использовали интерфейсы, поддерживаемые вашей подсистемой управления рабочим процессом

Итоги

Как иерархической системе архитектуре ПО присущ ряд свойств, важнейшими из которых являются:

- вертикальная соподчиненность, заключающаяся в последовательном упорядоченном расположении взаимодействующих компонент, составляющих данный комплекс программ;
- компоненты одного уровня обеспечивают реализацию функций компонент следующего уровня;
- каждый уровень иерархии реализуется через функции компонент более нижних уровней;
- каждый компонент знает о компонентах более низких уровней и ничего не знает о компонентах более высоких уровней;
- право вмешательства и приоритетного воздействия на компоненты любых уровней со стороны компонент более высоких иерархических уровней;
- взаимозависимость действий компонент верхних уровней от реакций на воздействия и от функционирования компонент нижних уровней, информация о которых передается верхним уровням.