Архитектура программного обеспечения инфокоммуникационных систем

Раздел 2

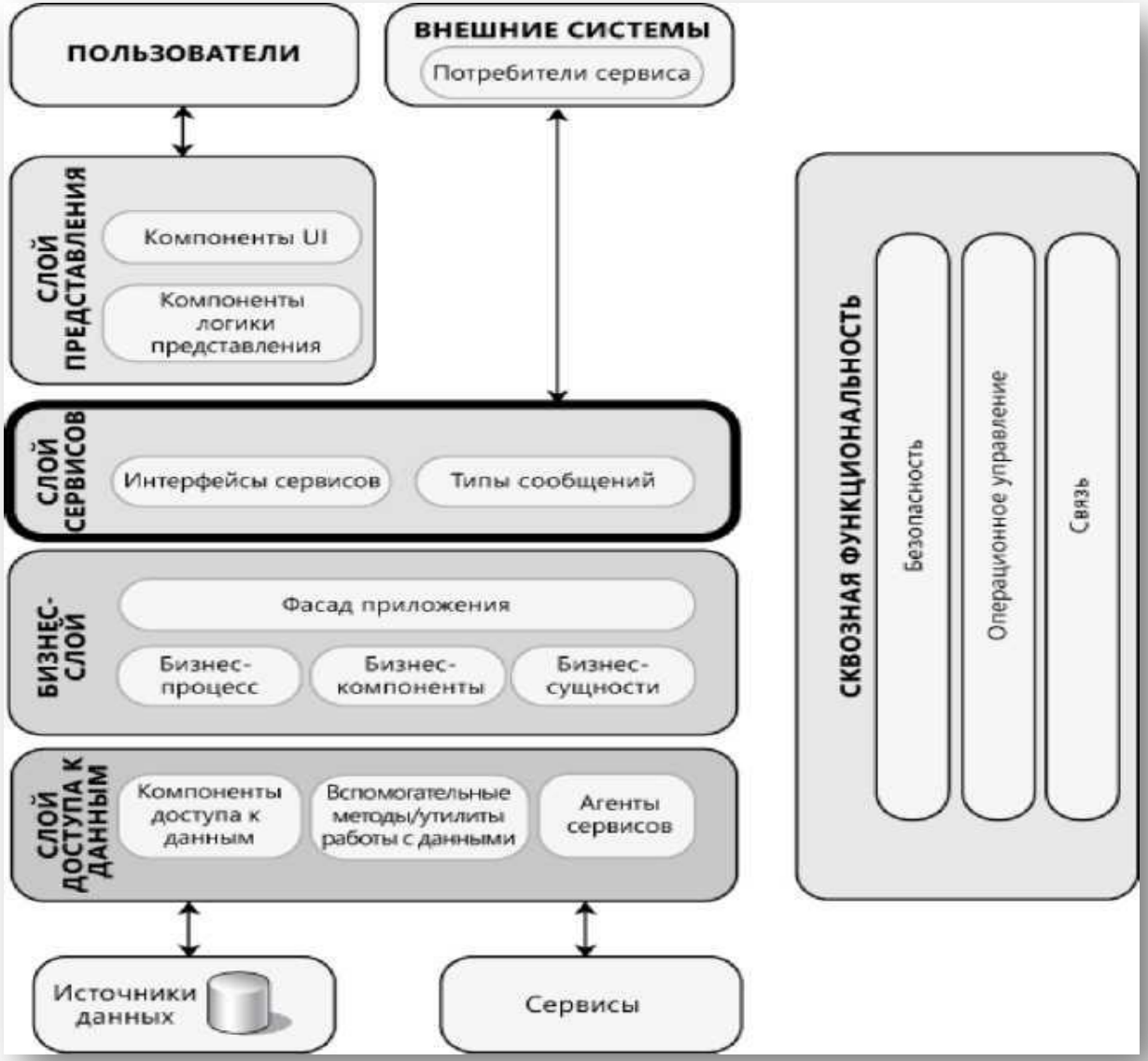
Основы проектирования многослойных приложений

Логическое представление архитектуры многослойной системы

Логическое представление архитектуры многослойной системы

- **Слой представления**
- содержит ориентированную на пользователя функциональность, которая отвечает за реализацию взаимодействием пользователя с системой
- **Бизнес-слой (слой бизнес-логики)**
- реализует основную функциональность системы и инкапсулирует связанную с ней бизнес-логику. Обычно состоит из компонентов, некоторые из которых предоставляют интерфейсы сервисов, доступные для использования другими участниками взаимодействия.
- **Слой доступа к данным**
- обеспечивает доступ к данным, хранящимся в рамках системы, и данным, предоставляемым другими сетевыми системами. Доступ может осуществляться через сервисы.

Включение слоя сервисов в приложение



- Слой сервисов обеспечивает альтернативное представление, позволяющее клиентам использовать другой механизм для доступа к приложению

Этапы проектирования многослойной структуры

- Шаг 1 - Выбор стратегии разделения на слои.
- Шаг 2 - Выбор необходимых слоев.
- Шаг 3 - Принятие решения о распределении слоев и компонентов.
- Шаг 4 - Выяснение возможности сворачивания слоев.
- Шаг 5 - Определение правил взаимодействия между слоями.
- Шаг 6 - Определение сквозной функциональности.
- Шаг 7 - Определение интерфейсов между слоями.
- Шаг 8 - Выбор стратегии развертывания.
- Шаг 9 - Выбор протоколов связи.

Слой представления

Содержит

- компоненты реализующие и отображающие пользовательский интерфейс
- компоненты управляющие взаимодействием с пользователем
- элементы управления для ввода данных пользователем и их отображения.

Компоненты пользовательского интерфейса:

- визуальные элементы приложения, используемые для отображения данных пользователю и приема пользовательского ввода

Компоненты логики представления:

- код приложения, определяющий поведение и структуру приложения и не зависящий от конкретной реализации пользовательского интерфейса.

При реализации шаблона Separated Presentation могут использоваться следующие компоненты логики представления:

- Презентатор (Presenter),
- Модель презентации (Presentation Model)
- Модель Представления (View Model).

Рекомендации по проектированию слоя представления

- Выбирайте соответствующий тип приложения
  - Решение должно приниматься на основании требований, предъявляемых к приложению, и ограничений, накладываемых организацией или средой
- Выбирайте соответствующую технологию UI
- Используйте соответствующие шаблоны
  - Separated Presentation, MVC, MVP, Supervising Presenter
- Разделяйте функциональные области
  - Используйте специальные компоненты UI для формирования визуального представления, отображения и взаимодействия с пользователем
- Учитывайте рекомендации по проектированию пользовательского интерфейса
  - удобство и простота доступа, локализация и удобство использования, интерактивность UI
- Придерживайтесь принципов ориентированного на пользователя проектирования
  - используйте опросы, исследования, интервью для выбора варианта пользовательского интерфейса

Бизнес-слой

Включает следующие компоненты:

- Фасад приложения
  - необязательный компонент обеспечивает упрощенный интерфейс для компонентов бизнес-логики, сочетая множество бизнес-операций в одну, упрощает использование бизнес-логики
- Компоненты бизнес-логики
  - извлечение, обработка, преобразование и управление данными приложения; применение бизнес-правил и политик и обеспечение непротиворечивости и действительности данных
    - Компоненты бизнес-процесса
      - определяют и координируют долгосрочные многоэтапные бизнес-процессы
    - Компоненты бизнес-сущностей
      - инкапсулируют бизнес-логику и данные, необходимые для представления в приложении объектов реального мира, таких как заказчики (Customers) или заказы (Orders)

Рекомендации по проектированию бизнес-слоя

- Убедитесь в необходимости отдельного бизнес-слоя
  - способствует повышению удобства обслуживания приложения
- Определитесь с ответственностями и потребителями бизнес-слоя
  - решение о том, какие задачи должен выполнять бизнес-слой, и каким образом будет предоставляться доступ к нему
- Не смешивайте в бизнес-слое компоненты разных типов
  - бизнес-слой - средство избежать смешения кода представления и доступа к данным с бизнес-логикой, чтобы отделить бизнес-логику от логики представления и доступа к данным и упростить тестирование бизнес-функциональности
- Сократите количество сетевых вызовов при доступе к удаленному бизнес-слою
  - Используйте большие пакеты для передачи данных по сети
- Избегайте тесного связывания между слоями
  - При создании интерфейса бизнес-слоя применяйте принципы абстракции для максимального ослабления связывания

Слой доступа к данным

Включает следующие компоненты:

- Компоненты доступа к данным.
  - Эти компоненты абстрагируют логику, необходимую для доступа к базовым хранилищам данных. Они обеспечивают централизацию общей функциональности доступа к данным, что способствует упрощению настройки и обслуживания приложения.
- Агенты сервисов.
  - реализуют компоненты доступа к данным, которые изолируют меняющиеся требования вызова сервисов от приложения и могут обеспечивать дополнительные сервисы

Рекомендации по проектированию слоя доступа к данным

- Правильно выберите технологию доступа к данным
- Используйте абстракцию для реализации слабо связанного интерфейса слоя доступа к данным
- Инкапсулируйте функциональность доступа к хранилищу данных в слое доступа к данным
- Примите решение о том, как будет выполняться сопоставление сущностей приложения со структурами источника данных
- Рассмотрите возможность объединения структур данных
- Примите решение о том, как будет реализовано управление подключениями
- Определите, как будут обрабатываться исключения, возникающие при обработке данных
- Учтите риски безопасности
- Сократите количество сетевых вызовов и обращений к базе данных
- Учтите требования производительности и масштабируемости

Шаблоны для слоя представления

Категория	Шаблоны проектирования
Кеширование	Cache Dependency (Кэш с зависимостью). Использует внешние данные для определения состояния данных, хранящихся в кэше. Page Cache (Кэш страниц). Улучшает время ответа динамических Веб-страниц
Композиция и компоновка	Composite View (Составное представление). Сочетает отдельные представления в композитное представление. Шаблон Presentation Model (Model-View-ViewModel). Разновидность шаблона Model-View-Controller (MVC), приспособленная для современных платформ разработки UI Template View (Представление по шаблону). Реализует представление общего шаблона и создает представления на базе этого шаблонного представления. Transform View (Представление с преобразованием). Преобразует данные, переданные на уровень представления, в HTML для отображения в UI. Two-Step View (Двухэтапное представление). Преобразует модель данных в логическое представление без какого-либо специального форматирования и затем преобразует это логическое представление, добавляя необходимое форматирование.
Управление исключениями	Exception Shielding (Экранирование исключений). При возникновении исключения предотвращает предоставление сервисом данных о его внутренней реализации.
Навигация	Application Controller (Контроллер приложений). Обработка навигации между окнами. Front Controller (Контроллер запросов). Шаблон только для Веб, консолидирующий обработку запросов путем направления всех запросов через один объект-обработчик, который можно изменять во время выполнения с помощью декораторов. Page Controller (Контроллер страниц). Принимает ввод из запроса и обрабатывает его для конкретной страницы или действия Веб-сайта. Command (Команда). Инкапсулирует обработку запроса в отдельный командный объект с обычным интерфейсом выполнения.
Взаимодействие с пользователем	Asynchronous Callback (Асинхронный обратный вызов). Выполняет длительные задачи в отдельном потоке, выполняющемся в фоновом режиме, и обеспечивает потоку функцию для обратного вызова по завершении выполнения задачи. Chain of Responsibility (Цепочка обязанностей). Предоставляет возможность обработать запрос нескольким объектам, устраняет возможность связывания отправителя запроса с его получателем

Шаблоны для бизнес-слоя

Категория	**Шаблоны проектирования**
Компоненты бизнес-слоя	Application Fagade (Фасад приложения). Централизует и агрегирует поведение для обеспечения унифицированного слоя сервисов. Chain of Responsibility (Цепочка обязанностей). Предоставляя возможность обработать запрос нескольким объектам, устраняет возможность связывания отправителя запроса с его получателем. Command (Команда). Инкапсулирует обработку запроса в отдельный командный объект с общим интерфейсом выполнения.
Бизнес-сущности	Domain Model (Модель предметной области). Набор бизнес- объектов, представляющих сущности предметной области и отношения между ними. Entity Translator (Транслятор сущностей). Объект, преобразующий типы данных сообщения в бизнес-типы для запросов и выполняющий обратные преобразования для ответов. Table Module (Модуль таблицы). Единый компонент, реализующий бизнес-логику для всех строк таблицы или представления базы данных.
Рабочие процессы	Data-Driven Workflow (Управляемый данными рабочий процесс). Рабочий процесс, включающий задачи, последовательность выполнения которых определяется значениями данных в рабочем процессе или системе. Human Workflow (Рабочий процесс, управляемый оператором). Рабочий процесс, включающий задачи, выполняемые вручную. Sequential Workflow (Последовательный рабочий процесс). Рабочий процесс, включающий задачи, выполняющиеся в определенной последовательности, когда выполнение одной задачи запускается только после завершения предыдущей. State-Driven Workflow (Управляемый состоянием рабочий процесс). Рабочий процесс, включающий задачи, последовательность выполнения которых определяется состоянием системы.

Шаблоны для слоя доступа к данным

Категория	Шаблоны проектирования
Общие	Active Record (Активная запись). Включает объект доступа к данным в сущность предметной области. Data Mapper (Преобразователь данных). Реализует слой преобразования между объектами и структурой базы данных, используемый для перемещения данных из одной структуры в другую. Data Transfer Object (Объект передачи данных). Объект, в котором сохраняются данные, передаваемые между процессами, что обеспечивает сокращение необходимого числа вызовов методов. Domain Model (Модель предметной области). Набор бизнес-объектов, представляющих сущности предметной области и отношения между ними. Query Object (Объект запроса). Объект, представляющий запрос к базе данных. Repository (Хранилище). Представление источника данных в памяти, работающее с сущностями предметной области. Row Data Gateway (Шлюз записи данных). Объект, выступающий в роли шлюза к отдельной записи Table Data Gateway (Шлюз таблицы данных). Объект, выступающий в роли шлюза к таблице или представлению источника данных и выполняющий сериализацию всех запросов на выбор, вставку, обновление и удаление. Table Module (Модуль таблицы). Единый компонент, реализующий бизнес-логику для всех строк таблицы или представления базы данных
Пакетная обработка	Parallel Processing (Параллельная обработка). Позволяет обрабатывать множество пакетных операций одновременно, чтобы сократить время обработки. Partitioning (Секционирование). Разбивает большие пакеты, чтобы обрабатывать их параллельно.
Транзакции	Capture Transaction Details (Перехват данных транзакции). Создает объекты базы данных, такие как триггеры и теневые таблицы, для записи всех изменений, вносимых в ходе транзакции. Coarse Grained Lock (Блокировка крупных фрагментов данных). Одной блокировкой блокирует набор взаимосвязанных объектов. Implicit Lock (Неявная блокировка). Использует код инфраструктуры для запроса блокировки от имени кода, выполняющего доступ к совместно используемым ресурсам. Optimistic Offline Lock (Оптимистическая блокировка в автономном режиме). Pessimistic Offline Lock (Пессимистическая блокировка в автономном режиме). Предотвращает конфликты, вынуждая транзакцию блокировать данные перед их использованием. Transaction Script (Сценарий транзакции). Организует бизнес-логику каждой транзакции в одну процедуру, обращаясь к базе данных напрямую либо через тонкую оболочку над базой данных

Проектирование компонентов приложения

Применяйте принципы **SOLID** при проектировании классов, входящих в компонент:

- *Принцип единственности ответственности* (**S**ingle responsibility). Класс должен отвечать только за один аспект.
- *Принцип открытости/закрытости* (**O**pen/closed principle). Классы должны быть расширяемыми без необходимости доработки.
- *Принцип замещения Лискова* (**L**iskov substitution principle). Подтипы и базовые типы должны быть взаимозаменяемы.
- *Принцип отделения интерфейса* (**I**nterface segregation principle). Интерфейсы классов должны быть клиент-специфическими и узконаправленными. Классы должны предоставлять разные интерфейсы для клиентов, имеющих разные требования к интерфейсам.
- *Принцип инверсии зависимостей* (**D**ependency inversion principle). Зависимости между классами должны заменяться абстракциями. Абстракции не должны зависеть от деталей - детали должны зависеть от абстракций.

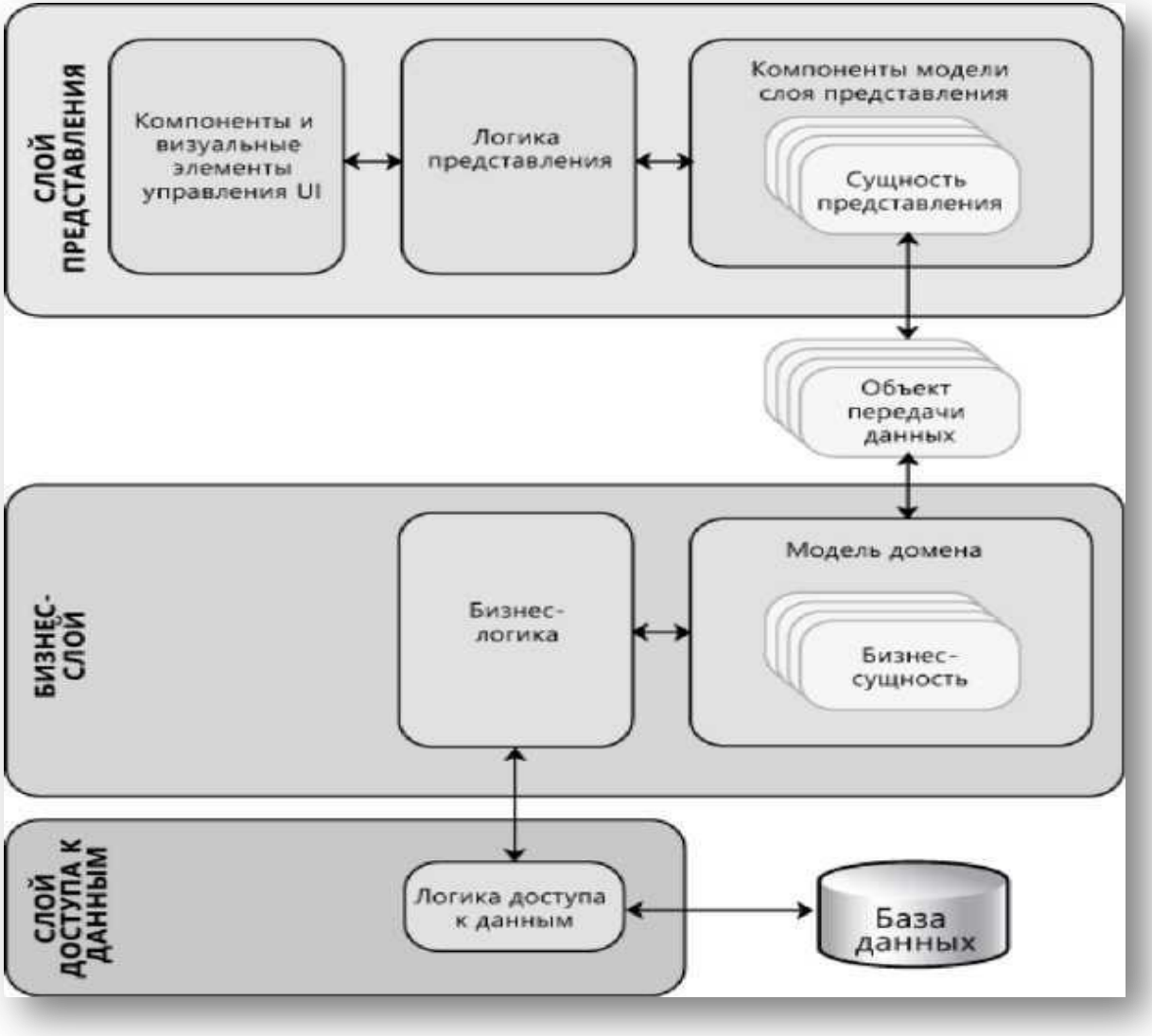
Проектирование компонентов приложения

- Проектируйте сильно связные компоненты
  - избегайте смешения в компонентах бизнес-слоя логики доступа к данным и бизнес-логики
- Компонент не должен зависеть от внутренних деталей других компонентов
- Продумайте, как компоненты будут взаимодействовать друг с другом
  - используются сценарии развертывания
- Не смешивайте код сквозной функциональности и прикладную логику приложения
- Применяйте основные принципы компонентного архитектурного стиля
  - компоненты должны быть пригодными для повторного использования, заменяемыми, расширяемыми, инкапсулированными, независимыми и не зависеть от контекста

Проектирование компонентов пользовательского интерфейса и компонентов логики представления

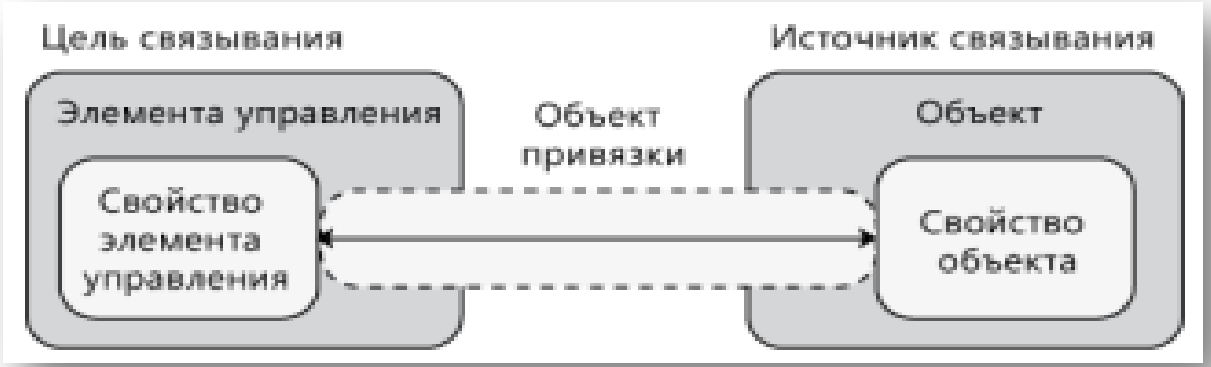
1. Понимание предъявляемых к UI требований
   - требования к UI определяются функциональностью, которую должно поддерживать приложение, и ожиданиями пользователей
2. Выбор необходимого типа UI
   - мобильные приложения, насыщенные клиентские мобильные приложения, Веб или тонкие клиентские мобильные приложения, насыщенные клиентские приложения, консольные приложения
3. Выбор технологии UI
   - Microsoft .NET Compact Framework, ASP.NET, WPF, Windows Forms, XBAP, Silverlight, ASP.NET Web Forms, ASP.NET Web Forms с AJAX, ASP.NET MVC, ASP.NET Dynamic Data
4. Проектирование компонентов представления
   - компоненты пользовательского интерфейса – визуальные ЭУ
   - компоненты логики представления – не визуальные ЭУ
   - компоненты модели представления – представляют данные от бизнес-слоя

Компоненты модели представления и сущности представления



Проектирование компонентов пользовательского интерфейса и компонентов логики представления

5. Определение требований к привязке данных
   - Привязка данных обеспечивает возможность создания связи между элементами управления пользовательского интерфейса и данными или логическими компонентами приложения



6. Выработка стратегии обработки ошибок
   - Компоненты UI должны реализовывать соответствующую стратегию обработки ошибок для обеспечения стабильности приложения и положительного впечатления при взаимодействии с пользователем
7. Определение стратегии валидации

Проектирование компонентов бизнес - слоя

- 1. Выбор компонентов бизнес-слоя, которые будут использоваться в приложении
- 2. Принятие ключевых решений по компонентам бизнес-слоя
  - Размещение, Связывание, Взаимодействие
- 3. Выбор соответствующей поддержки транзакций
- 4. Выработка стратегии обработки бизнес-правил
  - в зависимости от шаблона проектирования, применяемого для их реализации
  - отделить бизнес- правила от бизнес-сущностей,
- 5. Выбор шаблонов, соответствующих требованиям
  - Adapter (Адаптер), Command (Команда), Decorator (Декоратор), Chain of Responsibility (Цепочка обязанностей), Dependency Injection (Внедрение зависимостей), Fagade (Фасад), Factory (Фабрика) Transaction Script (Сценарий транзакции)

Проектирование компонентов данных

- 1. Выбор технологии доступа к данным
  - ADO.NET Entity Framework, ADO.NET Data Services Framework, ADO.NET Core, LINQ to XML, ADO.NET Sync Services
- 2. Принятие решения о методе извлечения и хранения бизнес-объектов источника данных
  - Object/Relational Mapping, O/RM)
- 3. Выбор способа подключения к источнику данных
  - открывайте подключения к источнику данных как можно позже и закрывайте их как можно раньше
  - осуществляйте транзакции через одно подключение
  - используйте пул подключений и оптимизируйте производительность
  - избегайте использования системных или пользовательских DSN для хранения данных подключения
  - предусмотрите логику повторного подключения для случаев разрыва соединения
  - по возможности используйте пакетные команды
- 4. Выработка стратегий обработки ошибок источника данных
- 5. Проектирование объектов агентов сервисов (необязательный)

Показатели качества

Категория	Показатель качества
Качество дизайна	Концептуальная целостность
	Удобство и простота обслуживания
	Возможность повторного использования
Качество времени выполнения	Доступность
	Возможность взаимодействия
	Управляемость
	Производительность
	Надежность
	Масштабируемость
	Безопасность
Качество системы	Обеспеченность технической поддержкой
	Тестируемость
Качество взаимодействия с пользователем	Удобство и простота использования

Проектирование общей функциональности приложения

- Сквозная функциональность – общая функциональность, которую нельзя отнести к конкретному слою или уровню:
  - аутентификация
  - авторизация
  - кэширование
  - связь
  - управление исключениями
  - протоколирование
  - инструментирование
  - валидация
- Эти функции оказывают влияние на все приложение и, по возможности, должны реализовываться централизованно

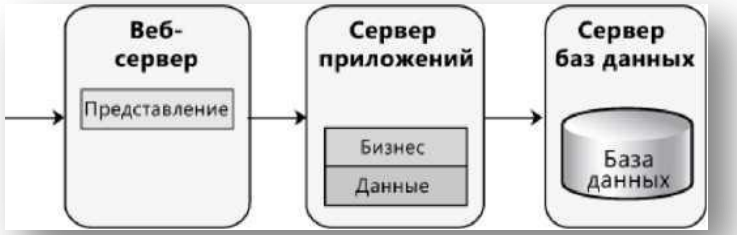
Стратегии развертывания приложений

При выборе стратегии развертывания:

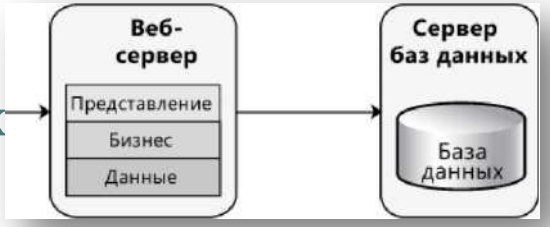
- Изучите целевую физическую инфраструктуру развертывания.
- Исходя из инфраструктуры развертывания, выявите ограничения архитектуры и дизайна.
- Выявите, какое влияние на безопасность и производительность разрабатываемой системы будет оказывать инфраструктура развертывания

Модель развертывания:

- распределенное
  - все слои приложения располагаются на разных физических уровнях

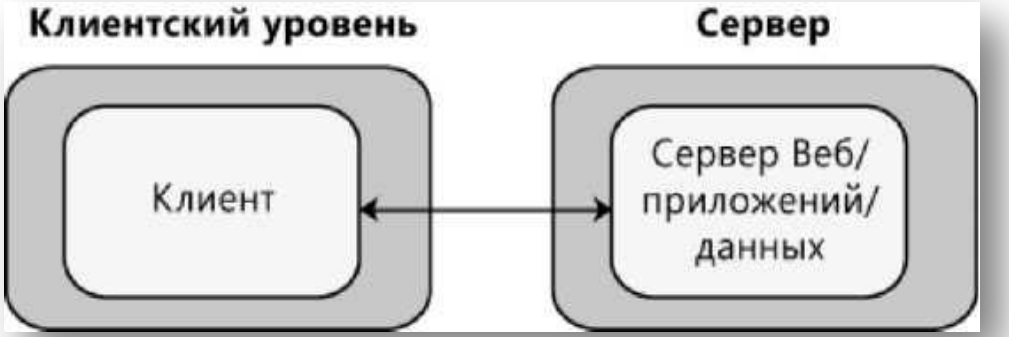


- нераспределенное
  - вся функциональность и слои приложения, кроме функциональности хранения данных располагаются на одном сервере



Шаблоны распределенного развертывания

- Развертывание клиент-сервер



- n-уровневое развертывание