Описание архитектуры информационной системы

автоматизации процессов сбора, обработки и хранения отчетности

1. Общая информация, технологические основы

1.1. Информационная система автоматизации процессов сбора, обработки и хранения отчетности, разработанная ООО «Инфостандарт» реализована по классической клиент-серверной модели с учетом особенностей, которые накладываются применением технологии ODANT.

1.2. Особенности реализации системы с использованием технологии ODANT:

Объекты могут иметь сложную древовидную структуру с неограниченным количеством уровней и полей на каждом уровне, включая вложенные массивы (таблицы).

Структура объектов описывается в классах, экземплярами которых они являются, и в которых они хранятся или отображаются.

Объекты могут ссылаться друг на друга, наследоваться и объединяться в древовидные иерархии.

Домены - структурные элементы, формирующие иерархию хранения и необходимые для группировки и изоляции классов.

Если сравнивать с РСУБД, то можно считать, что домены это отдельные базы данных, обладающие собственной структурой и системой безопасности.

Домены бывают 3 типов:

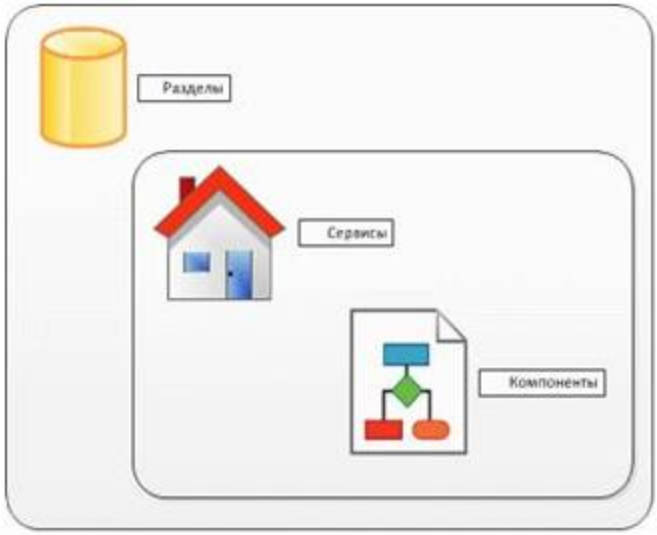
- разделы,

- сервисы

- компоненты.

Системные домены содержат в себе классы с системными данными и функционалом, например - систему безопасности.

Пример (Изобр.1):



Каждый элемент odant - хост, домен, класс, объект, имеет глобальный идентификатор (присваивается системой) в виде 15-значного кода _id (например, 1CEEAB285FDB6FC)

Из иерархической последовательности идентификаторов строится полный идентификатор _fullid, указывающий точное расположение каждого элемента в глобальной сети.

Конфигурация для каждого пользователя формируется динамически, из набора доступных для него элементов.

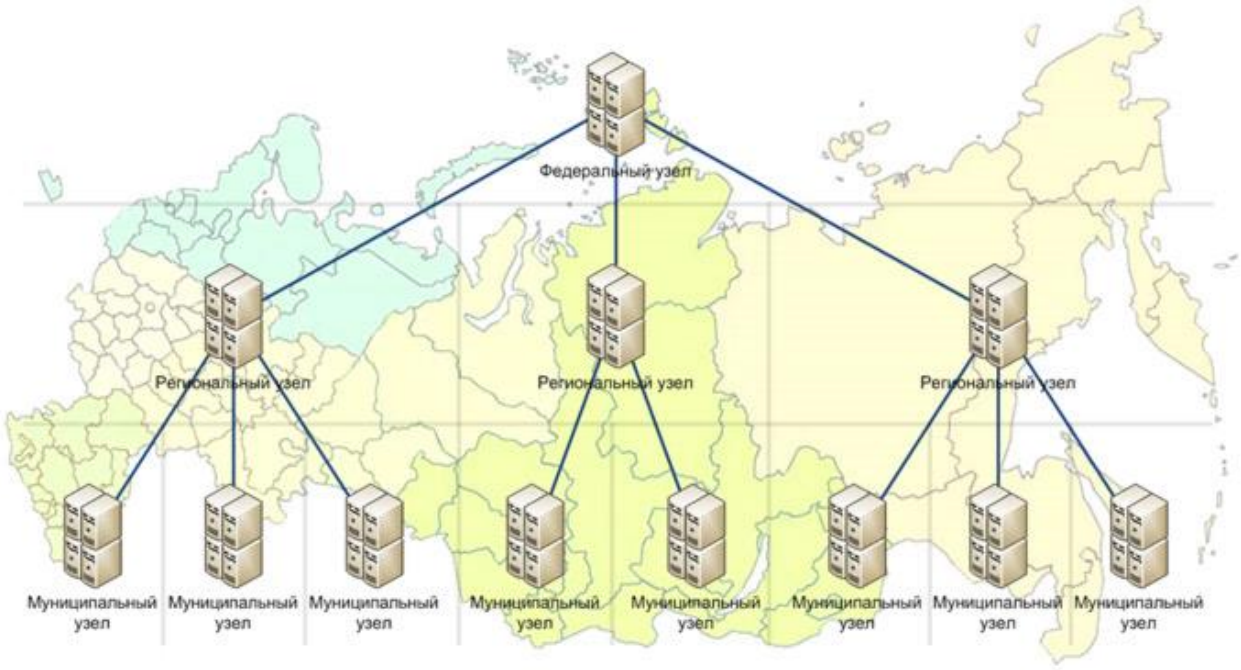
Хост – это отдельный физический сервер.

Каждый хост имеет собственный IP-адрес или URL.

Хосты могут объединяться в бизнес-сети, т.е. каждый хост может предоставлять другим хостам свои структуры и/или объекты данных.

В данном случае хост – это место развертывания проекта Системы автоматизации процессов сбора, обработки и хранения отчетности.

Пример (Изобр.2):



Все данные хранятся в XML-подобном бинарном формате ODB.

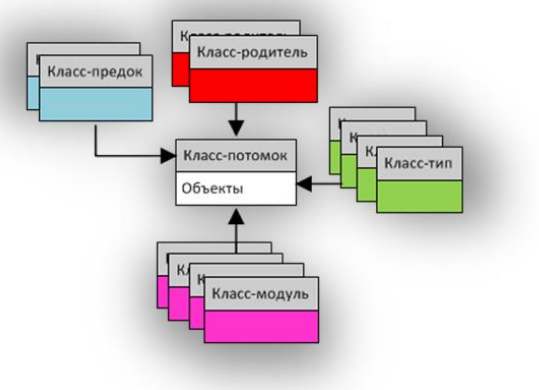
Работа с данными ведется с использованием w3c-стандартного языка запросов XQuery.

Доступ к ресурсам и структурам узлов сети осуществляется через REST API команды с глобальной адресацией.

Для работы напрямую с объектными данными не требуются какие-либо ORM библиотеки. При создании базы моделирование данных происходит в объектных терминах, поскольку данная СУБД использует универсальную объектную модель разметки документа - ODBML (Object DataBase Markup Language). Единый язык объектной разметки в совокупности с уникальной глобальной идентификацией данных позволяет системам, созданным разными разработчиками для разных целей, обладать самым труднодостижимым (для традиционных SQL/NOSQL технологий) видом совместимости - совместимостью на уровне данных.

Классы являются одновременно структурными, информационными и функциональными элементами, типами данных для полей, а также могут предоставлять свой функционал соседним классам.

Классы могут объединяться в домены по функциональному, прикладному, отраслевому или другим принципам (Изобр.3):



Компоненты бывают трех типов: модули, решения и конфигурации.

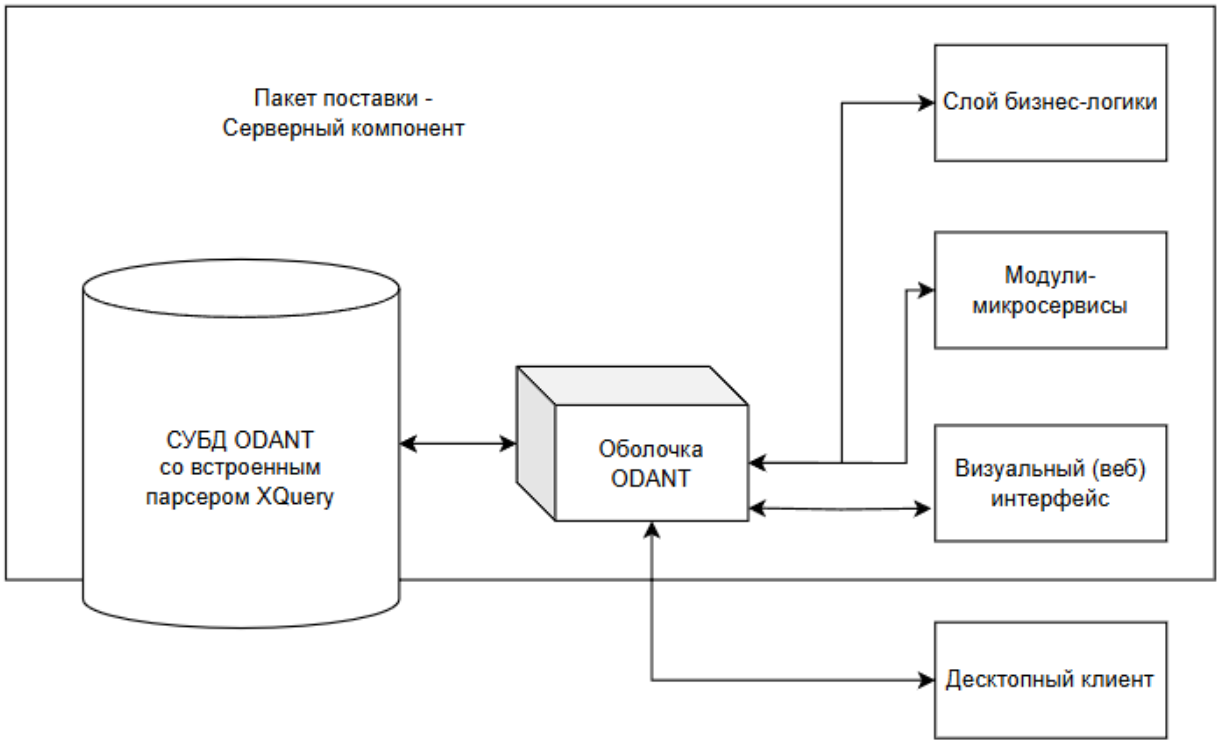
Компоненты могут быть опубликованы в общем разделе любого сервера в сети, а затем установлены в хранилища (опять же, на любой сервер).

Компоненты, как и классы, поддерживают распределенное наследование.

Подробнее: https://csc.odant.org/?page_id=18

2. Визуальное представление архитектуры системы

В общем виде реализацию можно представить следующим образом (Изобр.4):



СУБД ODANT выполняет функции не только хранения информации в виде ODB (XML-подобных бинарных файлов), но и предоставляет ряд платформенных функций:
- Хранение и исполнение запросов к данным, а также результатов индексирования, журналирования и пакетирования запросов встроенными средствами.

- Выступает как репозиторий бизнес-логики, которая хранится также внутри СУБД согласно парадигме технологии.

- Имеет ряд встроенных (платформенных) решений по контролю доступа к данным (пользователи, ролевая модель), а также обеспечению общих платформенных функций.

Слой оболочки ODANT предоставляет API как в виде традиционных REST-запросов, так и http запросов по внутренним стандартам платформы. Запросы работают по стандарту HTTP2.0 и имеют элементы сокетного соединения.

Оболочка не имеет самостоятельной роли и выполняет роль контроллера только в объемах своих задач.

Слой бизнес-логики реализован в виде настроек (конфигурации) внесенной на уровне платформы и хранящейся в СУБД. Настройки (конфигурация) описывают модель данных, правила установленные для модели данных, запросы к данным, триггеры и другие правила описывающие поведения по событиям (запросам), а также запросы получающие, обрабатывающие и сохраняющие измененные данные (логика) и способы (модификаторы) их вызова. Также слой логики частично включает в себя правила отображения информационных сущностей в интерфейса и правила их обработки.

Слой модулей (микросервисов) включает в себя отдельные (подключаемые) компоненты системы, в том числе внешней разработки, которые выполняют сервисную (узкофункциональную) роль и запускаются (разворачиваются) как отдельные элементы общей системы (подсистемы, сервисы, службы). Могут быть реализованы в виде .NET, JavaScript (Node.JS), Python и подобных систем, а также могут быть запущены в виде контейнеров (расширяемость и изолированность) в целях безопасности и гибкости системы. Компоненты не заменяют собой ядро системы (Слои СУБД, Оболочки и Бизнес-логики).

Визуальный интерфейс системы реализован в виде JavaScript-кода, а также производных и связанных HTML и CSS файлов, и содержит в себе адаптированное веб-приложение, которое направляет команды к веб-серверу системы и отрисовывает (предоставляет) пользовательский веб-интерфейс в зависимости от поступающих ему данных из слоя СУБД и правил бизнес-логики из слоя Бизнес-логики.

Десктопный клиент реализован на технологии ASP.NET и работает под Windows, а также при помощи специальных средств под Linux ОС. Десктопный клиент является дополнительным средством работы с системой и повторяет логику основного интерфейса – веб-клиента.

Правила работы клиентской части идентичны. Веб-маршрутизатор (рекомендован Nginx) обеспечивает направление запросов и ответов на уровне окружения системы, однако во всех случаях безопасность обеспечивается силами внешних средств и самой ОС. Аутентификация, как ранее было описано, обеспечивается средствами СУБД (платформы).

3. Дополнительная информация по теме

Подробнее про технологические и архитектурные основы серверного API можно прочитать здесь:

https://csc.odant.org/?p=1210

Подробнее про стандарты языка запросов XQuery можно прочитать здесь:

https://www.w3.org/TR/xquery-31/

Подробнее про применяемые на уровне веб-клиента подходы по работе с JavaScript, браузером и безопасностью можно прочитать здесь:

https://odajs.org/