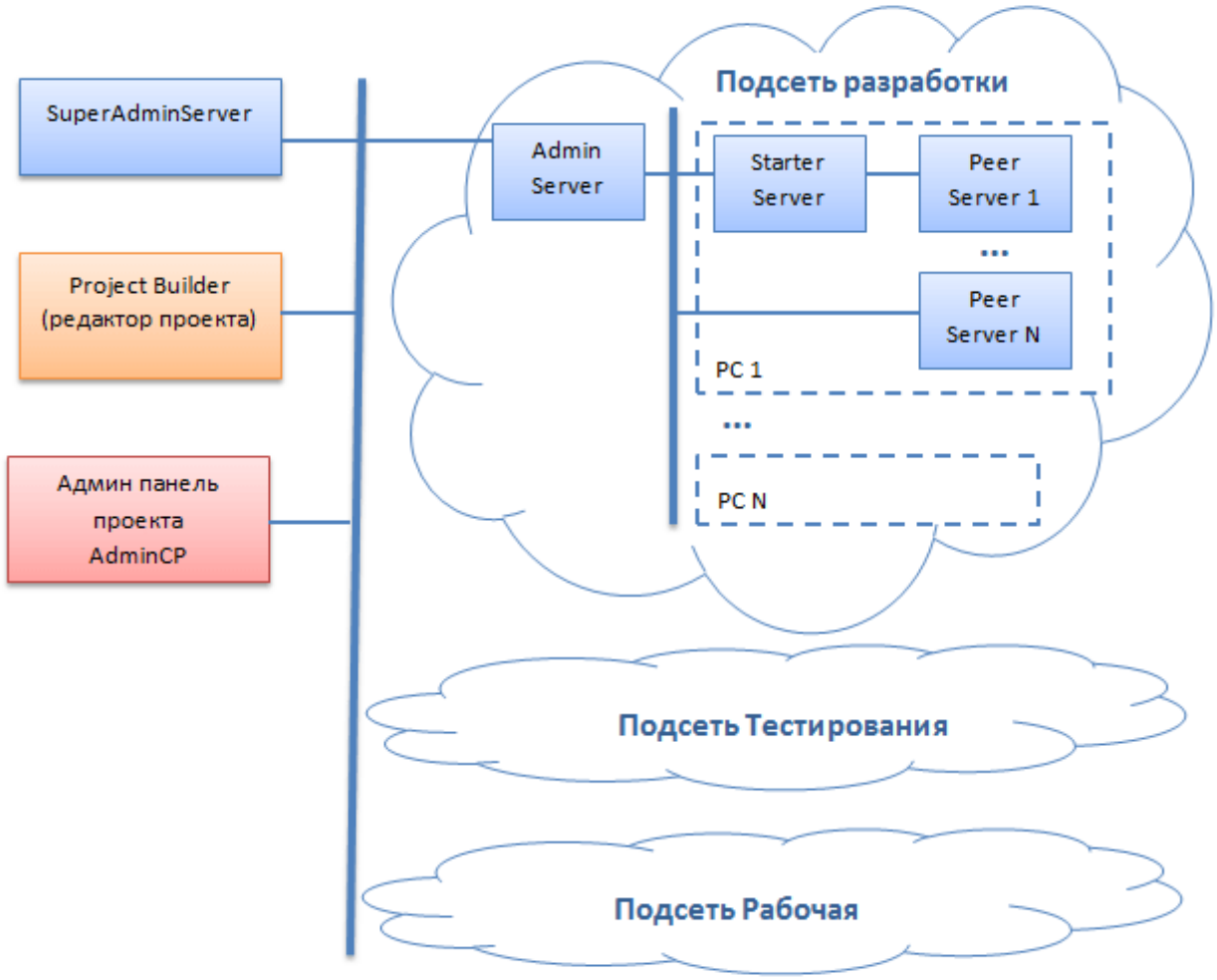
# CrystalEngine

## Структура и принцип работы

Серверное решение, на основе CrystalEngine, легко создается и быстро масштабируется. Все действия по построению выполняются с помощью визуальных редакторов, и администрирует через Admin Control Panel.




Структурная схема серверов проекта

Обычно реализуется минимум 3 подсети для разработки проекта

1) Подсеть разработчика (в ней происходит разработка структуры и модулей логики)
2) Подсеть тестирования – происходит тестирование проекта клиент-сервер
3) Рабочая подсеть с реальными клиентами

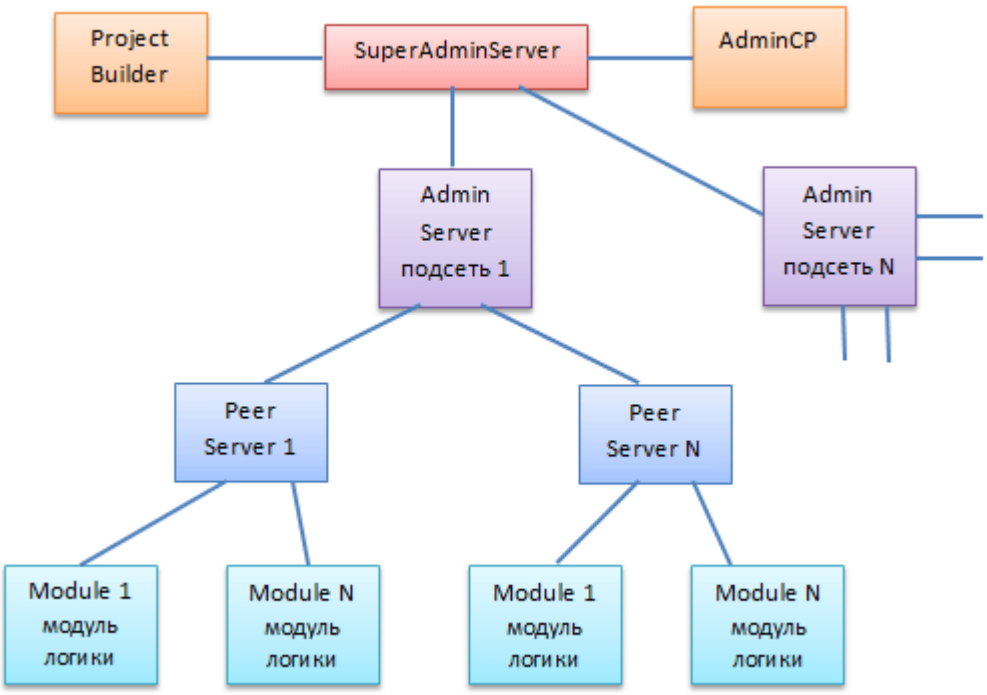
## Состав пакета

В полной версии в состав пакета входят приложения:

- **SuperAdminServer.exe** – предназначен для администрирования подсетей, работы с редактором серверной части (ProjectBuilder), администрирования серверов
- **AdminServer.exe** – служит для администрирования свой подсети
- **StarterServer.exe** – для запуска и остановки рабочих приложений (серверов)
- **PeerServer.exe** – основной, рабочий сервер, запускает и останавливает модули логики, предоставляет транспортные протоколы и сокеты (TCP UDP), в нем происходит работа логики проекта
- **ProjectBuilder.exe** – основной редактор серверной части, позволяет визуально, без написания кода создать структуру данных всего проекта, автоматически создает таблицы в базе данных SQL и файловые хранилища, команды работы с клиентом и межмодульная коммуникация
- **AdminCP.exe** – визуальный редактор для администрирования серверов всего решения, создание рабочих серверов (PeerServer), подключение модулей логики, назначение нужной версии серверной библиотеки для подсетей, работа с архивом

Все приложения могут запускаться как windows сервисы так и как обычные приложения.

Для понимания серверной иерархии обратите внимание на схему ниже



На самом верху находится Супер админ сервер, который производит администрирование всей серверной части и работает с редакторами. Он дает указания Админ серверам а те в свою очередь администрируют своими рабочими серверами. Рабочие сервера управляют своими модулями логики. Такая структура позволяет очень быстро **масштабировать** серверную часть вне зависимости, где находятся сервера физически. Например, при большом наплыве пользователей админ-сервер дает указания и его стартер-сервера запустят дополнительное количество рабочих серверов для обработки клиентов. Либо система может находиться в «облаке» например, Amazon и в этом случает - Админ сервер напрямую взаимодействует с АПИ Amazon и запускает дополнительные мощности. Точно также и при оттоке пользователей.

## Технические требования

Минимальная конфигурация ПК:
Процессор Intel P4 или равнозначный AMD
RAM 500 Mb
HDD 200 Mb
OS Windows server 2003 x86
SQL MySQL, MSSQL
.NET framework 4

Оптимальная конфигурация ПК:
Процессор Intel i7 или равнозначный AMD
RAM 8 Gb
HDD 500 Gb
OS Windows server 2008 x64
SQL MSSQL (возможно бесплатная версия)
.NET framework 4

Накладные расходы серверной части очень малы, поэтому практически вся мощность компьютера отдается на работу логики проекта.
В системе повсеместно используется параллельные потоки обработки данных, поэтому, чем больше система будет иметь встроенных ядер CPU – тем быстрее будет происходить работа логики. На сегодня оптимальная конфигурация ПК стоит порядка 100$ в месяц ( i7 8Gb RAM Windows 2008 x64)

## Требования к персоналу

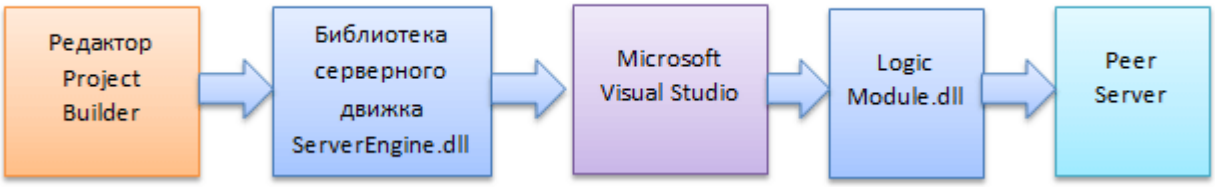
Серверный движок повсеместно использует визуальные редакторы. Настройку и установку серверной части может провести программист среднего уровня. Глубоких познаний в сетевом программировании не требуется.
Для разработки модулей логики требуется разработчик с познанием в математике и хорошим логическим мышлением. То есть в большей степени математик и логик нежели гуру-программист, так, как вся структура объектов выполняется и генерируется с помощью редакторов. Разработчик логики лишь вызывает их в скрипте логики и не задумывается где они хранятся и как доставляются и кэшируются. Все эти функции CrystalEngine выполняет автоматически сам. Например, разработчик для получении имени юзера вызывает лишь одну строку Root.Users[id].Name при этом этот пользователь может быть подключен на другом ПК. Доставку и кеширование данных имени движок выполнит сам.

## Скорость разработки

При использовании среды разработки с визуальными редакторами и генераторами исходного кода скорость разработки увеличивается в 2-3 раза. Разработчику достаточно 3-4 раза кликнуть мышкой и редактор сам сгенерирует десятки строк кода, которые уже отлажены и проверены на работоспособность.

## Порядок разработки

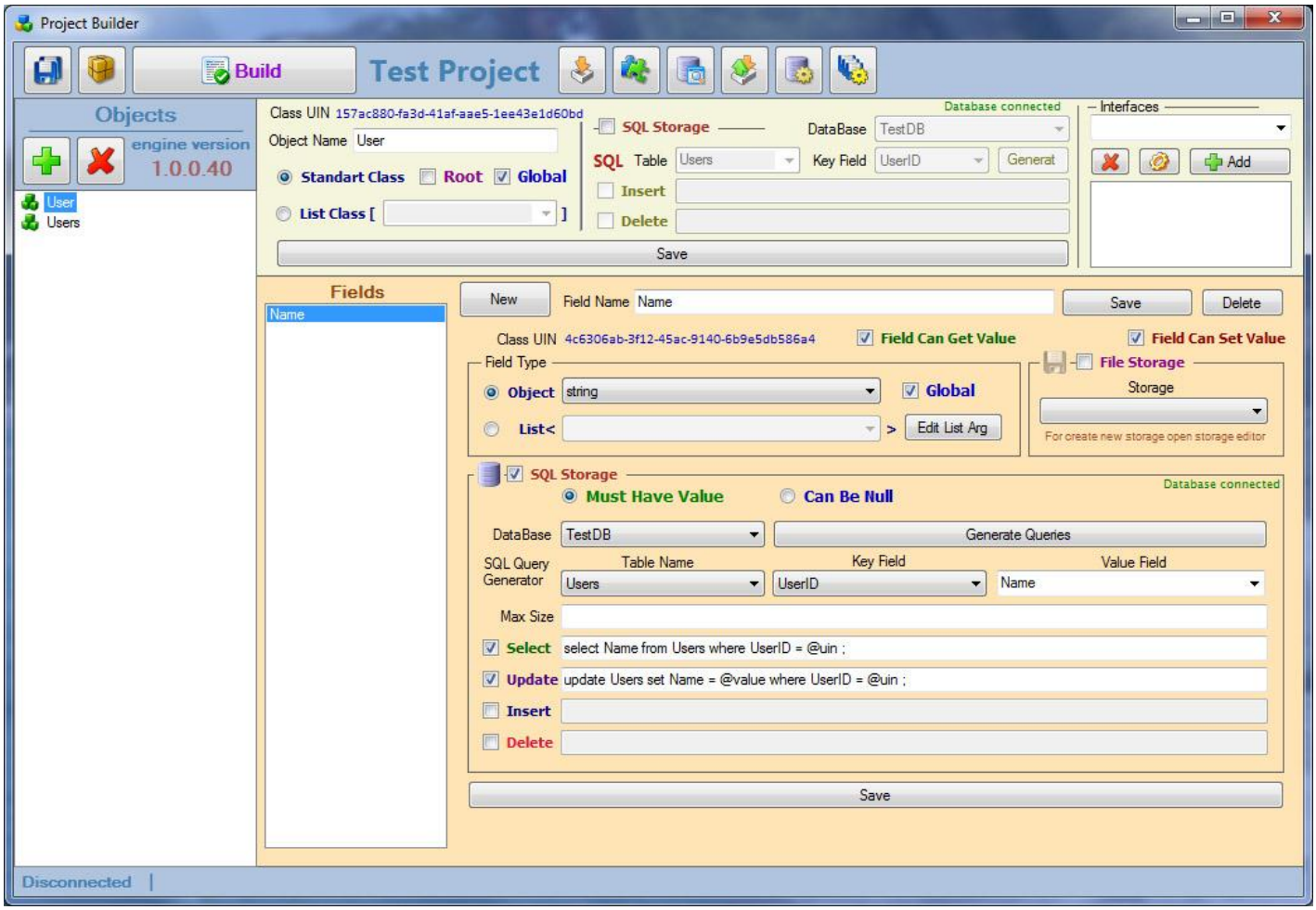
1) Разработчик устанавливает SuperAdminServer и редактор ProjectBuilder
2) С помощью редактора ProjectBuilder создает структуру объектов и команды (все структуры и исходные коды хранятся в месте где установлен SuperAdminServer)
3) Нажатием на одну кнопку «Build» происходит компиляция объектов и команд в библиотеку «ServerEngine.dll» при этом библиотека автоматически подгружается на ПК разработчика. Разработчик подключает библиотеку в Microsoft Visual Studio и с помощью этого лучшего на сегодняшний день отладчика пишет скрипты логики проекта.
4) После создания модулей логики «.dll» разработчик опять же с помощью визуального редактора (AdminCP) подгружает созданные модули на свой сервер и запускает их в работу



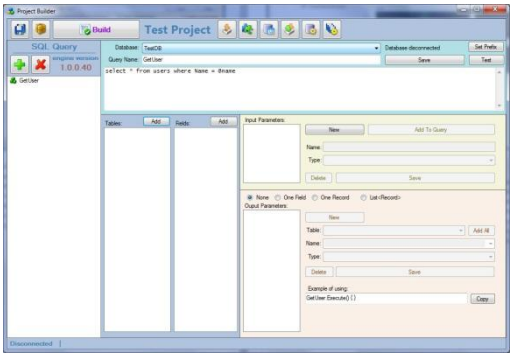

Цепочка создания серверного кода

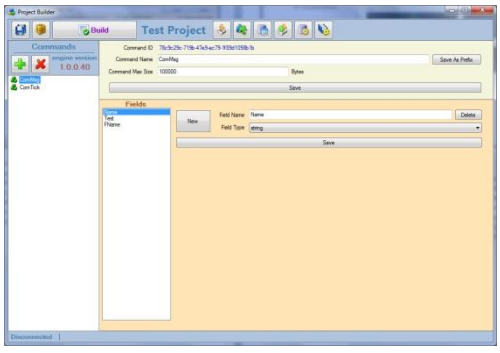
Использованием визуальных редакторов, генераторов кода (код на выходе уже отлажен и стабилен) и применение мощного инструмента Visual Studio для написания скриптов логики достигается высокая надежность серверного кода и снижаются риски проекта со стороны серверной части.

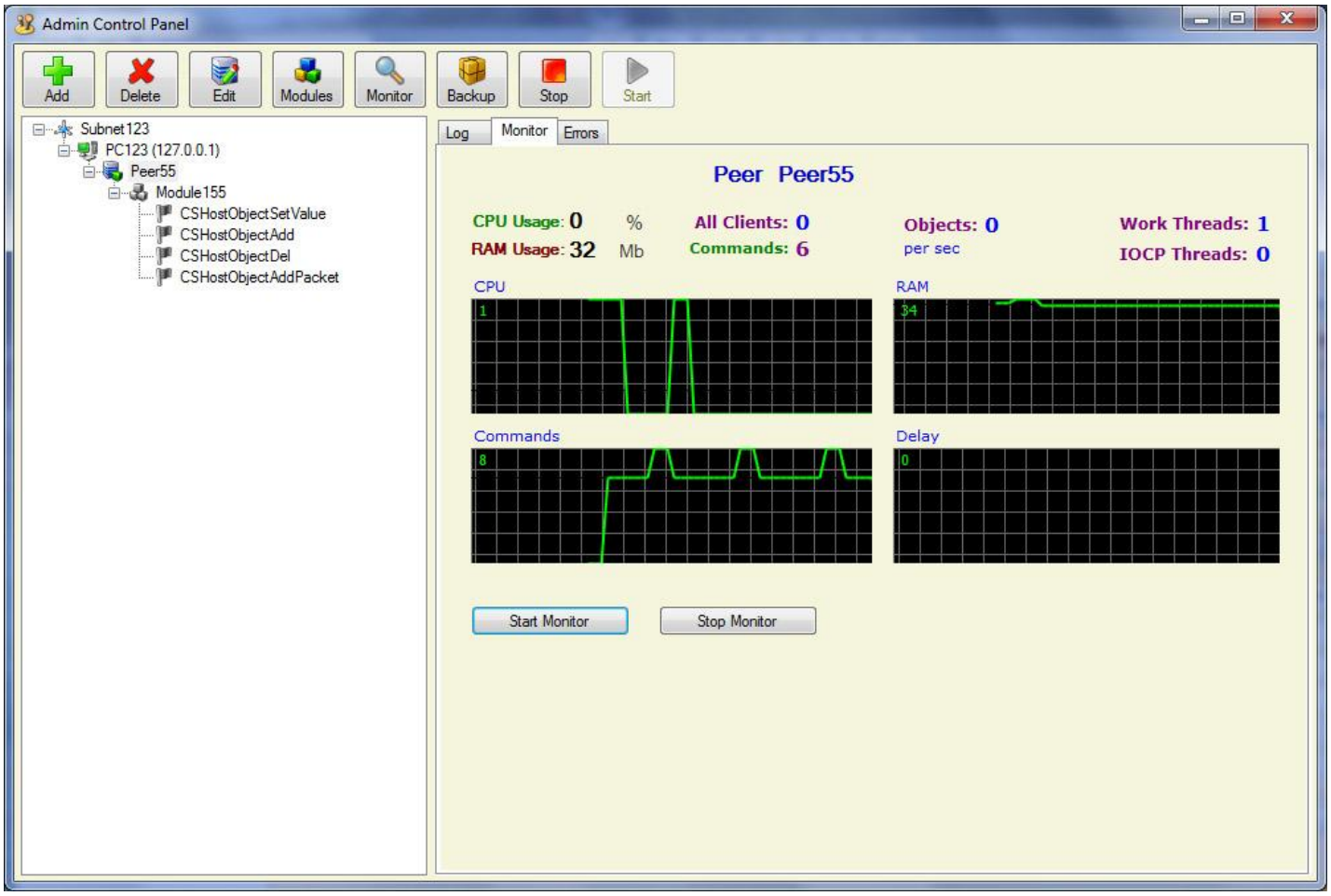
## Скриншоты редакторов




Редактор объектов «Project Builder». При наличии множества настроек, создание объекта происходит очень быстро благодаря уже приготовленным шаблонам и «мастера» создания объектов (Wizard). При этом создание объекта сводится лишь к написанию его имени и выбора его типа, все остальные параметры генерируются автоматически. В дальнейшем при необходимости можно изменить любой параметр. Все редакторы сведены в одно окно и переключаются кнопками сверху. Все редакторы модульные и написаны с учетом структуры движка.



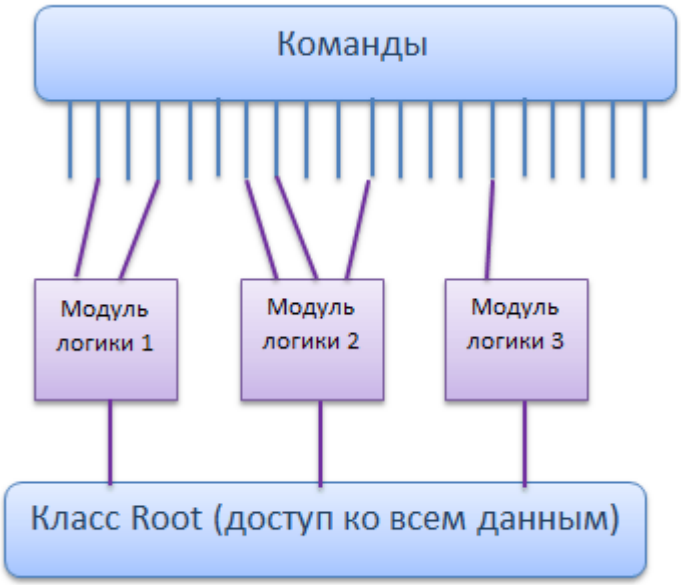





Редактор «AdminCP» с открытой закладкой Монитор сервера Peer55
слева находится дерево серверов, где сразу можно видеть все сервера системы и определить работает в данный момент сервер (подсвечен) или нет (затенен). Путем простого нажатия на кнопки старт и стоп можно запустить или остановить любой сервер. Также можно запустить в работу или перегрузить любой модуль логики на сервер, при этом не останавливая сервер – «**горячий рестарт**», что особенно важно, когда необходимо обновить модуль и при этом не сбрасывая в офлайн всех подключённых клиентов. Также в каждом модуле показаны все команды, которые он обрабатывает.

## Структура модулей логики

Все модули логики представляют собой библиотеки dll (сборки .NET) Каждый модуль логики реализует функции обработки данных, которые доставляются сгенерированной ProjectBuilder библиотекой «ServerEngine.dll». Вызов в работу каждой функции происходит через команды, приходящие от клиента либо от других модулей логики. Если говорить по-простому, то модуль логики подключается к нужным командам и ждет их срабатывания.




При старте – модуль подключается к нужным командам и ждет сигнала к обработке. При остановке – отключается. Через статический класс Root (виден отовсюду) можно получить любые данные :

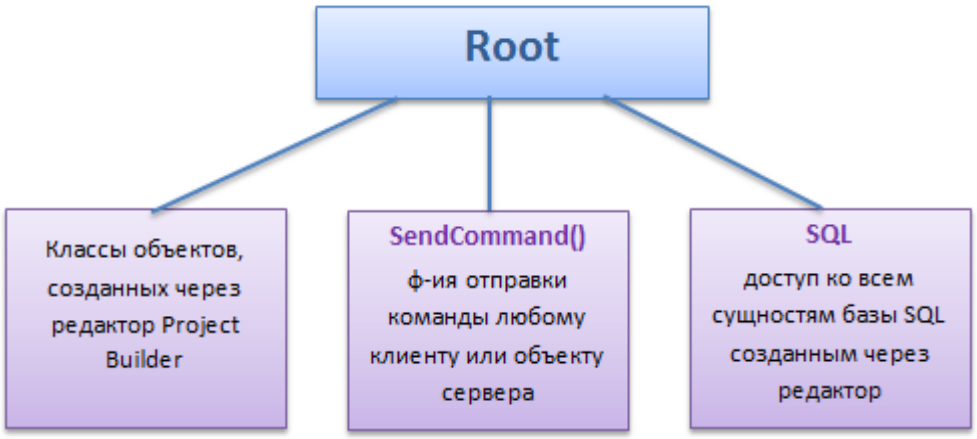
- Из кэша
- Из SQL
- Из файлового хранилища

Всего одной строкой кода

Можно отправить любую команду любому клиенту или другому модулю, любому объекту данных.

## Структура Класса Root

Доступ к любым данным всей серверной части реализован через один класс, что очень упрощает и ускоряет разработку. Достаточно в коде написать слово Root. И редактор сам подскажет все созданные вами объекты и функции.



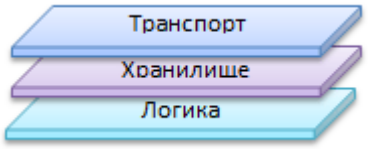
## Перечень основных достоинств движка

- Очень **гибкая структура**, позволяющая создавать самые различные решения. Например, легко можно сделать оптимальный сервер, как для локационных игр, так и для больших бесшовных игровых миров.

- **Любая клиентская часть**. Возможность использования в качестве клиентской части любого игрового движка Unity3D, Flash, UDK, и др. На любой платформе Windows, MacOS, Linux

- Нет ограничений в создании логики проекта. Разработчик может использовать **любые инструменты** среды .NET либо других компиляторов (C++, Delphi).

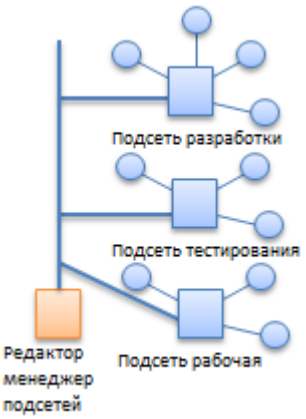
- Транспортная система, система хранения данных отделены от логики, поэтому можно легко менять логику проекта, не изменяю структуру транспорта и хранения данных




- Использование быстрой, распределенной базы **NoSQL в RAM** как оперативное хранилище данных (с которыми непосредственно работает логика проекта), позволяет достигнуть высокой скорости работы с данными. Данные просто хранятся в памяти, непосредственно в сервере. База данных SQL используется только как постоянное хранилище объектов

- **Локальная транспортная система** (связь между серверами) построена на динамических пулах подключений, которые при увеличении нагрузки автоматически добавляют новые подключения, а при уменьшении закрывают. Подключения являются полностью дуплексными, то есть могут одновременно передавать и принимать данные. Отдельно на прием и на передачу.

- **Доменная структура** серверного решения, позволяет легко создать несколько подсетей. Например, подсеть разработчиков, подсеть тестирования, подсеть рабочего проекта. Используя менеджер подсетей, можно легко устанавливать в каждую подсеть различные версии сборок серверного движка, при необходимости сделать откат на более раннюю и стабильную сборку.




- **Резервное хранение** всех версий сборок движка и структур объектов. Автоматически, после каждого нажатия на кнопку «Build» происходит автоматическое сохранение текущей версии сборки. Как уже скомпилированных библиотек движка (для быстрой установки в подсети), так и файлов самих структур (объектов, команд, запросов, сущностей SQL). Все файлы в резервном хранилище удобно разделены в каталогах по дате (год, месяц, день). Доступ и администрирование файлов структур осуществляется через «Backup Manager», который входит в состав Project Builder.

- **Автоматическое создание таблиц и полей** в базе данных SQL для хранения значений создаваемых объектов. Больше нет необходимости думать над структурой базы данных и иметь программиста баз данных. Приложение «Project Builder» автоматически создаст базу данных, таблицы и поля для хранения данных ваших объектов. Удобный «мастер» по созданию объекта и его полей, автоматическое заполнение полей в редакторе и генератор запросов к БД, сэкономит время для работы над логикой проекта. Для «advanced» разработчиков возможна ручная коррекция всех параметров.

- **Удобный доступ к объектам** системы из модулей логики. Разработчик пишет логику проекта в удобной и настроенной под себя среде разработки Microsoft Visual Studio. Каждый модуль логики компилируется в .NET сборку (.dll) и реализует очень простой интерфейс для подключения и отключения от серверной системы. Доступ ко всем объектам серверной части реализуется через статический класс «Root», который доступен отовсюду. Для примера допустим, у нас есть в системе список пользователей «Users», в котором хранятся объекты «User» и имеют поле «Name» хранящее имя пользователя. Для получения имени любого пользователя, который может храниться на любом компьютере серверной части или в Базе SQL в модуле логики достаточно написать одну строку Root.Users[id].Name где id это идентификатор объекта в системе.

- **Прозрачная система идентификации**. В серверном решении в качестве

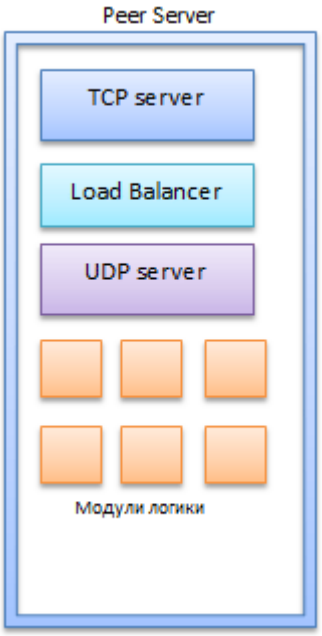
идентификатора всех объектов используется структура Guid, которая повсеместно используется компанией Microsoft в операционных системах. Использование данной структуры, помогает исключить возможность повторения номера и позволяет использовать обобщенные прозрачные списки объектов в базе NoSQL

- **Удобные структуры для передачи команд**. Локальный и глобальный обмен данными в системе производится через структуру команд. Поэтому много внимания уделено не только удобному отображению структуры, но и быстрой упаковке и распаковке данных (низкоуровневая работа с памятью). Отправить команду пользователю, объекту или модулю логики можно через одну универсальную функцию. Root.SendCommand(id, command); где id это может быть идентификатор пользователя, объекта или модуля в системе. Так, как система идентификаторов прозрачна, то сервер сам определит, кому необходимо отправить данную команду.

- **Взаимодействие модулей логики** осуществляется через команды. Каждый модуль должен реализовать у себя две функции «Start()» и «Stop()» в которых он подписывается и отписывается на получение необходимых команд. Модуль может подписаться на команду, в общем, то есть если была отправлена кем-то нецелевая, общая команда, то все модули получат ее. Так и целевая команда с указанием идентификатора получателя. Например, модуль подписался на команду «ComMove» для объекта с идентификатором «id» и обработкой в функции «onMove()». В коде это выглядит следующим образом ComMove.AddEvent(onMove, id); Обязательно наличие парного метода «RemoveEvent()» для отключения подписки на команду. Это необходимо для «горячей» замены модулей. Так, как, не отписавшись от команды, система не даст заменить модуль.

- **«Горячая» замена модулей логики.** В серверном движке предусмотрена горячая замена модулей логики, без перезагрузки приложения сервера, и как следствие без отключения пользователей. Остановка, подгрузка новой версии сборки в модуль и запуск осуществляется нажатием на кнопки «Start» и «Stop» в приложении «Admin Control Panel». При этом сервер продолжает работать

- **Унифицированные сервера**. Каждый подсеть (домен) серверного решения состоит из нескольких типов приложений. Это «AdminServer» «StarterServer» «PeerServer». «AdminServer» предназначен для администрирования других членов своего домена, работы с редактором «Project Builder» и «Admin Control Panel», проверки лицензии и связи с другими доменами. Идентификация подсети осуществляется через «AdminServer». «StarterServer» предназначен для запуска и остановки рабочих приложений серверного решения («PeerServer»)на своем ПК. «PeerServer» является основной рабочей единицей серверной части. Изначально «PeerServer» является «пустым» контейнером для размещения модулей логики и транспортных единиц. Можно представить его как пустую коробку, в которую разработчик может поместить: Балансировщик нагрузки «LoadBalancer», «TCPServer» «UDPServer» любые модули логики, созданные, разработчиком. Таким образом унифицированный «PeerServer» разработчик может превратить либо в Балансировщик нагрузки, либо в транспортный сервер, либо в сервер обработки команд, либо в сервер расчета логики, либо во все вместе.




На данном этапе реализуется гибкость в построении логической структуры серверной части и при правильном ее выборе обуславливает быструю и стабильную работу всей системы в целом для данного конкретного проекта. Ведь очевидно, что оптимальная структура для большого бесшовного мира очень отличается от структуры проекта с небольшими локациями. Возможность реализации любой логической структуры является одним из основных преимуществ использования CrystalEngine.

- **Программные сущности для организации SQL запросов**. Для удобного вызова SQL запросов из кода логики проекта, реализована возможность оперирования исключительно инструментами кода. Функциями и классами. Написав в редакторе «SQL Editor», который входит в состав Project Builder, и оттестировав SQL запрос, автоматически сгенерируется функция для данного запроса и структура результата. В результате код проверяется на наличие ошибок еще до запуска приложения. Вызов запроса осуществляется из кода логики через класс «Root» например ResultUser user = Root.SQL.GetSQLUser(name); в примере происходит вызов запроса выборки с одним входным параметром «name».

- **Связь редактора и среды разработки. Система «OneClick»**. Связь редактора ProjectBuilder и среды разработки Visual Studio осуществляется через библиотеки движка ServerEngine.dll и ClientEngine.dll. Библиотеки автоматически

компилируются на сервере и копируются на ПК разработчика, при этом они устанавливаются в GAC глобальный кеш сборок и доступны для подключения к любым проектам разработчика в VisualStudio на закладке «NET». При внесении изменений в структуру проекта, разработчик лишь нажимает одну кнопку «Build» в редакторе и в VisualStudio доступны все обновления. Что *на много повышает скорость разработки проекта*.

- **Групповая разработка проекта**. Серверный движок CrystalEngine поддерживает групповую разработку на уровне редактора «Project Builder» Все изменения вносимые одним разработчиком становятся видны другим.