**Введение**

ЕГАИС - единая государственная автоматизированная информационная система. ЕГАИС - это огромная база данных, в которой хранится вся информация обо всей алкогольной продукции, когда - либо производимой на территории России, либо ввозимой из других стран. В этой базе содержится информация о составе, производителе, крепости, объеме каждой выпущенной или ввезенной единице продукции. ЕГАИС - это некий инструмент контроля за полным циклом всего оборота алкогольной продукции в России.

ЕГАИС охватывает все этапы производства и оборота алкоголя. Задача системы - необходимость обеспечить контроль от выхода единицы продукции с производства, продажи дистрибьюторами и торговыми предприятиями до ее попадания в корзину потребителя. С 2015 года контроль осуществляется на всех перечисленных этапах.

Начало проекта начинается в 2006 году, когда система по сути и была создана. Уже порядка 14 лет она внедряется на заводах производителей, в несколько этапов:

К 2012 году все процессы учета были отлажены и к системе были подключены все производители и импортеры алкогольной продукции.

* к 2012 году все процессы учета были отлажены и к системе были подключены все производители и импортеры алкогольной продукции;
* c 1 ноября 2015 года обязательным является подключение всех оптовых компаний;
* c 1 июля 2016 года обязательным является подключение всех розничных магазинов;
* c 1 января на всех розничных предприятиях должен быть автоматизирован процесс подтверждения факта закупки алкогольной продукции;
* c 1 июля 2017 года к системе в обязательном порядке должны быть подключены все розничные точки в удаленных населенных пунктах страны;
* с 2018 года подключение к ЕГАИС производителей и импортеров этилового спирта алкогольной и спиртосодержащей продукции.

Регулирующим органом и исполнителем проекта является Федеральная Служба Росалкогольрегулирования (ФС РАР), а главным разработчиком информационной системы является - ФГУП "ЦентрИнформ".

Боевой единицей системы (носителем всей информации о составе, производителе, дате выпуска или ввоза в страну, времени и месте розлива, объеме тары и других уникальных характеристиках каждой единицы продукции) является федеральная специальная акцизная марка (ФСМ). Производитель или импортер оклеивают ей продукцию и сразу передавая эту информацию в ЕГАИС. Марка содержит буквенно-цифровую информацию, которая закодирована также в 2D - штрихкоде PDF417 или micro PDF417 (Тот же принцип работы системы и на рынке табачной продукции).

Информации передается в ЕГАИС при производстве, отгрузке дистрибьютерам, продаже в розничные сети и покупке конечным пользователем, сканер считывает штрих код на акцизной или федеральной специальной марке и передает информацию в кассовую программу, совместимую с модулем ЕГАИС.

Модуль ЕГАИС принимает данные и с помощью программного ключа электронной подписи (его получает предприятие-лицензиат на каждый торговый объект) информация шифруется и передается в систему ЕГАИС, установленную у ФС РАР.

Главным компонентом системы ЕГАИС является Универсальный Транспортный Модуль (УТМ) – это программное обеспечение обеспечивающее обмен данными с базой данных ЕГАИС посредством открытого API интерфейса. API интерфейс УТМ достаточно подробно описан в официальной документации. Все дальнейшее взаимодействие строится на интеграции основной учетной системы организации производителя и УТМ.

На рынке программного обеспечения есть явный лидер в разработке приложений для интеграции и взаимодействия с ЕГАИС - АО "ЦентрИнформ". Крупные разработчики учетных систем (1С, Монолит, Аванкор, Петроглиф и т.д.) так же разрабатывают и поддерживают модули для работы с ЕГАИС в своих продуктах с привычным интерфейсом и поддержкой, но стоит отметить, что ЕГАИС постоянно развивается, изменения в системе происходят регулярно, проявляются особенности бизнес процессов, доработка модулей под требования РАР достаточно дорогая, для небольших производственных компаний это достаточно чувствительная статья расходов. Так же стоит отметить, что зачастую интеграцию приходится поддерживать между продуктами разных компаний разработчиков, что также может стать проблемой.

Решением описанных проблем может стать промежуточный модуль, между программным обеспечением организации производителя и API УТМ. Архитектура такого модуля должна поддерживать работу с API УТМ и с другой стороны, быть проста в интеграции с основной учетной системой организации производителя.

Работа с УТМ представляет собой обмен XML-документами, построенными по определенной схеме. На данный момент УТМ поддерживает список из 70 типов документов разных версий. Для каждого типа УТМ предусмотрена отдельная директория и посредствам протокола HTTP происходит обмен данными в формате XML.

Промежуточный модуль для обмена данными с ЕГАИС должен состоять из двух частей:

* интеграционная часть – реализована на языке SQL и описывает бизнес процессы организации производителя;
* приложение обмена - приложение типа Windows Service - служба, которая опрашивает базу данных и взаимодействует с УТМ сервером посредствам протокола HTTP. Данная технология является частью инфраструктуры Windows и позволяет создавать долговременные исполняемые приложения, которые запускаются в собственных сеансах Windows.

Данное взаимодействие позволяет большую часть логики, а именно: подготовка данных из учетной системы организации, формирование XML-документов, реализация бизнес логики; разрабатывать на языке SQL, что дает возможность построения гибкой и расширяемой системы при правильной организации архитектуры базы данных.

Структура базы данных системы для обмена данными с ЕГАИС должна позволять легко разрабатывать дополнительные модули исходя из особенностей корпоративной информационной системы и видения разработчиком бизнес процессов и с другой стороны не должна навязывать большое количество сущностей для функционирования приложения, так как особенности предметной области различных организаций, работающих с ЕГАИС будут коренным образом отличаться.

**Глава 4. Разработка архитектуры системы**

**4.1 Проектирования главного модуля системы**

**4.1.1 Описание главного модуля**

Главным модулем системы является приложение типа Windows Service - служба, которая опрашивает базу данных и взаимодействует с УТМ сервером по протоколу HTTP.

Структурно, приложение разбито на две части: библиотеку классов UTM\_ExchangeLibrary и приложение службы UTM\_ExchangeService. В библиотеке классов UTM\_ExchangeLibrary реализуется основная логика модуля и определяются основные сущности системы. Библиотекаможет использоваться вне службы, например, для реализации консольного приложения для отправки документов в ручном режиме или тестовом режиме системы при разработке новых версий модулей.

Служба UTM\_ExchangeService запускает цикл с определенной частотой (timeout) и последовательно выполняет следующие процессы приложения:

* сканирование всех УТМ серверов на активность – служба опрашивает каждый УТМ сервер системы, описанный в таблице srv.UTM, если сервер не отвечает на HTTP запрос, то данный узел исключается из дальнейшей обработки;
* удаление файлов с сервера УТМ – обработанные файлы в УТМ удаляются автоматически (для каждого типа обмена установлен свой период удаления, например, для файлов АСИУ удаление происходит каждые три дня). Помимо автоматического удаления, система позволяет удалять файлы на сервере в указанный промежуток времени;
* получение и обработка входящих документов;
* отправка документов в УТМ.

Получение и отправка документов реализуется в асинхронном режиме.

Здесь дальше про настроечную таблицу

**4.1.2 UTM\_ExchangeLibrary**

Библиотека классов UTM\_ExchangeLibrary определяет основные типы и методы, которые необходимы для работы UTM\_ExchangeService.

Разделение основного кода приложения от службы позволяет более гибко разрабатывать дополнительный функционал системы, путем разработки новых библиотек и подключения их к службе, не загромождая при этом основной код новой логикой.

Вторым преимуществом такого подхода является независимое тестирование библиотеки. Для приложения Windows Serviceне доступнастандартная отладка сред разработки до момента установки службы и подключения отладчика к процессу службы. Библиотеку классов можно тестировать в любом типе приложений, например, подключить к консольному приложению и проводить тестирование.

Нужно использовать паттерны!

Паттерны: Fluent Builder, Command, db gateaway

**4.1.3 UTM\_ExchangeService**

UTM\_ExchangeService представляет собой приложение Windows Service или по-другому служба NT. Приложения Windows Service позволяют создавать долговременные исполняемые программы, которые запускаются в собственных сеансах Windows. Для этих служб не предусмотрен пользовательский интерфейс. Они могут запускаться автоматически при загрузке компьютера, их также можно приостанавливать и перезапускать в диспетчере служб Windows. Благодаря этому службы идеально подходят для использования на сервере, а также в ситуациях, когда необходимы долго выполняемые процессы, которые не мешают работе пользователей на том же компьютере. Службы могут выполняться в контексте безопасности определенной учетной записи пользователя, которая отличается от учетной записи вошедшего в систему пользователя или учетной записи компьютера по умолчанию.

**4.2 Проектирование базы данных**

Для разработки ExchangeUTMService естественно использовать жесткую структуру приложения, изменения в которой после компиляции проекта станут невозможными. Клиенту не требуется знать все детали реализации службы для разработки дополнительного функционала системы. Являясь сердцем приложения, ExchangeUTMService инкапсулирует в себе весь функционал необходимый для работы с УТМ.

Другая ситуация с разработкой базы данных приложения. Структура базы должна навязывать минимальное количество сущностей для функционирования приложения, так как особенности предметной области различных организаций, работающих с ЕГАИС будут коренным образом отличаться.

Структура базы данных должна позволять клиенту разрабатывать дополнительные модули исходя из особенностей корпоративной информационной системы и видения разработчиком бизнес процессов.

Для объектов базы данных введены две схемы, схема интеграции – int, и схема сущностей, непосредственно работающих со службой приложения ExchangeUTMService - srv. Типы данных в полях таблиц на схеме 4.2.1 ориентированы на СУБД MSSQL Server, но приложение не привязано к конкретной системе управления базами данных, поддержка конкретной СУБД, а также названия процедур из схемы srv, настраиваются в конфигурации службы ExchangeUTMService.

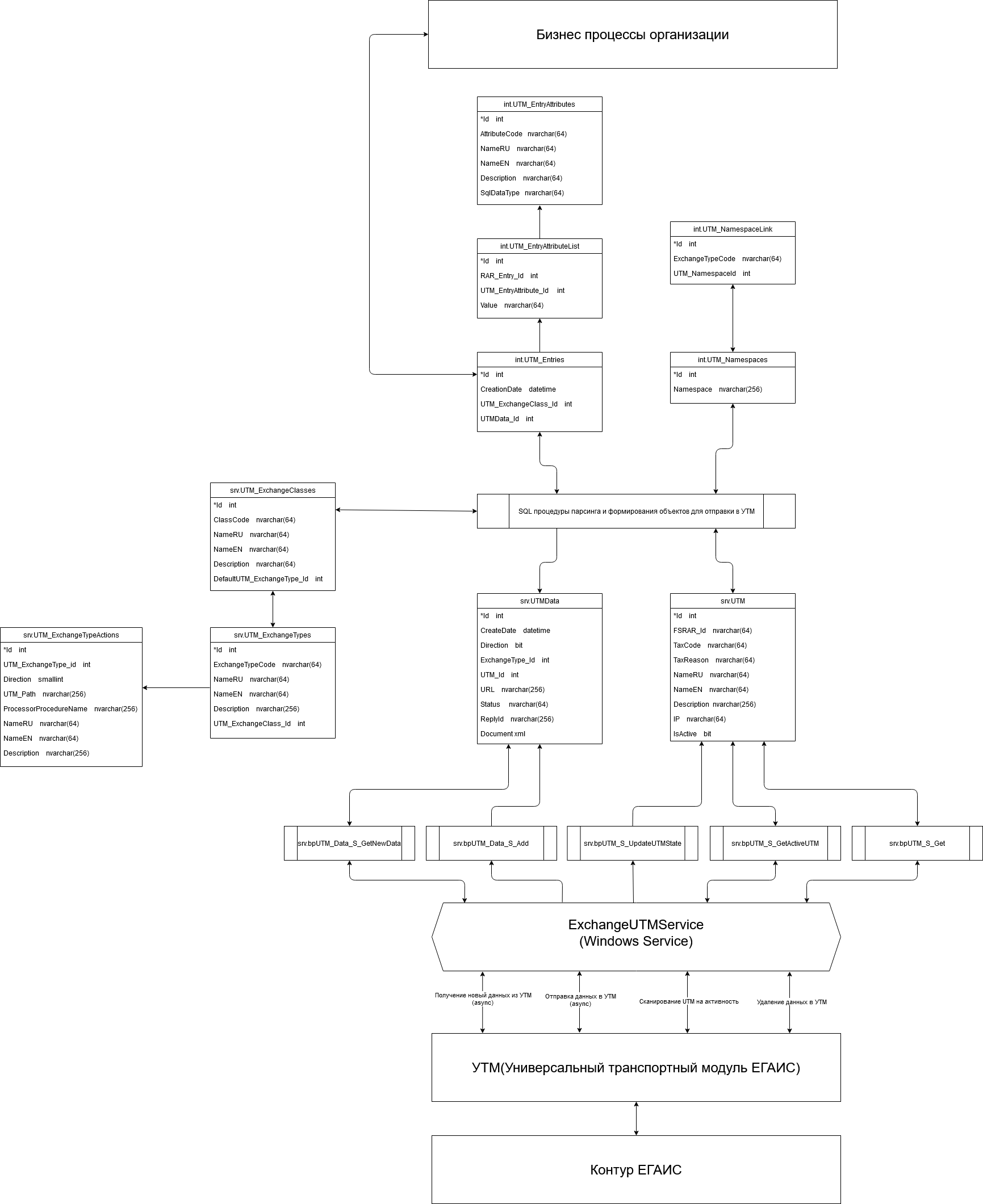


Схема 4.2.1. Базовая структура базы данных

Таблица srv.UTM представляет собой объект, описывающий конкретный УТМ сервер и содержащий всю необходимую информацию для отправки и получения данных.

Описание полей таблицы srv.UTM:

* FSRAR\_Id – уникальный идентификатор организации в РАР, FSRAR ID хранится в УТМ и необходим для всех документов, которые УТМ передаёт в ЕГАИС;
* TaxCode – ИНН организации;
* TaxReason – КПП организации;
* Description – текстовое описание УТМ;
* IP - уникальный сетевой адрес УТМ;
* IsActive – флаг активности УТМ.

Сущность srv.UTMData играет роль буферной таблицы обмена в системе. Данные, обмен которыми происходит с УТМ, на уровне приложения представляют собой документы в XML формате. Методом POST протокола HTTP документы отправляются на API УТМ. Сервис ExchangeUTMService ничего не знает о данных в документе, а выполняет в данном контексте только роль транспорта, помещает входящие документы в таблицу srv.UTMData и получает готовые к отправке документы из нее. Парсинг и сереализацию XML документов клиент может реализовать любой технологией. В данной работе я буду использовать встроенные средства языка T-SQL.

Помимо XML документов таблица srv.UTMData содержит необходимые метаданные, дающие некоторое представление о документе.

Описание полей таблицы srv.UTMData:

* CreateDate – дата и время создания записи;
* Direction – направление движения документа(входящий/исходящий);
* ExchangeType\_Id – внешний ключ на таблицу srv.UTM\_ExchangeType;
* UTM\_Id – внешний ключ на таблицу srv.UTM, определяющий сервер УТМ к которому принадлежит данный документ;
* URL – полный адрес ресурса(документа) на УТМ;
* Status – текущий статус документа;
* ReplyId – идентификатор документа;
* Document – документ XML.

Полное описание API УТМ есть в руководстве «Техническая документация для организаций оптовой и розничной торговли» на официальном сайте РАР\*. Формат XML документов для примеров данной работы также будет подробно описан в приложении 2.

УТМ определяет некоторое количество поддерживаемых типов обменов, версии которых меняются от версии приложения, также, в последующих версиях могут добавляться новые типы обменов. Весь список доступных типов обмена для версии УТМ 4.0.2(последняя доступная версия приложения на момент написания НИР) находится в приложении \*\*.

Каждый тип обмена, по своей сути, определяет некоторый бизнес процесс организации: отправка информации об отгрузке(WayBill), запрос справочника контрагентов(QueryClients), отчет о производстве(RepProducedProduct) и т.д. В последующих, новых версиях для типов обменов могут вводится новые поля для данных, которые организация обязана предоставлять в ЕГАИС посредством отправки документа в УТМ. Все типы обмена в системе разделены на классы и должны храниться в таблице srv.UTM\_ExchangeClasses. Конкретные версии типов обмена находятся в таблице srv.UTM\_ExchangeTypes.

Описание полей таблицы srv.UTM\_ExchangeClasses:

* ClassCode – код класса;
* NameRU – наименование класса кириллицей;
* NameEN – наименование класса латиницей;
* Description - описание класса;
* DefaultUTM\_ExchangeType\_Id – тип обмена, установленный для класса обмена по умолчанию.

Описание полей таблицы srv.UTM\_ExchangeTypes:

* ExchangeTypeCode – код обмена с УТМ;
* NameRU – наименование кода кириллицей;
* NameEN – наименование кода латиницей;
* Description – описание кода;
* UTM\_ExchangeClass\_Id – внешний ключ на таблицу srv.UTM\_ExchangeClass.

На схеме 4.2.1 обозначен блок: «SQL процедуры парсинга и формирования объектов для отправки в УТМ», представляющий собой набор процедур для формирования исходящих и парсинга входящих документов. В зависимости от требующегося функционала, количество таких процедур может значительно меняться. В следующих главах данной работы будет реализован функционал для отправки документов исходящих поставок и парсинга входящих поставок.

Для каждого используемого кода обмена с УТМ должны быть определены как минимум две процедуры: процедура формирования и процедура парсинга. Таблица srv.UTM\_ExchangeTypeActions содержит названия процедур – процессоров и ссылку на конкретный тип обмена с УТМ.

Описание полей таблицы srv.UTM\_ExchangeTypeActions:

* UTM\_ExchangeType\_id - внешний ключ на таблицу srv.UTM\_ExchangeType;
* Direction – направление движения документа(входящий/исходящий);
* UTM\_Path – путь до каталога в УТМ для типа обмена;
* ProcessorProcedureName – название процедуры – процессора;
* NameRU – наименование действия кириллицей;
* NameEN – наименование действия латиницей;
* Description – описание действия.

Каждый отправленный документ из корпоративной информационной системы в УТМ должен фиксироваться в системе и иметь набор определенных описательных полей. Сущность документа в системе отражает таблица int.UTM\_Entries, содержащая минимальное количество полей для описания документа, такие как: дата создания, ссылку на таблицу классов обмена с УТМ - srv.UTM\_ExchangeClasses, и ссылку на буферную таблицу с данными, необходимыми для отправки в УТМ - srv.UTMData.

Описание полей таблицы int.UTM\_Entries:

* CreationDate - дата и время создания записи;
* UTM\_ExchangeClass\_Id - внешний ключ на таблицу srv.UTM\_ExchangeClass;
* UTMData\_Id - внешний ключ на таблицу - srv.UTMData.

В зависимости от процесса, документ может включать значительно большее количество описательных полей, содержащих данные необходимые для отправки. Документ отгрузки(WayBill) включает информацию об отправителе и получателе продукции, данные об отгружаемой продукции, дате отгрузке и отправке и т. д., а запрос остатков(QueryRests) содержит только уникальный идентификатор организации в РАР. Создание таблиц под каждый тип документа повлечет за собой разрастание базы данных, а при появлении новых версий обменов затруднит разработку нового функционала и поддержку новых версий документов. Исходя из этого, все необходимые данные документа представлены в виде атрибутов и должны хранится в таблице int.UTM\_EntryAttributeList, которая содержит ссылку на int.UTM\_Entries и строковое поле Value для хранения значения. Все возможные атрибуты заполняются в таблицу – справочник int.UTM\_EntryAttributes.

Описание полей таблицы int.UTM\_EntryAttributeList:

* RAR\_Entry\_Id – внешний ключ на таблицу int.UTM\_Entries;
* UTM\_EntryAttribute\_Id - внешний ключ на таблицу int.UTM\_EntryAttributes;
* Value – значение атрибута.

Описание полей таблицы int.UTM\_EntryAttributes:

* AttributeCode – системный код атрибута;
* NameRU – наименование атрибута кириллицей;
* NameEN – наименование атрибута латиницей;
* Description – описание атрибута;
* SqlDataType - тип данных значения атрибута.

Листинг с кодом создания таблиц для построения базовой структуры базы данных представлен в приложении 1.

Для работы сервис ExchangeUTMService должен получать и обмениваться данными с базой посредством набора стандартных SQL - хранимых процедур. Названия процедур, непосредственно работающие со службой, прописаны в конфигурационном файле приложения.

Базовые SQL – хранимые процедуры для обмена со службой:

* srv.bpUTM\_Data\_S\_GetNewData – процедура, получает из таблицы srv.UTMData готовые к отправке документы и передает их службе;
* srv.bpUTM\_Data\_S\_Add – процедура, производит вставку входящего документа со всеми необходимыми атрибутами в таблицу srv.UTMData;
* srv.bpUTM\_S\_UpdateUTMState – процедура, которая обновляет состояние УТМ сервера в базе;
* srv.bpUTM\_S\_GetActiveUTM – процедура, которая передает в службу только активные объекты УТМ;
* srv.bpUTM\_S\_Get – процедура, получает все объекты УТМ из базы.

Перечисленных процедур достаточно для непосредственного взаимодействия со службой: передачей и получением данных из УТМ, опрос состояния УТМ сервером и работой с объектами УТМ. Данные процедуры имеют строгий интерфейс, изменить который после компиляции приложения не будет возможности. SQL код данных процедур может модифицироваться в зависимости от требований производительности и бизнес процессов компании, но входящие и исходящие параметры процедур не должны меняться.

**C# и платформа .NET Framework**

Главный модуль системы, приложение типа Windows Service и библиотека классов описывающая основной функционал системы разработана с помощью платформы .NET Framework на языке программирования C# компании Microsoft.

Платформа .NET Framework – это технология, поддерживающая создание и выполнение приложений на базе семейства операционных систем Windows, а также многочисленных операционных систем разработки не Microsoft, таких как Mac OS X и различные дистрибутивы Linux. Наиболее популярными поддерживаемыми платформой языками являются C#, VB.NET, C++, F#.

Платформа .NET Frameworkсостоит из общеязыковой среды выполнения (среды CLR или Common Language Runtime) и библиотеки классов .NET Framework.

Сердце платформы .NET – исполняющая среда CLR, которая представляет собой коллекцию служб, которые требуются для выполнения скомпилированной единицы кода. Исполняющая среда .NET обеспечивает единый уровень выполнения, который разделяется всеми языками и платформами, ориентированными на платформу .NET.

Основной механизм CLR физически имеет вид библиотеки по имени mscoree.dll (также известной как общий механизм выполнения исполняемого кода объектов (Common Object Runtime Execution Engine). Когда на сборку производится ссылка с целью ее использования, библиотека mscoree.dll загружается автоматически и, в свою очередь, загружает требуемую сборку в память. Исполняющая среда отвечает за решение множества задач. Первая и наиглавнейшая задача — определение местоположения сборки и нахождение запрошенного типа в двоичном файле за счет чтения содержащихся в нем метаданных. Затем CLR размещает тип в памяти, преобразует связанный с ним CIL-код в специфичные для платформы инструкции, производит все необходимые проверки безопасности и после этого выполняет нужный код.

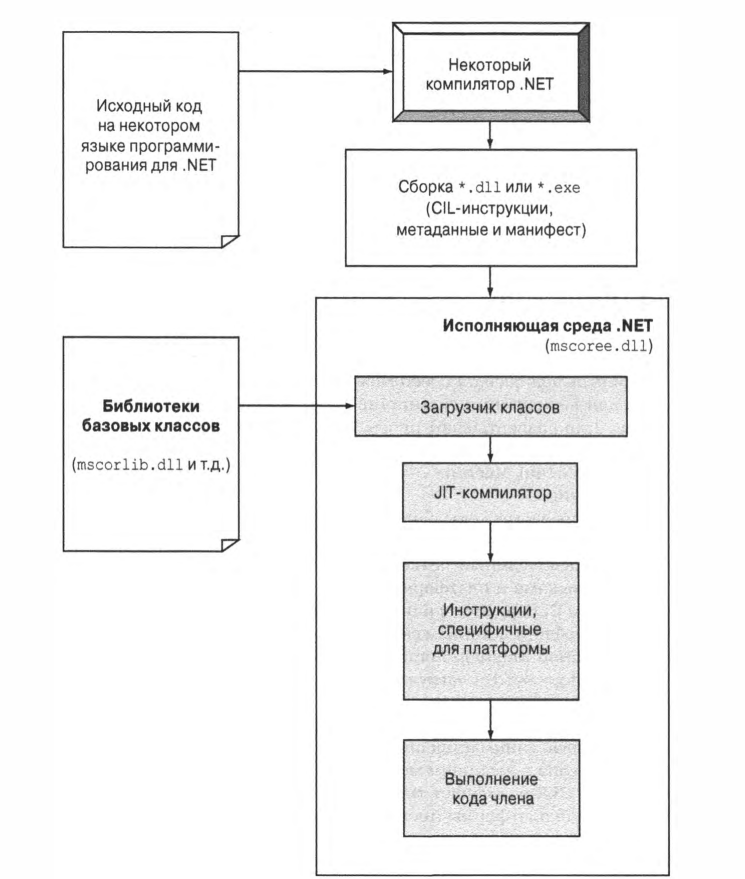


Схема ХХХ - Механизм CLR

В дополнение к загрузке специальных сборок и созданию специальных типов, среда CLR при необходимости будет взаимодействовать с типами, содержащимися в библиотеках базовых классов .NET. Хотя вся библиотека базовых классов разделена на отдельные сборки, главной среди них является сборка mscorlib.dll. Эта сборка содержит большое количество основных типов, которые инкапсулируют широкий спектр общих задач программирования, а также основные типы данных, используемые всеми языками .NET. При построении решений .NET доступ к этой сборке предоставляется автоматически.

**СУБД MSSQL Server и Transact-SQL (T-SQL)**

Основная часть бизнес-логики приложения описывается на языке SQL на стороне базы данных. Система не привязана к конкретной СУБД, для корректной работы UTM\_ExchangeService требуется модуль - библиотека, которая реализует основной функционал для работы с конкретной СУБД, а также название настроечной таблицы в базе данных, где описаны основные хранимые процедуры для работы службы.

В данной работе база данных разработана с использованием СУБД MSSQL Server и языка T-SQL. В главе «Нужна глава для разработки библиотеки C# для работы с Postgresql, а вообще хорошо бы придумать как все это будет работать)))» будет описано создание модуля для поддержки работы с СУБД PostgreSQL.

MSSQL Server — система управления реляционными базами данных разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов с расширениями.

MSSQL Server является одной из наиболее популярных систем управления базами данных для корпоративных приложений. Долгое время MSSQL Server был исключительно системой управления базами данных для Windows, однако начиная с версии 16 система доступна и на дистрибутивах Linux.

В основном MSSQL Server реализуется как клиент-серверная система и может иметь двухзвенную установку либо трехзвенную установку. Независимо от варианта установки, программное обеспечение и базы данных MSSQL Server размещаются на центральном компьютере, который называется сервер базы данных. Пользователи системы работают на отдельных компьютерах, которые называются клиенты.

T-SQL — диалект языка SQL, который предоставляет некоторые расширения при работе с MSSQL Server, такие как: обработка исключений, циклы, курсоры и ветвление.

Важной особенностью MSSQL Server является возможность интеграции со средой CLR, что позволяет разрабатывать функционал на языках платформы .NET Framework, который средствами T-SQL реализовать невозможно. Примером может стать реализация функционала для работы с такими популярными Web-протоколами, как: HTTP, FTP, POP и т.д.

**ПРИЛОЖЕНИЕ 1**

**create** **schema** integration**;**

**GO**

**create** **schema** srv**;**

**GO**

**create** **table** integration**.**UTM\_EntryAttributes**(**

Id int **identity(**1**,**1**)** **primary** **key**

**,**AttributeCode nvarchar**(**64**)** **not** **null**

**,**NameRU nvarchar**(**64**)**

**,**NameEN nvarchar**(**64**)**

**,**Description nvarchar**(**64**)**

**,**SqlDataType nvarchar**(**64**)** **not** **null)**

**create** **table** integration**.**UTM\_EntryAttributeList**(**

Id int **identity(**1**,**1**)** **primary** **key**

**,**UTM\_Entry\_Id int

**,**UTM\_EntryAttribute\_Id int

**,Value** nvarchar**(**64**)**

**,constraint** FK\_UTM\_EntryAttributeList\_UTM\_Entries **foreign** **key(**RAR\_Entry\_Id**)**

**references** integration**.**UTM\_Entries**(**Id**)**

**,constraint** FK\_UTM\_EntryAttributeList\_UTM\_EntryAttributes **foreign** **key(**UTM\_EntryAttribute\_Id**)**

**references** integration**.**UTM\_EntryAttributes**(**Id**))**

**create** **table** integration**.**UTM\_Entries**(**

Id int **identity(**1**,**1**)** **primary** **key**

**,**CreationDate datetime

**,**UTM\_ExchangeClass\_Id int

**,**UTMData\_Id int

**,constraint** FK\_UTM\_Entries\_UTM\_ExchangeClasses **foreign** **key(**UTM\_ExchangeClass\_Id**)**

**references** integration**.**UTM\_ExchangeClasses**(**Id**)**

**,constraint** FK\_UTM\_Entries\_UTMData **foreign** **key(**UTMData\_Id**)**

**references** integration**.**UTMData**(**Id**))**

**create** **table** integration**.**UTM\_NamespaceLink**(**

Id int **identity(**1**,**1**)** **primary** **key**

**,**UTM\_ExchangeType\_Id int

**,**UTM\_NamespaceId int

**,constraint** FK\_UTM\_NamespaceLink\_UTM\_ExchangeTypes **foreign** **key(**UTM\_ExchangeType\_Id**)**

**references** integration**.**UTM\_ExchangeTypes**(**Id**)**

**,constraint** FK\_UTM\_NamespaceLink\_UTM\_Namespaces **foreign** **key(**UTM\_NamespaceId**)**

**references** integration**.**UTM\_Namespaces**(**Id**))**

**create** **table** integration**.**UTM\_NamespaceLink**(**

Id int **identity(**1**,**1**)** **primary** **key**

**,**Namespace nvarchar**(**256**))**

**create** **table** srv**.**UTM\_ExchangeTypes**(**

Id int **identity(**1**,**1**)** **primary** **key**

**,**ExchangeTypeCode nvarchar**(**64**)**

**,**NameRU nvarchar**(**64**)**

**,**NameEN nvarchar**(**64**)**

**,**Description nvarchar**(**256**)**

**,**UTM\_ExchangeClass\_Id int

**,constraint** FK\_UTM\_ExchangeTypes\_UTM\_ExchangeClasses **foreign** **key(**UTM\_ExchangeClass\_Id**)**

**references** srv**.**UTM\_ExchangeClasses**(**Id**))**

**create** **table** srv**.**UTM\_ExchangeClasses**(**

Id int **identity(**1**,**1**)** **primary** **key**

**,**ClassCode nvarchar**(**64**)**

**,**NameRU nvarchar**(**64**)**

**,**NameEN nvarchar**(**64**)**

**,**Description nvarchar**(**256**)**

**,**DefaultUTM\_ExchangeType\_Id int

**,constraint** FK\_UTM\_ExchangeClasses\_UTM\_ExchangeTypes **foreign** **key(**DefaultUTM\_ExchangeType\_Id**)**

**references** srv**.**UTM\_ExchangeTypes**(**Id**))**

**create** **table** srv**.**UTM\_ExchangeTypeActions**(**

Id int **identity(**1**,**1**)** **primary** **key**

**,**UTM\_ExchangeType\_id int

**,**Direction bit

**,**UTM\_Path nvarchar**(**256**)**

**,**ProcessorProcedureName nvarchar**(**256**)**

**,**NameRU nvarchar**(**64**)**

**,**NameEN nvarchar**(**64**)**

**,**Description nvarchar**(**256**)**

**,constraint** FK\_UTM\_ExchangeTypeActions\_UTM\_ExchangeTypes **foreign** **key(**UTM\_ExchangeType\_id**)**

**references** srv**.**UTM\_ExchangeTypes**(**Id**))**

**create** **table** srv**.**UTM**(**

Id int **identity(**1**,**1**)** **primary** **key**

**,**FSRAR\_Id nvarchar**(**64**)**

**,**TaxCode nvarchar**(**64**)**

**,**TaxReason nvarchar**(**64**)**

**,**NameRU nvarchar**(**64**)**

**,**NameEN nvarchar**(**64**)**

**,**Description nvarchar**(**256**)**

**,**IP nvarchar**(**64**)**

**,**IsActive bit**)**

**create** **table** srv**.**UTMData**(**

Id int **identity(**1**,**1**)** **primary** **key**

**,**CreateDate datetime

**,**Direction bit

**,**UTM\_ExchangeType\_id int

**,**UTM\_Id int

**,**URL nvarchar**(**256**)**

**,**Status nvarchar**(**64**)**

**,**ReplyId nvarchar**(**256**)**

**,**Document xml

**,constraint** FK\_UTMData\_UTM\_ExchangeTypes **foreign** **key(**UTM\_ExchangeType\_id**)**

**references** srv**.**UTM\_ExchangeTypes**(**Id**)**

**,constraint** FK\_UTMData\_UTM **foreign** **key(**UTM\_Id**)**

**references** srv**.**UTM**(**Id**))**