НАЦИОНАЛЬНЫЙ ЦЕНТР ПРАВОВОЙ ИНФОРМАЦИИ

**РЕСПУБЛИКИ БЕЛАРУСЬ**

УТВЕРЖДЕН

**ЗЩ. 1530.37477375-0083-ЛУ**

**ИНФОРМАЦИОННО-ПОИСКОВАЯ СИСТЕМА**

**”ЭТАЛОН-ONLINE“**

**Рабочий проект**

**Описание программы**

**ЗЩ. 1530.37477375-0083**

Листов 36

2013СОДЕРЖАНИЕ

[1 Общие сведения 3](#_Toc360524509)

[2 Функциональное назначение 3](#_Toc360524510)

[3 Описание логической структуры 4](#_Toc360524511)

[3.1 Модуль построения ПОД 4](#_Toc360524512)

[3.1.1 Процесс-обработчик 9](#_Toc360524513)

[3.1.2 Служба-балансировщик 15](#_Toc360524514)

[3.1.3 Служба-менеджер 15](#_Toc360524515)

[3.2 Модуль ”Поиск“ 16](#_Toc360524516)

[3.3 Механизм построения ПОЗ 17](#_Toc360524517)

[3.4 Интерфейсная часть модуля ”Поиск“ 18](#_Toc360524518)

[4 Формирование информационного обеспечения системы из лингвистической базы данных на SQL -сервере 20](#_Toc360524519)

[4.1 Используемое программное обеспечение 21](#_Toc360524520)

[4.2 Формирование входных данных из SQL-ЛБД в текстовом формате 23](#_Toc360524521)

[4.3. Выходные данные 26](#_Toc360524522)

[5 Используемое системное программное обеспечение и средства разработки 27](#_Toc360524523)

[6 Описание комплекта поставки 28](#_Toc360524524)

[7 Описание комплекта исходных модулей 29](#_Toc360524525)

[8 Установка, вызов и загрузка 31](#_Toc360524526)

# 1 Общие сведения

Информационно-поисковая система ”ЭТАЛОН-ONLINE“ (далее – ИПС ”ЭТАЛОН-ONLINE“) представляет собой двуязычную систему поиска для ресурсов правовой информации в сети Интернет на основе современных методов компьютерной лингвистики.

# 2 Функциональное назначение

Основными функциями ИПС ”ЭТАЛОН-ONLINE“ являются:

1. Построение поисковых образов документов (далее – ПОД) для банков данных ”Законодательство Республики Беларусь“, ”Решения органов местного управления и самоуправления“, ”Международные договоры“, ”Судебная практика“ и ”Правоприменительная практика“. ПОД представляет собой структуру, в которой каждое слово документа описывается отдельно. Для описания слова используются следующие характеристики:

* каноническая форма (начальная форма слова, например, для имени существительного канонической формой является форма именительного падежа единственного числа мужского рода);
* номер предложения от начала текста, содержащего данное слово;
* номер слова в предложении;
* информативность слова (информативность в данном случае зависит от части речи, к которой относится слово);
* номер символа, с которого данное слово начинается в предложении.

Таким образом, ПОД содержит информацию, необходимую для осуществления поиска, формирования подсветки в тексте документа слов из поискового запроса, выборки короткой фразы из текста (сниппета), по которой пользователь смог бы сделать вывод о соответствии данного документа его требованиям, не открывая документ.

1. Осуществление интеллектуального поиска по текстам документов, а также расширенного поиска по реквизитам. При осуществлении поиска ИПС ”ЭТАЛОН-ONLINE“ выполняет анализ введенной пользователем фразы на естественном языке, в том числе осуществляет:

* перевод запроса пользователя с белорусского языка на русский либо с русского языка на белорусский;
* построение поискового образа запроса (далее – ПОЗ);
* поиск релевантных запросу документов с учетом введенных пользователем в форму расширенного поиска дополнительных атрибутов (вид акта, номер документа и т.д.);
* сортировку результатов поиска с учетом соответствия запросу (релевантности), даты принятия (издания), вида документа.

Также удобными функциями, предоставляемыми ИПС ”ЭТАЛОН-ONLINE“ являются формирование и отображение сниппетов, а также подсветка в тексте документа слов из поискового запроса пользователя.

1. Предоставление пользователям широкого выбора форм оплаты за пользование ИПС ”ЭТАЛОН-ONLINE“:

* оплата по безналичному расчету с заключением договора в печатной форме;
* оплата с использованием систем электронных платежей WebMoney и EasyPay без необходимости заключать договор в печатной форме;
* наиболее упрощенная форма подключения на одни сутки без регистрации с помощью SMS-сообщения, отправленного на короткий номер;
* оплата с помощью системы ”Единое расчетное информационное пространство“ (далее – ЕРИП) также без необходимости заключать договор в печатной форме.

# 3 Описание логической структуры

Система двуязычного полнотекстового поиска ИПС ”ЭТАЛОН-ONLINE“ состоит из нескольких модулей, представленных на рисунке 3.1.

## 3.1 Модуль построения ПОД

Процесс построения ПОД представляет собой обработку текста документа лингвистическим процессором и занесение полученных данных в соответствующие таблицы базы данных etalon1\_index, функционирующей под управлением Microsoft SQL Server 2008. Основными таблицами для хранения ПОД являются Index и VS\_Index (рисунок 3.2).

Таблица VS\_Index представляет собой справочник, в котором хранятся описания всех слов, встречающихся в текстах Эталонного банка данных правовой информации (далее – ЭБДПИ).

В таблице Index хранятся записи, относящиеся к размещению конкретного слова в текстах документа.



Рисунок 3.1 − Структурно-функциональная схема двуязычного полнотекстового поиска



Рисунок 3.2 – Структура таблиц Index и VS\_Index

Рассмотрим более подробно структуру данных таблиц.

Структура таблицы **VS\_Index**:

|  |  |  |
| --- | --- | --- |
| **Наименование столбца** | **Тип** | **Назначение** |
| word | строковый | слово на русском или белорусском языке |
| lingvocode | строковый | лингвистический код слова (далее ЛГК) |
| lang | целое без знака | язык (возможные значения: 0 – русский; 1 – белорусский) |
| hash | двоичный | хэш слова, вычисленный по алгоритму MD5 |
| frequency | целое без знака | частота встречаемости слова в текстах |

Структура таблицы **Index**:

|  |  |  |
| --- | --- | --- |
| **Наименование столбца** | **Тип** | **Назначение** |
| id | целое без знака | идентификатор записи, автоматически формируемый при вставке информации |
| reg\_num | строковый | номер регистрации документа в НЦПИ |
| hash\_real | двоичный | хэш слова (связь с полем hash таблицы VS\_Index) |
| hash\_canon | двоичный | хэш канонической формы слова (связь с полем hash таблицы VS\_Index) |
| sentence | целое без знака | номер предложения в тексте документа, в котором находится слово |

|  |  |  |
| --- | --- | --- |
| **Наименование столбца** | **Тип** | **Назначение** |
| position | целое без знака | позиция слова в предложении |
| position\_info | целое без знака | позиция слова в предложении относительно только информативных слов |
| isinfo | логический | признак информативности слова |
| offset | целое без знака | смещение слова относительно начала текста в байтах |

В процессе разработки и тестирования модуля построения ПОД были выявлены некоторые проблемы, в частности – сравнительно малая скорость индексации за счёт работы в однопоточном режиме, не позволяющем использовать все вычислительные ресурсы многоядерной платформы.

Поэтому с целью увеличения скорости индексирование выполняется в многопоточном режиме. Однако библиотека RBTagger.dll, выполняющая лингвистическую обработку и разбор текста документа, а также используемые для работы данной библиотеки словари, не поддерживают работу в многопоточном режиме. Для решения этой проблемы используются несколько копий программы для индексирования, использующих каждая свою копию библиотеки и словарей (каждую копию данной программы далее будем называть процессом-обработчиком). Процесс-обработчик представляет собой консольное .NET-приложение на языке C#. Для обеспечения функционирования процессов-обработчиков необходимо, чтобы в архитектуре модуля присутствовали части, отвечающие за их корректный запуск, проверку работы и сохранение результатов в таблицы базы данных.

При этом необходимо обеспечить возможность функционирования разных частей модуля на разных серверах в сети НЦПИ. В связи с этим, было решено реализовать модуль в виде распределённого приложения с использованием технологии Windows Communications Foundation (далее – WCF). Технология WCF предоставляет возможность строить приложения, осуществляющие обмен данными посредством сетевых протоколов, в частности – TCP/IP.

Таким образом, мы получаем приложение, которое состоит из нескольких частей:

* Процессов-обработчиков – блоков, выполняющих обращение к библиотеке RBTagger.dll для разбора текста и формирования объекта DataTable, содержащего поисковый индекс документа.
* Службы-инициализатора, отвечающей за запуск процессов-обработчиков.
* Службы-балансировщика, распределяющей задания на обработку текстов между процессами-обработчиками.
* Службы-менеджера, представляющей собой управляющую часть модуля.

Физически модуль построения ПОД представляет собой решение, состоящее из 14 проектов, сгруппированных в 5 разделов: CommonLib, ControlAgent, Shedule, Worker и WorkerAgent. На следующем рисунке представлен вид окна Solution Explorer:

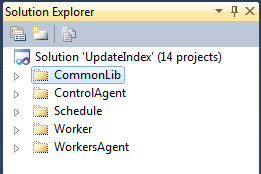


Рисунок 3.3 – Вид окна Solution Explorer

Служба-инциализатор, служба-балансировщик и служба-менеджер реализованы в виде разработанных на языке C# Windows-служб. Благодаря этому функционирование модуля индексатора происходит в фоновом режиме. Службы модуля могут быть настроены таким образом, что будут запускаться автоматически в случае перезагрузки сервера, что позволяет исключить необходимость ежедневного отслеживания их статуса. Службы модуля сохраняют информацию о своей работе в лог-файлах, что позволяет получать информацию о задачах, выполняемых службами в каждый момент времени.

Служба-менеджер ежедневно по заданному расписанию выполняет формирование партии документов, для которых необходимо получить поисковый образ. В случае если партия сформирована и если ее размер равен или превышает заданный, выполняется обращение к службе-инициализатору, осуществляющей запуск необходимого количества процессов-обработчиков. Данное количество также является настраиваемым параметром и может меняться в зависимости от объемов данных, которые необходимо проиндексировать.

### 3.1.1 Процесс-обработчик

Процесс-обработчик занимается непосредственной обработкой текста документа с помощью лингвистического процессора. После запуска каждый процесс направляет запрос службе-балансировщику для получения регистрационного номера документа и приступает к непосредственной обработке текста данного документа.

Входными параметрами для обработчика является собственно сам текст, признак языка (русский, белорусский, смешанный) и признак формата текста (HTML или Plain text). После успешной обработки текста документа обработчик сохраняет ПОД в таблицах базы данных и запрашивает следующее задание.

Технологическая схема работы процедуры индексации не предусматривает наличия какого-либо постоянного числа процессов обработчиков: для обеспечения процесса индексации достаточно одного такого процесса, а при необходимости увеличения скорости обработки нескольких документов достаточно запустить требуемое число обработчиков.

В связи с тем, что все наиболее сложные операции по обработке текстов осуществляются процессами-обработчиками, общая надёжность системы существенно повышается по сравнению с однозвенной схемой работы. Фатальная ошибка одного обработчика в случае возникновения лишь уменьшит общую скорость индексации, в крайнем случае (если ни одного процесса-обработчика не останется) индексация остановится без какой-либо потери данных.

Обобщённая блок-схема алгоритма обработки текста документа представлена на рисунке 3.4.

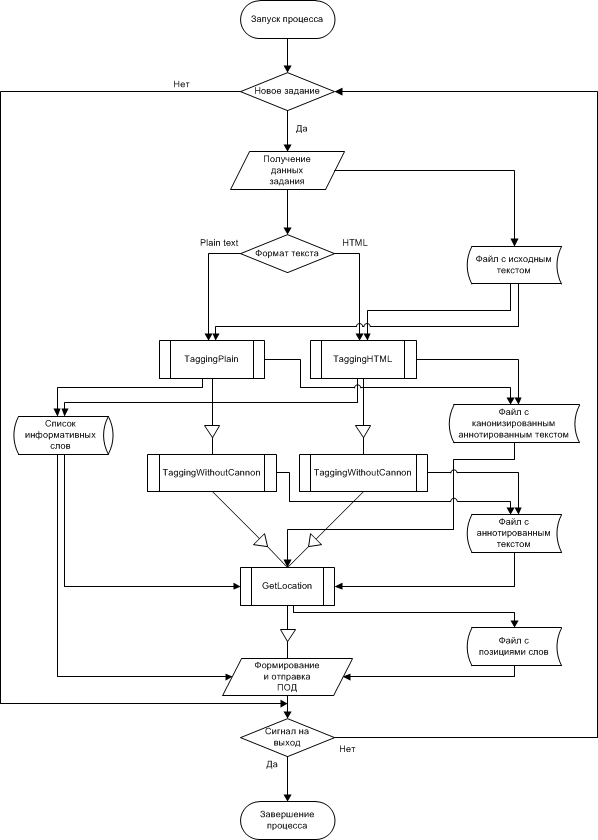


Рисунок 3.4 – Обобщённая блок-схема алгоритма обработки текста документа

Рассмотрим более подробно методы, выполняющие лингвистическую обработку текста.

**Метод TaggingPlain**

Метод применяется для обработки текстов в формате Plain text.

Сигнатура метода:

string TaggingPlain(string InFName,   
string OutFName,   
int Lang)

Метод возвращает строку, представляющую собой список информативных слов документа, приведённых к каноническому виду, с соответствующими весами и, в случае смешанного текста (текста, имеющего фрагменты на русском и белорусском языках), код языка для каждого слова. Кроме того, он же создаёт файл, в котором содержится канонизированный аннотированный текст документа.

Метод принимает следующие параметры:

* string InFName – имя файла с исходным текстом документа;
* string OutFName – имя файла для сохранения канонизированного аннотированного текста документа;
* int Lang – признак языка документа (0 – русский, 1 – белорусский, 2 – смешанный).

Выходная строка формируется по следующему шаблону:

слово\_код[пробел]вес\n

где слово – канонизированное слово текста,

код – ЛГК,

вес – вещественное число с 6 цифрами после запятой,

\n – символ конца строки.

например:

республика\_NNIFO∙1.000000\nуказ\_NNIMO∙0.545455\n

В случае смешанного текста в выходную строку добавляется дополнительный параметр, и шаблон выглядит как:

слово\_код[пробел]вес[пробел]{0|1}\n

где слово – канонизированное слово текста,

код – аббревиатура соответствующего ЛГК,

вес – вещественное число с 6 цифрами после запятой,

0/1 –код языка (0 – русский, 1 – белорусский)

\n – символ конца строки.

например:

рэспублiка\_NNIFO\_1∙1.000000\nуказ\_NNIMO\_0∙0.545455\n

Файл с канонизированным аннотированным текстом имеет следующий формат:

слово\_код[пробел]…\n…

например:

указ\_NNIMO∙президент\_NNAMO∙республика\_NNIFO∙беларусь\_NPIFO\n

В случае смешанного текста этот файл формируется несколько иначе: слово\_код\_{0|1}[пробел]…\n…

Например,

статья\_NNIFO\_0∙1\_DCO∙.\_.\n

закон\_NNIMO\_1∙рэспублiка\_NNIFO\_1∙беларусь\_NPIFO\_1\n

В канонизированном аннотированном тексте код языка указывается только для слов, которые относятся к частям речи.

**Метод TaggingHTML**

Метод применяется для обработки текстов в формате HTML.

Сигнатура метода:

string TaggingHTML(string InFName,   
string OutFName,   
int Lang)

Входные и выходные параметры метода такие же, как и в методе TaggingPlain.

**Метод** **TaggingWithoutCannon**

Метод применяется для обработки текстов в форматах HTML и Plain text.

Сигнатура метода:

void TaggingWithoutCannon( string InFName,  
string OutFName,  
int html,  
int Lang)

Метод выполняет аннотирование текста лексико-грамматическими кодами без приведения слов в каноническую форму.

Метод принимает следующие параметры:

* string InFName – имя файла с исходным текстом документа;
* string OutFName – имя файла для сохранения аннотированного текста документа;
* int html – признак формата документа (0 – Plain tex, 1 – HTML);
* int Lang – признак языка документа (0 – русский, 1 – белорусский, 2 – смешанный).

Формат выходного файла аналогичен описанным выше:

слово\_код[пробел]…\n…

Единственное отличие состоит в том, что слова не приводятся к канонической форме и регистр их написания не меняется, например:

УКАЗ\_NNIMO∙ПРЕЗИДЕНТА\_NNAMG∙РЕСПУБЛИКИ\_NNIFG∙БЕЛАРУСЬ\_NPIFO

В случае смешанных текстов для каждого слова, относящегося к частям речи, дополнительно указывается код языка:

слово\_код\_{0|1}[пробел]…\n…

например:

Статья\_NNIFO\_0∙1\_DCO∙.\_.\n  
Закон\_NNIMO\_1∙Рэспублiкi\_NNIFG\_1∙Беларусь\_NPIFA\_1\n

**Метод** **GetLocation**

Метод применяется для получения позиций информативных слов; эти данные в дальнейшем используются для формирования сниппетов в результатах поиска и отображения слов, релевантных поисковому запросу.

Сигнатура метода:

void GetLocation( string Text,  
string InFName1,  
string InFName2,  
string OutFName)

Используя список информативных слов документа, файлы с аннотированным канонизированным текстом и аннотированным текстом без канонизации, метод генерирует файл, содержащий список информативных слов с указанием всех позиций, начиная с которых слово встречается в аннотированном тексте. Нумерация позиций начинается с единицы для каждого абзаца. Номер абзаца также указывается.

Позиция, с которой слово встречается в тексте, указывается для аннотированного текста без канонизации слов. Т.е. несмотря на то, что полученный список содержит канонизированные слова с указанием соответствующего ЛГК, позиции соответствуют реальным грамматическим формам данного слова в тексте.

Порядок следования слов в полученном файле и в передаваемом списке информативных слов сохранен.

Метод принимает следующие параметры:

* string Text – список информативных слов, полученный методами TaggingPlain или TaggingHTML;
* string InFName1 – имя файла с канонизацией слов, полученного методами, упомянутыми выше;
* string InFName2 – имя файла с аннотированным текстом, полученным после работы метода TaggingWithoutCannon;
* string OutFName – имя выходного файла с результатами работы.

Результатом работы метода является выходной файл со списком слов с позициями. В случае одноязычного текста список имеет следующий формат:

слово\_код[пробел]<абзац>|<позиция>,…,<абзац>|<позиция>,\n

где слово – канонизированное слово текста,

код – аббревиатура соответствующего ЛГК,

<абзац>|<позиция> – пара, содержащая список позиций, начиная с которых слово в нужной грамматической форме встречается в тексте. Первым указывается абзац, где содержится данное слово, затем через вертикальную черту ”|“ – позиция начала слова в соответствующем абзаце. Каждая такая пара отделяется от предшествующей запятой ”,“. Запятая стоит и после последней пары.

\n – символ конца строки.

Например:

юридический\_JJMO∙98|73,\n  
являться\_VILC∙28|42,89|50,102|121,107|130,123|97,133|52,\n  
язык\_NNIMO∙42|318,112|226,187|61,\n

Для смешанных текстов дополнительно указывается код языка каждого конкретного слова (за исключением цифр), и формат списка выглядит следующим образом:

слово\_код\_{0|1}[пробел]<абзац>|<позиция>,…,<абзац>|<позиция>\n

где слово – канонизированное слово текста,

код – аббревиатура соответствующего ЛГК,

0/1 – код языка (0 – русский, 1 – белорусский),

<абзац>|<позиция> – пара, содержащая список позиций, начиная с которых слово в нужной грамматической форме встречается в тексте.

\n – символ конца строки.

Например:

87\_DCO∙3|144,\n  
акт\_NNIMO\_0∙3|63,\n  
беларусь\_NPIFO\_1∙2|33,2|162,3|97,\n  
реестр\_NNIMO\_0∙3|32,\n

### 3.1.2 Служба-балансировщик

Данный процесс реализован в виде windows-службы, которая, с одной стороны, воспринимает запросы от службы-менеджера и принимает её задания, а с другой стороны ведёт учёт зарегистрировавшихся процессов-обработчиков и распределяет среди них имеющиеся задания. В случае отсутствия свободных обработчиков приём заданий временно прекращается.

Служба-балансировщик обладает высокой надёжностью за счёт исполнения в управляемой среде и простоты алгоритма функционирования.

### 3.1.3 Служба-менеджер

Служба-менеджер реализована в виде windows-службы. Служба менеджер осуществляет управление схемой работы алгоритма индексации. Она осуществляет обращение ко всем остальным службам по определенному расписанию и осуществляет мониторинг их состояния.

Поскольку все вычислительно ёмкие операции вынесены в процессы-обработчики, данная процедура является высоконадёжной.

Обобщённая схема алгоритма работы процедуры-менеджера приведена на рисунке 4.



Рисунок 3.5 – Обобщённая схема алгоритма работы процедуры-менеджера

## 3.2 Модуль ”Поиск“

Модуль ”Поиск“ включает в себя всю функциональность, необходимую для осуществления интеллектуального поиска по текстам документов и расширенного поиска по реквизитам. При осуществлении интеллектуального поиска ИПС ”ЭТАЛОН-ONLINE“ выполняет построение ПОЗ, поиск релевантных запросу документов, с учетом введенных пользователем в форму расширенного поиска дополнительных атрибутов, сортировку результатов с учетом соответствия запросу (релевантности), даты принятия (издания), вида документа, формирует сниппеты и подсветку.

Модуль ”Поиск“ реализован в виде ASP.NET http-handler на языке C#. Помимо осуществления функций поиска, данный сервис обеспечивает также получение информации из справочников ”Вид акта (документа)“, ”Орган принятия (издания)“, ”Ключевые слова“, ”Государства-участники“, а также доступ к рубрикам Единого правового классификатора.

Сервис может использоваться сторонними организациями для предоставления своим пользователям поисковых возможностей ИПС ”ЭТАЛОН-ONLINE“ на своих сайтах. Доступ к сервису осуществляется по протоколу HTTP методом POST (адрес сервиса: http://etalonline.by/api/XMLProcessor.ashx). Полная спецификация сервиса приводится в документе ”Описание сервиса“.

## 3.3 Механизм построения ПОЗ

Механизм построения ПОЗ в общих чертах схож с построением ПОД. Фактически, ПОЗ представляет собой список информативных слов запроса, с соответствующими ЛГК. Поскольку при поиске документов нужно реализовать двуязычность, то поисковый запрос перед разбором переводится (исходный язык определяется автоматически), и список информативных слов строится для обоих вариантов. Все слова в списке представлены в канонической форме и идут в алфавитном порядке.

Для формирования ПОЗ приложение использует COM-объект, реализованный на языке программирования C++ в виде динамически загружаемой библиотеки. Реализация объекта позволяет обрабатывать запросы в многопользовательском режиме, что существенно при использовании его в web-приложении.

Перед обработкой запросов анализатор должен быть инициализирован; при этом в память загружаются словари. В случае использования анализатора в приложении ASP.NET, инициализация должна быть произведена в методе Application\_Start вызовом метода Init. Сигнатура метода:

void ISemanticProcessor.Init(string sLdbPath)

где sLdbPath – путь к каталогу ЛБЗ.

После инициализации запросы могут быть обработаны методом ParseQuery. Сигнатура метода:

string ISemanticProcessor.ParseQuery(string sQuery)

где sQuery – поисковый запрос пользователя.

Данный метод возвращает ПОЗ в виде нуль-терминированной строки, содержащей список информативных слов, приведенных к каноническому виду и упорядоченных по алфавиту. Каждому слову приписан его ЛГК (через символ подчеркивания). Сначала приводится список слов для русского языка, затем, через точку запятой – для белорусского.

Например, при запросе ”Как пройти в библиотеку?“ результат работы метода будет выглядеть следующим образом:

библиотека\_NNIFO\n  
пройти\_VPC\n  
;бiблiятэка\_NNIFO\n  
прайсцi\_VPC\n

## 3.4 Интерфейсная часть модуля ”Поиск“

Интерфейсная часть ИПС ”ЭТАЛОН-ONLINE“ представляет собой сайт ASP.NET, разработанный с использованием технологии AJAX и библиотеки JQuery. На следующем рисунке представлен вид окна Solution Explorer:



Рисунок 3.6 – Вид окна Solution Explorer

Помимо сервиса для осуществления поиска, для функционирования ИПС ”ЭТАЛОН-ONLINE“ используются дополнительные сервисы:

Сервис для получения карточки и текста документа.

Сервис, предоставляющий функции для работы с пользователями, в том числе:

* вход пользователя в ИПС ”ЭТАЛОН-ONLINE“;
* регистрация пользователя в ИПС ” ЭТАЛОН-ONLINE“;
* получение и обновление информации о пользователе (фамилия, имя, отчество, номер телефона, адрес электронной почты) для отображения ее в личном кабинете;
* смена пароля пользователя;
* взаимодействие с платежными системами для выписки счета.

Для обмена информацией с сервисами используется формат XML.

# 4 Формирование информационного обеспечения системы из лингвистической базы данных на SQL -сервере

Лингвистическая база данных (ЛБД) ИПС ЭТАЛОН логически состоит из следующих частей:

* лексико-грамматического классификатора белорусского и русского языков;
* аннотированного словаря белорусского языка;
* аннотированного словаря русского языка;
* аннотированного белорусско-русского словаря;
* аннотированного русско-белорусского словаря;
* базы данных распознающих лингвистических моделей, которая содержит так называемые паттерны, формализующие языковую компетенцию в целях автоматического анализа текста, в том числе лексико-грамматического анализа.

Одноязычные словари используются при решении задачи морфологических преобразований слов в пределах одной парадигмы (перехода от любой словарной формы к ее канонической форме). Двуязычные словари реализуют функцию машинного перевода запросов, вводимых при поиске, с русского языка на белорусский и с белорусского языка на русский.

Физическая организация ЛБД представляет собой совокупность файлов как в текстовом, так и в специальном, бинарном, формате, поэтому по отношению к ЛБД используется понятие «бинарного словаря» или «бинарной ЛБД». Данный словарь формируется из лингвистической базы данных на SQL-сервере (SQL-ЛБД) путем выгрузки данных в особый текстовый формат, который описан в разделе 5.

Для формирования бинарного словаря нужен также корпус текстов на русском и белорусском языках как аннотированный, так и с оригинальными текстами в текстовом формате для снятия статистики.

Все словари и тексты должны быть представлены в кодировке CP1251, причем особое внимание следует уделить кодированию белорусской буквы *«і»* для текстов на белорусском языке в соответствии с государственным стандартом Республики Беларусь СТБ 956-2004 «Информационные технологии. Системы обработки информации. Наборы 8- и 16-битных кодированных символов белорусского алфавита. Інфармацыйныя тэхналогіі. Сістэмы апрацоўкі інфармацыі. Наборы 8- і 16-бітных кадзіраваных сімвалаў беларускага алфавіта», который был утвержден и введен в действие с 1 июля 2005 года.

## 4.1 Используемое программное обеспечение

Программное обеспечение, используемое для решения задачи формирования бинарной ЛБД, находится в каталоге с именем RBDictcompiler и до выполнения программ содержит:

* папку ldbscr для размещения входных данных;
* папку Release с исполнимыми файлами формирования бинарной ЛБД;
* папку Scripts со вспомогательными скриптами на языке PERL, также необходимыми для формирования бинарной ЛБД;
* файл Makefile – файл, содержащий набор инструкций для программы make, позволяющей автоматизировать процесс выполнения различных действий;
* файл MakeLDB.bat– файл, запускающий процесс поэтапного формирования бинарной ЛБД.

Для формирования бинарной ЛБД достаточно запустить на выполнение указанный bat-файл.

После завершения формирования бинарной ЛБД в каталог RBDictcompiler добавятся папки:

* папка ldb, в которой будут размещаться файлы бинарной ЛБД;
* временные папки tmp и tmp\_ldb с промежуточными данными, которые следует удалить;
* папка log с сообщениями об ошибках и времени выполнения программ в случае их успешной работы.
* 4.2 Входные данные
* Входные данные для формирования бинарной ЛБД находятся в каталоге ldbscr, который, в свою очередь, содержит:
* папку bin – набор утилит, необходимых для обеспечения процесса компиляции ЛБД;
* папку Corpora – наборы корпусов текстов в текстовом формате для сбора статистики, которая используется при анализе текста и для выбора переводных эквивалентов. Эта папка содержит подпапки BY и Ru для белорусского и русского языков соответственно. Каждая из этих подпапок в свою очередь имеет 3 подпапки:
* подпапка plain – содержит исходные тексты на соответствующем языке до их «раскраски» (имена файлов произвольные);
* подпапка tagged – «раскрашенные» тексты на соответствующем языке (имена файлов произвольные);
* подпапка stat содержит исходный файл с начальными статистическими данными.
* папку Dictionary, содержащую базовые двуязычные словари в исходном формате. В ней находятся:
* подпапка Bilingual непосредственно содержит двуязычные словари. В этой папке уже находятся файлы Punkt (знаки препинания) и Preposition (предлоги), и именно в эту папку следует поместить файлы, получаемые из SQL-ЛБД;
* подпапка Prefix с файлом русско-белорусских префиксов, получаемым из SQL-ЛБД;
* файл TagPriority – текстовый файл с классификатором, содержащим коды порядке, который соответствует расположению кодов в парадигме исходных словарей в текстовом формате;
* файл TranslationFilters.txt – текстовый файл с правилами коррекции таблиц перевода: > из таблицы удаляется русско-белорусский перевод, < удаляется белорусско-русский перевод (только переводы, в базовых словарях слова остаются). Могут быть только однословные слова;
* папку TagSet с ресурсами для построения классификатора и структур описания каждого кода в виде текстовых файлов:
* файл Par.txt описания кодов парадигм;
* файл WF.txt описания кодов словоформ;
* файл link.txt– связи кодов парадигм и словоформ;
* файл Tags.stat с частотой встречаемости тегов – используется при снятии омонимии, когда правилами ее разрешить не удается; формируется из SQL-ЛБД
* файл TagSet.h – вспомогательный, используется для формирования файла TagList.data;
* папку Translator с ресурсами систем лингвистического анализа текста и системы перевода в виде текстовых файлов:
* файлы BRTransCorrect.txt, RBTransCorrect.txt Par.txt с правилами коррекции перевода белорусско-русского и русско-белорусского соответственно;
* файлы BelPriorityRules.txt, RusPriorityRules.txt с приоритетами слов при выборе переводного эквивалента вне зависимости от частоты слов для белорусского и русского языков соответственно;
* файл PrepCorrection.src – с правилами коррекции предлогов в зависимости от контекста.

В эту же папку помещаются следующие файлы, получаемые из SQL-ЛБД:

* файлы BelWBD.txt, RusWBD.txt с правилами распознавания границ предложений для белорусского и русского языков соответственно;
* файлы BelSBD.txt, RusSBD.txt с правилами распознавания границ предложений для белорусского и русского языков соответственно;
* файлы VerbRuling\*.src – 4 файла глагольного управления (предложного и беспредложного) для белорусского и русского языков соответственно;
* файл TransDependTreeBuilder.lua с наборами правил анализа по технологии LUA (порядок их вызова).

## 4.2 Формирование входных данных из SQL-ЛБД в текстовом формате

Для формирования бинарной ЛБД в каталог ldbscr нужно выполнить выгрузку из SQL-ЛБД в следующие текстовые файлы:

* в подпапку Bilingual папки Dictionary выгрузить все части речи, за исключением предлогов и знаков препинания (файлы Preposition и уже находятся в папке). Выгрузка двуязычного словаря выполняется из таблиц WordBasis , WordTermination, TerminationGroup, WordBasisTree и таблиц соответствия при переводе Trans, TransWF, TransMW1, TransMW2.

Слова по частям речи выгружаются в так называемые двуязычные парадигмы. Не должно быть парадигм длиннее 32 слов.

**Структура парадигмы двуязычного словаря приведена на рисунке.**

**НАЧАЛО**

***–КОД ПАРАДИГМЫ(рус.яз)= КОД ПАРАДИГМЫ(бел.яз)***

*Словоформа1(рус.яз.)\_КОД=Словоформа1(бел.яз.)\_КОД*

**Словоформа2(рус.яз.)\_КОД=Словоформа2(бел.яз.)\_КОД**

**Словоформа3(рус.яз.)\_КОД=Словоформа3(бел.яз.)\_КОД**

**Словоформа(Х-1)(рус.яз.)\_КОД=Словоформа(Х-1)(бел.яз.)\_КОД**

**Словоформа(Х)(рус.яз.)\_КОД=Словоформа(Х)(бел.яз.)\_КОД**

**КОНЕЦ**

Рисунок 4.1 – Структура парадигмы русско-белорусского словаря

Например, для русско-белорусского словаря парадигма имеет вид:

***–NJІР=NJІР***

***данные\_О=даныя\_О***

***данных\_G=даных\_G***

***данным\_D=даным\_D***

***данные\_А=даныя\_А***

***данными\_І=данымі\_І***

***данных\_R=даных\_R***

***+***

где NJІР – имя существительное, субстантивированное прилагательное, нарицательное, употребляемое только во множественном числе,

О, G, D, А, І, R – соответственно обозначают именительный, родительный, дательный падежи.

Порядок расположения кодов в парадигме определяется файлом *link.txt* папки *TagSet.* Формат записи в этом файле:

***<общий код парадигмы>[\t<словоизменительный код>]***

Отдельно формируются файлы для глагола, имени существительного и прилагательного, имен собственных. Имена файлов – произвольные.

Для всех причастий, в том числе и со словосочетаниями – имя файла *Participle\_sum.* Для всех остальных словосочетаний (вне зависимости от части речи) имя единого выходного файла *Word\_comb\_sum.* Оставшиеся части речи помещаются в файл с именем *Others\_sum.*

Парадигмы не обязательно отсортированы по алфавиту.

* в подпапку *Prefix* папки *Dictionary* выгружаются русско-белорусские префиксы. Имя файла *Prefix*, формат:

***<русский префикс>=<белорусский префикс>***

**Выгрузка словаря выполняется из таблицы *WordPrefix.***

* в папку *TagSet* записывается файл *Tags.stat* со статистикой кодов. Выгрузка выполняется из таблицы *TagList.*

Формат записи:

***<значение частоты>\t<полный код слова>***

При этом значение частоты выровнено по правому краю, полный код – по левому. Например:

**18251***\t***CC**

**11923***\t***INR**

**10329***\t***NNING**

**9750***\t***NNIFG**

**7375***\t***NNIMG**

* в папку *Translator*  выгружаются в текстовом формате файлы *BelWBD.txt, RusWBD.txt, BelSBD.txt, RusSBD.txt. TransDependTreeBuilder.lua* (см. Входные данные). Выгрузка выполняется из таблицы правил *Rules.*

Файлы глагольного управления также выгружаются в текстовые файлы с именами:

* + - *VerbRulingBel.src* и *VerbRulingBelPrep.src* – беспредложное и предложное глагольное управление для белорусского языка;
    - *VerbRulingRus.src* и *VerbRulingRusPrep.src* – беспредложное и предложное глагольное управление для русского языка.

Выгрузка выполняется из таблицы *VerbRuling.*

Формат записи:

***глагол\t[+***|***–\t…]***

где «+» - есть управление

«–» - нет управления

Для беспредложного управления последовательность данных знаков соответствует следующим четырем падежам: родительному, дательному, винительному, творительному. Для предложного управления – пяти: родительному, дательному, винительному, творительному и предложному.

Например*, беречься\t+\t-\t-\t-*

## 4.3. Выходные данные

Физически ЛБД в бинарном формате после завершения выполнения комплекса программ по формированию бинарного словаря располагается в папке *LDB* и содержит следующие бинарные файлы:

1) Базовые словари:

*– BelDictionary.dat* – словарь белорусского языка;

*–  RusDictionary.dat* – словарь русского языка.

2)*TagList.data* – лексико-грамматический классификатор для белорусского и русского языков.

3)Таблицы соответствия парадигм при переводе:

*–  TranslationBR.tbl* – при переводе с белорусского языка на русский;

*–  TranslationRB.tbl* – при переводе с русского языка на белорусский.

4)Таблицы глагольного управления:

для белорусского языка:

*– VerbRulingBelPrep.bin* – предложное управление;

*– VerbRulingBel.bin* – беспредложное управление;

для русского языка:

*– VerbRulingRuslPrep.bin* – предложное управление;

*– VerbRulingRus.bin* – беспредложное управление.

5)Файлы со статистикой словоупотреблений для предметной области:

*– law\_by.stat* – для белорусского языка;

*– law\_ru.stat* – для русского языка.

а также текстовые файлы:

1)*TagList* – лексико-грамматический классификатор для белорусского и русского языка со статистикой встречаемости кодов.

2)*TagPriority* – вспомогательный классификатор с кодами парадигм в порядке, который соответствует расположению кодов в парадигме исходных словарей в текстовом формате.

3)Файлы с правилами распознавания границ слов:

*– BelWBD.txt* – для белорусского языка;

*– RuslWBD.txt* – для русского языка.

4)Файлы с правилами распознавания границ предложений:

*– BelSBD.txt* – для белорусского языка;

*– RusSBD.txt* – для русского языка.

5)Файл *TransDependTreeBuilder.lua*, содержащий текст скриптов технологии LUA, реализующих многопоточность при работе машинного перевода.

# 5 Используемое системное программное обеспечение и средства разработки

Все модули системы представляют собой программы, разработанные на платформе .NET на языке C#, данные хранятся в базе данных, функционирующей под управлением СУБД Microsoft SQL Server 2008 и выше. В программах применяются следующие технологии:

Windows Communication Foundation (WCF) – технология компании Microsoft, используемая для построения распределенных приложений. В данном случае WCF используется для осуществления обмена данными между подмодулями модуля построения ПОД.

Windows Workflow Foundation (WWF) – технология компании Microsoft для управления рабочими процессами. В данном случае используется для задания логической последовательности выполнения операций в модуле построения ПОД.

Windows-сервисы – приложения, автоматически запускаемые системой при запуске Windows и выполняющиеся вне зависимости от статуса пользователя.

ASP.NET – технология Microsoft для создания web-приложений и web-сервисов. Данная технология применяется для разработки web-сайта ИПС ”ЭТАЛОН-ONLINE“, а также для создания ashx-страниц (handlers), позволяющих избежать использования полнофункциональных web-страниц.

Программирование клиентских скриптов с помощью javascript с использование библиотеки JQuery.

AJAX – технология построения интерактивных пользовательских интерфейсов, предоставляющая возможность выполнять обмен данными между web-страницей, загруженной в браузер пользователя, и web-сервером без перезагрузки страницы.

# 6 Описание комплекта поставки

Описание комплекта поставки модуля построения ПОД:

Папка **ControlAgent**, содержащая файлы:

ControlAgentSetup.msi

setup.exe

Папка **ScheduleService**, содержащая файлы:

ScheduleServiceSetup.msi

setup.exe

Папка **Worker**, содержащая файлы:

ComputePositionsLib.dll

ComputePositionsLib.pdb

ControlAgent.dll

ControlAgent.pdb

CreationSDI.dll

CreationSDI.pdb

RBTagger.dll

RQTrans.dll

Utils.dll

Utils.pdb

Worker.exe

Worker.exe.config

Worker.pdb

Worker.vshost.exe

Worker.vshost.exe.config

Worker.vshost.exe.manifest

В паке **Worker** также содержится подпапка **ldb** со следующими файлами:

BelDictionary.dat

BelSBD.txt

BelWBD.txt

LangWeight.dat

law\_by.stat

law\_ru.stat

Program.cs

RusDictionary.dat

RusSBD.txt

RusWBD.txt

TagList

TagList.data

TagPriority

TransDependTreeBuilder.lua

TranslationBR.tbl

TranslationRB.tbl

VerbRulingBel.bin

VerbRulingBelPrep.bin

VerbRulingRus.bin

VerbRulingRusPrep.bin

Папка **WorkersAgent**, содержащая файлы:

setup.exe

WorkersAgentSetup.msi

# 7 Описание комплекта исходных модулей

Описание комплекта исходных файлов модуля построения ПОД:

Проект **ComputePositionsLib**

|  |  |
| --- | --- |
| **Имя файла** | **Описание** |
| ComputePositions.cs | Модуль для пересчета позиций слов в тексте. |

Проект **ControlAgent**

|  |  |
| --- | --- |
| **Имя файла** | **Описание** |
| ControlAgent.cs | Реализация интерфейса службы-балансировщика. |
| IControlAgent.cs | Интерфейс службы-балансировщика. |

Проект **ControlAgentService**

|  |  |
| --- | --- |
| **Имя файла** | **Описание** |
| ControlAgentService.cs | Реализация службы-балансировщика в виде windows-службы. |

**ControlAgentSetup** представляет собой проект инсталляции службы-балансировщика.

**CreationSDI**

|  |  |
| --- | --- |
| **Имя файла** | **Описание** |
| CreationSDI.cs | .NET оболочка библиотеки RBTagger.dll. |

**ScheduleLib**

|  |  |
| --- | --- |
| **Имя файла** | **Описание** |
| ScheduleWorkflow.cs | Реализация службы-менеджера. |

**ScheduleService**

|  |  |
| --- | --- |
| **Имя файла** | **Описание** |
| ScheduleService.cs | Реализация службы-менеджера в виде windows-службы. |

**ScheduleServiceSetup** представляет собой проект инсталляции службы-менеджера.

**UtilsLibrary**

|  |  |
| --- | --- |
| **Имя файла** | **Описание** |
| Utils.cs | Модуль, содержащий вспомогательные функции. |

**Worker**

|  |  |
| --- | --- |
| **Имя файла** | **Описание** |
| Document.cs | Реализация процесса-обработчика. |
| SQLProcessor.cs | Модуль для обращения к базе данных SQL Server. |

**WorkersAgent**

|  |  |
| --- | --- |
| **Имя файла** | **Описание** |
| IWorkersAgent.cs | Интерфейс службы-инициализатора. |
| WorkersAgent.cs | Реализация интерфейса службы-инициализатора. |

**WorkersAgentService**

|  |  |
| --- | --- |
| **Имя файла** | **Описание** |
| WorkersAgentService.cs | Реализация службы-инициализатора в виде windows-службы. |

**WorkersAgentSetup** представляет собой проект инсталляции службы-инициализатора.

Описание комплекта исходных файлов модуля ”Поиск“:

|  |  |
| --- | --- |
| **Имя файла** | **Описание** |
| Card.aspx.cs | Карточка документа (страница добавлена для обратной совместимости с предыдущей версией). |
| Default.aspx.cs | Главная страница сайта. |
| Print.aspx.cs | Версия для печати результатов поиска. |
| PrintText.aspx.cs | Версия для печати текста документа. |
| Text.aspx.cs | Текст документа (страница добавлена для обратной совместимости с предыдущей версией) |
| Chambers.ashx | Сервис для реализации функций работы с пользователями. |
| Control.ashx | Сервис для реализации функции ”Документы на контроле“. |
| Document.ashx | Сервис для отображения текстов и карточек документов. |
| Login.ashx | Сервис для реализации функции входа в систему. |
| Registration.ashx | Сервис для реализации функции регистрации пользователя в системе. |
| XMLProcessor.ashx | Сервис для реализации функций интеллектуального поиска по текстам документов. |

# 8 Установка, вызов и загрузка

Для установки модуля построения ПОД необходимо выполнить установку на серверах НЦПИ трех windows-служб и скопировать в папку на сервере несколько копий процессов обработчиков вместе с отдельными копиями бинарных словарей. Для каждой windows-службы разработан инсталлятор. Процесс инсталляции во всех случаях выглядит аналогично. Рассмотрим его на примере установки службы-инициализатора:

1. Запускаем setup.exe. После этого должно появиться следующее окно:

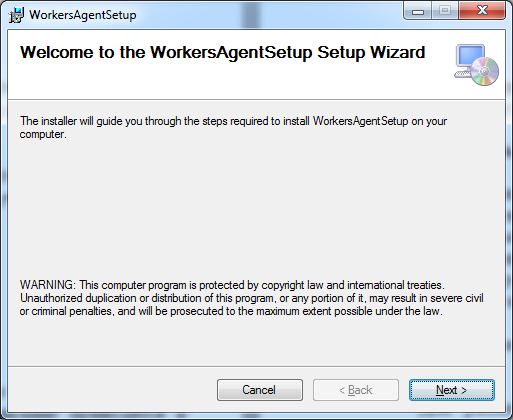


Рисунок 8.1 – Начало установки службы-инициализатора

1. Нажимаем ”Next“. В появившемся диалоговом окне выбираем папку, куда будет выполнена установка программы.

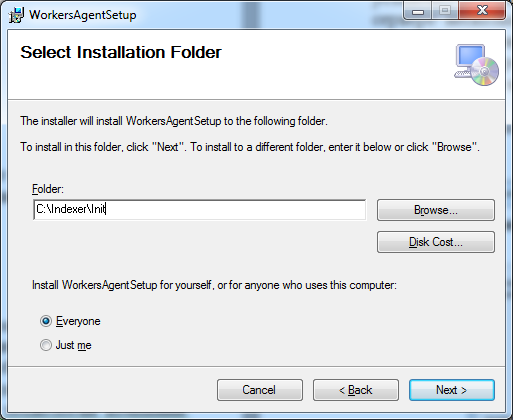


Рисунок 8.2 – Выбор папки на диске, в которую будет выполнена установка программы

1. Нажимаем ”Next“. В результате должно появиться следующее диалоговое окно:

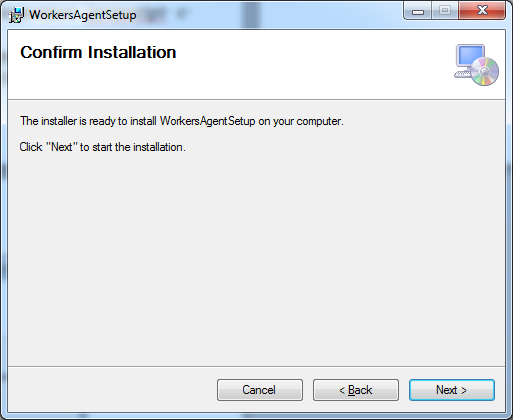


Рисунок 8.3 – Переход к непосредственной установке службы-инициализатора

1. Нажимаем ” Next“. В результате будет выполнена установка.

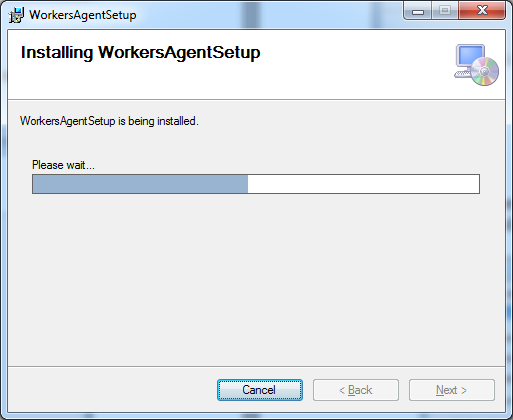


Рисунок 8.4 – Непосредственная установка службы-инициализатора

1. После окончания процесса установки должно появиться следующее окно:

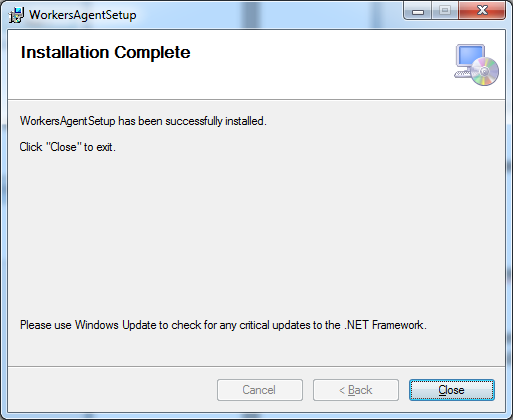


Рисунок 8.5 – Окончание установки службы-инициализатора

1. Нажимаем кнопку ”Close“.

Установленную службу можно увидеть с помощью утилиты ” Панель управления“ → ”Администрирование“ → ”Службы“:

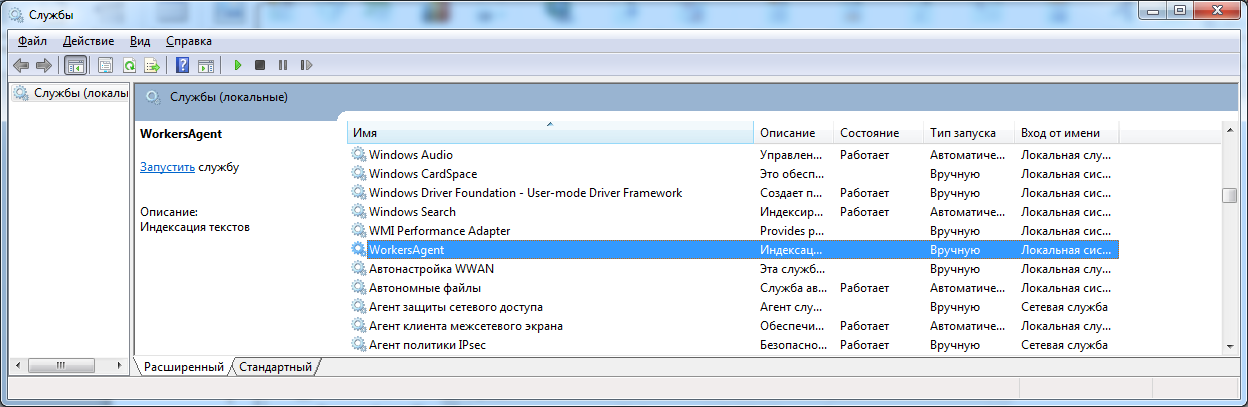


Рисунок 8.6 – Окно ”Службы“

Здесь же можно настроить тип запуска службы и действия, которые необходимо выполнять при сбое.

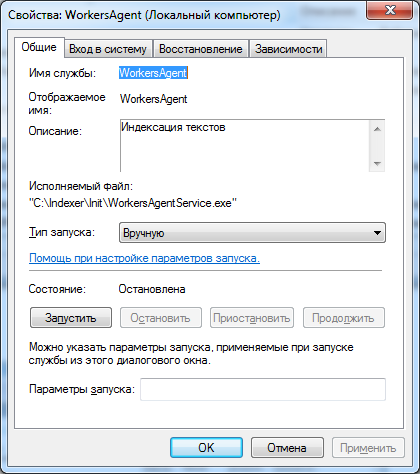


Рисунок 8.7 – Вкладка ”Общие“ окна свойств установленной службы

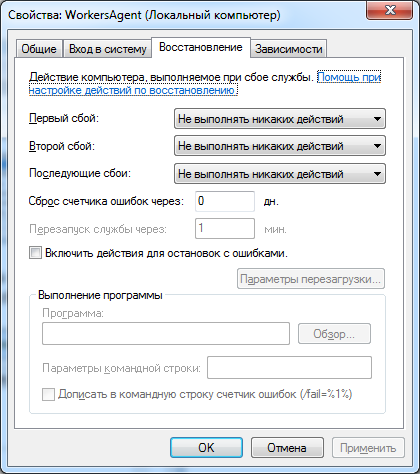


Рисунок 8.8. – Вкладка ”Восстановление“ окна свойств установленной службы

Удалить сервис можно в ”Панель управления“ → ”Программы и компоненты“.

Для размещения поискового сервиса и сайта <http://etalonline.by> необходимо выполнить операцию Publish Web Site.

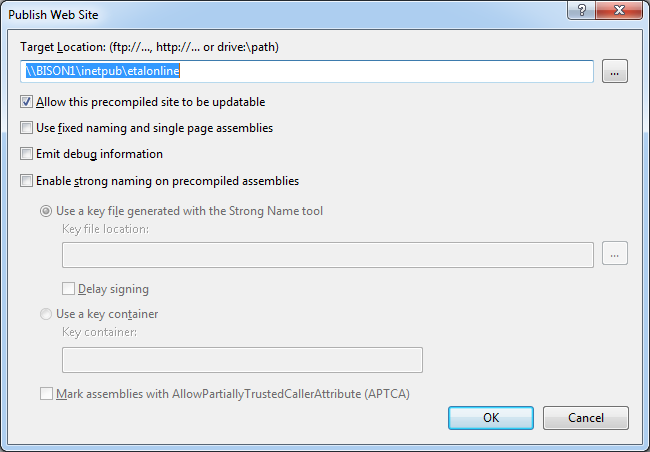


Рисунок 8.9 – Публикация сайта