Система автоматического обновления баз данных - KodUp v.1.0.

Техническая документация.

|  |  |  |
| --- | --- | --- |
| Дата | Описание | Автор |
| 08/29/2016 | Первая версия документации. | Провалов Л.В. |
| 09/09/2016 | Обновление | Провалов Л.В. |
| 10/19/2016 | Обновление | Провалов Л.В. |

# Общие характеристики

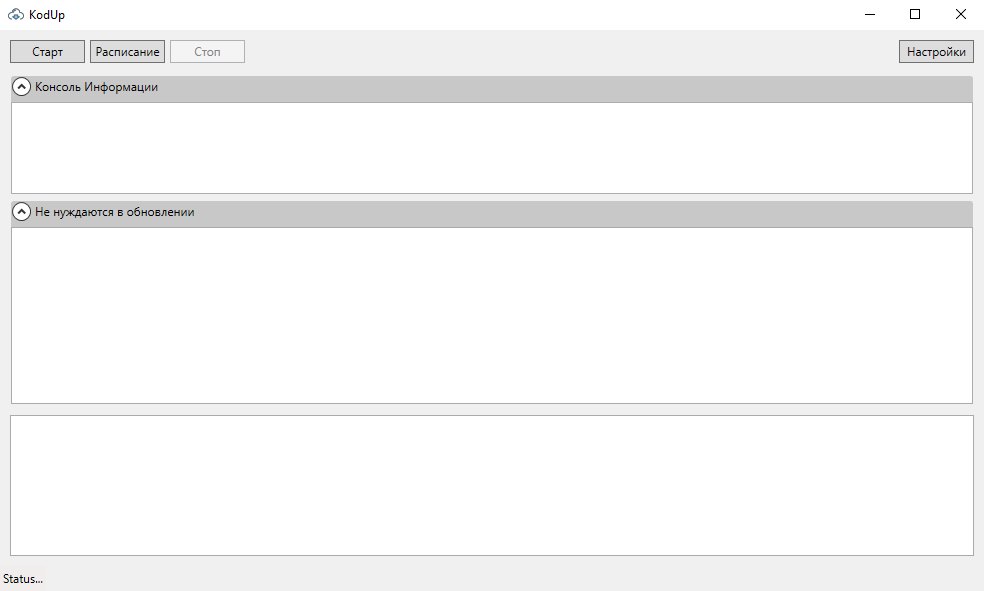
|  |  |
| --- | --- |
| Среда разработки и тестирования | Microsoft Visual Studio Professional 2015 |
| Версия .Net framework | .net framework 4.0 |
| Библиотеки | Crypt.dll |
| Сторонние библиотеки | [NLog.dll](#_Log) |

# Архитектура

# Графический интерфейс пользователя.

Программа состоит из 2х основныз окон. Main Window и Settings Window.

## Main Window



Программа стартует с вызова конструктора главного окна. В нём происходит инициализация компонентов окна, создание процесса обновления интерфейса (FetchDownloaderManager), а так же запуск инициализации настроек (FirstStartInitialization).

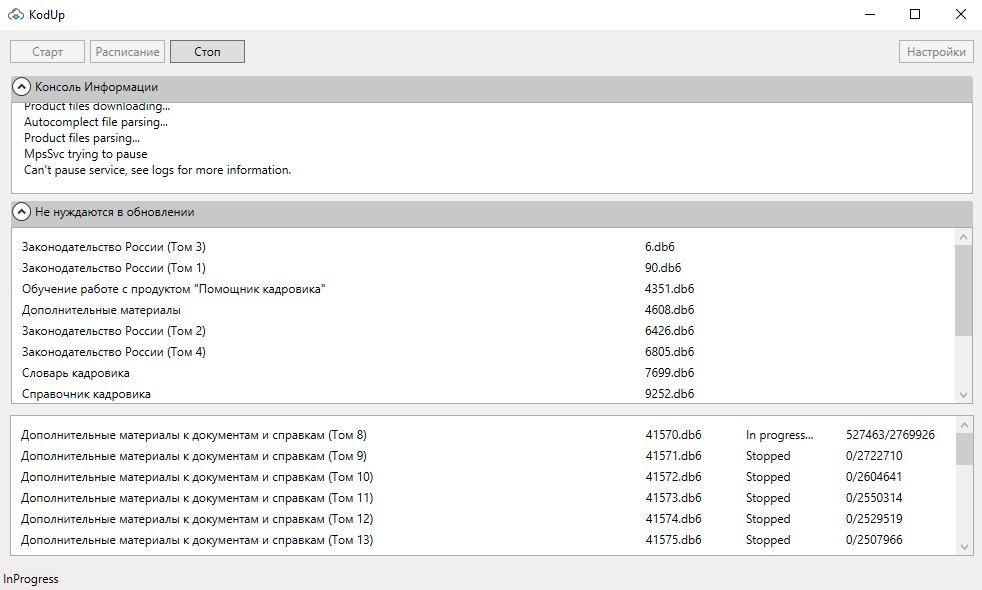
При старте программы, в зависимости от валидности настроек, принимается решение об открытии окна настроек.

Основные методы данного окна работают с объектом (DownloaderManager) :

DownloaderManager.StartAsync();

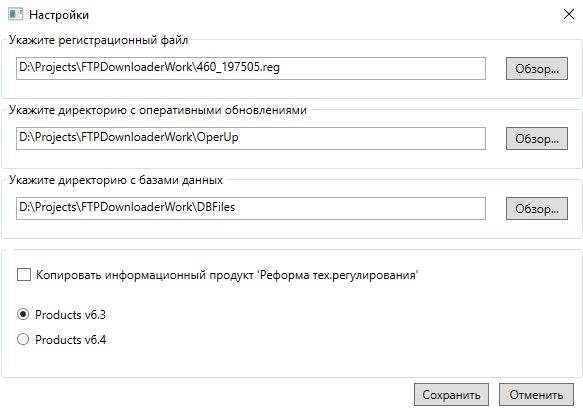
DownloaderManager.Stop();

При старте обновления, происходит заполнение рабочих полей. В консоль информации выводиться информация о ходе работы программы обновления. Так же выводиться информация о файлах не нуждающихся в обновлении и формируется список обновляемых файлов, который обновляется в реальном времени каждую секунду.



## Setting Window

Окно реализует графическое представление модуля конфигурации. Позволяет задать пути к директориям и файлам используемым в программе. При нажатии кнопки “Сохранить” происходит проверка валидности настроек и сохранение выбранных настроек в модуле конфигурации и закрытие окна настроек.



При не валидных настройках запуск программы обновления будет недоступен.

# Модули программы

Программа KodUp состоит из набора модулей реализованных в виде отдельных классов. Основная структура состоит из следующих модулей:

* **DownloaderManager**  
  Отвечает за чтение настроек, загрузку и разбор фалов autocomplect и products, построение очереди загрузки и списка файлов не требующих обновления. Описывает общий алгоритм построения очереди загрузки и реализует его асинхронное выполнение. Так же отвечает за очистку директории OperUp и обновление данных в ней, остановку и запуск windows сервисов.
* **LoadingManager**  
  Реализует очередь загрузки с асинхронным методом её выполнения.
* **DownloadFile**  
  Класс являющийся абстракцией для сущности обновляемого файла. Реализует методы асинхронной загрузки нового файла во временную копию (имя-файла\_copy), методы замены старой версии. Инкапсулирует непосредственный модуль загрузки файла (FTPDownloader).
* **FTPDownloader**  
  Класс реализующий асинхронную загрузку файлов, поддерживающую возобновление докачки.
* **FTPWorker, FTPClient**  
  Вспомогательные модули реализующие методы отправки отчетов на FTP сервер. И другие методы для получения данных с FTP (Получение списков фалов и т.п.)
* **ReportWriter**  
  Модуль для заполнения отчета о ходе выполнения обновления. Формирует файл отчета report.txt с дальнейшей отправкой на FTP сервер.
* **Log**  
  Модуль логирования процесса работы. Использует библиотеку NLog.
* **XML**  
  Набор классов описывающих структуру Autocomplect и Products файлов. Позволяют загружать в память данные из xml и осуществлять поиски необходимых данных.
* **Configuration, FtpConfiguration**Классы реализующие работу с конфигурационными файлами системы.

## Downloader Manager

Содержит списки файлов ожидающих обновление и не нуждающихся в обновлении. (DownloadingItems и NNTD\_DownloadingItems соответственно).

Основной метод асинхронная загрузка обновления (StartAsync) создаёт поток выполнения загрузки (Processing).

Метод Processing реализует основной алгоритм обновления:

* DelayedStartFunc  
  Ожидание начала загрузки, с возможностью отмены старта.
* Initialize  
  Инициализация загрузки, построение очереди.
  + Чтение не зашифрованного файла конфигурации (AppConfig.xml)
  + Чтение регистрационного \*.reg файла, получение данных о клиенте.
  + Создание экземпляров модулей, FtpWorker, LoadingManager
  + GetDBListWithSize  
    Получение данных о файлах БД на FTP сервере.
  + GetListOfToms  
    Загрузка файлов автокомплектов и продуктов с FTP, составление списка томов для загрузки.
  + Определение необходимости загрузки тома и формирование очереди загрузки в модуле LoadingManager
* StopKTServices  
  Попытка остановки windows сервисов.
* Ожидание загрузки файлов модулем LoadingModel.
* При условии загрузки всех обновлений, очистка директории OperUp и загрузка обновлений.
* Попытка возобновления работы windows сервисов.

## Loading Manager

Модуль реализующий очередь загрузки файлов, позволяющий добавлять файлы для загрузки формируя очередь загрузки. Реализующий асинхронную загрузку данных(BeginAsync). А так же метод вызывающие завершение текущего потока загрузки(Cancel), метод получения статуса обновляемых данных (GetStatuses).

Основной метод (BeginAsync) создает поток выполняющий последовательный перебор загружаемых фалов и их последовательную загрузку.

## DownloadFile

Класс инкапсулирующий объект класса FileDownloader и абстрагирующий сущность обновляемого файла. Основные методы данного класса DestinationFileDownloadedLength – получение размера загружаемого файла, сначала копии, а затем и обновленного. OverwriteDestinationFile – замена обновляемого файла на загруженный, установка дат создания и обновления файла.

## FTPDownloader

Реализация загрузки файла с FTP сервера с поддержкой дозагрузки. BeginAsync – метод запускающий асинхронный процесс скачивания данных с FTP сервера. При возникновении web ошибок при загрузке, попытки загрузки будут производиться с периодом CYCLE\_SLEEPING\_TIME равным 10 секундам.

ResumeFtpFileDownload – основной метод загрузки файла.

Создаётся запрос на FTP сервер к необходимому для скачивания файлу. На локальном диске проверяется наличае недокачанного файла (с суффиксом \_copy), если он существует, то определяется его размер и с FTP запрашивается файл с отступом по количеству байт равному размеру недокачанного файла на локальном клиенте. Происходит потоковое копирование файла. По окончанию загрузки потоки закрываются, генерируется событие сигнализирующее об окончании загрузки. В результате которого вызывается метод OverwriteDestinationFile класса DownloadFile.

При возникновении исключительных ситуаций при загрузке, генерируются соответствующие события вызывающие пропуск загрузки и вывод сообщений в логи и в UI приложения. При возникновении web ошибок связанных с сетью предпринимается попытка повторной загрузки файла, количество попыток и временная задержка между ними настраивается в конфигурации системы. При невозможности загрузить файл (непример нет доступа к файлу) попытки загрузки прекращаются до следующего запуска приложения.

## ReportWriter

Модуль осуществляет запись данных в текстовый файл report.txt тем самым формируя отчет о процессе загрузки. По окончании загрузки или при её остановке, файл report.txt отправляется на FTP сервер в папку с отчетами, при этом имя файла устанавливается в соответствии с форматом: ИД Пользователя\_dГодМесяцДеньtЧасыМинутыСекунды, дата соответствует времени формирования отчёта.

## Log

Класс реализующий логирование процесса выполнения программы. При реализации использовалась библиотека NLog. (<https://github.com/NLog/NLog/wiki/Tutorial#configuration>).

Процесс логирования предоставляет возможность настройки уровня (детализации) логирования. Для этого надо изменить конфигурационный файл NLog.config

В теге

<rules>

<logger name="\*" minlevel="Trace" writeTo="logfile" />

</rules>

Необходимо поменять атрибут minlevel на одно из следующих значений:

Trace – выводить самый подробный лог из возможных.

Info  – выводить только основные логи.

Error – выводить только сообщения об ошибках.

## Модуль конфигурации (Configuration)

Представляет собой модель конфигурации (ConfigurationModel) и реализует чтение и запись файла конфигурации (AppConfig.xml). Файл конфигурации представляет собой файл формата \*.xml хранящий различные настройки программы.

<RegFile></RegFile> - путь до регистрационного файла \*.reg (Настраивается через UI).

<OperationalUpdateDirectory></OperationalUpdateDirectory> - директория оперативных обновлений. (Настраивается через UI).

<DBDirectory></DBDirectory> - директория с базами данных требующих обновления. (Настраивается через UI).

<IsTechnicalRegulationReform></IsTechnicalRegulationReform> - Следует ли загружать реформу технического регулирования. (Настраивается через UI).

<ConnectionInitFile> </ConnectionInitFile> - Указывает путь к шифрованному файлу конфигурации с настройками FTP сервера.

<UseProxy></UseProxy> - Флаг использования прокси сервера.

<UsePassiveFTP></UsePassiveFTP> - Использовать пассивный FTP.

<ProxyAddress /> - Адрес прокси сервера.

<DelayedStart></DelayedStart> - Время отложенного старта. (Настраивается через UI).

<KTServices></KTServices> - Список windows сервисов, которые следует остановить перед началом обновления. Указываются через запятую (ИмяСервера1,ИмяСервера2,ИмяСервера3) или остаётся пустым.

<ProductVersion></ProductVersion> - Версия файла продуктов. (Настраивается через UI).

<AutoStart></AutoStart> - Автоматический запуск обновления при старте (true/false).

<CountOfRepeat>5</CountOfRepeat> - Число повторений при проблемах с соединением.

<RepeatDalay>10000</RepeatDalay> - Задержка в мс между попытками загрузить файл.

## Модуль конфигурации FTP сервера (FtpConfiguration)

Представляет собой модель конфигурации FTP сервера (), модуль шифрования данных и реализует чтение, расшифровку и предоставление данных конфигурации FTP сервера.

Для шифрофки/дешифровки используется модуль CryptRC4 реализующий алгоритм потокового шифрования RC4.

При загрузке модуля происходит чтение зашифрованного файла FTP конфигурации указанного в модуле конфигурации (ConnectionInitFile, по умолчанию “FtpConfiguration.dat”) и его расшифровка. Расшифрованный файл представляет собой файл формата \*.xml и имеет следующую структуру:

<FtpSourcePath></FtpSourcePath> - указывается ip адрес FTP сервера.

<DBPath></DBPath> - директория на FTP сервере содержащая базы данных для обновления.

<AutocomplectsPath></AutocomplectsPath> - директория на FTP с файлами автокомплектов.

<ProductsPath></ProductsPath> - директория на FTP с файлами продуктов.

<ProductFileNames>Products\_6.3\_1627.xml</ProductFileNames> - Список файлов продуктов на сервере.

<ProductFileNames>Products\_6.4\_1627.xml</ProductFileNames>

<ReportsPath>update\_logs</ReportsPath> - директория на FTP сервере в которую записываются отчеты о запусках обновлений.

<User>signal</User> - пользователь.

<Password>fvu90m</Password> - пароль.

<ClearFolder>operup/new</ClearFolder> - директория на FTP из которой берутся файлы при обновлении OperUP.

Файл конфигурации разбирается и происходит заполнение модели (FtpConfigurationModel) в памяти.

Файл FTP конфигурации подготавливается отдельно и шифруется с помощью приложения **CryptEncoder.exe**. Зашифрованный файл имеет расширение \*.dat

## XML

Набор классов реализующих скачивание и разбор файлов autocomplects и products. Основные модули ProductsParser и AutoComplectsParser. Работают по принципу десериализации xml файлов. В папке models находятся классы описывающие структуру объектов десериализации.