**КОМИТЕТ ПО ОБРАЗОВАНИЮ ПРАВИТЕЛЬСТВА САНКТ-ПЕТЕРБУРГА**

**САНКТ- ПЕТЕРБУРГСКОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ   
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ   
  
«КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»**

Специальность 09.02.03 «Программирование в компьютерных системах»

**УТВЕРЖДАЮ**

**Зам. директора по УМР**

**\_\_\_\_\_\_\_\_\_\_\_\_ Венедиктова О.Н.**

**«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.**

**ДИПЛОМНЫЙ ПРОЕКТ**

Пояснительная записка

**Тема: Создание информационной системы**

Группа 463

Рецензент

Нормоконтроль Согласовано О.Н. Венедиктова

Консультант по экономической части Согласовано Т.В. Львова

Руководитель дипломного проекта­­­­\_\_\_\_\_\_СОГЛАСОВАНО И.П. Смирнова

Выполнил А.В. Дорогобед

САНКТ-ПЕТЕРБУРГ, 2020

ОГЛАВЛЕНИЕ

[1. ОБЩАЯ ЧАСТЬ 4](#_Toc42286873)

[1.1. Цель разработки 4](#_Toc42286874)

[1.2. Анализ рынка подобных программных решений 4](#_Toc42286875)

[1.3. Анализ методов разработки. 8](#_Toc42286879)

[1.4. Анализ среды разработки 12](#_Toc42286882)

[2. СПЕЦИАЛЬНАЯ ЧАСТЬ 29](#_Toc42286894)

[2.1. Техническое задание 29](#_Toc42286895)

[2.2. Описание функционально-логической структуры данных 31](#_Toc42286900)

[2.3. Описание алгоритма 34](#_Toc42286905)

[2.4. Программная реализация 35](#_Toc42286907)

[2.5. Инструкция пользователя 38](#_Toc42286910)

[2.6. Анализ результатов работы 39](#_Toc42286911)

[3. ЭКОНОМИЧЕСКАЯ ЧАСТЬ. 40](#_Toc42286912)

[3.1. Расчёт затрат на выполнение проекта 40](#_Toc42286913)

[3.2. Расчет предполагаемой прибыли 44](#_Toc42286920)

[ЗАКЛЮЧЕНИЕ 45](#_Toc42286921)

[ЛИТЕРАТУРА 46](#_Toc42286922)

[ПРИЛОЖЕНИЕ А – Исходный текст программы 48](#_Toc42286923)

**ВВЕДЕНИЕ**

Тема дипломного проекта: Разработка информационной системы. Данный диплом подтверждает освоение следующих компетенции профессиональных модулей:

ПМ.01. Разработка программных модулей программного обеспечения для компьютерных систем.

ПК 1.1. Выполнять разработку спецификаций отдельных компонент.

ПК 1.2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК 1.4. Выполнять тестирование программных модулей.

ПК 1.5. Осуществлять оптимизацию программного кода модуля.

ПК 1.6. Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций.

ПК 3.2. Выполнять интеграцию модулей в программную систему.

ПК 3.3. Выполнять отладку программного продукта с использованием специализированных программных средств.

ПК 3.5. Производить инспектирование компонент программного продукта на предмет соответствия стандартам кодирования.

ПК 3.6. Разрабатывать технологическую документацию.

Мозг человека легче всего воспринимает информацию представленную графически. И из-за этого, большой популярностью пользуются различные сервисы по просмотру видео. Безусловным лидером среди них является YouTube.

Кроме тех людей, которые интересуются только тем, чтобы найти то видео, которое им нужно, немалое количество людей желают создавать собственные видеоматериалы, выкладывать их на YouTube и получать за это деньги. Данный программный продукт предназначен для того, чтобы помочь людям, желающим создавать видеоконтент и продвигать его на YouTube.

Для того, чтобы объяснить, чем данная программа может помочь, сначала нужно объяснить, как работает оценивание качества видео в YouTube. YouTube оценивает качество видео, отталкиваясь от очень многих параметров. Большинство этих параметров строятся из реакции пользователей на видео и на них может повлиять только то, насколько качественно сделано само видео. Но, существует у каждого видео также такой параметр, как ключевые слова. Этот параметр представляет собой некоторое количество слов и/или фраз, по которым YouTube определяет, должно это видео выводиться в поисковом запросе или нет. Определяет он это простым сравнением поискового запроса и всех ключевых слов по очереди. Но что делать, если тот, кто выкладывает видео, не знает, какие слова использовать в качестве ключевых? Тут может помочь данный программный продукт, который способен подсказать лучшие ключевые слова для продвижения своего видео. Безусловно, существует множество программ, которые предлагают такой же или даже больший функционал, но они, как правило, ориентированы на тех, кто занимается созданием видеоконтента профессионально, и вследствие этого, имеют стоимость, как правило, больше тысячи рублей, и обычно распространяются по модели платной подписки. Для любительского создания видео для YouTube подобные траты не оправданы. Данный программный продукт не имеет такого богатого функционала, как другие программы подобной направленности, но зато и стоит он очень дёшево.

# ОБЩАЯ ЧАСТЬ

В этом разделе описаны средства разработки, использованные при создании продукта, указаны цели разработки дипломного проекта, проанализированы различные средства разработки.

## Цель разработки

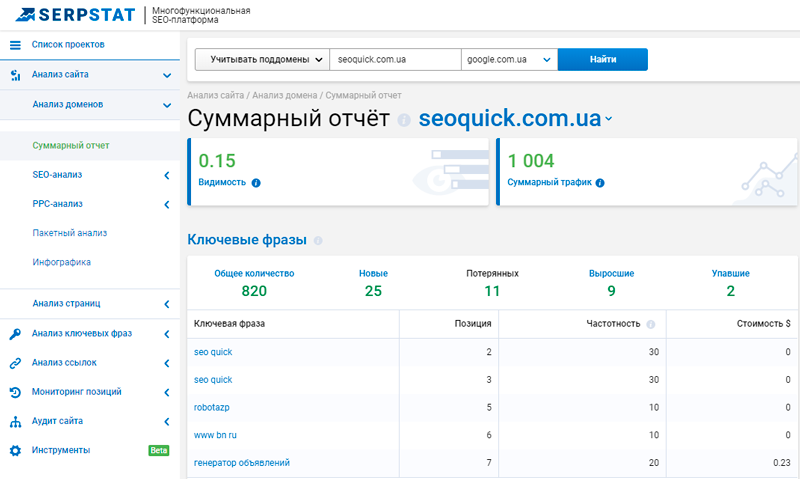
Целью разработки данного дипломного проекта является создание программного продукта (информационная система), способного выполнять следующие задачи:

* подбор ключевых слов на основе названия будущего видео;
* сохранение результатов подбора ключевых слов.

## Анализ рынка подобных программных решений

### Облачный сервис SERPSTAT

Облачный сервис SERPSTAT является сервисом с очень богатым функционалом, предлагающим своим пользователям не только помощь в продвижении на YouTube, но и на множестве других платформ таких как Google AdWords и Яндекс Директ. И более того, данный сервис предлагает анализировать посещение сайтов-блогов, домены, семантику сайтов. Но из-за настолько широкого функционала, и цена SERPSTAT очень высока: 19$ (1360,92 рублей по состоянию на 2020.05.24) за один месяц подписки. Интерфейс сервиса представлен на рисунке 1.

Рисунок 1 – Интерфейс SERPSTAT

Плюс:

* очень широкие возможности, подходящие, пожалуй, любому контентмейкеру.

Минус:

* какие возможности такая и цена. 19$ в месяц за эту программу согласится отдавать только тот, кто профессионально занимается контентмейкингом.

### Сайт Keyword Tool

Сайт keywordtool.io по сути отбирает ключевые слова, схожие с тем, которое вы ввели в строку поиска. Так же имеется платная подписка, при которой для найденных ключевых слов, отображаются различные метрики, например, количество поисковых запросов, в которых выводится данный тег. Но ни одна из метрик не доступна в бесплатном плане, а самый дешёвый платный план стоит 69$ (4942,29 рублей по состоянию на 2020.05.24) за один месяц подписки. И к тому же, данный сайт не выводит ключевые слова, которые схожие по лексическому значению или те, которые используются в видео совместно с ключевым словом, введённым в поисковую строку. Данный сайт выводит только те ключевые слова, которые содержат в себе введённое в поисковую строку ключевое слово (рисунок 2).

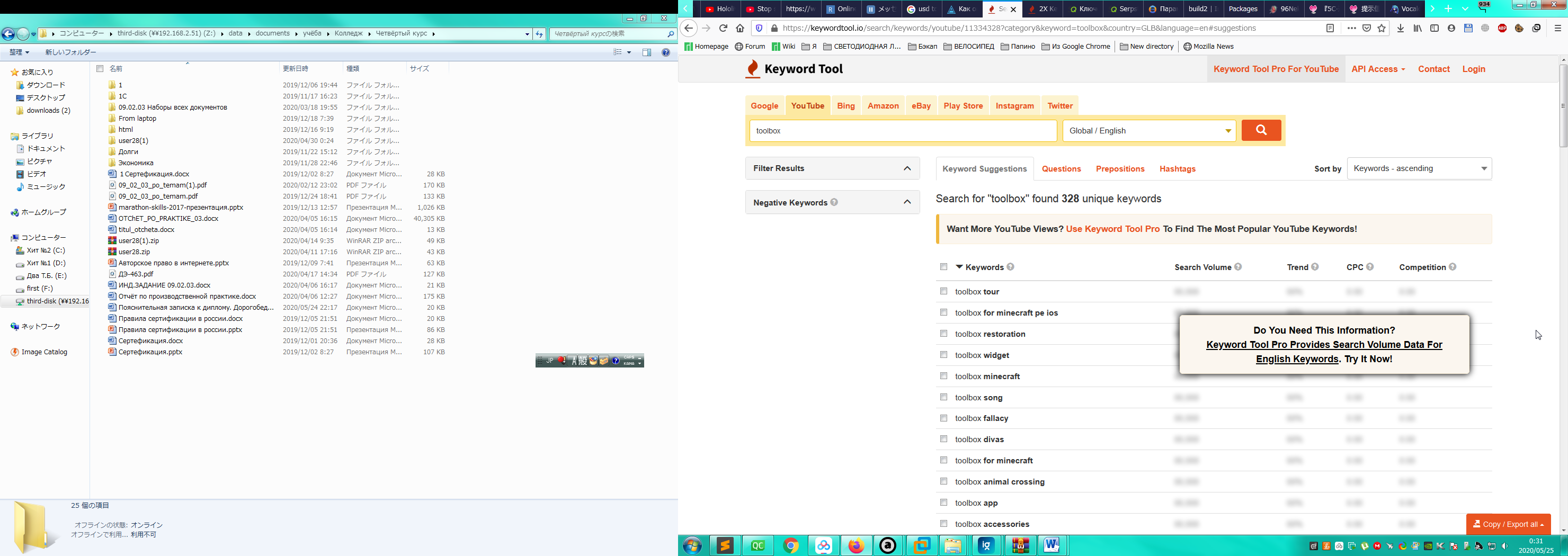


Рисунок 2 – Интерфейс Keyword Tool

Плюс:

* позволяет отбирать ключевые слова в зависимости от выбранного языка.

Минусы:

* подбирает слова только по похожести, в написании игнорируя похожесть смысла;
* платная подписка стоит очень дорого (минимум 69$);
* отношение цены и функционала намного хуже, чем у SERPSTAT.

### Сервис Ahrefs

Данный сервис предоставляет меньшее количество функций, чем SERPSTAT, но зато предоставляются более полно другие, тоже очень важные функции, такие как невероятно большой набор актуальных данных, круглосуточную техническую поддержку, обучающие материалы, приватное сообщество пользователей данного сервиса в Facebook. Но за столь глубокую проработку приложения придётся много платить: бесплатного плана у приложения нет, а самый дешёвый платный план стоит 99$ (7091,11 рублей по состоянию на 2020.05.25) в месяц. Так же данный сервис рекомендуют эксперты в сфере SEO и маркетинга и данным сервисом пользуются известные компании, такие как Adobe, Netflix, Uber. Интерфейс сервиса представлен на рисунке 3.

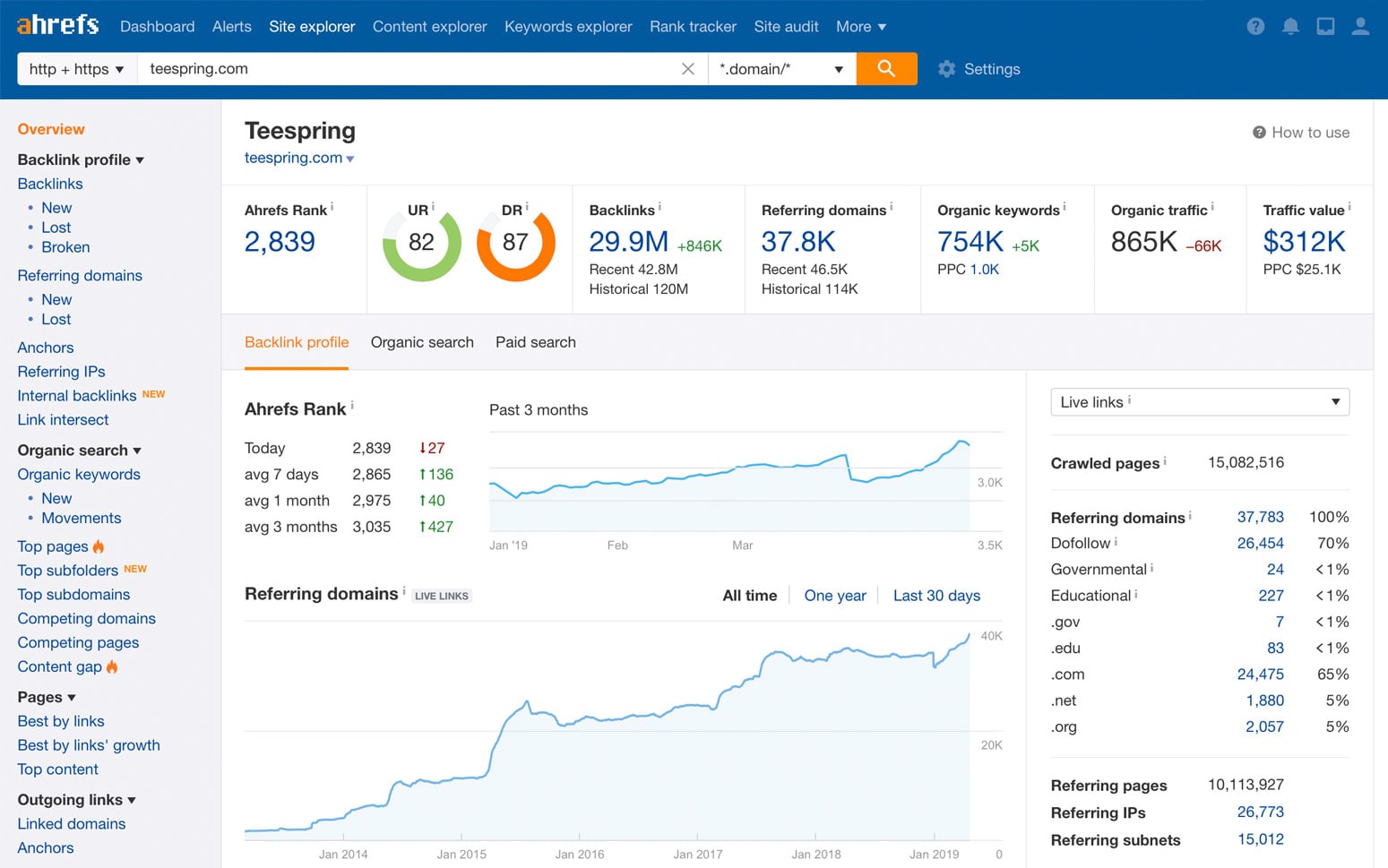


Рисунок 3 – Интерфейс Ahrefs

Плюсы:

* предоставляется техническая поддержка;
* невероятно широкий набор данных;
* приватное сообщество пользователей данного сервиса в Facebook;
* рекомендации экспертов в сфере SEO и маркетинга.

Минусы:

* не такой широкий функционал как у SERPSTAT;
* самая дешёвая цена за подписку 99$.

Преимущества разрабатываемого приложения заключаются в простоте использования и низкой стоимости. Данный программный продукт не имеет функционала, нацеленного на профессионалов, из-за чего он прост в освоении и использовании и имеет низкую стоимость.

## Анализ методов разработки

### Обзор методов решения

Во время разработки рассматривалась возможность создания приложения для платформ:

* ОС Windows;
* ОС на основе Linux;
* ОС MacOs.

OC Windows – данная платформа содержит наибольшее число потенциальных пользователей.

OC MacOS – данная платформа также содержит немалое количество потенциальных пользователей.

ОС на основе Linux –на данной платформе также могут быть потенциальные пользователи и, обычно, если каком либо языке программирования можно создавать приложения под MacOS, то в нём можно создавать приложения и под Linux.

### Выбор языка разработки

В данном разделе пояснительной записки к дипломному проекту рассматриваются языки программирования, подходящие для реализации данного дипломного проекта.

#### Язык программирования Java

Java — строго типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, для которой существует реализация виртуальной Java-машины. Дата официального выпуска — 23 мая 1995 года.

Плюсы:

* высокая скорость разработки;
* кроссплатформенность.

Минусы:

* низкая скорость работы программы;
* платное коммерческое использование;
* отсутствие нативного GUI.

#### Язык программирования C

С – язык программирования, разработанный в компании Bell Labs Деннисом Ритчи в начале семидесятых. Предшественником его стал язык B, который был разработан совместно с Кеном Томпсоном. Со временем, язык настолько усовершенствовался, что на нем решили переписать ядро операционной системы UNIX, что дало хороший старт для популяризации языка в системном программировании.

Плюсы:

* высокая скорость работы программы;
* низкое потребление памяти программой.

Минусы:

* низкая скорость разработки;
* сложнее большинства высокоуровневых языков программирования;
* отсутствие нативного GUI.

#### Язык программирования C++

C++ — компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.

Плюсы:

* поддержка нескольких парадигм программирования;
* высокая скорость работы программ;
* низкое потребление памяти программами.

Минусы:

* высокая сложность разработки;
* низкая скорость разработки;
* отсутствие нативного GUI.

#### Элементы выбранного языка

C++ относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C.

При объявлении переменной в C++ сначала указывается тип данных для этой переменной, а затем название.

Пример 1 – Объявление переменных:

* int i = 12;– объявление переменной a целого типа;
* double d = 12.3 – объявление переменной 64 разрядного типа с плавающей точкой;
* char c = 's' – объявление переменной символьного типа;
* std::string str = "word" – объявление переменной типа string;
* bool b = false – объявление логической переменной b.

Также можно использовать неявно типизированные переменные.

Пример 2 – Объявление неявно типизированных переменных:

* + auto intgr = 12 – объявление переменной a целого типа;
  + auto dbl = 12.3 – объявление переменной 64 разрядного типа с плавающей точкой;
  + auto str = "word" – объявление переменной типа string;
  + auto b = true – объявление логической переменной b.

## Анализ среды разработки

Для разработки информационной системы необходимо позаботиться о подходящей для этого среде разработки, удовлетворяющей всем потребностям.

### Среда разработки Eclipse

Eclipse (рисунок 4) является бесплатной программной платформой с открытым исходным кодом, контролируется организацией Eclipse Foundation. Она написана на языке программирования Java и основной целью её создания является повышение продуктивности процесса разработки программного обеспечения.

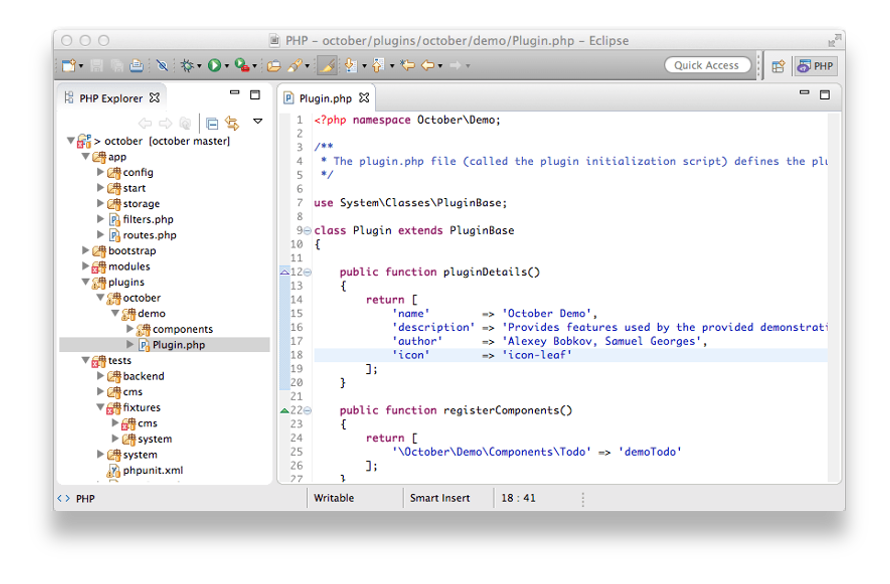


Рисунок 4 – Интерфейс Eclipse

Претендует на статус наиболее популярной Java IDE и является единственным конкурентом такой мощной платформы как NetBeans.

Но в отличие от NetBeans, который для создания элементов пользовательского интерфейса использует платформо-независимую библиотеку Swing, в Eclipse используется платформо-зависимая библиотека SWT - Standard Widget Toolkit.

IDE разработанные на базе платформы Eclipse применяются для создания программного обеспечения на различных языках программирования, так как Eclipse является платформой для разработки любых интегрированных сред программирования и расширений для себя же, по принципу "Плагины для Eclipse разрабатываются в самой Eclipse".

Интерфейс данной среды разработки представлен на рисунке 4.

Особенности Eclipse:

* кроссплатформенность - работает под операционными системами Windows, Linux, Solaris и Mac OS X;
* используя Eclipse можно программировать на множестве языков, таких как Java, C и C++, PHP, Perl, Python, Cobol и других;
* является фреймворком для разработки других инструментов и предлагает обширный набор API для создания модулей;
* используя подход RCP (Rich Client Platform) Eclipse является инструментом для создания практически любого клиентского программного обеспечения.

### Среда разработки Code::Blocks

Code::Blocks (рисунок 5) – это кроссплатформенная, бесплатная, Open Source среда разработки. В ней есть многие приятные вещи, которые современная среда разработки обязана уметь. Class browser, code completion, фолдинг (folding).

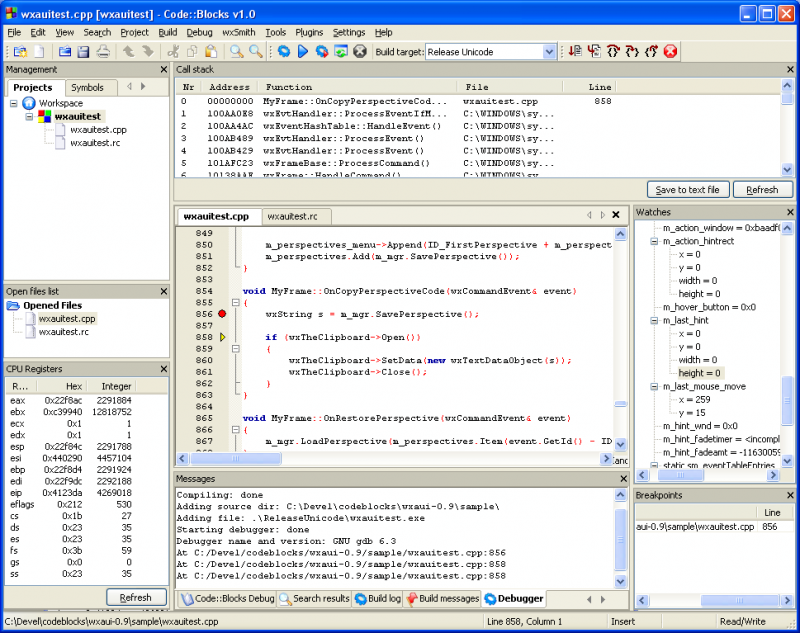


Рисунок 5 – Интерфейс Code::Blocks

Code::Blocks – это кроссплатформенная, бесплатная, Open Source среда разработки. В ней есть многие приятные вещи, которые современная среда разработки обязана уметь. Class browser, code completion, фолдинг (folding).

Есть импорт воркспейсов Microsoft Visual Studio и проектов Dev-C++. Насколько мне известно, популярная среда разработки Dev-C++ умеет работать только с GCC-компиляторами. Так же как и у Code::Blocks у них есть версия с MinGW GCC, есть версия вообще без компилятора. Но к Code::Blocks еще можно подключать другие (не обязательно GCC) компиляторы через приятный интерфейс.

В качестве отладчика по умолчанию CodeBlocks использует GDB — самый популярный отладчик для языка Си, который был разработан еще Ричардом Столменом.

Отладчик позволяет установить точки останова, и затем пошагово выполнять код с одновременным мониторингом переменных и массивов. Также можно отслеживать стеки вызываемых функций.

Количество возможностей и настроек среды CodeBlocks позволяют превратить эту среду в отличный инструмент программиста.

Интерфейс данной среды разработки представлен на рисунке 5.

Особенности Eclipse:

* бесплатность;
* поддержка большого количества компиляторов;
* наличие готовых шаблонов проектов;
* большое количество плагинов.

### Среда разработки GNAT Programming Studio

GNAT Programming Studio также известная как GPS это продвинутая среда разработки, очень сильно облегчающая жизнь разработчика программного обеспечения.

Интерфейс данной среды разработки представлен на рисунке 6.

Особенности GNAT Programming Studio:

* кроссплатформенная;
* присутствует браузер классов;
* бесплатная.

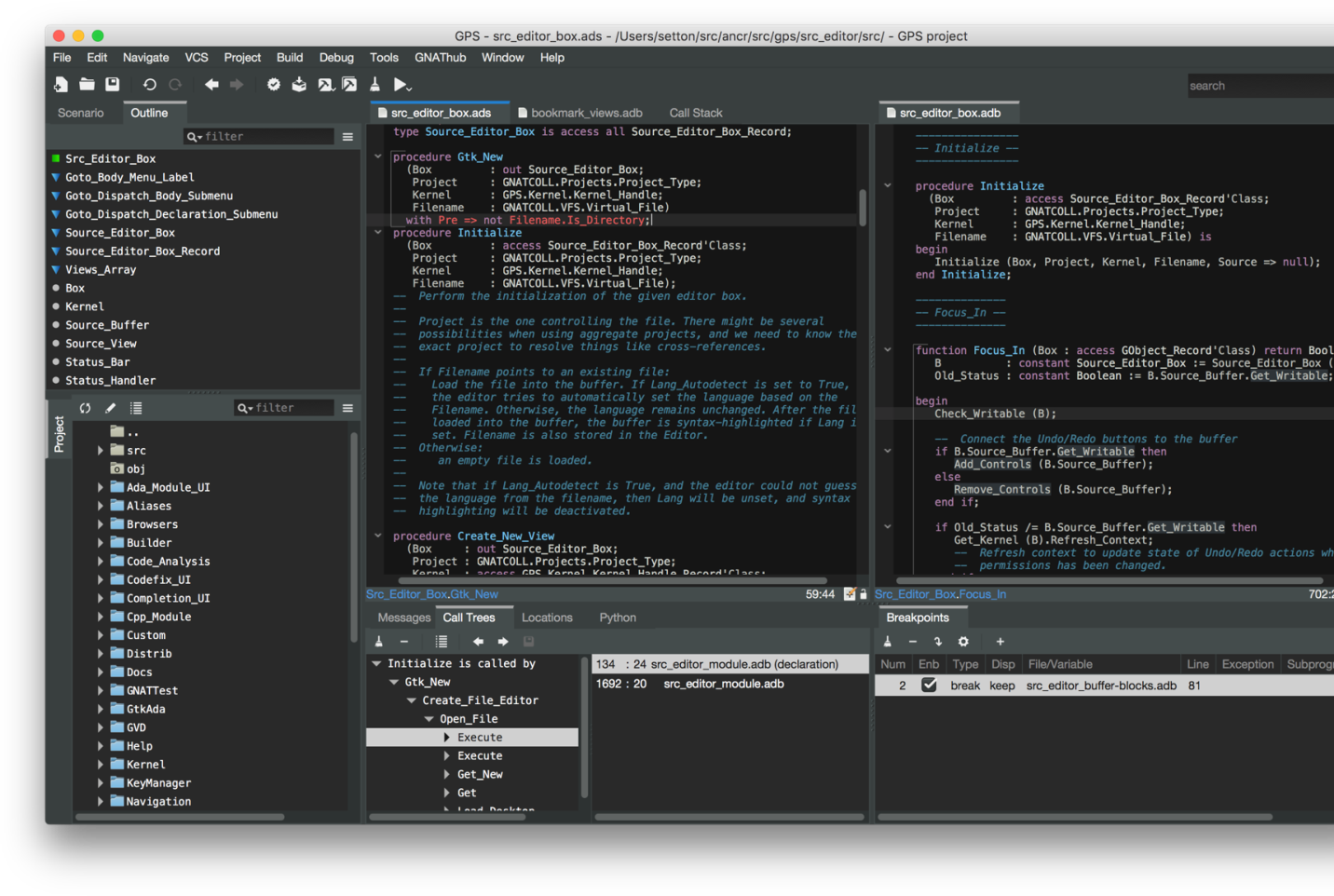


Рисунок 6 – Интерфейс GNAT Programming Studio

### Среда разработки CodeLite

CodeLite - это кроссплатформенная IDE с открытым исходным кодом для языков программирования C / C ++ / PHP и Node.js. Известно, что CodeLite работает на различных ОС, включая: Windows XP SP3, Windows 7 и 8, Debian / Ubuntu.

Интерфейс данной среды разработки представлен на рисунке 7.

Особенности CodeLite:

* поддержка SVN, интеграция с программами для поиска cscope и тестовым фреймворком UnitTest++;
* автодополнение кода, рефакторинг, подсветка синтаксиса для C/C++, XML, Makefile, Diff file и других языков;
* встроенная система генерации документации Doxygen;
* импорт проектов из Visual Studio;
* поддержка отладчика GDB;
* автогенерация make-файлов, возможность использования системы сборки qmake;
* встроенная система статического анализа С/C++ кода CppCheck.

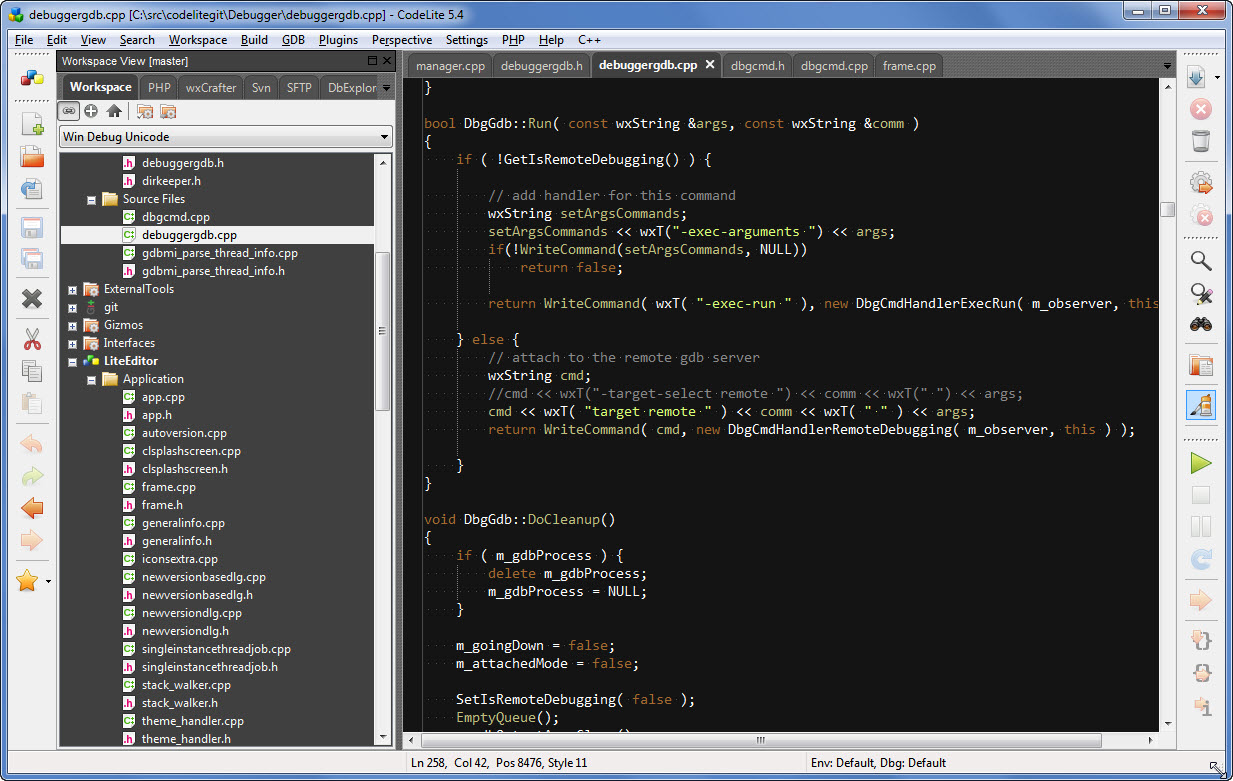


Рисунок 7 – Интерфейс CodeLite

### Среда разработки QtCreator

Qt Creator — это полнофункциональная интегрированная среда разработки (IDE) для приложений C ++, QML и JavaScript. Это комплексная кроссплатформенная IDE, которая разрешает пользователям масштабироваться выше кода и создавать подключенные устройства, приложения и пользовательские интерфейсы.

ПО представляет собой единую сложную консоль, которая позволяет разработчикам создавать приложения для множества встроенных платформ, настольных систем, мобильных устройств, включая iOS и Android.

В приложении имеется редактор кода, который легко интегрируется с другими инструментами для кодирования, проектирования, развертывания, тестирования, мониторинга.

Интерфейс данной среды разработки представлен на рисунке 8.

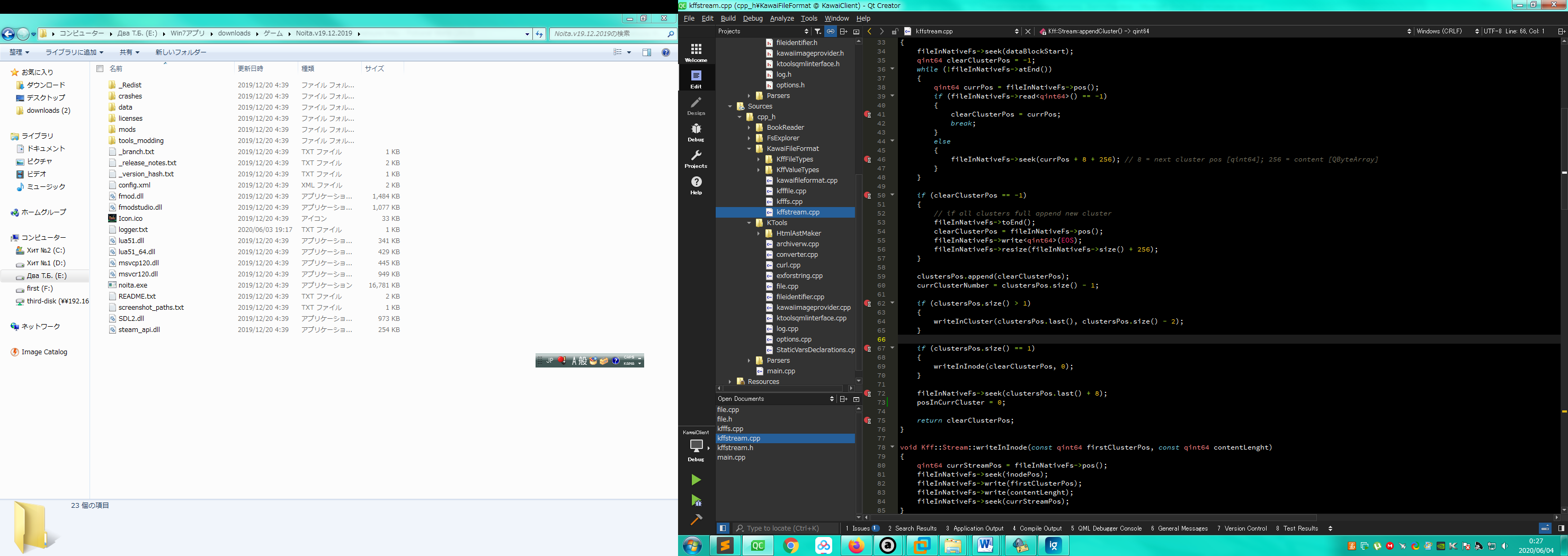


Рисунок 8 – Интерфейс Qt Creator

Возможности Qt Creator:

* редактор кода с автозаполнением и подсветкой синтаксиса;
* перетаскивание и дизайн пользовательского интерфейса;
* инструменты профилирования;
* визуальная отладка;
* интегрированный графический интерфейс;
* дизайнер форм;
* управление проектом.

### Среда разработки Dev-C++

Dev-C++ – это портативная (IDE) интегрированная среда разработки и компилятор для языка программирования C++. Это поддерживаемая версия Dev-C++, которая включает в себя обновленный компилятор MinGW и обновленный код. Благодаря GCC, Mingw или Cygwin в качестве набора компиляторов и библиотек, эта среда разработки C ++ была перенесена для использования в среде Windows.

Интерфейс данной среды разработки представлен на рисунке 9.

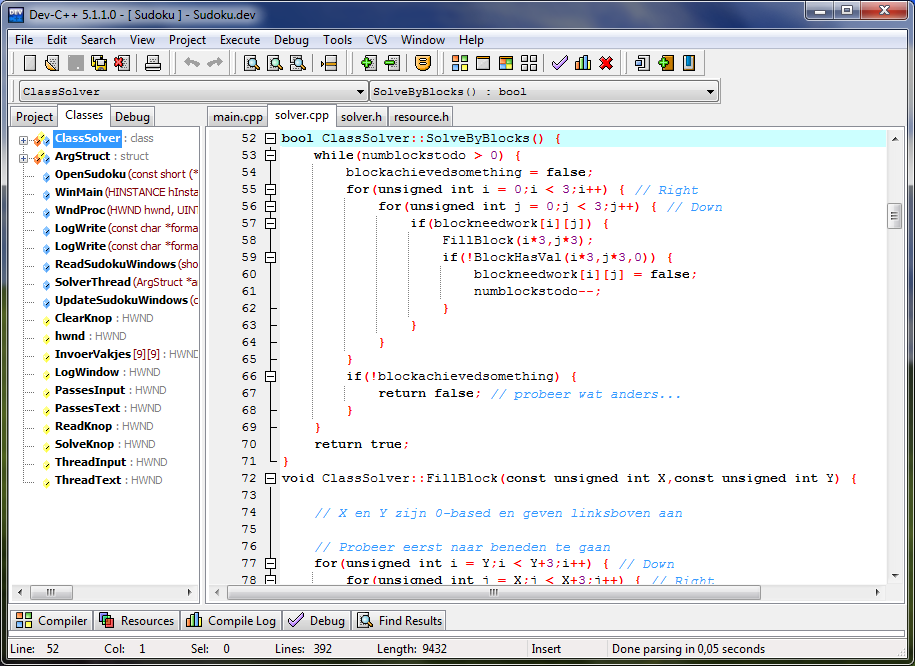


Рисунок 9 – Интерфейс Dev-C++

Возможности Dev-C++:

* удобный редактор с подсветкой синтаксиса, нумерацией строк, автоотступами и т.д.;
* возможность автозавершения кода для удобства работы и повышения производительности;
* заготовки кода и шаблоны для вставки;
* использование закладок в редакторе для быстрого перемещения по коду;
* экспорт исходных файлов или целого проекта в HTML или RTF для публикации исходных кодов на своём веб-сайте;
* встроенный менеджер проектов;
* импорт проектов из MS Visual C++;
* возможность настройки ассоциации файлов по расширению – c, cpp, h и т.д;
* в навигаторе классов два варианта обзора - просмотр функций, классов и их членов как для всего проекта, так и для текущего редактируемого файла;
* гибкая настройка рабочей среды, редактора и компилятора, большое количество различных опций;
* используется Mingw GCC компилятор, может работать с любым компилятором GCC;
* возможность отладки проекта - встроенный дебаггер GDB;
* возможна работа с CVS (скачивается отдельно);
* существует портабельная версия программы, не требующая установки;
* мультиязычный пользовательский интерфейс с поддержкой русского и украинского языков.

### Среда разработки C++Builder

C++Builder – программный продукт, инструмент быстрой разработки приложений (RAD), интегрированная среда программирования (IDE), система, используемая программистами для разработки программного обеспечения на языке C++.

Изначально разрабатывался компанией Borland Software, а затем её подразделением CodeGear, ныне принадлежащем компании Embarcadero Technologies.

C++Builder объединяет в себе комплекс объектных библиотек (STL, VCL, CLX, MFC и др.), компилятор, отладчик, редактор кода и многие другие компоненты. Цикл разработки аналогичен Delphi. Большинство компонентов, разработанных в Delphi, можно использовать и в C++Builder без модификации, но обратное утверждение не верно.

C++Builder содержит инструменты, которые при помощи drag-and-drop действительно делают разработку визуальной, упрощает программирование благодаря встроенному WYSIWYG-редактору интерфейса и пр.

Возможности C++Builder:

* быстрое создание приложений для Windows с VCL для обеспечения непревзойденной производительности;
* расширение визуальных пользовательских интерфейсов с использованием компонентов от разных сертифицированных партнеров;
* использование специфичных для конкретной платформы элементов управления для улучшения UX.

Интерфейс данной среды разработки представлен на рисунке 10.

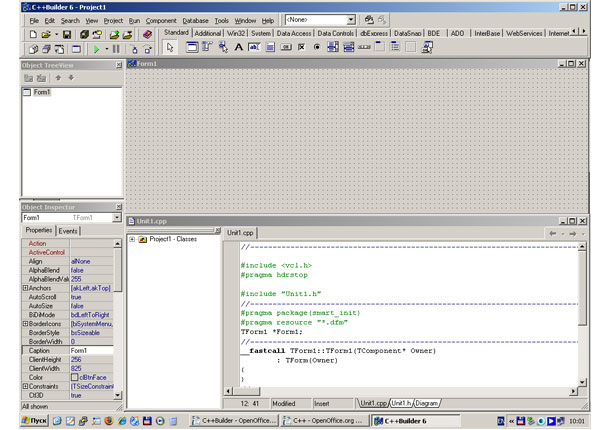


Рисунок 10 – Интерфейс C++Builder

### Среда разработки Anjuta

Anjuta – это гибкая интегрированная среда разработки для языков C и C++, которая включает ряд мощных средств для программирования. Среди них - средства управления проектом, мастера приложений, встроенный интерактивный отладчик, мощный редактор исходного кода со средствами просмотра и подсветкой синтаксиса.

Интерфейс данной среды разработки представлен на рисунке 11.

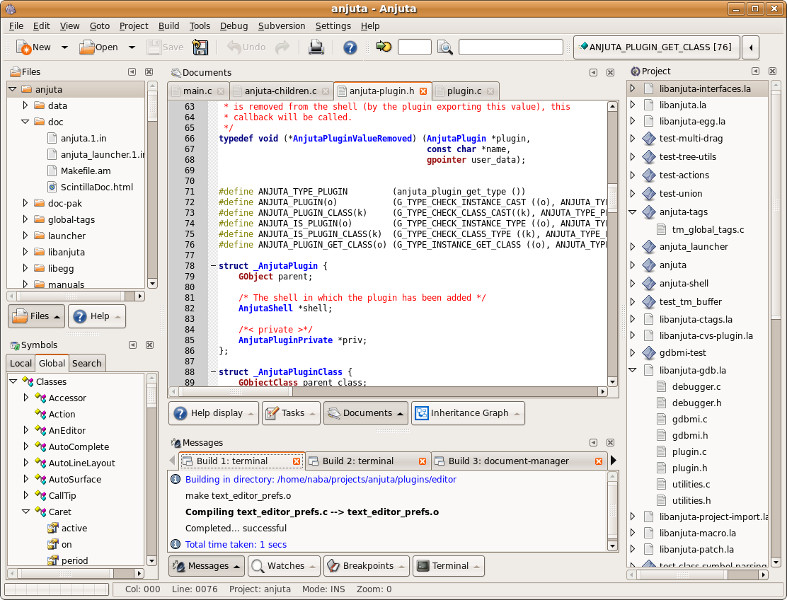


Рисунок 11 – Интерфейс Anjuta

Возможности Anjuta:

* простой интерфейс;
* наличие шаблонов проектов;
* интегрированный отладчик;
* поддержка языков программирования C, C++, Java, Javascript, Python.

### Среда разработки CLion

Интеллектуальная кроссплатформенная C / C ++ IDE. CLion - это новая среда разработки, которая помогает более продуктивно разрабатывать приложения на C и C ++ для Windows, Linux и OS X. Преймущества данной среды разработки в эффективной навигации по коду, умного завершения кода, «на лету» анализ с предложениями по быстрому исправлению, широкими возможностями генерации кода и превосходным рефакторингом кода, построенным на современной платформе IntelliJ.

Интерфейс данной среды разработки представлен на рисунке 12.

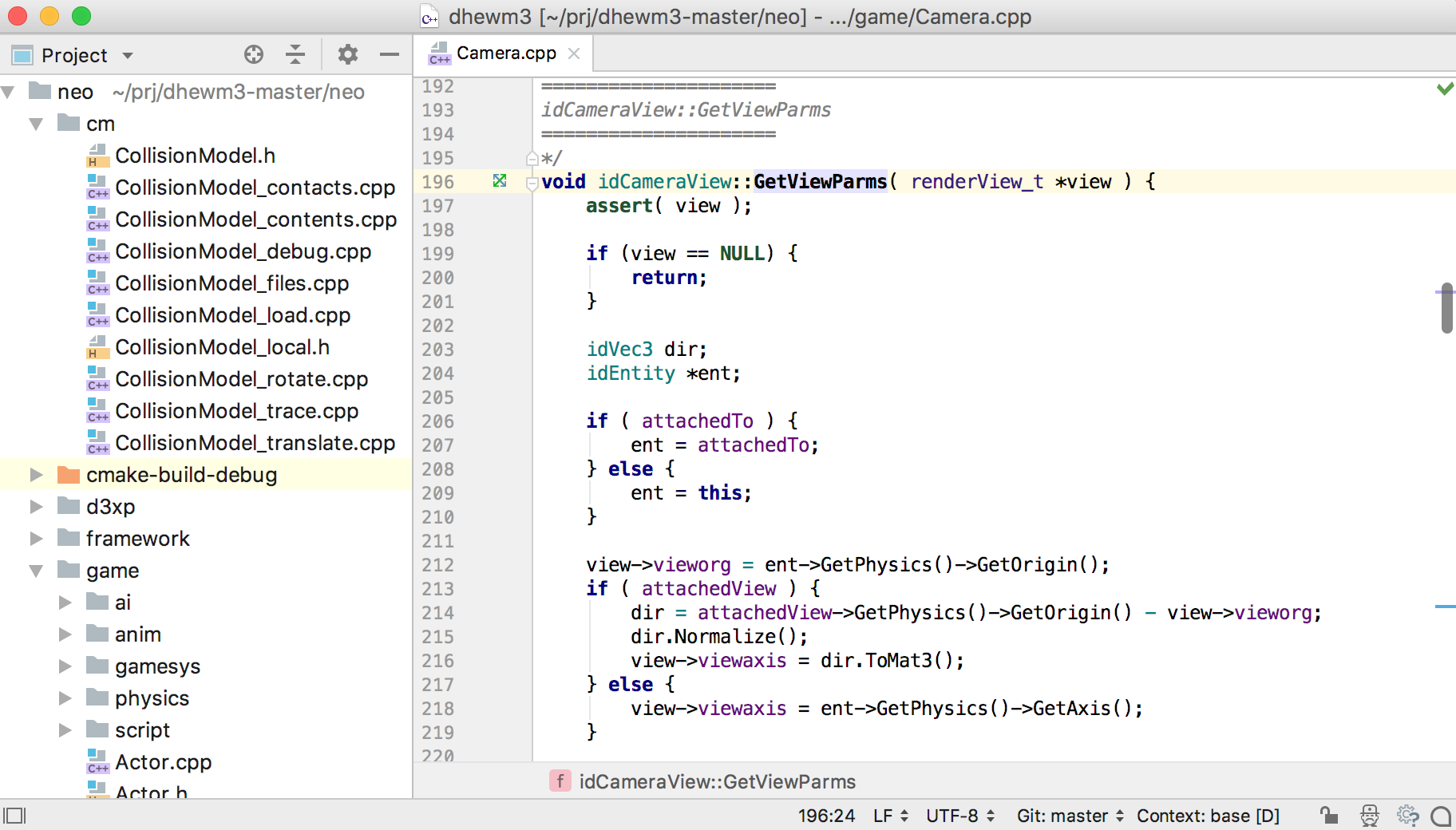


Рисунок 12 – Интерфейс CLion

Возможности CLion:

* удобная навигация по коду;
* продвинутая генерация кода;
* удобные средства рефакторинга;
* встроенный отладчик;
* создание нескольких конфигураций сборки проекта.

### Среда разработки MonoDevelop

MonoDevelop – свободная мультиплатформенная среда разработки, предназначенная для создания приложений на языках C#, C, C++, Java, Visual Basic.NET, CIL, Nemerle, Boo. На данный момент среда является частью проекта Mono (http://mono-project.com) и является одной из самых лучших IDE для разработки проектов на базе Mono.

MonoDevelop обладает всеми основными возможностями, необходимыми для современной интегрированной среды разработки.

Интерфейс данной среды разработки представлен на рисунке 13.

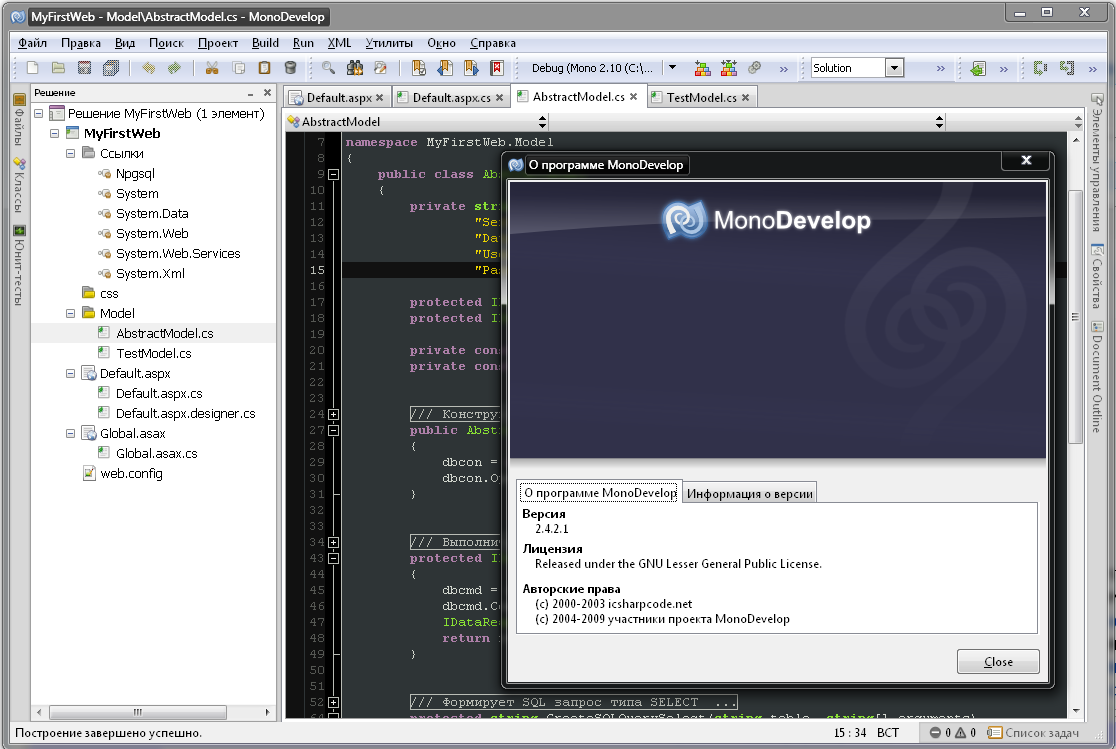


Рисунок 13 – Интерфейс MonoDevelop

Возможности MonoDevelop:

* настраиваемая подсветка синтаксиса;
* автоматическое дополнение кода;
* выделение блоков кода с возможностью их сворачивания/разворачивания;
* интеллектуальная работа с отступами в коде;
* возможности рефакторинга (переименование классов и методов, автоматическая реализация интерфейсов в производных классах);
* удобная навигация по коду (навигация по классам, методам, свойствам);
* визуальный редактор форм для проектов на Gtk#;
* создание нескольких раскладок интерфейса и переключение между ними;
* множество стандартных шаблонов проектов;
* возможность автоматического создания бинарных пакетов и архивов после компиляции исходного кода;
* работа с базами данных;
* создание приложений с GUI, поддерживающим несколько языков;
* интеграция с Subversion для управления исходным кодом;
* поддержка NUnit для создания Unit-тестов;
* автоматическое создание документации;
* расширение возможностей за счет дополнений и внешних инструментов.

### Среда разработки KDevelop

Kdevelop - среда разработки проекта KDE, активно развивается с 1998 г. Kdevelop помимо проектов на языке C, позволяет вести проекты практически на всех используемых в Linux языках: C++, Perl, Python, PHP, Java, Fuby, Ada, Bash, Pascal, Fortran. Эта среда позволяет интегрировать (технология KParts) различные текстовые редакторы, но основным редактором является Kate.

Kdevelop плотно интегрирован с Linux (в отличие от большинства других IDE он не является много-платформенным). Среда умеет генерировать начальные скелеты приложений. Отличительной особенностью Kdevelop (большим плюсом в некоторых случаях) является то, что среди таких шаблонов есть и проект модуля ядра (драйвера) Linux.

Интерфейс данной среды разработки представлен на рисунке 14.

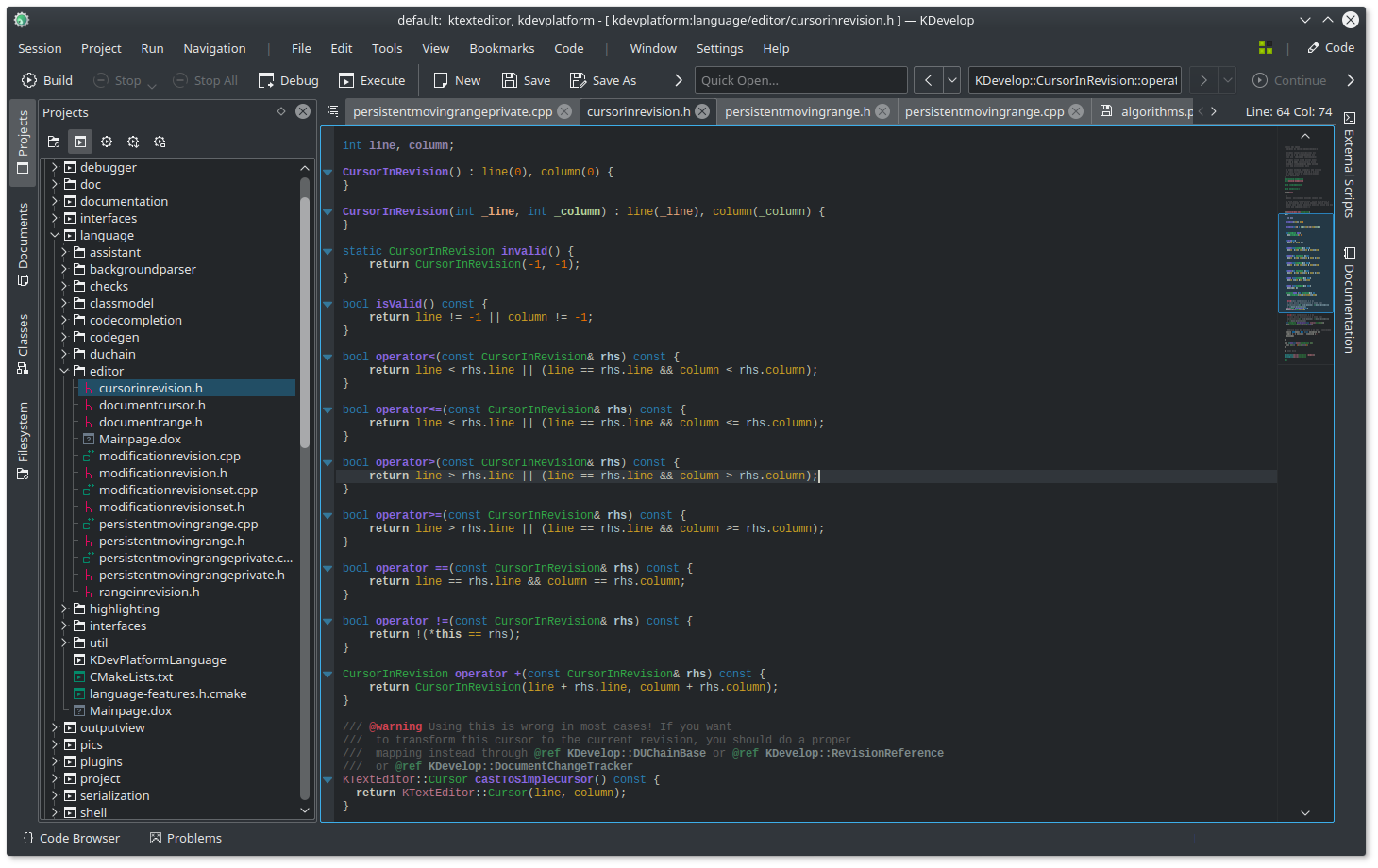


Рисунок 14 – Интерфейс Kdevelop

Особенности:

* различное выделение имён переменных и имён классов даже в тех случаях, когда другие IDE не способны различить их;
* сложная компиляция, которая происходит с учетом всех семантических особенностей кода;
* настраиваемая цветовая схема;
* выделение каждому проекту фиксированного кеша на диске (его размер зависит от размера самого проекта).

Для разработки данного программного продукта была выбрана среда разработки Qt Creator. Причина выбора данной среды разработки в том, что, так как C++ не имеет нативного GUI, для разработки данной программы необходим фреймворк предоставляющий GUI. И лучше всего на роль этого фреймворка подошёл Qt. И исходя из этих соображений, была выбрана та среда разработки, которая имеет наилучшую интеграцию с данным фреймворком.

# СПЕЦИАЛЬНАЯ ЧАСТЬ

В этом разделе описаны все методы, алгоритмы и процессы разработки дипломного проекта, а также техническое задание и инструкция пользователя.

## Техническое задание

В текущем разделе представлено техническое задание, содержащее постановку задачи, назначение программного продукта, основание для разработки, а также требования к конечному программному продукту.

Разработать приложение для ОС Windows 7 и новее, имеющее простой и интуитивный интерфейс, который выполняет основную функцию: искать ключевые слова на заданную тему.

### Назначение программного продукта

Данный программный продукт должен представлять пользователям удобный интерфейс для поиска ключевых слов.

### Основание для разработки

Основанием для разработки программы является задание на дипломное проектирование и задание на преддипломную практику.

### Требования к программе

В этом разделе описаны аппаратные требования необходимые для корректного функционирования программы.

#### Требования к функциональным характеристикам

Программа должна выполнять следующую функцию:

* поиск ключевых слов на заданную тему;

и иметь простой и интуитивный интерфейс.

Входные данные в приложении – это название видео, для которого требуется найти ключевые слова.

Выходные данные в приложении – это ключевые слова, подходящие для видео с указанным названием.

#### Требования к аппаратным и программным средствам

Для полноценного функционирования программы необходим доступ в Интернет. Остальные требования к данному программному продукту перечислены в таблице 1.

Таблица 1 – Системные требования для установки программы

|  |  |  |
| --- | --- | --- |
|  | Минимальные | Рекомендуемые |
| Операционная система (ОС) | Windows 7, Ubuntu 16.04 LTS (64 разрядная) | Windows 7 или новее, Ubuntu 16.04 LTS (64 разрядная) или новее |
| Память на диске | 150МБ | 150МБ |
| Оперативная память | 1ГБ | 1ГБ или больше |
| Процессор | Intel Core i3 540 | Intel Core i5 2300 |

### Требования к надёжности

Данный раздел определяет требуемый уровень надежности разрабатываемого приложения.

#### Требования к надёжному функционированию системы

Надёжное функционирование данного программного продукта обеспечивается выполнением следующих мер:

* бесперебойное питание компьютера;
* бесперебойное подключение к сети интернет.

#### Отказы из-за некорректных действий пользователя программного продукта

Для того, чтобы избежать критических ошибок, необходимо проверять данные введённые пользователем, оповещать его о вводе некорректных данных, и запросить ввод корректных данных.

## Описание функционально-логической структуры данных

В этой части пояснительной записки расположены разработанные схемы и диаграммы, описывающие алгоритмы разрабатываемого программного продукта.

### Структурная схема приложения

Структурная схема – это совокупность элементарных звеньев объекта и связей между ними, один из видов графической модели. Под элементарным звеном понимают часть объекта, системы управления и т. д., которая реализует элементарную функцию.

Структурная схема приложения представлена ниже на рисунке 15.

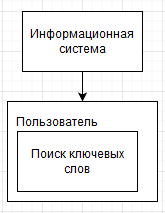


Рисунок 15 – Структурная схема приложения

### Функциональная схема приложения

Функциональная схема – документ, разъясняющий процессы, протекающие в отдельных функциональных цепях программы. Функциональная схема является поясняющим материалом отдельных видов процессов, протекающих в целостных функциональных блоках и цепях устройства.

Функциональная схема приложения представлена ниже на рисунке 16.

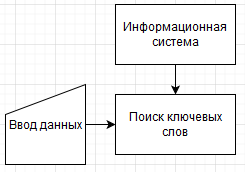


Рисунок 16 –Функциональная схема приложения

### Диаграммы методологии IDEF0

IDEF0 — методологии семейства ICAM (Integrated Computer-Aided Manufacturing) для решения задач моделирования сложных систем, позволяет отображать и анализировать модели деятельности широкого спектра сложных систем в различных разрезах. При этом широта и глубина обследования процессов в системе определяются самим разработчиком, что позволяет не перегружать создаваемую модель (Рисунок 17) излишними данными.

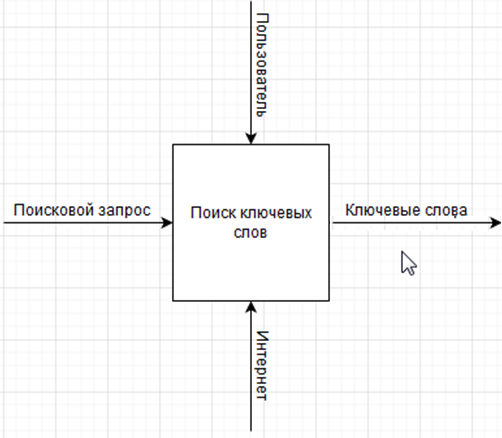


Рисунок 17 – Диаграмма методологии IDEF0

### Диаграмма вариантов использования (UML)

Диаграмма вариантов использования описывает функциональное назначение системы или, другими словами, все возможности приложения во время работы. Она является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки. У данной диаграммы будет только один взаимодействующий с ней актер – пользователь.

Построенная диаграмма вариантов использования приложения представлена на рисунке 18.

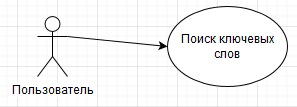


Рисунок 18 – Диаграмма вариантов использования

## Описание алгоритма

Этот раздел описывает алгоритмы приложения и алгоритмы для работы отдельных модулей программы.

### Общий алгоритм работы программы

При открытии приложения пользователь увидит пустую строку ввода и кнопку. В строку ввода ему необходимо ввести название видео, для которого он желает найти ключевые слова. После того как он введёт название видео ему необходимо нажать на кнопку и через некоторое время ниже строки ввода появятся подобранные программой ключевые слова.

Алгоритм работы программы представлен в виде блок-схемы на рисунке 19.

Блок-схема – распространенный тип схем, описывающий алгоритмы или процессы, изображая шаги в виде блоков различной формы, соединенных между собой стрелками.

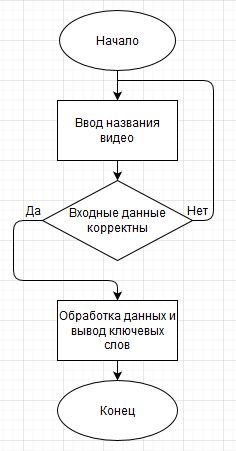


Рисунок 19 – Алгоритм работы программы

## Программная реализация

В данном разделе рассматривается программная реализация описанных ранее алгоритмов, а также проводится тестирование и отладка программы.

### Пользовательский интерфейс

Пользователь характеризует любое приложение следующими свойствами:

* удобство (понятность) интерфейса;
* быстродействие приложения;
* отказоустойчивость, связанная с ошибками пользователя.

Интерфейс пользователя считается удобным, если пользователь без дополнительных знаний и навыков может им пользоваться.

Далее будет рассмотрен интерфейс данного программного продукта.

На рисунке 20 представлен интерфейс данной программы.

Рисунок 20 – Интерфейс данного программного продукта

В верхней части интерфейса данного программного продукта расположены строка ввода и кнопка. В строку ввода вводится название видео, для которого необходимо найти ключевые слова. Далее, после того, как название введено, необходимо нажать по кнопке «найти ключевые слова», которая расположена справа от строки ввода, и приложение начнёт искать ключевые слова.

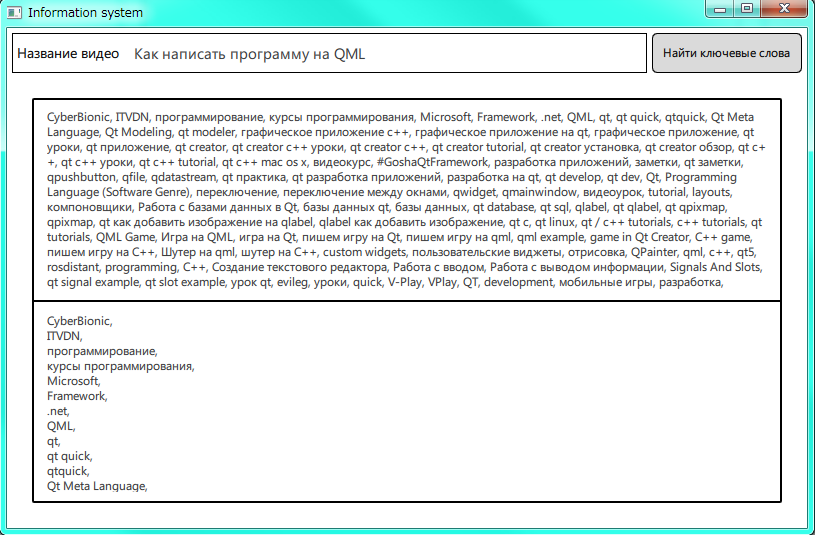


Рисунок 21 – Интерфейс данного программного продукта

После того как приложение нашло ключевые слова, (рисунок 21) они будут отображены в двух прямоугольниках, располагающихся ниже строки ввода и кнопки. В верхнем прямоугольнике ключевые слова отображаются подряд, разделённые только запятой. В нижнем прямоугольнике ключевые слова отображаются разделённые запятой и пробелом для того чтобы их было более удобно читать. После того как ключевые слова отобразились, пользователю остаётся только выделить их и копировать в своё видео.

### Тестирование и отладка программы

В данном разделе будут рассмотрены одни из важных этапов разработки: отладка и тестирование разработанного приложения.

Тестирование позволяет выявить большинство ошибок допущенных при создании программы.

В данном пункте будет описано тестирование программы по принципу «чёрного ящика».

Тестовый набор будет представлять собой набор фраз содержащих:

* символы различных языков;
* пробелы;
* специальные символы.

Также при тестировании будет протестирована реакция программы на пустой ввод.

При пустом вводе между строкой ввода и верхним прямоугольником появляется надпись «Пустой ввод неприемлем» и поиск ключевых слов не начинается. При любом ином вводе поиск ключевых слов начинается и выдаёт ожидаемый результат.

## Инструкция пользователя

В данном разделе описана инструкция пользователя, описывающая работу и функционал данного программного продукта.

При запуске программы пользователя встречает окно программы, представленное на рисунке 20 (страница 36). Из данного окна пользователь может выполнять в программе все необходимые действия. Строка ввода, находящаяся в левой верхней части окна программы, предназначена для ввода названия видео, для которого необходимо найти ключевые слова. Справа от строки ввода расположена кнопка, при нажатии на которую, начинается поиск ключевых слов. Между строкой ввода и двумя прямоугольниками появляется строка состояния, которая сообщает пользователю либо об ошибках, либо о том, что идёт поиск ключевых слов. Когда ключевые слова найдены, строка состояния исчезает. Далее под строкой ввода расположены два прямоугольника, в которых отображаются найденные ключевые слова. В верхнем прямоугольнике слова отображаются разделённые только запятой, а в верхнем – запятой и переносом строки.

Для того, чтобы найти нужные ключевые слова пользователю необходимо ввести название видео, для которого он желает найти ключевые слова в строку ввода и нажать на кнопку справа от строки ввода. Через некоторое время после этого программа найдёт ключевые слова для видео и пользователю останется только выбрать подходящие и копировать в видео.

## Анализ результатов работы

В данном разделе будут подведены итоги работы с дипломным проектом.

При разработке дипломного проекта были изучены язык программирования C++, фреймворк Qt и библиотека запросов к сети интернет Curl.

Также были произведены тестирование и отладка программы.

Для приложения был разработан интуитивно понятный интерфейс.

Подведя итог можно сказать, что для данного дипломного проекта было разработано полностью работоспособное приложение, отвечающее всем заявленным требованиям.

# ЭКОНОМИЧЕСКАЯ ЧАСТЬ

Данный дипломный продукт позволит облегчить выбор ключевых слов тем людям, которые создают видео на YouTube для его дальнейшего продвижения и популяризации.

Интерфейс приложения интуитивно понятен, что обеспечивает комфортное и эффективное пользование программой.

В данном разделе оценивается стоимость дипломного проекта и рассчитывается предполагаемая прибыль от реализации программы.

## Расчёт затрат на выполнение проекта

Разработка данного программного продукта потребовала определённых материальных, трудовых и временных затрат и следовательно эти затраты должны окупиться.

Для того чтобы данный дипломный проект считался экономически успешным, все затраты связанные с его разработкой, должны быть покрыты доходами от реализации программного продукта.

### Расчёт трудоёмкости

Трудоемкость выполнения программного продукта характеризуется перечнем основных этапов и видов работ, которые должны быть выполнены. Упорядочен данный перечень в соответствии со смысловым содержанием каждого вида работ и взаимосвязями между всеми видами работ.

Выделим основные этапы разработки приложения:

1. Подготовительный этап:

* сбор информации;
* выбор объектного построения программы;
* разработка общей методики создания продукта.

1. Основной этап:

* разработка основных алгоритмов;
* создание интерфейса;
* отладка.

1. Завершающий этап:

* подготовка технической документации;
* интеграция продукта;
* анализ работы;
* сдача продукта.

Трудоемкость выполнения работы оценивается экспертным путем в человеко-часах и представлена в таблице 2.

Таблица 2 – Трудоемкость выполнения работы

|  |  |  |
| --- | --- | --- |
| № п/п | Виды работ | Трудоёмкость |
| 1 | Составление ТЗ | 2 |
| 2 | Сбор информации и ознакомление с предметной областью | 5 |
| 3 | Выбор объектного построения программы | 5 |
| 4 | Разработка общей методики создания продукта | 5 |
| 5 | Разработка системы взаимодействия с потребителем | 10 |
| 6 | Разработка основного алгоритма | 35 |
| 7 | Создание интерфейса | 11 |
| 8 | Отладка | 8 |

Продолжение таблицы 2

|  |  |  |
| --- | --- | --- |
| 9 | Тестирование | 10 |
| 10 | Подготовка технической документации | 15 |
| 11 | Сдача продукта | 1 |
|  | Итого | 107 |

Суммарная трудоёмкость составила 107 чел./час.

### Расчёт расходов на заработную плату

На основе данных о трудоемкости и средней заработной плате по профессии рассчитываем основную заработную плату. Предположим, что заработная плата программиста без опыта работы составляет 20 тыс. руб. в месяц (22 рабочих дня, 8-ми часовой рабочий день) или 113 руб./час.

Таким образом, расходы на заработную плату по нашему проекту составляют:

113\*107=12091 руб.

Отчисления на заработную плату (Пенсионный Фонд, Фонд социального страхования, Фонд обязательного медицинского страхования) составляют 30%. В денежном выражении составляют:

12091\*0,3= 3627,3 руб.

### Расчёт затрат на материалы

Материалы, затраченные на создание проекта, приведены в таблице 3.

Таблица 3 – Материалы, затраченные на создание проекта

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Материальные ресурсы | Единицы измерения | Потребляемое количество | Цена за единицу, руб. | Сумма, руб. |
| Бумага формата А4 | упаковка | 1 | 190 | 190 |

Продолжение таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Папка | шт | 1 | 240 | 240 |
| Файлы | упаковка | 1 | 197 | 197 |
| Диск | шт | 1 | 60 | 60 |
| Печать | лист | 70 | 5 | 350 |
| Итого |  | | | 1037 |

### Расчет амортизации

Стоимость компьютера 27 тыс. руб. Используем его в течении 6 лет (72 мес.). Годовая амортизация составит в месяц 375 рублей. В час: 0,52 руб. Умножим на трудоёмкость:

0,52\*107 = 55,64 руб.

### Расчет электроэнергии

Персональный компьютер в среднем потребляет 0,3 кВт/час.

0,3\*107=32,1 кВт. Примерная стоимость 1 кВт/час около 3,54 руб.

Итого: 32,1\*3,54 = 113,63 руб.

### Расчет сметы затрат

Результаты расчета затрат приведены в таблице 4.

Таблица 4 – Результаты расчета затрат

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование статей затрат | Сумма, руб. |
| 1 | Расходные материалы | 1037 |
| 2 | Основная заработная плата | 12091 |
| 3 | Расходы на электроэнергию | 113,63 |
| 4 | Амортизационные отчисления | 55,64 |

Продолжение таблицы 4

|  |  |  |
| --- | --- | --- |
| 5 | Отчисления на ЗП | 3627,3 |
|  | Итого | 16924,57 |

## Расчет предполагаемой прибыли

Разработанный программный продукт актуален для тех, кто создаёт видео для YouTube.

Данная программа обладает следующими достоинствами:

* простой и удобный интерфейс;
* низкая стоимость.

Стоимость рыночных аналогов 1-5 тысяч рублей. При реализации данного продукта за 0,5 тысяч рублей 34 раза разработка полностью окупится, и прибыль будет составлять 75,43 рубля.

Созданный программный продукт позволяет человеку, создающему видео для YouTube более быстро, легко и эффективно подбирать ключевые слова, что положительно скажется на количестве просмотров.

В дальнейшем возможно добавление нового функционала по мере его необходимости.

# ЗАКЛЮЧЕНИЕ

При реализации проекта был проведён анализ предметной области, продумано назначение данного программного продукта, приобретены практические навыки по проектированию структуры, разработке и реализации проектирования информационной системы.

Данная информационная система предназначена для более удобного и эффективного подбора ключевых слов для видео, выкладываемого на YouTube. Программа имеет простой и удобный интерфейс, выполняет функцию поиска ключевых слов. Закрепление навыков происходит в ходе решения поставленной задачи – разработки бизнес-процесса, тестирования и отладки информационной системы.

Использование данной информационной системы упрощает поиск ключевых слов для видео, выкладываемого на YouTube, благодаря чему, видео чаще появляется в поисковой выдаче YouTube и рекомендациях пользователей. И это приводит к тому, что больше людей смотрят созданное видео.

Тот, кто выкладывает видео, может и сам придумать ключевые слова, но почти гарантированно он не придумает такое большое количество подходящих ключевых слов, которое данный программный продукт может выдать ему после всего десятка секунд анализа данных. И к тому же подбирать слова через данный программный продукт намного проще, что экономит силы и время, которое можно потратить на создание следующего видео.

# 

# ЛИТЕРАТУРА

1. Довек, Ж. Введение в теорию языков программирования / Довек Жиль, Леви Жан-Жак. — Москва: ДМК Пресс, 2013. — 134 с. - ISBN 978-5-94074-913-4. - Текст: электронный. - URL: https://znanium.com/catalog/document?id=341088 (дата обращения: 08.06.2020).
2. Дудецкий, В.Н. Объектно-ориентированные языки программирования: учеб. пособие : в 3 ч. Ч. I / В.Н. Дудецкий. — 2-е изд., стер. — Москва: ФЛИНТА, 2016. - 48 с. - ISBN 978-5-9765-2252-7. - Текст: электронный. - URL: https://znanium.com/catalog/document?id=343327 (дата обращения: 08.06.2020).
3. Дорогов, В. Г. Основы программирования на языке С : учеб. пособие / В.Г. Дорогов, Е.Г. Дорогова ; под ред. проф. Л.Г. Гагариной. — Москва : ИД «ФОРУМ» : ИНФРА-М, 2019. — 224 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-16-107656-9. - Текст: электронный. - URL: https://znanium.com/catalog/document?id=339306 (дата обращения: 08.06.2020).
4. Бедердинова, О. И. Программирование на языках высокого уровня : учеб. пособие / О.И. Бедердинова, Т.А. Минеева, Ю.А. Водовозова. — Москва : ИНФРА-М, 2019. — 159 с. - Текст: электронный. - URL: https://znanium.com/catalog/document?id=344897 (дата обращения: 08.06.2020).
5. ГОСТ Р 7.0.97-2016. Национальный стандарт Российской Федерации. Система стандартов по информации, библиотечному и издательскому делу. Организационнораспорядительная документация. Требования к оформлению документов" (ред. от 14.05.2018).
6. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления.
7. ГОСТ 7.9—95 Система стандартов по информации, библиотечному и издательскому делу. Реферат и аннотация. Общие требования.
8. <https://stackoverflow.com> – Англоязычный ресурс вопросов и ответов на тему программирования.
9. <https://github.com> – Веб-сервис для хостинга IT-проектов и их совместной разработки.
10. <https://app.diagrams.net/> - Сайт для создания диаграмм.
11. <https://curl.haxx.se/> - Сайт содержащий документацию для библиотеки Curl.
12. <https://doc.qt.io/> - Сайт содержащий документацию для фреймворка Qt.

# ПРИЛОЖЕНИЕ А – Исходный текст программы

**InformationSystem.pro**

QT += quick

CONFIG += c++17

# The following define makes your compiler emit warnings if you use

# any Qt feature that has been marked deprecated (the exact warnings

# depend on your compiler). Refer to the documentation for the

# deprecated API to know how to port your code away from it.

DEFINES += QT\_DEPRECATED\_WARNINGS

# You can also make your code fail to compile if it uses deprecated APIs.

# In order to do so, uncomment the following line.

# You can also select to disable deprecated APIs only up to a certain version of Qt.

#DEFINES += QT\_DISABLE\_DEPRECATED\_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \

cpp\_h/KTools/HtmlAstMaker/object.cpp \

cpp\_h/KTools/HtmlAstMaker/tag.cpp \

cpp\_h/KTools/StaticVarsDeclarations.cpp \

cpp\_h/KTools/converter.cpp \

cpp\_h/KTools/curl.cpp \

cpp\_h/KTools/exforstring.cpp \

cpp\_h/KTools/file.cpp \

cpp\_h/KTools/fileidentifier.cpp \

cpp\_h/KTools/ktoolsqmlinterface.cpp \

cpp\_h/KTools/log.cpp \

cpp\_h/KTools/options.cpp \

cpp\_h/interfaceforqml.cpp \

cpp\_h/parserclass.cpp \

main.cpp \

youtube.cpp

RESOURCES += qml.qrc

# Additional import path used to resolve QML modules in Qt Creator's code model

QML\_IMPORT\_PATH =

# Additional import path used to resolve QML modules just for Qt Quick Designer

QML\_DESIGNER\_IMPORT\_PATH =

# Default rules for deployment.

qnx: target.path = /tmp/$${TARGET}/bin

else: unix:!android: target.path = /opt/$${TARGET}/bin

!isEmpty(target.path): INSTALLS += target

HEADERS += \

cpp\_h/KTools/HtmlAstMaker/object.h \

cpp\_h/KTools/HtmlAstMaker/tag.h \

cpp\_h/KTools/converter.h \

cpp\_h/KTools/curl.h \

cpp\_h/KTools/enums.h \

cpp\_h/KTools/exforstring.h \

cpp\_h/KTools/file.h \

cpp\_h/KTools/fileidentifier.h \

cpp\_h/KTools/ktoolsqmlinterface.h \

cpp\_h/KTools/log.h \

cpp\_h/KTools/options.h \

cpp\_h/interfaceforqml.h \

cpp\_h/parserclass.h \

youtube.h

#Curl

LIBS += -L$$PWD/libs/curl-x64/lib/dll/ -llibcurl.dll

INCLUDEPATH += $$PWD/libs/curl-x64/include

DEPENDPATH += $$PWD/libs/curl-x64/include

**main.cpp**

#include <QGuiApplication>

#include <QQmlApplicationEngine>

#include <QQmlContext>

#include "cpp\_h/interfaceforqml.h"

#include <curl/curl.h>

int main(int argc, char \*argv[])

{

QTextCodec \*codec = QTextCodec::codecForName("UTF-8");

QTextCodec::setCodecForLocale(codec);

QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);

QGuiApplication app(argc, argv);

curl\_global\_init(CURL\_GLOBAL\_ALL);

QQmlApplicationEngine engine;

const QUrl url(QStringLiteral("qrc:/main.qml"));

QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,

&app, [url](QObject \*obj, const QUrl &objUrl) {

if (!obj && url == objUrl)

QCoreApplication::exit(-1);

}, Qt::QueuedConnection);

InterfaceForQml \*in = new InterfaceForQml();

engine.rootContext()->setContextProperty("inter", in);

engine.load(url);

return app.exec();

}

**main.qml**

import QtQuick 2.13

import QtQuick.Window 2.13

import QtQuick.Controls 2.2

import "qml/QmlIncludes"

import "qml"

Window {

visible: true

width: 800

height: 500

title: qsTr("Information system")

TextArea {

id: infoText

anchors.left: parent.left

anchors.right: parent.right

anchors.top: addressInput.bottom

height: 40

font.family: "Segoe UI"

readOnly: true

selectByMouse: true

}

KawaiLabelInputFieldCombination {

id: addressInput

anchors.left: parent.left

anchors.right: submitButton.left

anchors.top: parent.top

anchors.margins: 5

klifcLabel.text: "Название видео"

klifcLabel.font.family: "Segoe UI"

klifcTextField.font.family: "Segoe UI"

}

KawaiButton {

id: submitButton

anchors.top: parent.top

anchors.right: parent.right

anchors.margins: 5

labelText: "Найти ключевые слова"

widthC: 150

onClicked: {

if (addressInput.klifcTextField.text != "")

{

if (allDataPiecesHere)

{

inter.getData({url: addressInput.klifcTextField.text})

infoText.text = "Идёт обработка данных"

resultText.text = ""

lineByLineResultText.text = ""

keywordsCounter = {}

}

}

else

{

infoText.text = "Пустой ввод неприемлем"

}

}

}

Rectangle {

anchors.top: addressInput.bottom

anchors.bottom: parent.bottom

anchors.right: parent.right

anchors.left: parent.left

anchors.margins: 25

border.color: "Black"

border.width: 2

radius: 2

ScrollView{

id: resultScroll

anchors.top: parent.top

anchors.left: parent.left

anchors.right: parent.right

anchors.bottom: dividingLine.top

anchors.margins: 5

TextArea {

id: resultText

wrapMode: TextArea.WordWrap

font.family: "Segoe UI"

readOnly: true

selectByMouse: true

}

}

KawaiHorisontalLine {

id: dividingLine

anchors.verticalCenter: parent.verticalCenter

anchors.left: parent.left

anchors.right: parent.right

}

ScrollView {

id: lineByLineResultScroll

anchors.top: dividingLine.bottom

anchors.left: parent.left

anchors.right: parent.right

anchors.bottom: parent.bottom

anchors.margins: 5

TextArea {

id: lineByLineResultText

wrapMode: TextArea.WordWrap

font.family: "Segoe UI"

readOnly: true

selectByMouse: true

}

}

}

property var keywordsCounter: ({})

property bool allDataPiecesHere: true

Connections {

target: inter

onSingleData: {

allDataPiecesHere = data["last"]

if (data["last"])

infoText.text = ""

for (var i = 0; data["keywords"][i]; i++)

{

if (keywordsCounter[data["keywords"][i]])

keywordsCounter[data["keywords"][i]] += 1

else

keywordsCounter[data["keywords"][i]] = 1

}

var str = ""

var str2 = ""

var keys = Object.keys(keywordsCounter)

for (var i = 0; keys[i]; i++)

{

str += keys[i]

str2 += keys[i]

if (keys[i + 1])

{

str += ", "

str2 += ",\n"

}

}

resultText.text = str

lineByLineResultText.text = str2

}

}

}

**qml.qrc**

<RCC>

<qresource prefix="/">

<file>main.qml</file>

<file>qml/QmlIncludes/KawaiButton.qml</file>

<file>qml/QmlIncludes/KawaiDownloadBar.qml</file>

<file>qml/QmlIncludes/KawaiHorisontalLine.qml</file>

<file>qml/QmlIncludes/KawaiLabelInputFieldCombination.qml</file>

<file>qml/QmlIncludes/KawaiLoadTabButton.qml</file>

<file>qml/QmlIncludes/KawaiSwitchableWindows.qml</file>

<file>qml/QmlIncludes/KawaiVerticalLine.qml</file>

<file>qml/StatisticWindow.qml</file>

<file>js/Tools.js</file>

<file>resources/other/cacert.pem</file>

<file>qml/InfoAboutVideo.qml</file>

</qresource>

</RCC>

**youtube.cpp**

#include "youtube.h"

YouTube::YouTube() {}

void YouTube::extractInfo()

{

QMap<QString, QString> commentsChunk = {

{"Accept", "\*/\*"},

{"Host", "www.youtube.com"},

{"Content-Type", "application/x-www-form-urlencoded"},

{"Origin", "https://www.youtube.com"},

{"DNT", "1"},

{"Pragma", "no-cache"},

{"Cache-control", "no-cache"},

{"TE", "Trailers"},

};

QString dir = KTools::Options::configPath + "/tmp";

cc->currHeaderMode = KTools::Enums::Curl::HeaderMode::None;

cc->currCookieMode = KTools::Enums::Curl::CookieMode::GetAllTimes;

cc->cookieFileName = "cookieYoutube.txt";

cc->cookiePath = KTools::Options::configPath;

cc->setOptions();

QByteArray data = cc->request("https://www.youtube.com");

KTools::HtmlAst::Object htmlObj = KTools::HtmlAst::Object();

htmlObj.makeAst(data);

commentsChunk["X-YouTube-Client-Version"] = htmlObj.arrsAndObjs.objects[0].value("INNERTUBE\_CONTEXT\_CLIENT\_VERSION").toString();

commentsChunk["X-YouTube-Client-Name"] = htmlObj.arrsAndObjs.objects[0].value("INNERTUBE\_CONTEXT\_CLIENT\_NAME").toVariant().toString();

cc->currHeaderMode = KTools::Enums::Curl::HeaderMode::Custom;

cc->setHeader(commentsChunk);

cc->setOptions();

QJsonArray jsonArr = KTools::Converter::convert<QString, QJsonArray>(cc->request("https://www.youtube.com/results?search\_query=" + QUrl::toPercentEncoding(params.value("url").toString()) + "&sp=EgIQAQ%253D%253D&pbj=1"));

QJsonArray videosInfo = jsonArr[1].toObject().value("response").toObject().value("contents").toObject().value("twoColumnSearchResultsRenderer").toObject().value("primaryContents").toObject().value("sectionListRenderer").toObject().value("contents").toArray()[0].toObject().value("itemSectionRenderer").toObject().value("contents").toArray();

QVector<QString> videosId;

for (int i = 0; i < videosInfo.size(); i++)

{

if (!videosInfo[i].toObject().value("videoRenderer").isUndefined())

videosId.append(videosInfo[i].toObject().value("videoRenderer").toObject().value("videoId").toString());

}

for (int i = 0; i < videosId.size(); i++)

{

data = cc->request("https://www.youtube.com/watch?v=" + videosId[i]);

QVector<QVector<QVector<QString>>> regexResult;

QString videoHtmlPage = data;

QJsonObject singleResObj;

regexResult = QVector<QVector<QVector<QString>>>();

QVector<QString> patterns = {

{R"raw(,\\"keywords\\":(\[[^]]+\]))raw"}, // keywords

};

KTools::ExForString::executeRegex(videoHtmlPage, patterns, regexResult);

if (regexResult[0].size() > 0)

singleResObj["keywords"] = KTools::Converter::convert<QString, QJsonArray>(regexResult[0][0][1].replace("\\", ""));

if (videosId.size() == i + 1)

singleResObj["last"] = true;

else

singleResObj["last"] = false;

emit singleInfo(singleResObj);

QString nope;

}

emit infoExtracted(result);

}

**youtube.h**

#ifndef YOUTUBE\_H

#define YOUTUBE\_H

#include "cpp\_h/parserclass.h"

#include <QTimeZone>

#include <QEventLoop>

#include <QNetworkAccessManager>

#include <QNetworkReply>

class YouTube : public ParserClass

{

Q\_OBJECT

public:

YouTube();

QJsonObject params;

QJsonArray result;

public slots:

void extractInfo();

signals:

void infoExtracted(QJsonArray info);

void singleInfo(QJsonObject info);

};

#endif // YOUTUBE\_H

**interfaceforqml.cpp**

#include "interfaceforqml.h"

#include <QFutureWatcher>

InterfaceForQml::InterfaceForQml() {}

void InterfaceForQml::getData(const QJsonObject &params)

{

QThread \*thread = new QThread();

api = new YouTube();

api->params = params;

api->moveToThread(thread);

connect(thread, SIGNAL(started()), api, SLOT(extractInfo()));

//connect(api, SIGNAL(infoExtracted(QJsonArray)), this, SLOT(sendResult(QJsonArray)));

connect(api, SIGNAL(singleInfo(QJsonObject)), this, SLOT(sendSingleResult(QJsonObject)));

connect(api, SIGNAL(infoExtracted(QJsonArray)), thread, SLOT(quit()));

thread->start();

}

void InterfaceForQml::sendResult(QJsonArray data)

{

emit ytData(data);

}

void InterfaceForQml::sendSingleResult(QJsonObject data)

{

emit singleData(data);

}

**interfaceforqml.h**

#ifndef INTERFACEFORQML\_H

#define INTERFACEFORQML\_H

#include <QObject>

#include <QThread>

#include <QtConcurrent/QtConcurrent>

#include "youtube.h"

class InterfaceForQml : public QObject

{

Q\_OBJECT

public:

InterfaceForQml();

YouTube \*api;

QJsonObject \*par;

InterfaceForQml \*obj;

QJsonArray \*result;

QFutureWatcher<QJsonArray> futWat;

public slots:

void sendResult(QJsonArray data);

void getData(const QJsonObject &params);

void sendSingleResult(QJsonObject data);

signals:

void ytData(const QJsonArray &data);

void singleData(const QJsonObject data);

};

#endif // INTERFACEFORQML\_H

**parserclass.cpp**

#include "parserclass.h"

ParserClass::ParserClass()

{

defExt = new KTools::FileIdentifier();

cc = new KTools::Curl();

}

ParserClass::~ParserClass()

{

delete defExt;

delete cc;

}

void ParserClass::recExtractJson(const QJsonObject &rootObject, QString offset, QString &data)

{

QString tmp;

QStringList keysChain = rootObject.keys(); // Number; Null; Array; String; Object(aka array); Bool;

for (int i = 0; i < keysChain.length(); i++)

{

if (rootObject.value(keysChain[i]).isString()) // is string

data += offset + keysChain[i] + ": " + rootObject.value(keysChain[i]).toString() + '\n';

else if (rootObject.value(keysChain[i]).isObject()) // is object

{

data += offset + "\\+" + keysChain[i] + '\n';

recExtractJson(rootObject.value(keysChain[i]).toObject(), offset + '\t', data);

}

else if (rootObject.value(keysChain[i]).isNull()) // is null

data += offset + keysChain[i] + ": Is Null\n";

else if (rootObject.value(keysChain[i]).isBool()) // is bool

{

if (rootObject.value(keysChain[i]).toBool())

tmp = offset + keysChain[i] + ": True\n";

else

tmp = offset + keysChain[i] + ": False\n";

data += tmp;

}

else if (rootObject.value(keysChain[i]).isArray()) // is array

{

data += offset + "\\+" + keysChain[i] + '\n';

QJsonValue val = rootObject.value(keysChain[i]);

QJsonArray arr = val.toArray();

int c = 0;

offset += '\t';

foreach (const QJsonValue &v, arr)

{

if (v.isBool())

{

if (v.toBool())

tmp = offset + QString::number(c) + ": True\n";

else

tmp = offset + QString::number(c) + ": False\n";

data += tmp;

}

else if (v.isDouble())

data += offset + QString::number(c) + ": " + v.toString() + '\n';

else if (v.isNull())

data += offset + QString::number(c) + ": Is Null\n";

else if (v.isObject())

{

data += offset + "\\+" + QString::number(c) + '\n';

recExtractJson(v.toObject(), offset + '\t', data);

}

else if (v.isString())

data += offset + QString::number(c) + ": " + v.toString() + '\n';

c++;

}

offset.chop(1);

}

else if (rootObject.value(keysChain[i]).isDouble()) // is double (aka number)

data += offset + keysChain[i] + ": " + rootObject.value(keysChain[i]).toVariant().toString() + '\n';

}

offset.chop(1);

}

void ParserClass::writeJsonDataInFile(const QJsonObject &object, const QString &path, const QString &fileName)

{

QString jData;

jData = QString();

recExtractJson(object, "", jData);

KTools::File::writeFile(jData.toUtf8(), path, fileName);

}

QJsonObject ParserClass::downloadJson(const QString url)

{

QByteArray result = cc->performing(url.toUtf8());

QJsonObject object = KTools::Converter::convert<QString, QJsonObject>(result);

return object;

}

void ParserClass::downloadAndWriteFile(const QString &url, const QString &path, const QString &fileName)

{

QByteArray result = cc->performing(url.toUtf8());

KTools::File::writeFile(result, path, fileName);

}

QJsonArray ParserClass::downloadJsonAsArray(const QString &url)

{

QByteArray result = cc->performing(url.toUtf8());

QJsonArray arr = KTools::Converter::convert<QString, QJsonArray>(result);

return arr;

}

void ParserClass::delay(const int &seconds)

{

std::this\_thread::sleep\_for(std::chrono::seconds(seconds));

}

void ParserClass::downloadAndWriteFileWithDefinedExtension(const QString &url, const QString &path, const QString &fileName)

{

QByteArray fileString = cc->performing(url.toUtf8());

QString extension = defineExtension(fileString);

KTools::File::writeFile(fileString, path, fileName + extension);

}

QString ParserClass::defineExtension(const QByteArray &file)

{

QStringList tmp = defExt->identifyFileFromString(file);

return tmp[0];

}

QVector<QJsonObject> ParserClass::extractJsonObjectFromText(const QString &text)

{

QString pattern;

QVector<QString> regexResult;

QVector<QJsonObject> objects;

pattern = "({\".+})+"; // =\*({(\")(.)+})+

KTools::ExForString::executeRegex(text, pattern, regexResult);

for (int i = 0; i < regexResult.size(); i++)

{

QString tmp = regexResult[i];

QJsonObject obj = KTools::Converter::convert<QString, QJsonObject>(tmp);

if (!obj.isEmpty())

objects.push\_back(obj);

}

return objects;

}

void ParserClass::endDownloadingFunction(const int parserMode, const QJsonObject &data, const QVector<QByteArray> &binaryContent)

{

QList<int> mode;

//mode.push\_back(static\_cast<int>(parserType));

mode.push\_back(parserMode);

emit downloadingFinished(mode, data, binaryContent);

}

**parserclass.h**

#ifndef PARSERCLASS\_H

#define PARSERCLASS\_H

#include <QJsonObject>

#include <QJsonDocument>

#include <QJsonArray>

#include <QJsonValue>

#include <QVariant>

#include <ctime>

#include <thread>

#include <chrono>

#include "KTools/curl.h"

#include "KTools/log.h"

#include "KTools/fileidentifier.h"

#include "KTools/converter.h"

#include "KTools/file.h"

#include "KTools/HtmlAstMaker/object.h"

#include "KTools/options.h"

#include "KTools/enums.h"

class ParserClass : public QObject

{

Q\_OBJECT

public:

ParserClass();

~ParserClass();

KTools::Curl \*cc;

protected:

// JSON

void recExtractJson(const QJsonObject &rootObject, QString offset, QString &data);

QJsonObject downloadJson(const QString url);

QJsonArray downloadJsonAsArray(const QString &url);

// File IO operations

void downloadAndWriteFile(const QString &url, const QString &path, const QString &fileName);

void downloadAndWriteFileWithDefinedExtension(const QString &url, const QString &path, const QString &fileName);

// Other/Mixed

void writeJsonDataInFile(const QJsonObject &object, const QString &path, const QString &fileName);

void delay(const int &seconds);

QVector<QJsonObject> extractJsonObjectFromText(const QString &text);

QString defineExtension(const QByteArray &file);

void endDownloadingFunction(const int parserMode, const QJsonObject &data = QJsonObject(), const QVector<QByteArray> &binaryContent = QVector<QByteArray>());

// variables

QString currUrl;

QString rootPath; // in this paths writes log file

QVector<QByteArray> chunk; // header for curl requests

KTools::FileIdentifier \*defExt;

QString logPath;

const QString logFile = "log.txt";

QString basePath;

QString parserName;

signals:

void downloadingFinished(const QList<int> mode, const QJsonObject data, const QVector<QByteArray> binaryContent);

void logMessage(const QString message);

};

#endif // PARSERCLASS\_H

**converter.cpp**

#include "converter.h"

KTools::Converter::Converter() {}

template<typename T>

T KTools::Converter::byteArrayToT(const QByteArray &bArr)

{

QDataStream str(bArr);

T result;

str >> result;

return result;

}

template<>

const char\* KTools::Converter::byteArrayToT<const char\*>(const QByteArray &bArr)

{

return bArr.data();

}

template<>

QString KTools::Converter::byteArrayToT<QString>(const QByteArray &bArr)

{

return bArr;

}

template<>

QByteArray KTools::Converter::byteArrayToT<QByteArray>(const QByteArray &bArr)

{

return bArr;

}

template<typename QByteArray, typename E>

E KTools::Converter::convert(const QByteArray &data)

{

return byteArrayToT<E>(data);

}

template<>

QJsonObject KTools::Converter::convert<QString, QJsonObject>(const QString &data)

{

QJsonObject obj;

QJsonDocument doc = QJsonDocument::fromJson(data.toUtf8());

if (!doc.isNull())

{

if (doc.isObject())

obj = doc.object();

else

return {}; // document is not an object

}

else

return {}; // invalid JSON

return obj;

}

template<>

QJsonArray KTools::Converter::convert<QString, QJsonArray>(const QString &data)

{

QJsonArray arr;

QJsonDocument doc = QJsonDocument::fromJson(data.toUtf8());

if (!doc.isNull())

{

if (doc.isArray())

arr = doc.array();

else

return {}; // document is not an object

}

else

return {}; // invalid JSON

return arr;

}

template<>

QByteArray KTools::Converter::convert<qint8, QByteArray>(const qint8 &data)

{

return toByteArray<qint8>(data);

}

template<>

QByteArray KTools::Converter::convert<quint8, QByteArray>(const quint8 &data)

{

return toByteArray<quint8>(data);

}

template<>

QByteArray KTools::Converter::convert<qint16, QByteArray>(const qint16 &data)

{

return toByteArray<qint16>(data);

}

template<>

QByteArray KTools::Converter::convert<quint16, QByteArray>(const quint16 &data)

{

return toByteArray<quint16>(data);

}

template<>

QByteArray KTools::Converter::convert<qint32, QByteArray>(const qint32 &data)

{

return toByteArray<qint32>(data);

}

template<>

QByteArray KTools::Converter::convert<quint32, QByteArray>(const quint32 &data)

{

return toByteArray<quint32>(data);

}

template<>

QByteArray KTools::Converter::convert<qint64, QByteArray>(const qint64 &data)

{

return toByteArray<qint64>(data);

}

template<>

QByteArray KTools::Converter::convert<quint64, QByteArray>(const quint64 &data)

{

return toByteArray<quint64>(data);

}

template<>

QByteArray KTools::Converter::convert<float, QByteArray>(const float &data)

{

return toByteArray<float>(data);

}

template<>

QByteArray KTools::Converter::convert<double, QByteArray>(const double &data)

{

return toByteArray<double>(data);

}

template<>

unsigned char KTools::Converter::convert<char, unsigned char>(const char &data)

{

return static\_cast<unsigned char>(data);

}

template<>

QByteArray KTools::Converter::convert<QString, QByteArray>(const QString &data)

{

return data.toUtf8();

}

template<>

unsigned char KTools::Converter::convert<QString, unsigned char>(const QString &data)

{

const char \*tmp = convert<QString, QByteArray>(data).constData();

return convert<char, unsigned char>(\*tmp);

}

template<>

QString KTools::Converter::convert<QJsonObject, QString>(const QJsonObject &data)

{

QJsonDocument jDoc(data);

return jDoc.toJson();

}

template<>

QByteArray KTools::Converter::convert<QJsonObject, QByteArray>(const QJsonObject &data)

{

return convert<QJsonObject, QString>(data).toUtf8();

}

template<typename T>

QByteArray KTools::Converter::toByteArray(const T &data)

{

QByteArray byteArr;

QDataStream byteStream(byteArr);

byteStream << data;

return byteArr;

}

QString KTools::Converter::numberToUtf8(const quint16 &code)

{

QString resultChar = QString::fromUtf16(&code);

resultChar.resize(1);

return resultChar;

}

void KTools::Converter::percentEncodingToString(QString &encodedString)

{

bool forWhat;

for (int i = 0; i < encodedString.size();)

{

QChar currChar = encodedString[i];

if (currChar == "%")

{

QString code = encodedString.mid(i + 1, 2);

QString repl = QString((char)code.toInt(&forWhat, 16));

encodedString.replace("%" + code, repl);

if (i > 2)

i = i - 2;

}

else

{

++i;

}

}

}

void KTools::Converter::percentEncodingToString(QByteArray &encodedString)

{

QString str = encodedString;

bool forWhat;

for (int i = 0; i < str.size();)

{

QChar currChar = str[i];

if (currChar == "%")

{

QString code = str.mid(i + 1, 2);

QString repl = QString((char)code.toInt(&forWhat, 16));

str.replace("%" + code, repl);

if (i > 2)

i = i - 2;

}

else

{

++i;

}

}

encodedString = str.toUtf8();

}

void KTools::Converter::toNtfsCompatibleString(QString &data)

{

QVector<QString> wrongChars = {"\"", "|", "/", ":", "\*", "?", ">", "<"};

QVector<QString> voids;

for (int i = 0; i < wrongChars.size(); i++)

voids.push\_back("");

ExForString::deleteChars(data, wrongChars);

}

void KTools::Converter::convertHtmlEntities(QString &inp)

{

QVector<QString> htmlEntities = {"&amp;", "&quot;", "&apos;", "&it;", "&gt;", "&nbsp;"};

QVector<QString> rightSumbols = {"&", "\"", "'", "<", ">", " "};

ExForString::replace(inp, htmlEntities, rightSumbols);

}

void KTools::Converter::convertHtmlEntities(QByteArray &inp)

{

QVector<QByteArray> htmlEntities = {"&amp;", "&quot;", "&apos;", "&it;", "&gt;", "&nbsp;"};

QVector<QByteArray> rightSumbols = {"&", "\"", "'", "<", ">", " "};

ExForString::replace(inp, htmlEntities, rightSumbols);

}

void KTools::Converter::convertHtmlHexCodes(QString &data)

{

QVector<QVector<QVector<QString>>> regexResult;

QVector<QString> patterns;

patterns.push\_back("&#(\\d+);");

ExForString::executeRegex(data, patterns, regexResult);

for (int i = 0; i < regexResult[0].size(); i++)

{

regexResult[0][i][1] = numberToUtf8(regexResult[0][i][1].toUShort());

data.replace(regexResult[0][i][0], regexResult[0][i][1]);

}

convertHtmlEntities(data);

}

QString KTools::Converter::nationalEncodingToUtf8(const QByteArray &inputEncoding, const QByteArray &data)

{

QTextCodec \*codec = QTextCodec::codecForName(inputEncoding);

return codec->toUnicode(data);

}

template qint8 KTools::Converter::byteArrayToT<qint8>(const QByteArray&);

template quint8 KTools::Converter::byteArrayToT<quint8>(const QByteArray&);

template qint16 KTools::Converter::byteArrayToT<qint16>(const QByteArray&);

template quint16 KTools::Converter::byteArrayToT<quint16>(const QByteArray&);

template qint32 KTools::Converter::byteArrayToT<qint32>(const QByteArray&);

template quint32 KTools::Converter::byteArrayToT<quint32>(const QByteArray&);

template qint64 KTools::Converter::byteArrayToT<qint64>(const QByteArray&);

template quint64 KTools::Converter::byteArrayToT<quint64>(const QByteArray&);

template QByteArray KTools::Converter::toByteArray<qint8>(const qint8&);

template QByteArray KTools::Converter::toByteArray<quint8>(const quint8&);

template QByteArray KTools::Converter::toByteArray<qint16>(const qint16&);

template QByteArray KTools::Converter::toByteArray<quint16>(const quint16&);

template QByteArray KTools::Converter::toByteArray<qint32>(const qint32&);

template QByteArray KTools::Converter::toByteArray<quint32>(const quint32&);

template QByteArray KTools::Converter::toByteArray<qint64>(const qint64&);

template QByteArray KTools::Converter::toByteArray<quint64>(const quint64&);

**converter.h**

#ifndef KTOOLS\_CONVERTER\_H

#define KTOOLS\_CONVERTER\_H

#include "exforstring.h"

#include "log.h"

#include <QByteArray>

#include <QString>

#include <QJsonObject>

#include <QJsonDocument>

#include <QJsonArray>

#include <QVector>

#include <QRegularExpression>

#include <QTextCodec>

#include <QDataStream>

namespace KTools

{

class Converter

{

public:

Converter();

template<typename T, typename E>

static E convert(const T &data);

template<typename T>

static T byteArrayToT(const QByteArray &bArr);

template<typename T>

static QByteArray toByteArray(const T &data);

static QString numberToUtf8(const quint16 &code);

static void percentEncodingToString(QString &encodedString);

static void percentEncodingToString(QByteArray &encodedString);

static void toNtfsCompatibleString(QString &data);

static void convertHtmlEntities(QString &inp);

static void convertHtmlEntities(QByteArray &inp);

static void convertHtmlHexCodes(QString &data);

static QString nationalEncodingToUtf8(const QByteArray &inputEncoding, const QByteArray &data);

};

}

#endif // KTOOLS\_CONVERTER\_H

**curl.cpp**

#include "curl.h"

KTools::Curl::Curl()

{

if (!KTools::File::fileExist(fullCacertPath))

KTools::File::copyFile(pathToCacertInQrc, cacertPath, cacertFileName);

//OptionsHandler::rootProgramPath + "/Cookie/";

gCurlHandle = curl\_easy\_init();

handlesList["main"] = gCurlHandle;

currCookieMode = KTools::Enums::Curl::CookieMode::Void;

currHeaderMode = KTools::Enums::Curl::HeaderMode::Custom;

currRequestType = KTools::Enums::Curl::RequestType::Get;

currHttpVersion = KTools::Enums::Curl::HttpVersion::Http2;

currRequestMode = KTools::Enums::Curl::RequestMode::New;

}

KTools::Curl::~Curl()

{

if (errFileSetted)

unsetErrFile();

QMap<QString, CURL\*>::iterator end = handlesList.end();

for (QMap<QString, CURL\*>::iterator i = handlesList.begin(); i != end; ++i)

{

curl\_easy\_cleanup(i.value());

}

if (cookies)

{

//curl\_slist\_free\_all(cookies);

}

}

QByteArray KTools::Curl::performing(const char\* url)

{

ForProggress fp;

QByteArray buffer = "";

CURL \*curlHandle = curl\_easy\_init(); // init curl session

curl\_easy\_setopt(curlHandle, CURLOPT\_URL, url); // specifu url to get

curl\_easy\_setopt(curlHandle, CURLOPT\_WRITEFUNCTION, writeMemoryCallback); // send all data to this function

curl\_easy\_setopt(curlHandle, CURLOPT\_WRITEDATA, &buffer);

curl\_easy\_setopt(curlHandle, CURLOPT\_HTTPHEADER, header);

curl\_easy\_setopt(curlHandle, CURLOPT\_CAINFO, fullCacertPath.toStdString().c\_str());

curl\_easy\_setopt(curlHandle, CURLOPT\_PROXY\_SSL\_VERIFYHOST, 1);

curl\_easy\_setopt(curlHandle, CURLOPT\_PROXY\_SSL\_VERIFYPEER, 1);

curl\_easy\_setopt(curlHandle, CURLOPT\_ACCEPT\_ENCODING, "deflate, gzip");

curl\_easy\_setopt(curlHandle, CURLOPT\_FOLLOWLOCATION, 1L);

curl\_easy\_setopt(curlHandle, CURLOPT\_MAXREDIRS, 5L);

curl\_easy\_setopt(curlHandle, CURLOPT\_PROGRESSFUNCTION, XFerInfoFunctionCallback);

curl\_easy\_setopt(curlHandle, CURLOPT\_PROGRESSDATA, &fp);

curl\_easy\_setopt(curlHandle, CURLOPT\_NOPROGRESS, 0);

fp.th = this;

fp.timer = QElapsedTimer();

fp.timer.start();

res = curl\_easy\_perform(curlHandle);

curl\_easy\_cleanup(curlHandle);

//curl\_global\_cleanup();

if (res != CURLE\_OK)

{

return "Error in CURL";

}

else

{

return buffer;

}

}

quint64 KTools::Curl::writeMemoryCallback(char \*data, quint64 size, quint64 nmemb, QByteArray \*writerData)

{

if (writerData == NULL)

return 0;

writerData->append(data, size\*nmemb);

return size \* nmemb;

}

void KTools::Curl::setHeader(QVector<QByteArray> chunk, const KTools::Enums::Curl::SetHeaderMode mode)

{

header = NULL;

for (qint64 i = 0; i < chunk.size(); i++)

{

header = curl\_slist\_append(header, chunk[i]);

}

}

int KTools::Curl::XFerInfoFunctionCallback(void \*p, double dlTotal, double dlNow, double ulTotal, double ulNow)

{ // all here in bytes

ForProggress \*fp = static\_cast<ForProggress\*>(p);

if (fp->lastDlNow != dlNow || fp->lastDlTotal != dlTotal || fp->lastUlTotal != ulTotal || fp->lastUlNow != ulNow)

{

QList<double> list;

list.append(dlNow);

list.append(dlTotal);

list.append(ulNow);

list.append(ulTotal);

//KToolsQmlInterface::curlSendProgress(list, fp->timer.elapsed(), fp->th->downloaderType);

fp->lastDlNow = dlNow;

fp->lastDlTotal = dlTotal;

fp->lastUlTotal = ulTotal;

fp->lastUlNow = ulNow;

}

return CURLE\_OK;

}

QByteArray KTools::Curl::request(const QString &url)

{

if (currRequestMode == KTools::Enums::Curl::RequestMode::Old)

{

return performing(url.toStdString().c\_str());

}

QByteArray buffer;

std::string c = currPostParam.toStdString();

curl\_easy\_setopt(gCurlHandle, CURLOPT\_URL, url.toStdString().c\_str()); // specify url to get

curl\_easy\_setopt(gCurlHandle, CURLOPT\_WRITEDATA, &buffer);

if (currRequestType == KTools::Enums::Curl::RequestType::Post)

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_POSTFIELDS, c.c\_str());

curl\_easy\_setopt(gCurlHandle, CURLOPT\_POSTFIELDSIZE, currPostParam.size());

}

if (currHeaderMode == KTools::Enums::Curl::HeaderMode::Custom)

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_HTTPHEADER, header);

}

savedHeaderData.clear();

res = curl\_easy\_perform(gCurlHandle);

curl\_easy\_getinfo(gCurlHandle, CURLINFO\_RESPONSE\_CODE, &responseCode);

curl\_easy\_getinfo(gCurlHandle, CURLINFO\_COOKIELIST, &cookies);

if (res != CURLE\_OK)

{

KTools::Log::writeError("Error in request. URL: " + url, "CurlClass");

return "Error in CURL";

}

else

{

return buffer;

}

}

void KTools::Curl::restartSession()

{

if (gCurlHandle)

curl\_easy\_cleanup(gCurlHandle);

gCurlHandle = curl\_easy\_init();

}

quint64 KTools::Curl::headerCallback(char \*buffer, quint64 size, quint64 nitems, QByteArray \*userdata)

{

userdata->append(buffer, nitems \* size);

return nitems \* size;

}

void KTools::Curl::setOptions()

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_WRITEFUNCTION, writeMemoryCallback); // send all data to this function

curl\_easy\_setopt(gCurlHandle, CURLOPT\_CAINFO, fullCacertPath.toStdString().c\_str());

curl\_easy\_setopt(gCurlHandle, CURLOPT\_PROXY\_SSL\_VERIFYHOST, 1);

curl\_easy\_setopt(gCurlHandle, CURLOPT\_PROXY\_SSL\_VERIFYPEER, 1);

curl\_easy\_setopt(gCurlHandle, CURLOPT\_ACCEPT\_ENCODING, "gzip, deflate, br");

curl\_easy\_setopt(gCurlHandle, CURLOPT\_FOLLOWLOCATION, 1L);

curl\_easy\_setopt(gCurlHandle, CURLOPT\_MAXREDIRS, 5L);

curl\_easy\_setopt(gCurlHandle, CURLOPT\_USERAGENT, "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0");

if (currHttpVersion == KTools::Enums::Curl::HttpVersion::Http2)

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_HTTP\_VERSION, CURL\_HTTP\_VERSION\_2TLS);

}

/\*

\* This if else construction choose which request type do

\*/

setRequestType(currRequestType);

/\*

\* This if else construction choose what to do with cookies

\*/

if (currCookieMode == KTools::Enums::Curl::CookieMode::GetAndKeep) // Get cookie in first request and use it in next requests

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_COOKIEJAR, (cookiePath + cookieFileName).toStdString().c\_str());

currCookieMode = KTools::Enums::Curl::CookieMode::Keep;

}

else if (currCookieMode == KTools::Enums::Curl::CookieMode::Keep) // Use cookie from file

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_COOKIEFILE, (cookiePath + cookieFileName).toStdString().c\_str());

}

else if (currCookieMode == KTools::Enums::Curl::CookieMode::GetAllTimes) // Get new cookie in each request

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_COOKIEJAR, (cookiePath + cookieFileName).toStdString().c\_str());

curl\_easy\_setopt(gCurlHandle, CURLOPT\_COOKIEFILE, (cookiePath + cookieFileName).toStdString().c\_str());

}

else if (currCookieMode == KTools::Enums::Curl::CookieMode::Void) // Only starts cookie engine

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_COOKIEFILE, "");

}

/\*

\* This if construction choose what to do with header

\*/

if (currHeaderMode == KTools::Enums::Curl::HeaderMode::Custom) // Sets header

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_HTTPHEADER, header);

}

/\*

\* This if construction choose send progress data or not (on default emitProgress = false)

\*/

ForProggress fp;

if (emitProgress)

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_PROGRESSFUNCTION, XFerInfoFunctionCallback);

curl\_easy\_setopt(gCurlHandle, CURLOPT\_PROGRESSDATA, &fp);

curl\_easy\_setopt(gCurlHandle, CURLOPT\_NOPROGRESS, 0);

fp.th = this;

fp.timer = QElapsedTimer();

fp.timer.start();

}

/\*

\* This if construction choose set automatically refer field or not (on default autoRefer = true)

\*/

if (autoRefer)

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_AUTOREFERER, 1L);

}

/\*

\* This if construction choose set automatically refer field or not (on default autoRefer = true)

\*/

//savedHeaderData.clear();

if (saveHeader)

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_HEADERDATA, &savedHeaderData);

curl\_easy\_setopt(gCurlHandle, CURLOPT\_HEADERFUNCTION, headerCallback);

}

if (verboseOutput)

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_VERBOSE, 1L);

curl\_easy\_setopt(gCurlHandle, CURLOPT\_STDERR, stdErr);

}

}

void KTools::Curl::setRequestType(KTools::Enums::Curl::RequestType requType)

{

currRequestType = requType;

if (requType == KTools::Enums::Curl::RequestType::Get)

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_CUSTOMREQUEST, "GET");

//curl\_easy\_setopt(gCurlHandle, CURLOPT\_HTTPGET, 1L);

}

else if (requType == KTools::Enums::Curl::RequestType::Post)

{

curl\_easy\_setopt(gCurlHandle, CURLOPT\_CUSTOMREQUEST, "POST");

//curl\_easy\_setopt(gCurlHandle, CURLOPT\_POST, 1L);

}

}

void KTools::Curl::generateHeader()

{

header = NULL;

QMap<QString, QString>::iterator end = headerData.end();

for (QMap<QString, QString>::iterator i = headerData.begin(); i != end; ++i)

{

header = curl\_slist\_append(header, (i.key() + ": " + i.value()).toStdString().c\_str());

}

}

void KTools::Curl::setHeader(const QMap<QString, QString> &chunk)

{

/\*QMap<QString, QString>::const\_iterator end = chunk.end();

for (QMap<QString, QString>::const\_iterator i = chunk.begin(); i != end; ++i)

{

headerData[i.key()] = i.value();

}\*/

headerData = chunk;

generateHeader();

}

void KTools::Curl::setHeaderLine(const std::pair<QString, QString> &headerLine)

{

headerData[headerLine.first] = headerLine.second;

generateHeader();

}

void KTools::Curl::delHeaderLine(const QString &key)

{

headerData.remove(key);

generateHeader();

}

void KTools::Curl::setErrFile(const QString &path, const QString &fileName)

{

stdErr = fopen((path + "/" + fileName).toStdString().c\_str(), "w");

errFileSetted = true;

verboseOutput = true;

}

void KTools::Curl::unsetErrFile()

{

if (errFileSetted)

{

fclose(stdErr);

errFileSetted = false;

verboseOutput = false;

}

}

void KTools::Curl::addHandle(const QString &handleName)

{

CURL\* newHandle = curl\_easy\_init();

handlesList[handleName] = newHandle;

}

void KTools::Curl::swichHandle(const QString &handleName)

{

gCurlHandle = handlesList[handleName];

}

void KTools::Curl::setCookie()

{

/\*std::string str = "";

struct curl\_slist \*next = cookies;

while (next)

{

str += next->data;

str += "\n";

next = next->next;

}

curl\_easy\_setopt(gCurlHandle, CURLOPT\_COOKIELIST, str.c\_str());\*/

curl\_easy\_setopt(gCurlHandle, CURLOPT\_COOKIEFILE, (cookiePath + cookieFileName).toStdString().c\_str());

}

**curl.h**

#ifndef KTOOLS\_CURL\_H

#define KTOOLS\_CURL\_H

#include <curl/curl.h>

#include <string>

#include <vector>

#include <QDir>

#include <QFile>

#include <QObject>

#include <QElapsedTimer>

#include "file.h"

#include "enums.h"

#include "log.h"

#include "ktoolsqmlinterface.h"

namespace KTools

{

class Curl

{

public:

Curl();

~Curl();

QByteArray performing(const char\* url);

void setHeader(QVector<QByteArray> chunk, const KTools::Enums::Curl::SetHeaderMode mode = KTools::Enums::Curl::SetHeaderMode::Add);

QByteArray request(const QString &url);

void restartSession();

void setOptions();

void setRequestType(KTools::Enums::Curl::RequestType requType);

void setHeader(const QMap<QString, QString> &chunk);

void setHeaderLine(const std::pair<QString, QString> &headerLine);

void delHeaderLine(const QString &key);

void setErrFile(const QString &path, const QString &fileName);

void unsetErrFile();

void setCookie();

void addHandle(const QString &handleName);

void swichHandle (const QString &handleName);

KTools::Enums::Curl::CookieMode currCookieMode;

KTools::Enums::Curl::HeaderMode currHeaderMode;

KTools::Enums::Curl::RequestType currRequestType;

KTools::Enums::Curl::HttpVersion currHttpVersion;

KTools::Enums::Curl::RequestMode currRequestMode;

QString currPostParam;

static QString cookiePath;

QString cookieFileName;

bool emitProgress = false;

long responseCode; // response code for last request

bool autoRefer = true;

bool saveHeader = false;

QByteArray savedHeaderData;

FILE \*stdErr;

bool errFileSetted = false;

bool verboseOutput = false;

private:

CURLcode res;

static quint64 writeMemoryCallback(char \*data, quint64 size, quint64 nmemb, QByteArray \*writerData);

static int XFerInfoFunctionCallback(void \*p, double dlTotal, double dlNow, double ulTotal, double ulNow);

static quint64 headerCallback(char \*buffer, quint64 size, quint64 nitems, QByteArray \*userdata);

void generateHeader();

struct curl\_slist \*header;

struct curl\_slist \*cookies;

static QString cacertPath;

static QString cacertFileName;

static QString fullCacertPath;

static QString pathToCacertInQrc;

CURL \*gCurlHandle;

QMap<QString, QString> headerData;

QMap<QString, CURL\*> handlesList;

struct ForProggress

{

Curl \*th;

QElapsedTimer timer;

double lastDlTotal;

double lastUlTotal;

double lastDlNow;

double lastUlNow;

};

};

}

#endif // KTOOLS\_CURL\_H

**enums.h**

#ifndef KTOOLS\_ENUMS\_H

#define KTOOLS\_ENUMS\_H

#include <QMap>

#include <QObject>

#include <QQmlEngine>

namespace KTools::Enums

{

enum class LogType {

Custom,

Info,

Error,

Debug

};

namespace Curl {

enum class RequestType {

Get,

Post

};

enum class CookieMode {

None,

GetAllTimes,

GetAndKeep,

Keep,

Void

};

enum class HeaderMode {

None,

Custom

};

enum class SetHeaderMode {

Add,

New

};

enum class HttpVersion {

Http2,

Standart

};

enum class RequestMode {

Old,

New

};

}

};

#endif // KTOOLS\_ENUMS\_H

**exforstring.cpp**

#include "exforstring.h"

KTools::ExForString::ExForString() {}

void KTools::ExForString::replace(QString &inp, const QVector<QString> &whatReplace, const QVector<QString> &onWhatReplace)

{

std::string input = inp.toStdString();

for (int i = 0; i < whatReplace.size(); i++)

{

size\_t startPos = 0;

while ((startPos = input.find(whatReplace[i].toStdString(), startPos)) != std::string::npos)

{

input.replace(startPos, whatReplace[i].length(), onWhatReplace[i].toStdString());

startPos += onWhatReplace[i].length();

}

}

inp = QString::fromStdString(input);

}

void KTools::ExForString::replace(QByteArray &inp, const QVector<QByteArray> &whatReplace, const QVector<QByteArray> &onWhatReplace)

{

std::string input = inp.toStdString();

for (int i = 0; i < whatReplace.size(); i++)

{

size\_t startPos = 0;

while ((startPos = input.find(whatReplace[i].toStdString(), startPos)) != std::string::npos)

{

input.replace(startPos, whatReplace[i].length(), onWhatReplace[i].toStdString());

startPos += onWhatReplace[i].length();

}

}

inp = QByteArray::fromStdString(input);

}

void KTools::ExForString::deleteChars(QString &inp, const QVector<QString> &whatDelete)

{

QVector<QString> voids;

for (int i = 0; i < whatDelete.size(); i++)

voids.push\_back("");

replace(inp, whatDelete, voids);

}

void KTools::ExForString::executeRegex(const QString &data, const QVector<QString> &pattens, QVector<QVector<QVector<QString>>> &regexResult)

{

regexResult.resize(pattens.size());

for (int i = 0; i < pattens.size(); i++)

{

QRegularExpression re(pattens[i]);

QRegularExpressionMatchIterator reI = re.globalMatch(data);

for (int j = 0; reI.hasNext(); j++)

{

regexResult[i].resize(regexResult[i].size() + 1);

QRegularExpressionMatch match = reI.next();

for (int jj = 0; jj <= match.lastCapturedIndex(); jj++)

regexResult[i][j].push\_back(match.captured(jj));

}

}

}

void KTools::ExForString::executeRegex(const QString &data, const QString &pattern, QVector<QString> &result)

{

result.clear();

QRegularExpression re(pattern);

QRegularExpressionMatchIterator i = re.globalMatch(data);

while (i.hasNext())

{

QRegularExpressionMatch match = i.next();

result.append(match.captured(1));

}

}

void KTools::ExForString::executeRegex(const QString &data, const QString &pattern, QVector<double> &result)

{

result.clear();

QRegularExpression re(pattern);

QRegularExpressionMatchIterator i = re.globalMatch(data);

while (i.hasNext())

{

QRegularExpressionMatch match = i.next();

result.append(match.captured(1).toDouble());

}

}

void KTools::ExForString::executeRegex(const QString &data, const QString &pattern, QVector<qint64> &result)

{

result.clear();

QRegularExpression re(pattern);

QRegularExpressionMatchIterator i = re.globalMatch(data);

while (i.hasNext())

{

QRegularExpressionMatch match = i.next();

result.append(match.captured(1).toLong());

}

}

QVector<qint64> KTools::ExForString::getIntegerNumberFromString(const QString &str)

{

QVector<qint64> result;

executeRegex(str, "(\\d+.\\d+)", result);

return result;

}

QVector<double> KTools::ExForString::getDoubleNumberFromString(const QString &str)

{

QVector<double> result;

executeRegex(str, "(\\d+[.,]\\d+|\\d+)", result);

return result;

}

**exforstring.h**

#ifndef KTOOLS\_EXFORSTRING\_H

#define KTOOLS\_EXFORSTRING\_H

#include <QString>

#include <QVector>

#include <QRegularExpression>

namespace KTools

{

class ExForString

{

public:

ExForString();

static void replace(QString &inp, const QVector<QString> &whatReplace, const QVector<QString> &onWhatReplace);

static void replace(QByteArray &inp, const QVector<QByteArray> &whatReplace, const QVector<QByteArray> &onWhatReplace);

static void deleteChars(QString &inp, const QVector<QString> &whatDelete);

static void executeRegex(const QString &data, const QVector<QString> &pattens, QVector<QVector<QVector<QString>>> &regexResult);

static void executeRegex(const QString &data, const QString &pattern, QVector<QString> &result);

static void executeRegex(const QString &data, const QString &pattern, QVector<double> &result);

static void executeRegex(const QString &data, const QString &pattern, QVector<qint64> &result);

static QVector<qint64> getIntegerNumberFromString(const QString &str);

static QVector<double> getDoubleNumberFromString(const QString &str);

};

}

#endif // KTOOLS\_EXFORSTRING\_H

**file.cpp**

#include "file.h"

KTools::File::File() {}

bool KTools::File::open(const QString &path, const QIODevice::OpenMode &flags)

{

file = new QFile(path);

if (file->open(flags))

{

return true;

}

else

return false;

}

bool KTools::File::seek(const qint64 &pos)

{

return file->seek(pos);

}

qint64 KTools::File::size()

{

return file->size();

}

qint64 KTools::File::pos()

{

return file->pos();

}

template<typename T>

void KTools::File::write(const T &data)

{

file->write(KTools::Converter::toByteArray<T>(data));

}

template<>

void KTools::File::write(const QByteArray &data)

{

file->write(data);

}

template<typename T>

T KTools::File::read(const qint64 &lenght)

{

return KTools::Converter::byteArrayToT<T>(file->read(lenght));

}

template<typename T>

T KTools::File::read()

{

return KTools::Converter::byteArrayToT<T>(file->read(sizeof(T)));

}

bool KTools::File::writeFile(const QByteArray &data, const QString &directory, const QString &fileName, const QIODevice::OpenMode &flags)

{

QString correctPath = directory;

correctPath.replace("\\", "/").replace("//", "/");

if (correctPath[correctPath.size() - 1] == "/")

correctPath.chop(1);

makePath(correctPath);

QFile file(correctPath + '/' + fileName);

if (file.open(flags))

{

file.write(data);

}

else

{

//Logging::writeError("Can't open file. directory: " + directory + "; fileName: " + fileName, "NativeFs::writeFile()");

QFile::FileError ferr = file.error();

file.close();

return false;

}

file.close();

return true;

}

void KTools::File::makePath(const QString &path)

{

QDir dir = QDir(path);

bool qexi = dir.exists();

if (!qexi)

qexi = dir.mkpath("."); // if derictory does not exist make it

}

bool KTools::File::fileExist(const QString &path)

{

QFileInfo checkFile(path);

if (checkFile.exists() && checkFile.isFile())

return true;

else

return false;

}

bool KTools::File::dirExist(const QString &path)

{

QFileInfo checkFile(path);

if (checkFile.exists() && checkFile.isDir())

return true;

else

return false;

}

bool KTools::File::toEnd()

{

return seek(size());

}

bool KTools::File::atEnd()

{

return file->atEnd();

}

bool KTools::File::resize(const qint64 &localSize)

{

return file->resize(localSize);

}

bool KTools::File::copyFile(const QString &oldPathToFile, const QString &newPath, const QString &newFileName)

{

if (!dirExist(newPath))

makePath(newPath);

return QFile::copy(oldPathToFile, newPath + '/' + newFileName);

}

template<typename T>

T KTools::File::readFile(const QString &directory, const QString &fileName, const QIODevice::OpenMode &flags)

{

if (!fileExist(directory + '\\' + fileName))

return T();

QFile rFile(directory + '\\' + fileName);

if (!rFile.open(flags))

return T();

if (typeid (T) != typeid (QByteArray))

return KTools::Converter::byteArrayToT<T>(rFile.readAll());

return rFile.readAll();

}

template void KTools::File::write<qint16>(const qint16&);

template void KTools::File::write<qint64>(const qint64&);

template qint8 KTools::File::read<qint8>(const qint64&);

template qint16 KTools::File::read<qint16>(const qint64&);

template qint64 KTools::File::read<qint64>(const qint64&);

template QByteArray KTools::File::read<QByteArray>(const qint64&);

template qint64 KTools::File::read<qint64>();

template quint64 KTools::File::read<quint64>();

template qint16 KTools::File::read<qint16>();

template QString KTools::File::readFile<QString>(const QString&, const QString&, const QIODevice::OpenMode&);

template QByteArray KTools::File::readFile<QByteArray>(const QString&, const QString&, const QIODevice::OpenMode&);

**file.h**

#ifndef KTOOLS\_FILE\_H

#define KTOOLS\_FILE\_H

#include "converter.h"

#include <QFile>

#include <QIODevice>

#include <QDataStream>

#include <QDir>

#include "log.h"

namespace KTools

{

class File

{

public:

File();

bool open(const QString &path, const QIODevice::OpenMode &flags);

bool seek(const qint64 &pos);

bool toEnd();

qint64 size();

qint64 pos();

template<typename T>

void write(const T &data);

template<typename T>

T read(const qint64 &lenght);

template<typename T>

T read();

static bool writeFile(const QByteArray &data, const QString &directory, const QString &fileName, const QIODevice::OpenMode &flags = QIODevice::WriteOnly);

static void makePath(const QString &path);

static bool fileExist(const QString &path);

static bool dirExist(const QString &path);

static bool copyFile(const QString &oldPathToFile, const QString &newPath, const QString &newFileName);

template<typename T>

static T readFile(const QString &directory, const QString &fileName, const QIODevice::OpenMode &flags = QIODevice::ReadOnly);

bool atEnd();

bool resize(const qint64 &localSize);

private:

QFile \*file;

};

}

#endif // KTOOLS\_FILE\_H

**fileidentifier.cpp**

#include "fileidentifier.h"

KTools::FileIdentifier::FileIdentifier() {}

QStringList KTools::FileIdentifier::identifyFileFromFileSystem(const QString &path)

{

KTools::File file = KTools::File();

if (!file.open(path, QIODevice::ReadOnly))

{

return {"", "Permission denied", ""};

}

QByteArray fileSignature = file.read<QByteArray>(20);

QStringList fileType;

if (fileSignature.size() < 20)

{

fileType.push\_back("");

fileType.push\_back("File too small");

fileType.push\_back("");

}

else

{

fileType = fileTypeSelector(fileSignature);

}

return fileType;

}

QStringList KTools::FileIdentifier::identifyFileFromString(const QByteArray &file)

{

QByteArray fileSignature = cutQString(0, 20, file);

QStringList fileType = fileTypeSelector(fileSignature);

return fileType;

}

QStringList KTools::FileIdentifier::fileTypeSelector(const QByteArray &bytes)

{

QStringList result;

QStringList stringsForEquations;

stringsForEquations.push\_back("");

for (int i = 1; i < bytes.length(); i++)

stringsForEquations.push\_back(cutQString(0, i, bytes));

if (stringsForEquations[4] == "\xFF\xD8\xFF\xDB" || stringsForEquations[12] == "\xFF\xD8\xFF\xE0\x00\x10\x4A\x46\x49\x46\x00\x01" || stringsForEquations[4] == "\xFF\xD8\xFF\xEE")

{

result.push\_back(".jpg");

result.push\_back("JPEG image file");

result.push\_back("qrc:/resources/FSExplorer/img/fileTypeIcons/jpeg.png");

}

else if (stringsForEquations[8] == "\x89\x50\x4E\x47\x0D\x0A\x1A\x0A")

{

result.push\_back(".png");

result.push\_back("PNG image file");

result.push\_back("qrc:/resources/FSExplorer/img/fileTypeIcons/png.png");

}

else if (stringsForEquations[6] == "\x47\x49\x46\x38\x37\x61" || stringsForEquations[6] == "\x47\x49\x46\x38\x39\x61")

{

result.push\_back(".gif");

result.push\_back("GIF image file");

result.push\_back("qrc:/resources/FSExplorer/img/fileTypeIcons/gif.png");

}

else if (stringsForEquations[4] == "\x49\x49\x2A\x00" || stringsForEquations[4] == "\x4D\x4D\x00\x2A")

{

result.push\_back(".tiff");

result.push\_back("TIFF image file");

result.push\_back("qrc:/resources/FSExplorer/img/fileTypeIcons/tiff.png");

}

else

{

result.push\_back("");

result.push\_back("Unknown file format");

result.push\_back("");

result.push\_back("");

}

return result;

}

QByteArray KTools::FileIdentifier::cutQString(const int &from, const int &lenghtCuttedString, const QByteArray &string)

{

return string.mid(from, lenghtCuttedString);

}

**fileidentifier.h**

#ifndef KTOOLS\_FILEIDENTIFIER\_H

#define KTOOLS\_FILEIDENTIFIER\_H

#include "../KTools/file.h"

namespace KTools

{

class FileIdentifier

{

public:

FileIdentifier();

static QStringList identifyFileFromFileSystem(const QString &path);

static QStringList identifyFileFromString(const QByteArray &file);

private:

static QStringList fileTypeSelector(const QByteArray &bytes);

static QByteArray cutQString(const int &from, const int &lenghtCuttedString, const QByteArray &string);

};

}

#endif // KTOOLS\_FILEIDENTIFIER\_H

**ktoolsqmlinterface.cpp**

#include "ktoolsqmlinterface.h"

KToolsQmlInterface::KToolsQmlInterface() {}

void KToolsQmlInterface::optionsHandlerSetRootProgramPath(const QString path)

{

KTools::Options::setRootProgramPath(path);

}

void KToolsQmlInterface::optionsHandlerEmitRootProgramPath()

{

emit optionsHandlerSendRootProgramPath(KTools::Options::rootProgramPath);

}

void KToolsQmlInterface::optionsHandlerSlotGetParam(const QString pathToParam)

{

emit optionsHandlerSignalParam(pathToParam, KTools::Options::getParam(pathToParam));

}

void KToolsQmlInterface::optionsHandlerSlotSetParam(const QString pathToParam, const QString param)

{

KTools::Options::setParam(pathToParam, param);

}

**ktoolsqmlinterface.h**

#ifndef KTOOLSQMLINTERFACE\_H

#define KTOOLSQMLINTERFACE\_H

#include <QObject>

#include "options.h"

class KToolsQmlInterface : public QObject

{

Q\_OBJECT

public:

KToolsQmlInterface();

public slots:

// Start OptionsHandler

void optionsHandlerSetRootProgramPath(const QString path);

void optionsHandlerEmitRootProgramPath();

void optionsHandlerSlotSetParam(const QString pathToParam, const QString param);

void optionsHandlerSlotGetParam(const QString pathToParam);

// End OptionsHandler

public:

static KToolsQmlInterface \*obj;

signals:

// Start OptionsHandler

void optionsHandlerSendRootProgramPath(const QString path);

void optionsHandlerSignalParam(const QString pathToParam, const QVariant param);

// End OptionsHandler

};

#endif // KTOOLSQMLINTERFACE\_H

**log.cpp**

#include "log.h"

KTools::Log::Log() {}

void KTools::Log::writeCustomLog(const QString &message, const QString &from, const KTools::Enums::LogType &type)

{

QString dateTime = QDateTime::currentDateTime().toString("yyyy.MM.dd hh:mm:ss:zzz");

QString dt = '[' + from + ']' + dateTime + ' ' + message + '\n';

File::writeFile(dt.toUtf8(), Options::logRootPath + logTypePath[type], logFileName, QIODevice::Append | QIODevice::WriteOnly);

}

void KTools::Log::writeCustomLog(const QString &message, const QString &from, const KTools::Enums::LogType &type, const QString &path, const QString &fileName)

{

QString dateTime = QDateTime::currentDateTime().toString("yyyy.MM.dd hh:mm:ss:zzz");

QString dt = '[' + logTypeNames[type] + "][" + from + ']' + dateTime + ' ' + message + '\n';

File::writeFile(dt.toUtf8(), path, fileName, QIODevice::Append | QIODevice::WriteOnly);

}

void KTools::Log::writeInfo(const QString &message, const QString &from)

{

writeCustomLog(message, from, KTools::Enums::LogType::Info);

}

void KTools::Log::writeError(const QString &message, const QString &from)

{

writeCustomLog(message, from, KTools::Enums::LogType::Error);

}

void KTools::Log::writeDebug(const QString &message, const QString &from)

{

writeCustomLog(message, from, KTools::Enums::LogType::Debug);

}

**log.h**

#ifndef KTOOLS\_LOG\_H

#define KTOOLS\_LOG\_H

#include <QDateTime>

#include "file.h"

#include "options.h"

#include "enums.h"

namespace KTools

{

class Log

{

public:

Log();

private:

static QString logFileName;

static QMap<KTools::Enums::LogType, QString> logTypePath;

static QMap<KTools::Enums::LogType, QString> logTypeNames;

public:

static void writeCustomLog(const QString &message, const QString &from, const KTools::Enums::LogType &type = KTools::Enums::LogType::Custom);

static void writeCustomLog(const QString &message, const QString &from, const KTools::Enums::LogType &type, const QString &path, const QString &fileName);

static void writeInfo(const QString &message, const QString &from);

static void writeError(const QString &message, const QString &from);

static void writeDebug(const QString &message, const QString &from);

};

}

#endif // KTOOLS\_LOG\_H

**options.cpp**

#include "options.h"

KTools::Options::Options()

{

bool wtf = KTools::File::fileExist(configPath + configFile);

if (wtf)

{

rootProgramPath = KTools::File::readFile<QString>(configPath, configFile, QIODevice::ReadOnly | QIODevice::Text);

}

else

{

rootProgramPath = configPath;

KTools::File::writeFile(configPath.toUtf8(), configPath, configFile, QIODevice::WriteOnly | QIODevice::Text);

}

logRootPath = Options::rootProgramPath + "/log/";

KTools::Curl::cookiePath = Options::rootProgramPath + "/Cookie/";

if (!KTools::File::fileExist(rootProgramPath + "/Settings/configs.json"))

{

KTools::File::copyFile(":/resources/sampleFiles/configs.json", rootProgramPath + "/Settings", "configs.json");

}

QString fileContent = KTools::File::readFile<QString>(rootProgramPath + "/Settings", "configs.json");

configsObj = KTools::Converter::convert<QString, QJsonObject>(fileContent);

}

bool KTools::Options::save()

{

return KTools::File::writeFile(KTools::Converter::convert<QJsonObject, QByteArray>(configsObj), rootProgramPath, "configs.json");

}

void KTools::Options::setRootProgramPath(const QString &path)

{

rootProgramPath = path;

KTools::File::writeFile(rootProgramPath.toUtf8(), configPath, configFile, QIODevice::WriteOnly | QIODevice::Text);

save();

}

QString KTools::Options::getRootProgramPath()

{

return rootProgramPath;

}

QJsonValue KTools::Options::privateSetParam(QList<QString> pathToParam, QJsonValue currLevel, const QVariant param)

{

QString name = pathToParam[0];

pathToParam.pop\_front();

QJsonValue::Type jsValType = currLevel.type();

if (jsValType == QJsonValue::Type::Undefined)

{

KTools::Log::writeError("Attempt access to non existent parameter. name: " + name, "OptionsHandler::setParam()");

return QJsonValue();

}

if (pathToParam.size() > 1)

{

if (jsValType == QJsonValue::Type::Array)

{

QJsonArray jsVal = currLevel.toArray();

QJsonValueRef tmp = jsVal[name.toInt()];

jsVal[name.toInt()] = privateSetParam(pathToParam, tmp, param);

return jsVal;

}

else if (jsValType == QJsonValue::Type::Object)

{

QJsonObject jsVal = currLevel.toObject();

QJsonValueRef tmp = jsVal[name];

jsVal[name] = privateSetParam(pathToParam, tmp, param);

return jsVal;

}

else

{

KTools::Log::writeError("Wrong type. jsValType: " + static\_cast<QString>(jsValType) + ", name: " + name, "OptionsHandler::setParam");

}

}

else

{

if (jsValType != QJsonValue::Type::Array)

{

QJsonObject jsVal = currLevel.toObject();

jsVal[name] = QJsonValue::fromVariant(param);

return jsVal;

}

else if (jsValType == QJsonValue::Type::Array)

{

QJsonArray jsVal = currLevel.toArray();

jsVal[name.toInt()] = QJsonValue::fromVariant(param);

return jsVal;

}

}

KTools::Log::writeError("Something went wrong. name: " + name, "OptionsHandler::privateSetParam()");

return QJsonValue();

}

QJsonValue KTools::Options::privateGetParam(QList<QString> pathToParam, const QJsonValue &previousLevel)

{

QString name = pathToParam[0];

pathToParam.pop\_front();

QJsonValue::Type jsValType = previousLevel.type();

if (jsValType == QJsonValue::Type::Undefined)

{

KTools::Log::writeError("Attempt access to non existent parameter. name: " + name, "OptionsHandler::getParam()");

return QJsonValue();

}

if (pathToParam.size() > 1)

{

if (jsValType == QJsonValue::Type::Array)

{

QJsonArray jsVal = previousLevel.toArray();

QJsonValueRef tmp = jsVal[name.toInt()];

return privateGetParam(pathToParam, tmp);

}

else if (jsValType == QJsonValue::Type::Object)

{

QJsonObject jsVal = previousLevel.toObject();

QJsonValueRef tmp = jsVal[name];

return privateGetParam(pathToParam, tmp);

}

else

{

KTools::Log::writeError("Wrong type. jsValType: " + static\_cast<QString>(jsValType) + ", name: " + name, "OptionsHandler::getParam");

return QJsonValue();

}

}

else

{

if (jsValType != QJsonValue::Type::Array)

{

QJsonObject jsVal = previousLevel.toObject();

return jsVal[name];

}

else if (jsValType == QJsonValue::Type::Array)

{

QJsonArray jsVal = previousLevel.toArray();

return jsVal[name.toInt()];

}

}

KTools::Log::writeError("Something went wrong. name: " + name, "OptionsHandler::privateGetParam()");

return QJsonValue();

}

void KTools::Options::setParam(const QString &pathToParam, const QString &param)

{

QList<QString> list = pathToParam.split("/", QString::SplitBehavior::SkipEmptyParts);

QString name = list[0];

list.pop\_front();

QJsonValueRef jsVal = configsObj[name];

configsObj[name] = privateSetParam(list, jsVal, param);

save();

}

QVariant KTools::Options::getParam(const QString &pathToParam)

{

QList<QString> list = pathToParam.split("/", QString::SplitBehavior::SkipEmptyParts);

QString name = list[0];

list.pop\_front();

QJsonValue jsVal = configsObj[name];

return privateGetParam(list, jsVal);

}

**options.h**

#ifndef KTOOLS\_OPTIONS\_H

#define KTOOLS\_OPTIONS\_H

#include <QObject>

#include <QJsonObject>

#include <QStandardPaths>

#include <QFile>

#include <QJsonDocument>

#include <QFileInfo>

#include <QVariant>

#include "file.h"

#include "converter.h"

#include "enums.h"

#include "log.h"

#include "curl.h"

namespace KTools

{

class Options

{

public:

Options();

QString getRootProgramPath();

static QJsonObject configsObj;

static QString configPath;

static QString configFile;

static QString rootProgramPath;

static QString logRootPath;

static void setRootProgramPath(const QString &path);

static void setParam(const QString &pathToParam, const QString &param); // wrapper for privateSetParam() for C++

static QVariant getParam(const QString &pathToParam); // wrapper for privateGetParam() for C++

private:

static bool save(); // saving current settings in native file system

static QJsonValue privateSetParam(QList<QString> pathToParam, QJsonValue currLevel, const QVariant param); // private method because for it need wrapper

static QJsonValue privateGetParam(QList<QString> pathToParam, const QJsonValue &previousLevel); // private method because for it need wrapper

};

}

#endif // KTOOLS\_OPTIONS\_H

**StaticVarsDeclarations.cpp**

#include "options.h"

#include "log.h"

#include "curl.h"

#include "enums.h"

#include "ktoolsqmlinterface.h"

#include "fileidentifier.h"

// Start OptionsHandler

QString KTools::Options::configFile = "config.txt";

QString KTools::Options::configPath = QStandardPaths::writableLocation(QStandardPaths::ConfigLocation) + "/InformationSystem/";

QString KTools::Options::rootProgramPath = "";

QString KTools::Options::logRootPath = "";

QJsonObject KTools::Options::configsObj = QJsonObject();

// End OptionsHandler

// Start Log

QString KTools::Log::logFileName = QDateTime::currentDateTime().toString("yyyy.MM.dd") + ".log";

QMap<KTools::Enums::LogType, QString> KTools::Log::logTypePath = {

{KTools::Enums::LogType::Info, "/info/"},

{KTools::Enums::LogType::Debug, "/debug/"},

{KTools::Enums::LogType::Error, "/error/"},

{KTools::Enums::LogType::Custom, "/custom/"}

};

QMap<KTools::Enums::LogType, QString> KTools::Log::logTypeNames = {

{KTools::Enums::LogType::Info, "Info"},

{KTools::Enums::LogType::Debug, "Debug"},

{KTools::Enums::LogType::Error, "Error"},

{KTools::Enums::LogType::Custom, "Custom"}

};

// End Log

// Start CurlClass

QString KTools::Curl::cacertPath = QDir::temp().path() + "/InfSyst";

QString KTools::Curl::cacertFileName = "cacert.pem";

QString KTools::Curl::fullCacertPath = cacertPath + '/' + cacertFileName;

QString KTools::Curl::pathToCacertInQrc = ":/resources/other/cacert.pem";

QString KTools::Curl::cookiePath = "";

// End CurlClass

// Start KTolsQmlInterface

KToolsQmlInterface\* KToolsQmlInterface::obj = new KToolsQmlInterface();

// End KTolsQmlInterface

**object.cpp**

#include "object.h"

KTools::HtmlAst::Object::Object() {}

bool KTools::HtmlAst::Object::makeAst(const QString &data)

{

htmlText = &data;

qint32 pos = 0;

return setRootTag(readTag(pos, htmlText->size()));

}

KTools::HtmlAst::Tag& KTools::HtmlAst::Object::readTag(qint32 &pos, const qint32 &endPos)

{

Tag \*tagClass = new Tag();

while (pos < endPos)

{

if (htmlText->mid(pos, 2) == "<!")

{

++pos;

while (htmlText->at(pos) != "<")

++pos;

}

else if (htmlText->at(pos) == "<")

{

bool tagNameReaded = false;

bool tagClosed = false;

++pos; // skip < char

if (htmlText->at(pos) == "/")

{

tagClass->status = Tag::StatusEnum::TagClosingDetected;

return \*tagClass;

}

while (pos < endPos)

{

if (!tagClosed)

{

if (htmlText->at(pos) == ">")

{

tagClosed = true; // tag end detected

++pos;

}

else if (htmlText->at(pos).isSpace() || htmlText->at(pos) == "/")

{

++pos; // skip space char after < char

}

else if ((htmlText->at(pos).isLetter() || htmlText->at(pos) == "-") && !tagNameReaded)

{

readTagName(pos, \*tagClass);

tagNameReaded = true;

}

else if ((htmlText->at(pos).isLetter() || htmlText->at(pos) == "-") && tagNameReaded)

{

readTagAttributes(pos, \*tagClass);

tagClosed = true;

}

}

else

{

qint32 tmpPos = findTagEndingPart(pos, endPos, \*tagClass);

if (tagClass->getName().toLower() == "script")

{

readJs(tagClass->getInnerContent());

tagClass->status = Tag::StatusEnum::TagValid;

pos += tagClass->getInnerContent().size() + 9;

return \*tagClass;

}

if (tagClass->isSelfclosing())

{

while (pos < tmpPos)

{

Tag \*newTag = &readTag(pos, tmpPos);

if (newTag->status == Tag::StatusEnum::TagValid)

{

tagClass->addChildTag(\*newTag);

}

else if (newTag->status == Tag::StatusEnum::TagInvalid)

{

delete newTag;

}

else if(newTag->status == Tag::StatusEnum::TagClosingDetected)

{

++pos;

}

}

tagClass->status = Tag::StatusEnum::TagValid;

return \*tagClass;

}

else

{

tagClass->status = Tag::StatusEnum::TagValid;

return \*tagClass;

}

}

}

}

else

{

tagClass->status = Tag::StatusEnum::TagInvalid; // error handling

pos++;

return \*tagClass;

}

}

tagClass->status = Tag::StatusEnum::TagValid;

return \*tagClass;

}

bool KTools::HtmlAst::Object::readTagName(qint32 &pos, Tag &tagClass)

{

qint32 start = pos;

while (htmlText->at(pos).isLetter() || htmlText->at(pos) == "-" || htmlText->at(pos).isDigit())

{

++pos;

}

tagClass.setName(htmlText->mid(start, pos - start));

return true;

}

bool KTools::HtmlAst::Object::readTagAttributes(qint32 &pos, Tag &tagClass)

{

while (htmlText->at(pos) != ">" || htmlText->mid(pos, 2) == "/>") // while tag not closed

{

qint32 startKey = pos;

while ((htmlText->at(pos).isLetter() || htmlText->at(pos) == "-" || htmlText->at(pos) == ":") && htmlText->at(pos) != "=")

{

++pos;

}

qint32 lenKey = pos - startKey; // key lenght

QChar quiote = htmlText->at(pos + 1);

if (htmlText->at(pos) == "=" && (quiote == "\"" || quiote == "'"))

{

pos += 2;

qint32 startValue = pos;

while (htmlText->at(pos) != quiote)

{

++pos;

}

tagClass.addAttribute(htmlText->mid(startKey, lenKey), htmlText->mid(startValue, pos - startValue));

}

else

{

return false; // here should be error handling

}

++pos; // skip " char

while (htmlText->at(pos) == " " || htmlText->at(pos) == "\n" || htmlText->at(pos) == "\r" || htmlText->at(pos) == "\t") // skip spaces between attributes

++pos;

}

++pos;

return true;

}

bool KTools::HtmlAst::Object::setRootTag(Tag &tag)

{

if (tag.status == Tag::StatusEnum::TagValid)

{

rootTag = &tag;

return true;

}

else

return false;

}

qint32 KTools::HtmlAst::Object::findTagEndingPart(const qint32 &pos, const qint32 &endPos, Tag &tagClass, qint32 differenceEdAndOp)

{

qint32 tmpPos = pos; // this also ABSOLUTE end position

++tmpPos;

QString tagOpeningPart = "<" + tagClass.getName();

qint32 tagOpSize = tagOpeningPart.size();

QString tagEndPart = "</" + tagClass.getName() + ">"; // thank to this variable ast maker now works in 6 times faster

qint32 tagEdSize = tagEndPart.size();

qint32 sameTagCounterEd = 0;

qint32 sameTagCounterOp = 0;

if (htmlText->mid(pos, tagEdSize) == tagEndPart)

{

tagClass.setSelfclosingness(true);

return pos;

}

if (tagClass.getName() == "tr")

{

QVector<qint32> closingTagsPos;

QVector<qint32> openingTagsPos;

while (tmpPos < endPos + 1)

{

if (htmlText->mid(tmpPos, tagEdSize) == tagEndPart) // tag ending part detected

{

if (sameTagCounterOp == 0) // tag ending part for this tag

{

tagClass.setInnerContent(htmlText->mid(pos, tmpPos - pos));

tagClass.setSelfclosingness(true);

//Logging::writeDebug(tagClass.getName() + "\t" + tagClass.getInnerContent().replace("\n", "").replace("\r", ""), "HtmlObject");

break;

}

else // tag ending part not for this tag

{

--sameTagCounterOp;

++sameTagCounterEd;

++tmpPos;

}

}

else // not ending part, something else

{

if (htmlText->mid(tmpPos - 1, tagOpSize) == tagOpeningPart) // another opening tag part detected

{

++sameTagCounterOp;

--sameTagCounterEd;

}

++tmpPos;

}

}

}

else

{

while (tmpPos < endPos + 1)

{

if (htmlText->mid(tmpPos, tagEdSize) == tagEndPart) // tag ending part detected

{

if (sameTagCounterOp == 0) // tag ending part for this tag

{

tagClass.setInnerContent(htmlText->mid(pos, tmpPos - pos));

tagClass.setSelfclosingness(true);

//Logging::writeDebug(tagClass.getName() + "\t" + tagClass.getInnerContent().replace("\n", "").replace("\r", ""), "HtmlObject");

break;

}

else // tag ending part not for this tag

{

--sameTagCounterOp;

++sameTagCounterEd;

++tmpPos;

}

}

else // not ending part, something else

{

if (htmlText->mid(tmpPos - 1, tagOpSize) == tagOpeningPart) // another opening tag part detected

{

++sameTagCounterOp;

--sameTagCounterEd;

}

++tmpPos;

}

}

}

/\*if (!tagClass.isSelfclosing() && sameTagCounterOp + sameTagCounterEd != 0) // opening parts more than ending parts

{ //this occurs whrn optionally selfclosing tags writed as selfclosing

findTagEndingPart(pos, endPos, tagClass, sameTagCounterEd);

}\*/

return tmpPos;

}

KTools::HtmlAst::Object::JsReadStatus KTools::HtmlAst::Object::readJs(const QString &inner)

{

//QString inner = tagClass.getInnerContent();

/\*if (tagClass.getAttributeValue("src") != QString())

{

inner

}\*/

if (inner.size() == 0)

return JsReadStatus::NotFound;

qint32 bracesCount = 0;

qint32 startPos = 0;

bool inObject = false;

bool inArray = false;

for (qint32 i = 0; i < inner.size(); i++)

{

//if (bracesCount < 0)

// bracesCount = 0;

if (inner[i] == '[')

{

++bracesCount;

if (!inObject && !inArray)

{

inArray = true;

startPos = i;

}

}

else if (inner[i] == '{')

{

++bracesCount;

if (!inObject && !inArray)

{

inObject = true;

startPos = i;

}

}

else if (inner[i] == "]")

{

--bracesCount;

if (bracesCount == 0 && inArray)

{

QJsonArray jsArr = KTools::Converter::convert<QString, QJsonArray>(inner.mid(startPos, i - startPos + 1));

if (!jsArr.isEmpty())

{

arrsAndObjs.arrs.append(jsArr);

}

else

{

i = startPos + 1;

}

inArray = false;

}

}

else if (inner[i] == "}")

{

--bracesCount;

if (bracesCount == 0 && inObject)

{

QJsonObject jsObj = KTools::Converter::convert<QString, QJsonObject>(inner.mid(startPos, i - startPos + 1));

if (!jsObj.isEmpty())

{

arrsAndObjs.objects.append(jsObj);

}

else

{

i = startPos + 1;

}

inObject = false;

}

}

}

}

**object.h**

#ifndef KTOOLS\_HTMLAST\_OBJECT\_H

#define KTOOLS\_HTMLAST\_OBJECT\_H

#include <QString>

#include "tag.h"

#include "../log.h"

namespace KTools::HtmlAst

{

class Object

{

public:

Object();

bool makeAst(const QString &data);

Tag \*rootTag;

private:

bool escapedChar = false;

const QString \*htmlText;

qint32 tagStartPos;

Tag& readTag(qint32 &pos, const qint32 &endPos);

bool readTagName(qint32 &pos, Tag &tagClass);

bool readTagAttributes(qint32 &pos, Tag &tagClass);

bool setRootTag(Tag &tag);

qint32 findTagEndingPart(const qint32 &pos, const qint32 &endPos, Tag &tagClass, qint32 differenceEdAndOp = 0);

//qint32 findTagEndingPartSecondPass(qint32 pos, qint32 endPos, HtmlTag &tagClass, qint32 differenceEdAndOp);

struct JsArrsAndObjects

{

QVector<QJsonObject> objects;

QVector<QJsonArray> arrs;

};

enum class JsReadStatus

{

Success,

InvalidString,

NotFound

};

JsReadStatus readJs(const QString &inner);

public:

JsArrsAndObjects arrsAndObjs;

};

}

#endif // KTOOLS\_HTMLAST\_OBJECT\_H

**tag.cpp**

#include "tag.h"

KTools::HtmlAst::Tag::Tag()

{

childTags = QVector<Tag\*>();

}

KTools::HtmlAst::Tag::~Tag()

{

if (parentTagSetted)

delete parentTag;

for (qint32 i = 0; i < childTags.size(); i++)

delete childTags.at(i);

}

bool KTools::HtmlAst::Tag::isSelfclosing()

{

return selfclosing;

}

void KTools::HtmlAst::Tag::setSelfclosingness(bool sc)

{

selfclosing = sc;

}

void KTools::HtmlAst::Tag::setName(const QString &newName)

{

name = newName;

}

QString& KTools::HtmlAst::Tag::getName()

{

return name;

}

void KTools::HtmlAst::Tag::addAttribute(const QString &key, const QString &value)

{

attributes.insert(key, value);

}

QString KTools::HtmlAst::Tag::getAttributeValue(const QString &key)

{

return attributes.value(key);

}

QMap<QString, QString>& KTools::HtmlAst::Tag::getAttributes()

{

return attributes;

}

void KTools::HtmlAst::Tag::clearAttributes()

{

attributes.clear();

}

void KTools::HtmlAst::Tag::setInnerContent(const QString &content)

{

innerContent = content;

}

QString& KTools::HtmlAst::Tag::getInnerContent()

{

return innerContent;

}

void KTools::HtmlAst::Tag::addChildTag(Tag &localTag)

{

childTags.append(&localTag);

localTag.setParentTag(\*this);

++childTagCounter;

}

QVector<KTools::HtmlAst::Tag\*>& KTools::HtmlAst::Tag::getChildTags()

{

return childTags;

}

void KTools::HtmlAst::Tag::setParentTag(Tag &localTag)

{

parentTag = &localTag;

parentTagSetted = true;

}

KTools::HtmlAst::Tag& KTools::HtmlAst::Tag::getParentTag()

{

return \*parentTag;

}

KTools::HtmlAst::Tag& KTools::HtmlAst::Tag::find(const qint32 numb)

{

return \*childTags[numb];

}

qint32 KTools::HtmlAst::Tag::getChildTagCounter()

{

return childTagCounter;

}

bool KTools::HtmlAst::Tag::isExist(qint32 childTagIndex)

{

if (childTagIndex < childTags.size())

return true;

else

return false;

}

**tag.h**

#ifndef HTMLTAG\_H

#define HTMLTAG\_H

#include <QString>

#include <QMap>

namespace KTools::HtmlAst

{

class Tag

{

public:

Tag();

~Tag();

void setSelfclosingness(const bool sc);

void setName(const QString &newName);

void addAttribute(const QString &key, const QString &value);

void setInnerContent(const QString &content);

void addChildTag(Tag &localTag);

void setParentTag(Tag &localTag);

bool isSelfclosing();

QString& getName();

QString getAttributeValue(const QString &key);

QMap<QString, QString>& getAttributes();

QString& getInnerContent();

QVector<Tag\*>& getChildTags();

Tag& getParentTag();

qint32 getChildTagCounter();

void clearAttributes();

Tag& find(const qint32 numb); // returns child tag ny number

bool isExist(qint32 childTagIndex);

enum class StatusEnum

{

TagClosingDetected,

TagInvalid,

TagValid

};

StatusEnum status;

private:

bool selfclosing = false;

QString name;

QMap<QString, QString> attributes;

QString innerContent;

QVector<Tag\*> childTags;

Tag \*parentTag;

bool parentTagSetted = false;

qint32 childTagCounter = 0;

};

}

#endif // HTMLTAG\_H

**Tools.js**

function bytesToHumanReadable(bytes, mode)

{

var ext = "/s"

if (!ext === null)

ext = ""

if (bytes === 0)

return 0 + " Bit" + ext

var tmp = 0

if (bytes < 1024)

{

tmp = bytes + " Byte" + ext

}

else

{

tmp = (bytes / 1024);

if (tmp < 1024)

{

tmp = Math.round(tmp \* 1000) / 1000

tmp += " Kb" + ext

}

else

{

tmp = tmp / 1024

if (tmp < 1024)

{

tmp = Math.round(tmp \* 1000) / 1000

tmp += " Mb" + ext

}

else

{

tmp = tmp / 1024;

tmp = Math.round(tmp \* 1000) / 1000

tmp += " Gb" + ext

}

}

}

return tmp

}

function cutNumberByPercents(number, percents)

{

return (number / 100) \* percents;

}

function boolSwich(boolVariable) {

if (boolVariable)

boolVariable = false

else

boolVariable = true

return boolVariable

}

**KawaiButton.qml**

import QtQuick 2.0

import QtQuick.Controls 2.2

Button {

id:root

property string labelText: "Sample text"

property string onPressedColor: "Black"

property string onHoveredColor: "Gray"

property string standartColor: "#dadada"

property string borderColorC: "Black"

property string onPressedBorderColor: "Black"

property string onHoveredBorderColor: "Black"

property string labelTextColor: "Black"

//property string onPressedLabelTextColor: "Black"

//property string onHoveredLabelTextColor: "Black"

property int borderWidthC: 1

property int borderRadiusC: 5

property int heightC: 40

property int widthC: 100

property int labelTextWeight: Font.Normal

background: Rectangle {

id: buttonRectangle

implicitWidth: widthC

implicitHeight: heightC

border.color: borderColorC

border.width: borderWidthC

radius: borderRadiusC

color: standartColor

Text {

id: buttonLabel

anchors.centerIn: parent

text: labelText

color: labelTextColor

font.weight: labelTextWeight

font.family: "Segoe UI"

}

}

states: [

State {

name: "onPressedButtonRectangle"

when: root.pressed == true

PropertyChanges {

target: buttonRectangle

color: onPressedColor

border.color: onPressedBorderColor

}

},

State {

name: "onHoveredButtonRectangle"

when: root.hovered == true

PropertyChanges {

target: buttonRectangle

color: onHoveredColor

border.color: onHoveredBorderColor

}

}

]

}

**KawaiHorisontalLine.qml**

import QtQuick 2.0

Rectangle{

height: 2

border.color: "Black"

}

**KawaiLabelInputFieldCombination.qml**

import QtQuick 2.0

import QtQuick.Controls 2.5

Rectangle {

property alias klifcLabel: label

property alias klifcTextField: textField

property alias klifcDecorativeRectangle: decorativeRectangle

height: 40

width: 300

border.color: "Black"

Text {

id: label

anchors.left: parent.left

anchors.verticalCenter: parent.verticalCenter

anchors.margins: 5

font.pixelSize: 14

}

TextArea {

id: textField

anchors.right: parent.right

anchors.left: label.right

anchors.verticalCenter: parent.verticalCenter

anchors.margins: 5

background: Rectangle {

id: decorativeRectangle

//border.color: textField.activeFocus ? "Grey" : "Transparent"

}

font.pixelSize: 15

}

KawaiHorisontalLine {

anchors.top: textField.bottom

anchors.left: textField.left

anchors.right: textField.right

border.color: textField.activeFocus ? "Grey" : "Transparent"

}

}