**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

СОГЛАСОВАНО  
Профессор департамента программной инженерии, кандидат технических наук  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.М. Гринкруг  
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

УТВЕРЖДАЮ  
Академический руководитель образовательной программы «Программная инженерия»  
  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  
«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

**ПРОГРАММНЫЙ КОМПОНЕНТ OPENGLVIEWER БИБЛИОТЕКИ JAVABEANS-КОМПОНЕНТ ДЛЯ 3D-ГРАФИКИ**

**Пояснительная записка**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.01-01 81 01-1-ЛУ**

**Исполнитель**

Студент группы БПИ163

\_\_\_\_\_\_\_\_\_\_\_\_ / Д.Е. Крайнов /

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

**Москва 2019**

**УТВЕРЖДЕН**

**RU.17701729.04.01-01 81 01-1-ЛУ**

**ПРОГРАММНЫЙ КОМПОНЕНТ OPENGLVIEWER БИБЛИОТЕКИ JAVABEANS-КОМПОНЕНТ ДЛЯ 3D-ГРАФИКИ**

**Пояснительная записка**

**RU.17701729.04.01-01 81 01-1**

**Листов 24**

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

**Москва 2019**

СОДЕРЖАНИЕ

[1. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 4](#_Toc483401394)

[1.1. Наименование программы 4](#_Toc483401395)

[1.2. Документ, на основании которого ведется разработка 4](#_Toc483401396)

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 5](#_Toc483401397)

[2.1. Функциональное назначение 5](#_Toc483401398)

[2.2. Эксплуатационное назначение 5](#_Toc483401399)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 6](#_Toc483401400)

[3.1. Постановка задачи на разработку программы 6](#_Toc483401401)

[3.1.1. Клиентская часть 6](#_Toc483401402)

[3.1.2. Серверная часть 7](#_Toc483401403)

[3.2. Описание применяемых методов 6](#_Toc483401401)

[3.3. Описание алгоритма и функционирования программы 8](#_Toc483401404)

[3.3.1. Описание алгоритма программы 8](#_Toc483401405)

[3.3.2. Описание функционирования программы 9](#_Toc483401406)

[3.4. Описание и обоснование выбора метода организации входных и выходных данных 11](#_Toc483401407)

[3.4.1. Описание и обоснование выбора метода организации входных данных 11](#_Toc483401408)

[3.4.2. Описание и обоснование выбора метода организации выходных данных 11](#_Toc483401409)

[3.5. Описание и обоснование выбора технических и программных средств 11](#_Toc483401410)

[4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ 13](#_Toc483401411)

[4.1. Ориентировочная экономическая эффективность 13](#_Toc483401412)

[4.2. Предполагаемая потребность 13](#_Toc483401413)

[4.3. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами 13](#_Toc483401414)

[5. ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ 14](#_Toc483401415)

[Приложение 1 15](#_Toc483401416)

[Приложение 2 16](#_Toc483401417)

[Приложение 3 17](#_Toc483401418)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 24](#_Toc483401419)

# ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

* 1. Наименование программы  
     Наименование программы: «Программный компонент OpenGLViewer библиотеки JavaBeans-компонент для 3D-графики» («The OpenGLViewer Software Component for JavaBeans 3D Graphics Components Library»). Краткое название: “OpenGLViewer”.
  2. Документ, на основании которого ведется разработка  
     Приказ декана факультета компьютерных наук Национального исследовательского университета «Высшая школа экономики» № 2.3-02/1012-0 2 от 10.12.2018.

# НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

* 1. Функциональное назначение  
     Функциональным назначением компонента является предоставление возможности графического рендеринга моделей, представленных в виде графа сцены [1], с помощью технологии OpenGL [2] и графического процессора пользовательского компьютера.
  2. Эксплуатационное назначение  
     Программный компонент предлагается к эксплуатированию программистами в области компьютерной графики и 3D-моделистами в качестве инструмента для тестирования созданных графических моделей.

# ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

* 1. Постановка задачи на разработку программы  
     Компонент должен предоставить интерфейс для взаимодействия с ним другим JavaBeans-компонентам библиотеки, в составе которой он должен работать. В частности, компонент OpenGLViewer должен:
* получать объект 3D-сцены (граф сцены) для отображения;
* поддерживать собственный вид на переданную ему сцену;
* обеспечивать графическое отображение (рендеринг) этого вида на экране, используя возможности библиотеки OpenGL с помощью программного интерфейса Java OpenGL (JOGL) [3];
* предоставлять возможность параллельного наблюдения сцены с помощью нескольких экземпляров компонента;
* предоставлять возможность манипулирования сценой с помощью методов получения и модификации графа сцены и её настроек (геттеры и сеттеры).
  1. **Описание применяемых методов и алгоритмов**
* **Компонентная модель и спецификация JavaBeans**

Компонентно-ориентированное программирование – парадигма программирования, основная идея которой – представление и разработка программ как набора компонентов - независимых модулей, направленных на выполнение одной определенной задачи. Каждый компонент компилируется и подключается к программному проекту отдельно от остальных.

Выделяется несколько основных преимуществ данного подхода [4].

Во-первых, код, написанный с использованием компонентов, проще разрабатывать. В силу ограниченности действий, выполняемых каждым компонентом, становится проще понимать, как добиться той функциональности, какая требуется в финальной программе. По этой же причине легко создавать новые вариации программ с дополнительными функциями путём замены или добавления компонентов.

Во-вторых, изолированность и заточенность компонентов под конкретное действие позволяет проще отслеживать ошибки, возникающие в результате работы.

Наконец, компоненты проще тестировать, так как сразу понятно, что должен и не должен делать компонент.

Спецификация JavaBeans – по сути, реализация компонентной методологии в среде Java [5]. Bean – любой класс, написанный по нескольким правилам:

* в классе присутствует публичный конструктор без параметров;
* все поля – приватные, работа с ними осуществляется через публичные методы доступа (геттеры) и модификации (сеттеры);
* класс имплементирует интерфейс Serializable (должна присутствовать возможность преобразования экземпляра класса в поток байтов для сохранения в файловую систему, базу данных и т.п.);
* геттеры не принимают параметров, возвращают объект определенного типа, названы в стиле get\*(), где \* - название поля;
* сеттеры принимают параметр, ничего не возвращают, названы в стиле set\*(), где \* - название поля.

Подобная стандартизация классов нашла своё применение, в первую очередь, в приложениях, предназначенных для визуального построения программ. Повторное использование компонентов и возможность их конфигурации на лету, а также возможность использования событийной модели (оповещения при изменении свойств класса) – основные преимущества спецификации JavaBeans. В частности, виджеты библиотеки Swing для создания графических интерфейсов в Java являются «бинами».

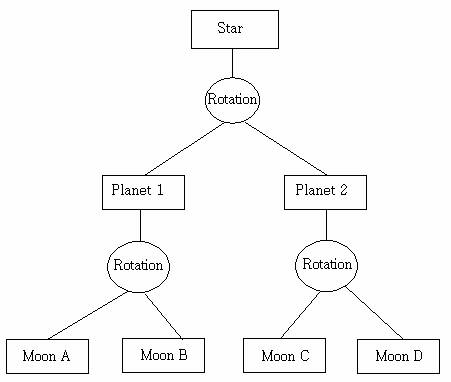
Набор компонентов OpenGLViewer разрабатывается в соответствии с данной спецификацией, что позволит использовать компоненты набора – двух- и трехмерные объекты и сцены, состоящие из данных объектов, а также экран просмотра и вспомогательные компоненты – в визуальных редакторах интерфейсов, фактически – без необходимости написания какого-либо кода.

* **Представление модели в виде графа сцены**

Любая графическая модель складывается из множества графических примитивов. Примитив – простейший геометрический объект, отображаемый на экране. Например, два прямоугольных треугольника могут образовать прямоугольник, множество квадратов в трехмерном пространстве – куб или сферу, и так далее.

В наборе разрабатываемых компонентов, помимо обычных примитивов (линия, треугольник), присутствуют сложные примитивы, упрощающие работу по созданию модели – куб, сфера, цилиндр и т. п. Для их получения не требуется описание множества более мелких примитивов, обычно их составляющих.

В свою очередь, экземпляры компонентов соединяются в одну структуру для удобства моделирования и отображения. Широко распространена практика использования в качестве такой структуры дерева, или «графа сцены».



*Рис. 1. Пример графа сцены*

На рис. 1 изображен простой граф сцены на примере планетной системы. Есть центральный, коренной объект – звезда (“Star”). Ко всем потомкам – объектам, логически связанным со звездой – применяется трансформация вращения вокруг родительского объекта (“Rotation”). В коде объектов применение вращения может быть запрограммировано как изменение координат планет, а параметры изменения можно получить как композицию информации о трансформации, примененной к данному объекту, и информации о трансформации объекта-родителя, чем будет обеспечиваться логическая связность модели.

В различных программных продуктах для программирования графики в режиме реального времени и создания игр трансформации традиционно производятся раз в единицу времени (частота обычно измеряется в кадрах в секунду) в специальном методе (например, в среде разработки для движка Unity таковым является метод Update(), вызываемый раз в кадр у всех объектов, являющихся производными от базового класса MonoBehaviour [6]).

Компоненты в данной разработке, представляющие графические примитивы, работают по такому же принципу. У них есть внутреннее состояние, которое можно менять либо вручную, либо путём вызова метода Update(), который имеется у всех компонентов, являющихся производными от основного класса графа сцены – GLObject.

## Описание алгоритма и функционирования программы

### Описание алгоритма программы

Ниже приводится краткая последовательность действий, выполняемых программой в процессе работы в автономном режиме:

* Пользователь запускает .jar-файл, содержащий программу;
* Программа создаёт окно, содержащее экземпляр компонента OpenGLViewer и вспомогательное меню для работы с компонентом, а также инициализирует пустой граф сцены;
* Пользователь может либо загрузить готовый граф из файла через меню, либо начать добавлять новые объекты в граф;
  + Во время добавления через меню каждого нового объекта пользователю будет предложено указать его начальные координаты, размер, цвет и другие параметры в зависимости от типа объекта;
* Пользователь может также настроить отображение осей координат (включено/выключено), отображение отладочного текста (включено/выключено) и состояние отображения сцены (рендеринг включен/выключен);
* Во время просмотра пользователь может пользоваться возможностью обзора сцены с помощью клавиш-стрелок, Shift и Ctrl для перемещения по всем осям координат и мышки для изменения угла обзора или приближения/отдаления сцены;
* После окончания работы со сценой пользователь может сохранить текущую сцену в файл (происходит сериализация графа сцены в формат json).

### Описание функционирования программы

Структура набора компонентов состоит из нескольких пакетов с классами. Основные из них:

* **objects**, содержащий компоненты, представляющие собой графические примитивы;
* **viewer**, содержащий компонент OpenGLViewer для просмотра сцен;
* **window**, содержащий компонент окна для запуска набора в автономном режиме.

### Пакет objects и класс GLObject

Основной класс пакета – **GLObject**. Это абстрактный класс, представляющий узел графа сцены, а также графический примитив, который будет отрисован. Все примитивы в пакете наследуются от данного класса. GLObject содержит ссылку на список дочерних объектов графа, тоже являющихся экземплярами GLObject. Объекту можно также присвоить идентификатор и текстуру.

GLObject содержит абстрактные методы draw(), который отвечает за отрисовку объекта на экране посредством команд OpenGL, и update(), который изменяет состояние объекта раз в единицу времени.

Для удобства отрисовки GLObject предоставляет метод drawTree(), который отрисовывает объект, вызывает метод update() и рекурсивно повторяет процедуру для всех потомков объекта, таким образом производя рендеринг всего дерева.

Помимо реализации метода, каждый дочерний класс в пакете содержит информацию о состоянии объекта (размер, цвет и т.д.) в виде дополнительных полей.

### Пакет viewer и класс OpenGLViewer

**OpenGLViewer** – основной компонент набора. Он обеспечивает подключение различных возможностей OpenGL (свет, текстурирование и т.д.), отрисовку, поддерживает вид на сцену (позицию в пространстве и угол поворота камеры), предоставляет интерфейс для изменения этого вида с помощью пользовательских команд, контролирует состояние рендеринга, показа отладочного текста и осей координат при необходимости.

OpenGLViewer содержит ссылку на корневую вершину графа сцены и раз в кадр производит обход этого графа в глубину, производя последовательную отрисовку объектов графа.

В пакете viewer также присутствует компонент **GLViewerCanvas** – обертка над OpenGLViewer, содержащая обработчики событий нажатия на клавиатуру и движения мышью.

### Пакет window и класс OpenGLTestFrame

Пакет window содержит единственный компонент **OpenGLTestFrame**, представляющий собой окно для работы с набором компонентов в автономном режиме.

## Описание и обоснование выбора метода организации входных и выходных данных

### Описание и обоснование выбора метода организации входных данных

В качестве входных данных в клиенте используются электронные документы в формате PDF. Данный формат является удобным и наиболее часто используемым форматом документов [3]. Отправка информации о входных данных (т.е. документах) на сервер организована с помощью встроенных в стандартную библиотеку C# методов работы с протоколом HTTP.

### Описание и обоснование выбора метода организации выходных данных

После обработки сервером и получения программой выходные данные для каждого обработанного документа отображаются в столбце «Состояние» и поле «Статус» и могут иметь один следующих видов:

* В случае успешной загрузки информации в базу умного контракта (возможность доступна только в полной версии программы):
  + столбец «Состояние»: «Загружен, дата загрузки: » + дата записи;
  + поле «Статус»: «Хэш документа «документ» записан в базу смарт-контракта»;
* В случае успешной верификации документа:
  + столбец «Состояние»: «Верифицирован, дата записи: » + дата записи;
  + поле «Статус»: «Документ «документ» верифицирован, дата записи: » + дата записи;
* В случае, если документ не верифицирован:
  + столбец «Состояние»: «Не верифицирован»;
  + поле «Статус»: «Документ «документ» не верифицирован».
  1. Описание и обоснование выбора технических и программных средств  
      Языком разработки клиентской части программы является C# ввиду существования определенных ограничений, связанных с разработкой программ как частей курсовых работ на первом курсе образовательной программы «Программная инженерия». Диаграмма классов клиентской части представлена в Приложении 1, описание назначения классов программы – в Приложении 2, описание методов и полей классов – в Приложении 3.

Серверная часть реализована при помощи следующего стека технологий:

* Apache – свободный кроссплатформенный веб-сервер;
* Ubuntu Linux – операционная система, на которой работает сервер;
* Ethereum – платформа для создания децентрализованных приложений на базе технологии Blockchain [2];
* testrpc – Ethereum-клиент, основанный на технологии Node.js и предназначенный для эмулирования блокчейна и развёртывания умных контрактов [2];
* Solidity – JavaScript-подобный язык программирования умных контрактов [2];
* Truffle – фреймворк, предоставляющий среду для компиляции, тестирования и развертывания умных контрактов [2];
* JavaScript – язык программирования, в данной программе предназначенный для создания скриптов, выполняющих запросы к контрактам;
* PHP – серверный язык программирования, в данной программе предназначенный для передачи данных от клиента и к клиенту, а также выполнения команд Truffle в командной оболочке операционной системы.

Были обозначены следующие минимальные требования к аппаратному и программному обеспечению, необходимые для работы программы:

* Клиентская часть:
  + Минимальная тактовая частота процессора – 1 Ггц;
  + Минимальный объем ОЗУ – 1 Гб;
  + Минимальное свободное место на жёстком диске – 3 МБ;
  + Интернет-канал, минимальная скорость соединения – 1 Мбит/сек;
  + Монитор, клавиатура и мышь;
  + ОС Windows 7 или выше;
  + Установленная среда .NET Framework 4.5.2 или выше.
* Серверная часть:
  + Минимальный объем ОЗУ – 1 Гб;
  + Минимальный объем свободного места на жёстком диске – 500 МБ;
  + Интернет-канал с минимальной пропускной способностью 100 Мбит/сек;
  + ОС Ubuntu Linux 16.04 или новее;
  + PHP 7.0;
  + Apache 2.2 или выше;
  + Установленные фреймворки Truffle и ethereumjs-testrpc [2].

# ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

* 1. Ориентировочная экономическая эффективностьДанный продукт потенциально позволит упростить работу с документами и уменьшить усилия, затрачиваемые на их хранение.
  2. Предполагаемая потребностьПрограмма может использоваться в учебных заведениях и организациях, где существует задача проверки подлинности дипломов и сертификатов.
  3. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогамиНа момент начала разработки аналогов открытого отечественного и зарубежного разрабатываемого программного обеспечения не нашлось.

# ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ

1. Understanding and Implementing Scene Graphs [Electronic resource]. // GameDev.net [Official website]. URL: <http://archive.gamedev.net/archive/reference/programming/features/scenegraph/index.html> (accessed: 14.04.2019)

2. OpenGL 2.1 Reference Pages [Electronic resource]. // The Khronos Group Inc [Official website]. URL: <https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/> (accessed: 14.04.2019)

3. JOGL – Java Binding for the OpenGL API [Electronic resource]. // JogAmp.org - Java graphics, audio, media and processing libraries exposing OpenGL, OpenCL, OpenAL and OpenMAX [Official website]. URL: <http://jogamp.org/jogl/www/> (accessed: 14.04.2019)

4. Advantages of component-based development [Electronic resource]. // ResearchGate [Official website]. URL: <https://www.researchgate.net/figure/Advantages-of-component-based-development_tbl1_220588371> (accessed: 14.04.2019)

5. JavaBeans Spec [Electronic resource]. // Oracle [Official website]. URL: <https://www.oracle.com/technetwork/articles/javaee/spec-136004.html> (accessed: 14.04.2019)

6. Scripting API: MonoBehaviour.Update() [Electronic resource]. // Unity [Official website]. URL: <https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html> (accessed: 14.04.2019)

# Приложение 1

**Описание и функциональное назначение классов**

|  |  |
| --- | --- |
| **Имя класса** | **Назначение** |
| MainForm | Представляет собой форму Windows, содержащую главное и единственное окно программы. Класс разделен на три файла (MainForm.cs, MainFormGeneralMethods.cs, MainFormDocumentMethods.cs) в целях декомпозиции и упрощения разработки.  • MainForm.cs содержит обработчики событий для кнопок, работающих с таблицей.  • MainFormGeneralMethods.cs содержит методы работы с формой и полем «Статус».  • MainFormDocumentMethods.cs содержит обработчики событий для кнопок, работающих с документами и коммуникацией с сервером, а также метод генерации хэш-суммы документа. |
| ServerConnection | Статический класс, предоставляющий методы коммуникации с удалённым сервером. |
| Program | Статический класс, содержащий единственный метод Main(), запускающий программу. |

# Приложение 3

**Описание и функциональное назначение полей/свойств и методов классов**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Класс MainForm** | | | | | | | |
| **Поля и свойства (если не указано иное, типы полей и свойств класса принадлежат пространству имён System.Windows.Forms). Примечание: в версии Light отсутствуют поля uploadSelectedButton и uploadAllButton.** | | | | | | | |
| Имя | Модификатор доступа | | Тип | | | Описание | |
| addButton | private | | Button | | | Кнопка «Добавить…». Событие Click вызывает метод addButton\_Click. | |
| selectAllButton | private | | Button | | | Кнопка «Отметить все». Событие Click вызывает метод selectAllButton \_Click. | |
| deselectAllButton | private | | Button | | | Кнопка «Сбросить все». Событие Click вызывает метод deselectAllButton \_Click. | |
| deleteButton | private | | Button | | | Кнопка «Удалить выбранные». Событие Click вызывает метод deleteButton \_Click. | |
| deleteAllButton | private | | Button | | | Кнопка «Удалить все». Событие Click вызывает метод deleteAllButton \_Click. | |
| uploadSelectedButton | private | | Button | | | Кнопка «Загрузить выбранные». Событие Click вызывает метод uploadSelectedButton \_Click. | |
| uploadAllButton | private | | Button | | | Кнопка «Загрузить все». Событие Click вызывает метод uploadAllButton \_Click. | |
| verifySelectedButton | private | | Button | | | Кнопка «Проверить выбранные». Событие Click вызывает метод verifySelectedButton \_Click. | |
| verifyAllButton | private | | Button | | | Кнопка «Проверить все». Событие Click вызывает метод verifyAllButton \_Click . | |
| exitButton | private | | Button | | | Кнопка «Выйти». Событие Click вызывает метод exitButton\_Click. | |
| documentsBox | private | | GroupBox | | | Контейнер элементов WinForms. Содержит все кнопки и таблицу documentList. | |
| documentsList | private | | DataGridView | | | Таблица, содержащая информацию о документах. Событие KeyDown вызывает метод documentsList\_KeyDown. | |
| Checkboxes | private | | DataGridViewCheckBoxColumn | | | Столбец documentList, содержащий информацию об отмеченных документах в виде элементов CheckBox. | |
| DocumentName | private | | DataGridViewTextBoxColumn | | | Столбец documentList, содержащий информацию о названиях документов. | |
| DocumentPath | private | | DataGridViewTextBoxColumn | | | Столбец documentList, содержащий информацию о расположении документов на компьютере. | |
| DocumentHash | private | | DataGridViewTextBoxColumn | | | Столбец documentList, содержащий информацию о хэш-суммах документов. | |
| DocumentState | private | | DataGridViewTextBoxColumn | | | Столбец documentList, содержащий информацию о названиях документов. | |
| docsSelectDialog | private | | OpenFileDialog | | | Диалог, предлагающий выбрать один или несколько файлов, подходящих под фильтр (возможен выбор только документов в формате PDF). | |
| statusBox | private | | GroupBox | | | Контейнер, содержащий элемент status. | |
| status | private | | RichTextBox | | | Поле «Статус», выводящее информацию об успешности обработки документов. | |
| components | private | | System.ComponentModel.IContainer | | | Обязательная переменная конструктора. | |
| **Методы. Примечание: в версии Light отсутствуют методы uploadSelectedButton\_Click и uploadAllButton\_Click.** | | | | | | | |
| Имя | | Модификатор доступа | | Тип | Параметры | | Описание |
| MainForm | | public | | void |  | | Конструктор формы. |
| InitializeComponent | | private | | void |  | | Требуемый метод для поддержки конструктора. |
| Dispose | | protected | | void | bool disposing | | Освобождает все используемые ресурсы. |
| MainForm\_Load | | private | | void | object sender, System.EventArgs e | | Вызывает проверку соединения во время загрузки формы,  закрывает приложение, если соединения нет. |
| addButton\_Click | | private | | void | object sender, System.EventArgs e | | Вызывает окно выбора файлов и добавляет их в список с помощью AddToGrid(). |
| AddToGrid | | private | | void | string path | | Добавляет в таблицу документ, расположенный по указанному пути. |
| selectAllButton\_Click | | private | | void | object sender, System.EventArgs e | | Отмечает все файлы в списке. |
| deselectAllButton\_Click | | private | | void | object sender, System.EventArgs e | | Снимает отметку со всех файлов в списке. |
| deleteButton\_Click | | private | | void | object sender, System.EventArgs e | | Удаляет отмеченные файлы из списка. |
| deleteAllButton\_Click | | private | | void | object sender, System.EventArgs e | | Удаляет все файлы из списка. |
| documentList\_DragEnter | | private | | void | object sender, System.EventArgs e | | Используется в связке с documentList\_DragDrop для переноса документов  в таблицу перетаскиванием. |
| documentList\_DragDrop | | private | | void | object sender, System.EventArgs e | | Используется в связке с documentList\_DragEnter для переноса документов  в таблицу перетаскиванием. |
| documentList\_MouseDown | | private | | void | object sender, System.MouseEventArgs e | | Снимает фокус с ячеек по нажатию на серую область таблицы. |
| documentList\_KeyDown | | private | | void | object sender, System.EventArgs e | | Удаляет выделенные файлы из списка по нажатию Delete. |
| documentList\_MouseDoubleClick | | private | | void | object sender, System.MouseEventArgs e | | Изменяет отметку документа на противоположную по двойному нажатию на его строчку в таблице. |
| UpdateStatus | | private | | void | string text, System.Drawing.Color | | Добавляет указанную строку в поле status, используя выбранный цвет или цвет по умолчанию, если он не указан. |
| status\_TextChanged | | private | | void | object sender, System.EventArgs e | | Автоматически прокручивает текст поля вниз для удобства просмотра. |
| exitButton\_Click | | private | | void | object sender, System.EventArgs e | | Выходит из программы. |
| uploadSelectedButton\_Click | | private | | void | object sender, System.EventArgs e | | Вызывает CallUpload() для отмеченных документов, выводит результат записи их хэшей в базу смарт-контракта. |
| uploadAllButton\_Click | | private | | void | object sender, System.EventArgs e | | Вызывает CallUpload() для всех документов, выводит результат записи их хэшей в базу смарт-контракта. |
| ChangeControlsState | | private | | void | bool enabled | | Включает или выключает кнопки приложения в зависимости от параметра. |
| CallUpload | | private | | Task | string action, bool applyToAll | | Вызывает ServerConnection.UploadDocument() для указанного действия. |
| GenerateHash | | internal | | string | string path | | Возвращает SHA256-хэш файла, расположенного по указанному пути. |
| ModifyState | | private | | void | string docPath, string state | | Модифицирует состояние документа, расположенного по указанному пути. |
| verifySelectedButton\_Click | | private | | void | object sender, System.EventArgs e | | Вызывает CallUpload() для отмеченных документов, выводит результат их верификации. |
| verifyAllButton\_Click | | private | | void | object sender, System.EventArgs e | | Вызывает CallUpload() для всех документов, выводит результат их верификации. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Класс ServerConnection** | | | | | | |
| **Поля и свойства** | | | | | | |
| Имя | Модификатор доступа | | Тип | | Описание | |
| serverIp | private | | String | | IP сервера. | |
| dateFormat | private | | String | | Формат вывода даты и времени. | |
| **Методы** | | | | | | |
| Имя | Модификатор доступа | Тип | | Параметры | | Описание |
| CheckConnection | internal | bool | |  | | Проверяет наличие соединения к серверу. |
| UploadDocument | internal | Task | | string path, string action, Action<string, Color> Update, Action<string, string> ModifyState | | Загружает хэш документа по данному пути на сервер для дальнейшего использования смарт-контрактом и выдает результат выполнения указанного действия. |
| SetServerIp | internal | bool | |  | | Получает IP-адрес сервера из конфигурационного файла. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Класс Program** | | | | |
| **Методы** | | | | |
| Имя | Модификатор доступа | Тип | Параметры | Описание |
| Main | private | void |  | Запускает программу. |

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |