**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: «Программная модель умных контрактов» («The Smart Contracts Program Model»). Краткое название: «Умные контракты».
  2. Документ, на основании которого ведется разработка  
     Приказ Национального исследовательского университета «Высшая школа экономики» № 2.3-02/0812-01 от 08.12.2016 «Об утверждении тем, руководителей курсовых работ студентов образовательной программы Программная инженерия факультета компьютерных наук».

# НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

* 1. Функциональное назначение  
     Функциональным назначением программы является реализация системы работы с электронными версиями дипломов в PDF-формате, а именно для реализации возможности проверки подлинности конкретного документа путем поиска в базе умного контракта.
  2. Эксплуатационное назначение  
     Возможности программы предполагают её использование в образовательных учреждениях, выдающих электронные версии дипломов вместе с обычными, а также в прочих организациях, где существует необходимость проведения проверок подлинности различных документов в электронном формате. Клиентская часть программы выполняется на персональных компьютерах.

# ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

* 1. Постановка задачи на разработку программы  
     Клиент программы предоставляется в двух версиях – полной и версии Light с неполным функционалом. В разработанной программе должны быть реализованы следующие возможности:
     1. Клиентская часть- Добавление в программу информации о документах, с которыми будет вестись работа, через кнопку «Добавить…» или путём перетаскивания иконок документов на таблицу с использованием способа Drag-and-drop;

- Вывод информации о документах в таблицу программы в следующем формате:

- название документа;

- путь к документу на компьютере;

- SHA256-сумма документа;

- состояние документа.

- Выбор/снятие выбора со всех документов, с которыми должна вестись работа, по нажатию соответствующих кнопок («Отметить все»/«Сбросить все»), выбор конкретных документов для последующей работы с ними;

- Удаление выбранных или всех документов из списка по нажатию соответствующих кнопок («Удалить выбранные»/«Удалить все») или по нажатию клавиши Delete;

- Отправка SHA256-сумм выбранных или всех документов для проверки их подлинности по нажатию клавиш «Проверить выбранные» или «Проверить все»;

- Вывод информации о работе с документами в поле «Статус» и в столбце «Состояние» таблицы.

* **Дополнительный функционал полной версии программы:**

- Отправка SHA256-сумм выбранных или всех документов в базу смарт-контракта по нажатию клавиш «Загрузить выбранные» или «Загрузить все».

* + 1. Серверная часть  
       - Получение SHA256-сумм документов из программы;   
       - Интеракция со смарт-контрактом, получающим на вход SHA256-сумму документа и дату записи, записывающим её в базу и сообщающим об успешности выполнения задачи;  
       - Интеракция со смарт-контрактом, получающим на вход SHA256-сумму документа, проводящим поиск суммы в базе и сообщающим об успешности нахождения суммы, а также передающим дату записи, если таковая имеется;  
       - Передача информации об успешности выполненных операций клиентской части программы.
  1. **Описание применяемых методов**
* Блокчейн и умные контракты

Программный продукт должен являться реализацией механизма умных контрактов в конкретной предметной области (верификация дипломов образовательных учреждений). Умные контракты (смарт-контракты) – алгоритмы, выполняющиеся и хранящиеся поверх децентрализованных сред (блокчейнов), за выполнением условий которых следят все участники сети. Блокчейн (Blockchain) – технология, реализующая распределённую базу транзакций, формирующих блоки и цепочки блоков согласно определенным правилам. Отличительной особенностью блокчейнов является тот факт, что в базах такого типа невозможно, или крайне сложно вычислительно, изменить или удалить записи, так как копия блокчейна имеется у каждого участника сети, все участники постоянно обмениваются информацией о транзакциях друг с другом и для добавления новых записей необходимо их подтверждение большинством участников сети. Данные особенности технологии позволяют найти ей применение в сфере работы с документами, в частности, с применением смарт-контрактов.

* Хэш-суммы

В основе программы лежат алгоритмы по обработке смарт-контрактами хэш-сумм различных документов. Хэш-сумма (хэш), или контрольная сумма, является результатом преобразования массива входных данных произвольной длины в выходную строку фиксированной длины по определенному алгоритму. Криптографические хэш-функции должны отвечать дополнительным условиям, в частности, они должны быть достаточно стойкими к таким методам криптоанализа, как поиск коллизий. Коллизия хэшей – явление, когда два различных массива входных данных отображаются в одинаковый выходной хэш, причём коллизии – обязательное явление в силу бесконечности возможных входных данных, которым противопоставляется строка фиксированной длины. Если хэш-функция достаточно стойкая, поиск коллизий является крайне вычислительно сложной проблемой, таким образом, вероятность наличия одинакового хэша у двух разных документов крайне мала. Данный факт можно использовать для решения проблемы проверки подлинности документов.

В качестве хэш-функции была выбрана криптографическая хэш-функция SHA-256 из семейства SHA-2, являющаяся достаточно устойчивой к криптоанализу и широко распространённой [1]. После выбора функции остаётся проблема хранения базы хэшей. Её предлагает решить модель распределённой базы данных, которой и являются вышеописанные блокчейны. Смарт-контракты в данной модели являются надстройками над блокчейном, работающими с хэш-суммами документов. Данный программный продукт предлагает реализацию модели.

## Описание алгоритма и функционирования программы

### Описание алгоритма программы

Ниже приводится краткая последовательность действий, выполняемых программой в процессе обработки одного документа (здесь и далее слова «документ» и «диплом» являются взаимозаменяемыми, если не указано иное):

* Для выбранного путём добавления в программу документа вычисляется SHA256-сумма, вместе с остальными данными о документе заносящаяся в таблицу программы;
* Хэш выбранного документа вместе с текущей датой отправляется программой на сервер;
* Сервер получает хэш и дату и, в зависимости от выбранного пользователем действия, либо записывает их в базу контракта, либо ищет хэш в базе и выводит дату записи в базу, если он найден (функция записи отсутствует в версии Light программы);
* Сервер передает выходные данные в программу;
* Программа выводит результаты проведённых операций в таблицу и поле «Статус».

### Описание функционирования программы

* + - 1. **Клиентская часть**

Клиентская часть программы целиком написана на языке программирования C# и включает в себя три класса: MainForm, ServerConnection и Program. Графический интерфейс разработан с помощью технологии Windows Forms (WinForms). Версия Light клиента была разработана в целях возможности её передачи лицам, не имеющим права записывать собственные документы в базу без должных на то полномочий.

* + - * 1. **Класс 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.
  + - * 1. **Класс ServerConnection**

Статический класс, предоставляющий методы коммуникации с удалённым сервером.

* + - * 1. **Класс Program**

Статический класс, содержащий единственный метод Main(), запускающий программу.

* В комплект программы также входит конфигурационный файл (config.ini), автоматически создающийся программой при его отсутствии и содержащий IP-адрес сервера, содержащего серверную часть программы.
  + - 1. **Серверная часть**

Серверная часть программы основана на стеке технологий и языков программирования, подробно описанном в пункте 3.4 данного документа. Связь с клиентской частью осуществляется посредством скриптов на языке PHP; необходимые вычисления и работа умных контрактов производятся с помощью платформы Ethereum, Ethereum-клиента testrpc, эмулирующего блокчейн, фреймворка Truffle, предназначенного для разработки и тестирования смарт-контрактов, языков программирования Solidity и JavaScript [2].

## 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].

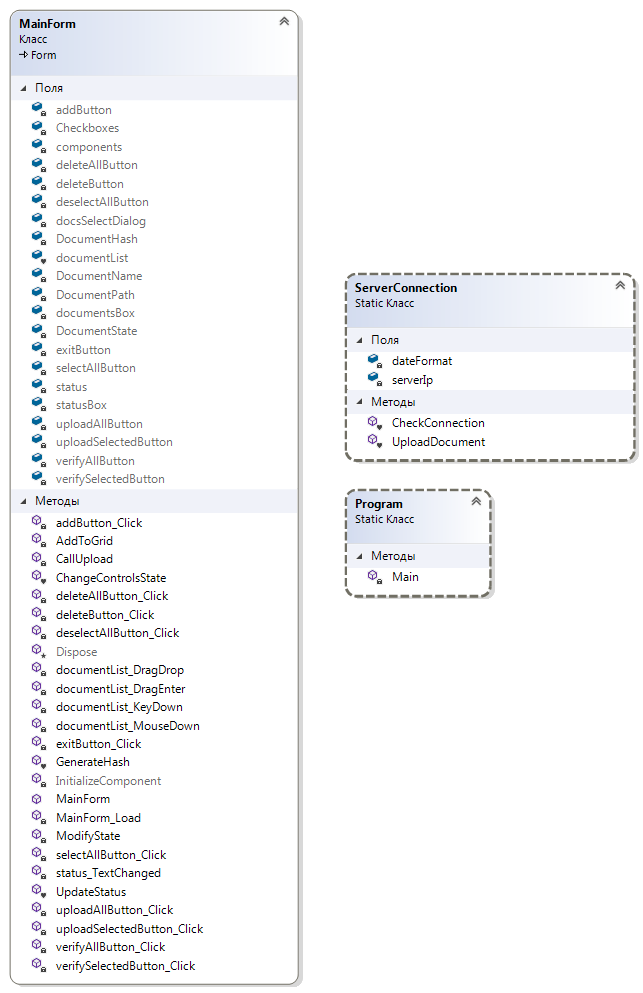
# ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

* 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. SHA1 vs SHA256 – KeyCDN Support [Electronic resource]. // KeyCDN Support [Official website]. URL: <https://www.keycdn.com/support/sha1-vs-sha256/> (accessed: 19.05.2017)
2. The Hitchhiker’s Guide to Smart Contracts in Ethereum [Electronic resource]. // OpenZeppelin – Medium [Official website]. URL: <https://medium.com/zeppelin-blog/the-hitchhikers-guide-to-smart-contracts-in-ethereum-848f08001f05> (accessed: 19.05.2017)
3. The 8 most popular document formats on the web [Electronic resource]. // Duff Johnson Strategy & Communications. [Official website]. URL: <http://duff-johnson.com/2014/02/17/the-8-most-popular-document-formats-on-the-web/> (accessed: 19.05.2017).
4. [Russian] Руководство по Solidity [Электронный ресурс]. // Ethereum Wiki [Официальный сайт]. URL: <https://github.com/ethereum/wiki/wiki/%5BRussian%5D-%D0%A0%D1%83%D0%BA%D0%BE%D0%B2%D0%BE%D0%B4%D1%81%D1%82%D0%B2%D0%BE-%D0%BF%D0%BE-Solidity> (дата обращения: 19.05.2017)

# Приложение 1

**Диаграмма классов**

**Примечание: в приложениях 1, 2 и 3 рассматривается только клиентская часть программы.**

Рис. 1. Диаграмма классов

# Приложение 2

**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 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |