**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Научный руководитель  доцент департамента  программной инженерии  факультета компьютерных наук,  канд. техн. наук  Родригес Залепинос Р.А.  **«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  2018 г. | УТВЕРЖДЕНО  Академический руководитель  образовательной программы  «Программная инженерия»  профессор департамента программной инженерии, канд. техн. наук  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** В.В. Шилов  **«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  2018 г. |

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

**ПРОГРАММА ОБНАРУЖЕНИЯ ИЗМЕНЕНИЙ ЗЕМЛЕПОЛЬЗОВАНИЯ ПО МУЛЬТИСЕНОСОРНЫМ СПУТНИКОВЫМ ДАННЫМ**

**Пояснительная записка**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.503390-01 81 01-1-ЛУ**

|  |  |
| --- | --- |
|  | Исполнитель:  студент группы БПИ152  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** /А.А. Лукин/  **«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  2018 г. |

УТВЕРЖДЕН

RU.17701729.503390-01 81 01-1-ЛУ

**ПРОГРАММА ОБНАРУЖЕНИЯ ИЗМЕНЕНИЙ ЗЕМЛЕПОЛЬЗОВАНИЯ ПО МУЛЬТИСЕНСОРНЫМ СПУТНИКОВЫМ ДАННЫМ**

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

**Пояснительная записка**

**RU.17701729.503390-01 81 01-1**

**Листов 33**

# **АННОТАЦИЯ**

В данном документе приведена пояснительная записка для «Программы обнаружения изменений землепользования по мультисенсорным спутниковым данным». Данная программа предназначена для обнаружения изменений землепользования по спутниковым снимкам.

Оформление программного документа «Пояснительная записка» произведено по требованиям ГОСТ 19.404-79 «Пояснительная записка. Требования к содержанию и оформлению» [1].

**Содержание**

[Аннотация 2](#_Toc482558782)

[1. Введение 4](#_Toc482558783)

[1.1. Наименование разработки 4](#_Toc482558784)

[1.2. Документы, на основании которых ведется разработка 4](#_Toc482558785)

[2. Назначение и область применения 5](#_Toc482558786)

[2.1. Назначение программного модуля 5](#_Toc482558787)

[2.2. Область применения программы 5](#_Toc482558788)

[2.3. Место подсистемы как компоненты сервиса 5](#_Toc482558789)

[3. Технически характеристики 7](#_Toc482558790)

[3.1. Постановка задачи для разработки программы 7](#_Toc482558791)

[3.2. Описание алгоритмов и функционирования подсистемы 7](#_Toc482558792)

[3.2.1. Генерация умных контрактов 7](#_Toc482558793)

[3.2.2. Компиляция сгенерированного умного контракта и его публикация в сеть Ethereum 24](#_Toc482558794)

[3.3. Описание и обоснование выбора метода организации входных и выходных данных 25](#_Toc482558795)

[3.3.1. Описание метода организации входных и выходных данных 25](#_Toc482558796)

[3.4. Описание и обоснование выбора состава технических и программных средств 25](#_Toc482558797)

[3.4.1. Состав технических и программных средств 25](#_Toc482558798)

[3.4.2. Обоснование выбора технических и программных средств 26](#_Toc482558799)

[4. Ожидаемые технико-экономические показатели 27](#_Toc482558800)

[4.1. Предполагаемая потребность 27](#_Toc482558801)

[4.2. Ориентировочная экономическая эффективность 27](#_Toc482558802)

[5. Список использованной литературы 28](#_Toc482558803)

[Приложение 1 Терминология 30](#_Toc482558804)

[Приложение 2 Описание и функциональное назначение файлов 31](#_Toc482558805)

[Приложение 2 Описание и функциональное назначение переменных и методов 32](#_Toc482558806)

# **Введение**

## **1.1. Наименование разработки**

Наименование программного продукта – «Программа обнаружения изменений землепользования по мультисенсорным спутниковым данным».

Наименование программного продукта на английском языке – «Program for Land-Use Change Detection using Multisensor Satellite Data».

**1.2. Документы, на основании которых ведется разработка**

Разработка программы ведется на основании приказа Национального исследовательского университета «Высшая Школа Экономики» № 2.3-02/1804-01 «Об изменении тем, руководителей курсовых работ студентов образовательной программы Программная инженерия факультета компьютерных наук» от 18.04.2017.

# **Назначение и область применения**

## **2.1. Назначение программного модуля**

Программа предназначена для детектирования изменений землепользования на основе мультисенсорных спутниковых данных, предоставляемых спутниками программы Sentinel 2 Европейского космического агентства.

Виды землепользования должны быть представлены следующими классами:

1. Водные ресурсы.
2. Пахотные земли и пастбища.
3. Застроенные области.
4. Публикация авторами работы для оценивания
5. Лесные массивы.

Программа должна визуализировать изменения землепользования по всем классам в совокупности с указанием видом изменения землепользования, а также по изменению каждого класса индивидуально.

Программа предоставляет возможность сохранения результатов обработки данных в виде гео-векторных фалов и отчета в формате pdf.

## **2.2. Область применения программы**

Программный продукт может применяться учеными в сферах экологии и урбанистики, а также в области учета землепользования государственными органами, занимающимися контролем пользования земельными участками.

# **Технические характеристики**

## **3.1. Постановка задачи для разработки программы**

Задачей разработки программы является создание программного модуля для генерации умных контрактов для организации экспертной деятельности. Генерируемые умные контракты должны обеспечивать выполнение следующих функций приложения:

1. Публикация авторами работы для оценивания
2. Добавление рецензий экспертами
3. Просмотр выложенных авторами работ
4. Просмотр выложенных рецензий на определенную работу
5. Просмотр всех выложенных рецензий определенного эксперта
6. Управление доступом пользователей к функциям приложения
7. Получение вознаграждения экспертами за публикацию отзыва

Программный модуль должен демонстрировать пользователю сгенерированный контракт и опубликовать его в блокчейн сети Ethereum с помощью Ethereum-кошелька пользователя.

## **3.2. Описание алгоритмов и функционирования подсистемы**

### **3.2.1. Генерация умных контрактов**

Генерация умных контрактов происходит при помощи POST запроса со страницы сервиса организации экспертной деятельности, который содержит в себе необходимую для генерации умного контракта информацию, которая включает в себя следующие данные:

* Список адресов учетных записей Ethereum кошельков, чьи владельцы могут отправлять работы на оценку, в случае ограничения данного круга лиц, иначе данное значение должно быть не определено. Список адресов учетных записей в случае ограничения можно в дальнейшем изменить.
* Список адресов учетных записей Ethereum кошельков, чьи владельцы являются экспертами данной «площадки», в случае ограничения данного круга лиц, иначе данное значение должно быть не определено. Список адресов учетных записей в случае ограничения можно в дальнейшем изменить
* Список адресов учетных записей Ethereum кошельков, чьи владельцы могут просматривать опубликованные работы и оценки, в случае ограничения данного круга лиц, иначе данное значение должно быть не определено. Список адресов учетных записей в случае ограничения можно в дальнейшем изменить
* Список параметров оценивания должен содержать хотя бы один параметр. Объекты параметров должны содержать два поля: тип и уникальное имя. Допустимые типы параметров – строка или целое число.
* Флаг выборочного тестирования, для определения варианта, когда эксперты могут оценивать только некоторые опубликованные работы авторов. Данный вариант возможен только при условии, что круг лиц авторов и экспертов ограничен.
* Значение вознаграждения, получаемого экспертами за работу, определяется целым числом, соответствующему сумме вознаграждения автора в Wei (минимальная дробная часть основной валюты Ethereum Ether, 1 Ether = 1018 Wei). В дальнейшем данное значение можно изменить. Данный вариант возможен только при условии, что круг лиц авторов и экспертов ограничен. Иначе данное значение должно быть не определено.

Генератор умных контракта является модулем Node.js сервера. Генерация умного контракта производиться путем его составления из шаблонов функций контракта, в зависимости от входных данных генератора.

В качестве языка, на котором написан умный контракт, выбран язык умных контрактов Ethereum Solidity, так как данный язык является наиболее популярным и поддерживаемым сообществом языком умных контрактов Ethereum.

Работы авторов идентифицируются адресом учетной записи автора, именем проектов и результатом хэш-функции SHA-256 на файлы проекта авторов.

В структуру отзыва эксперта входят адрес эксперта и автора работы, над которым производиться экспертиза, соответствующий ей имя и результат хэш-функции, а также значения параметров, указанных администратором «площадки» при публикации умного контракта в сеть Ethereum.

На рис.2 представлена блок-схема алгоритма генерации умных контрактов.

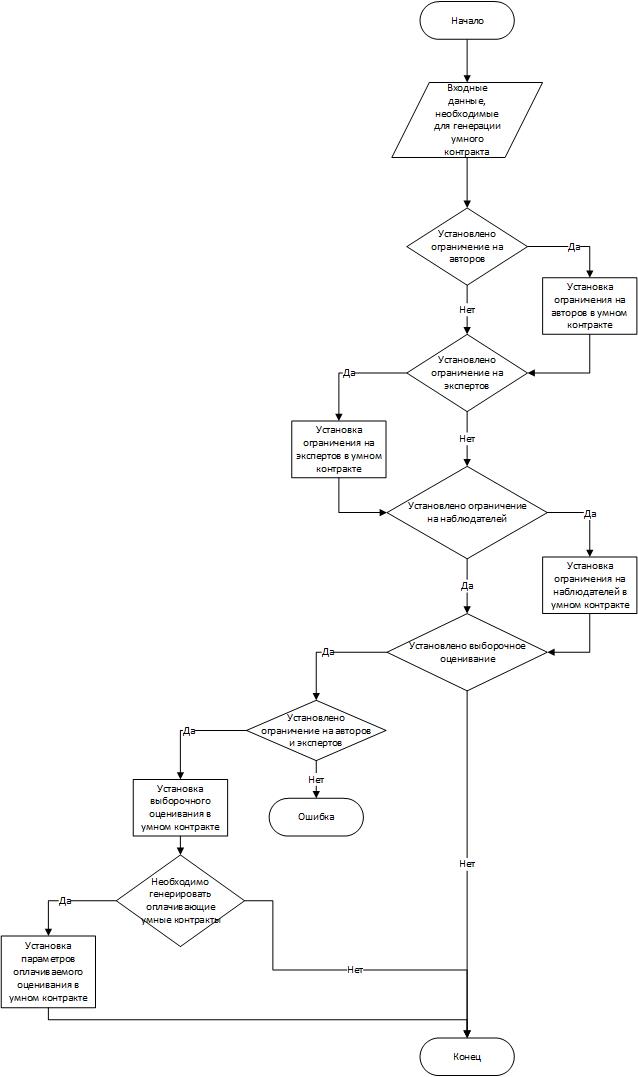


Рис. 2 Блок-схема алгоритма генерации умных контрактов

#### **3.2.1.1. Умный контракт без ограничений**

Ниже представлен вариант текста умного контракта, генерируемого подсистемой, в котором не установлены ограничения на лиц, способный отправлять проекту на оценку, проводить экспертизу и просматривать опубликованные проекты и результаты их оценок с единственным критерием оценки – целочисленным значением:

*contract ExpertActivityOrganisationService*

*{*

*// Структура проектов*

*struct Project*

*{*

*bytes32 workHash;*

*string name;*

*}*

*// Отображение адреса учетной записи автора в массив проектов*

*mapping(address => Project[]) public projects;*

*// Проверка существование проекта*

*function checkProject(address \_author, string \_projectName, bytes32 \_projectHash) private returns (bool)*

*{*

*if (projects[\_author].length == 0)*

*return false;*

*Project[] currentProjects = projects[\_author];*

*for (uint i = 0; i < currentProjects.length; i++)*

*if (currentProjects[i].workHash == \_projectHash && sha3(currentProjects[i].name) == sha3(\_projectName))*

*return true;*

*return false;*

*}*

*// Публикация прокта для оценки*

*function addProject(bytes32 \_projectHash, string \_projectName) returns (bool)*

*{*

*// Checking for resending of project*

*if (checkProject(msg.sender, \_projectName, \_projectHash))*

*return false;*

*projects[msg.sender].push(Project(\_projectHash, \_projectName));*

*return true;*

*}*

*// Структура отзыва*

*struct Review*

*{*

*address reviewer;*

*address author;*

*int mark;*

*bytes32 workHash;*

*string name;*

*}*

*// Отображение адреса автора в отзывы*

*mapping (address => Review[]) public reviewsByAuthor;*

*// Отображение адреса эксперта в отзывы*

*mapping (address => Review[]) public reviewsByReviewer;*

*// Публикация отзыва*

*function addReview(string CommentOne, int mark, address \_author, string \_projectName, bytes32 \_projectHash) returns (bool)*

*{*

*if (checkProject(\_author, \_projectName, \_projectHash)){*

*reviewsByAuthor[\_author].push(Review(msg.sender, \_author, CommentOne, mark, \_projectHash, \_projectName));*

*reviewsByReviewer[msg.sender].push(Review(msg.sender, \_author, CommentOne, mark, \_projectHash, \_projectName));*

*return true;*

*}*

*retrun false;*

*}*

*}*

Согласно приведенному выше примеру сгенерированного умного контракта, рассчитаем порядок функций добавления работы на оценку и результата оценивания в умный контракт. Порядок данных функций зависит от порядка функции проверки публикации проекта автора checkProject.

Согласно примеру сгенерированного умного контракта, порядок функции проверки публикации проектов равен *O(Number of Sent Projects)*, где *Number of Sent Projects* – количество отправленных определенным автором работ, т.к. согласно «Yellow Paper»[19] и п.3.2.1.8, все операции требуют фиксированного количества gas для выполнения и функция имеет единственный цикл, зависящий от количества отправленных определенным автором работ.

Функция публикации проектов авторов для оценивания *addProject* поддерживает уникальность опубликованных проектов с помощью функции *checkProject*, а операции получения значения из отображения и добавления элемента в конец массива требуют фиксированного количества gas согласно «Yellow Paper»[19], следовательно, порядок функции публикации проектов авторов равен *O(Number of Sent Projects)*, где *Number of Sent Projects* – количество выложенных работ автором, производящим транзакцию публикации работы в умном контракте, следовательно стоимость транзакции отправления работы на проверку зависит от количества добавленных выполняющему транзакцию автором работ.

Аналогично, порядок функции добавления отзыва экспертом *addReview* зависит от порядка функции проверки существования проекта *checkProject*, следовательно максимальная сложность данной функции равен *O(Number of Sent Projects)*, где *Number of Sent Projects* – число выложенных проектов автором, чей адрес учетной записи кошелька Ethereum соответствует адресу, указанному экспертом при проведении транзакции публикации отзыва, следовательно, максимальное количество gas, которое требуется для выполнения транзакцию зависит от количества выложенных автором проектов.

#### **3.2.1.2. Умный контракт с ограничением автором проектов**

Ниже представлен вариант текста умного контракта, генерируемого подсистемой, в котором установлено ограничение на круг лиц, способных отправлять проекту на оценку, а круг лиц, способных проводить экспертизу и просматривать опубликованные проекты и результаты их оценок не ограничен. Контракту соответствует единственный критерии оценивания – целочисленное значение:

*contract ExpertActivityOrganisationService*

*{*

*address private owner; // Владелец контракта*

*function ExpertActivityOrganisationService() // Конструктор контракта*

*{*

*owner = msg.sender;*

*authors[0xDd5dA50721Ba3C4e26Ada7AAA73cF9c6c26d9a86] = true;*

*}*

*struct Project // Струтура проектов*

*{*

*bytes32 workHash;*

*string name;*

*}*

*// Отображение адреса в массив проектов*

*mapping(address => Project[]) public projects;*

*// Отображение адреса в доступ к отправлению проектов*

*mapping(address => bool) private authors;*

*// Провека на возможноть добаления проектов*

*function checkAuthor(address \_author) private returns (bool)*

*{*

*return authors[\_author];*

*}*

*// Предоставление доступа новому автору*

*function addAuthor(address \_author) public returns (bool)*

*{*

*if (msg.sender != owner)*

*return false;*

*authors[\_author] = true;*

*return true;*

*}*

*// Запрет автору добавлять работы*

*function removeAuthor(address \_author) public returns (bool)*

*)*

*{*

*if (msg.sender != owner)*

*return false;*

*authors[\_author] = false;*

*return true;*

*}*

*// Проверка сущестования проекта*

*function checkProject(address \_author, string \_projectName, bytes32 \_projectHash) private returns (bool)*

*{*

*if (projects[\_author].length == 0)*

*return false;*

*Project[] currentProjects = projects[\_author];*

*for (uint i = 0; i < currentProjects.length; i++)*

*if (currentProjects[i].workHash == \_projectHash && sha3(currentProjects[i].name) == sha3(\_projectName))*

*return true;*

*return false;*

*}*

*// Публикация нового прокта*

*function addProject(bytes32 \_projectHash, string \_projectName) public returns (bool)*

*{*

*if (!checkAuthor(msg.sender)) return false;*

*if (checkProject(msg.sender, \_projectName, \_projectHash))*

*return false;*

*projects[msg.sender].push(Project(\_projectHash, \_projectName));*

*return true;*

*}*

*// Структура оценки*

*struct Review*

*{*

*address reviewer;*

*address author;*

*int mark;*

*bytes32 workHash;*

*string name;*

*}*

*// Отображение адреса автора в отзывы*

*mapping (address => Review[]) public reviewsByAuthor;*

*// Отображение адреса эксперта в отзывы*

*mapping (address => Review[]) public reviewsByReviewer;*

*// Публикация отзыва*

*function addReview(int mark, address \_author, string \_projectName, bytes32 \_projectHash) public returns (bool)*

*{*

*if (checkProject(\_author, \_projectName, \_projectHash)){*

*reviewsByAuthor[\_author].push(Review(msg.sender, \_author, mark, \_projectHash, \_projectName));*

*reviewsByReviewer[msg.sender].push(Review(msg.sender, \_author, mark, \_projectHash, \_projectName));*

*return true;*

*}*

*return false;*

*}*

*}*

Согласно приведенному выше примеру сгенерированного умного контракта, а также «Yellow Paper»[19] порядок конструктора умного контракта ExpertActivityOrganisationService, функций добавления addAuthor и удаления removeAuthor авторов работ, а также функция проверки доступа автора к публикации проектов checkAuthor является константой, следовательно, требуют фиксированной комиссии за выполнение транзакций. Данные функции доступны исключительно владельцу умного контракта.

Функция проверки публикации проекта checkProject и добавление отзыва addReview аналогичны случаю генерации умных контрактов без ограничений и имеют максимальную сложность *O(Number of Sent Projects)*, где *Number of Sent Projects* – число выложенных проектов автором.

Функция добавления проектов авторами *addProject* зависит от функций проверки существования проекта *checkProject* и функции проверки доступа к публикации проектов *checkAuthor,* следовательно, порядок добавления проектов авторами равен *O(Number of Sent Projects)*, где *Number of Sent Projects* – число выложенных проектов автором, выполняющему транзакцию добавления проекта. Это означает, что стоимость выполнения функции добавление проекта, при ограничении на круг лиц, имеющим доступ к выполнению данной функции, на всемирной виртуальной машине Ethereum зависит от количества выложенных автором работ.

#### **3.2.1.3. Умный контракт с ограничением круга экспертов**

Ниже представлен вариант текста умного контракта, генерируемого подсистемой, в котором установлено ограничение на адреса экспертов, а круг лиц, способных отправлять проекты на оценивание и просматривать опубликованные проекты и результаты их оценок не ограничен. Контракту соответствует единственный критерии оценивания – целочисленное значение:

*contract ExpertActivityOrganisationService*

*{*

*address private owner; // Адрес владельца контракта*

*// Конструктор умного контракта*

*function ExpertActivityOrganisationService()*

*{*

*owner = msg.sender;*

*reviewers[0xDd5dA50721Ba3C4e26Ada7AAA73cF9c6c26d9a86] = true;*

*}*

*// Структура проектов*

*struct Project*

*{*

*bytes32 workHash;*

*string name;*

*}*

*// Отображение адреса в массив проектов*

*mapping(address => Project[]) public projects;*

*// Проверка существования проекта*

*function checkProject(address \_author, string \_projectName, bytes32 \_projectHash) private returns (bool)*

*{*

*if (projects[\_author].length == 0)*

*return false;*

*Project[] currentProjects = projects[\_author];*

*for (uint i = 0; i < currentProjects.length; i++)*

*if (currentProjects[i].workHash == \_projectHash && sha3(currentProjects[i].name) == sha3(\_projectName))*

*return true;*

*return false;*

*}*

*// Добавление нового проекта*

*function addProject(bytes32 \_projectHash, string \_projectName) public returns (bool)*

*{*

*// Checking for resending of project*

*if (checkProject(msg.sender, \_projectName, \_projectHash))*

*return false;*

*projects[msg.sender].push(Project(\_projectHash, \_projectName));*

*return true;*

*}*

*// Стурктора отзыва*

*struct Review*

*{*

*address reviewer;*

*address author;*

*int mark;*

*bytes32 workHash;*

*string name;*

*}*

*// Отображение адреса автора в отзывы*

*mapping (address => Review[]) public reviewsByAuthor;*

*// Отображение адреса эксперта в отзывы*

*mapping (address => Review[]) public reviewsByReviewer;*

*// Отображение адреса к флагу доступа к публикации отзывов*

*mapping(address => bool) private reviewers;*

*// Проверка доступа эксперта*

*function checkReviewer(address \_reviewer ) private returns (bool)*

*{*

*return reviewers[\_reviewer];*

*}*

*// Добаление нового эксперта*

*function addReviewer(address \_reviewer)* *public returns (bool)*

*{*

*// Checking for caller*

*if (msg.sender != owner)*

*return false;*

*reviewers[\_reviewer] = true;*

*return true;*

*}*

*// Ограничение доступа эксперта к публикации отзывов*

*function removeReviewer(address \_reviewer)* *public returns (bool)*

*{*

*if (msg.sender != owner)*

*return false;*

*reviewers[\_reviewer] = false;*

*return true;*

*}*

*// Добавление отзыва*

*function addReview(int mark, address \_author, string \_projectName, bytes32 \_projectHash)* public returns (bool)

*{*

*if (!checkReviewer(msg.sender )) throw;*

*if (checkProject(\_author, \_projectName, \_projectHash)){*

*reviewsByAuthor[\_author].push(Review(msg.sender, \_author, mark, \_projectHash, \_projectName));*

*reviewsByReviewer[msg.sender].push(Review(msg.sender, \_author, mark, \_projectHash, \_projectName));*

*return true;*

*}*

*return false;*

*}*

*}*

Согласно приведенному выше сгенерированному умному контракту, конструктор умного контракта *ExpertActivityOrganisationService*, метод проверки существования проекта *checkProject* и метод добавление нового проекта имеют порядок, аналогичный варианту сгенерированного контракта без ограничений (п. 3.2.1.1.).

Методы добавление и удаления доступа эксперта к публикации отзывов, *addReviewer* и *removeReviewer* соответственно, принимающие в качестве параметра адрес учетной записи кошелька Ethereum эксперта, имеют порядок, аналогичный методам добавления и удаления авторов в умном контракте с ограничением на авторов (п.3.2.1.2). Доступ к данным функциям имеет только владелец контракта – администратор площадки.

Т.к. максимальная сложность метода проверки существования проекта зависит от количества выложенных автором работ, а функция проверки доступа эксперта – константа, то максимальная сложность функции публикации отзыва равна *О(Number of Sent Projects)*, где *Number of Sent Projects* – количество работ, выложенных автором, чей проект оценивается. Следовательно, комиссия за выполнение данной функции зависит от количества работ, выложенных автором, чей проект оценивается.

#### **3.2.1.4. Умный контракт с ограничением на наблюдателей**

Ниже представлена часть варианта текста умного контракта, генерируемого подсистемой, в котором установлено ограничение на адреса наблюдателей, а круг лиц, способных отправлять проекты на оценивание и оценивать опубликованные проекты не ограничен. Контракту соответствует единственный критерии оценивания – целочисленное значение. Ниже не показаны функции, относящиеся к процессу добавления проектов и отзывов, так как они аналогичны функциям, представленным в п.3.2.1.1.

*address private owner; // Владелец умного контракта*

*// Конструктор умного контракта*

*function ExpertActivityOrganisationService()*

*{*

*owner = msg.sender;*

*viewers[0xDd5dA50721Ba3C4e26Ada7AAA73cF9c6c26d9a86] = true;*

*}*

*// Отображение адреса в флаг доступа к просмотру проектов и оценок*

*mapping(address => bool) private viewers;*

*// Проверка доступа к просмотру*

*function checkViewer(address \_viewer) private returns (bool)*

*{*

*return viewers[\_viewer];*

*}*

*// Добавление нового наблюдателя*

*function addViewer(address \_viewer)*

*{*

*if (msg.sender != owner)*

*throw;*

*viewers[\_viewer] = true;*

*}*

*// Удаление наблюдателя*

*function removeViewer(address \_viewer)*

*{*

*if (msg.sender != owner)*

*throw;*

*viewers[\_viewer] = false;*

*}*

*// Просмотр проекта*

*function showProjects(address \_author, uint i) returns (string, bytes32)*

*{*

*if(!checkViewer(msg.sender))*

*throw;*

*return (projects[\_author][i].name, projects[\_author][i].workHash);*

*}*

*// Просмотр рецензий по адресу автора проектов*

*function showReviewsByAuthor(address \_author, uint i) returns (address, int, string, bytes32)*

*{*

*if(!checkViewer(msg.sender))*

*throw;*

*Review curReview = reviewsByAuthor[\_author][i];*

*return (curReview.reviewer, curReview.mark, curReview.name, curReview.workHash);*

*}*

*// Просмотр рецензий по адресу эксперта*

*function showReviewsByReviewer(address \_reviewer, uint i) returns (address, int, string, bytes32)*

*{*

*if (!checkViewer(msg.sender))*

*throw;*

*Review curReview = reviewsByReviewer[\_reviewer][i];*

*return (curReview.author, curReview.mark, curReview.name, curReview.workHash);*

*}*

Согласно приведенному выше тексту умного контракта, порядок конструктора умного контракта *ExpertActivityOrganisationService*, функции проверки доступа наблюдателя *checkViewer*, добавления и удаление доступа к просмотру (*addViewer* и *removeViewer*) является константой, следовательно, требуют фиксированного количества gas.

Функции демонстрации проектов и оценок возвращают кортежи с данными проектов и оценок, вызов данных функций из внешнего интерфейса Dapp имеет порядок *O(Number\_of\_Data), Number\_of\_Data –* количество работ или оценок в соответствующем отображении.

#### **3.2.1.5. Умный контракт с выборочным оцениванием проектов**

Ниже представлена часть варианта текста умного контракта, генерируемого подсистемой, в котором установлено ограничение на оценивание работ, каждому эксперту могут назначаться проекты, которые они могут оценивать. Для контроля учетных записей, данная функция возможна только при условии ограничения адресов авторов проектов и экспертов. Контракту соответствует единственный критерии оценивания – целочисленное значение. Ниже не показаны функции, относящиеся к процессу добавления проектов, так как они аналогичны функциям, представленным в п.3.2.1.2.

*// Отображение адреса эксперта в отображение адрессов авторов, чьи работы нужно оценить, в массив проектов для оценки*

*mapping(address => mapping(address => Project[])) private selectedReviewingMap;*

*// Функция проверки доступа эксперта, а так же проверки назначения опредленной работы для экспертизы автором*

*function* *checkReviewer(address \_reviewer , address \_author, string \_projectName, bytes32 \_projectHash, bool flag) private returns (bool)*

*{*

*if (!reviewers[\_reviewer])*

*return false;*

*Project[] curProjects = selectedReviewingMap[\_reviewer][\_author];*

*if (curProjects.length == 0)*

*return false;*

*for (uint i = 0; i < curProjects.length; i++)*

*if (curProjects[i].workHash == \_projectHash && sha3(curProjects[i].name) == sha3(\_projectName))*

*{*

*if (flag)*

*{*

*uint last = curProjects.length - 1;*

*if (i == last)*

*{*

*delete curProjects[last];*

*curProjects.length--;*

*}*

*else*

*{*

*curProjects[i].workHash = curProjects[last].workHash;*

*curProjects[i].name = curProjects[last].name;*

*delete curProjects[last];*

*curProjects.length--;*

*}*

*}*

*return true;*

*}*

*return false;*

*}*

*// Назанчение проекта эксперту на оценивания*

*function setProjectToReview(address \_reviewer, address \_author, string \_projectName, bytes32 \_projectHash)* public returns (bool)

*{*

*if (msg.sender != owner)*

*return false;;*

*if (!checkProject(\_author, \_projectName, \_projectHash))*

*return false;;*

*if (checkReviewer(\_reviewer, \_author, \_projectName, \_projectHash, false))*

*return false;*

*selectedReviewingMap[\_reviewer][\_author].push(Project(\_projectHash, \_projectName));*

*return true;*

*}*

*// Добавление отзыва*

*function addReview(int mark, address \_author, string \_projectName, bytes32 \_projectHash)* public returns (bool)

*{*

*if (!checkReviewer(msg.sender , \_author, \_projectName, \_projectHash, true)) throw;*

*if (checkProject(\_author, \_projectName, \_projectHash)){*

*reviewsByAuthor[\_author].push(Review(msg.sender, \_author, mark, \_projectHash, \_projectName));*

*reviewsByReviewer[msg.sender].push(Review(msg.sender, \_author, mark, \_projectHash, \_projectName));*

*return true;*

*}*

*return false;*

*}*

Согласно приведенному выше тексту умного контракта, порядок функции добавление существующего проекта на процедуру оценивания определенному эксперту *setProjectToReview* является зависит от функции проверки публикации проекта checkProject, следовательно порядок функции *setProjectToReview* равен *О(Number of Sent Projects)*, где *Number of Sent Projects* – количество работ, выложенных автором, чей проект должен будет оценивается. Выполнения данной функции доступно только владельцу контракта (администратору площадки).

Функция проверки доступа эксперта, а также проверки назначения определенной работы для экспертизы автором *checkReviewer* имеет сложность *О(Number of Sent Projects)*, где *Number of Sent Projects* – количество работ, выложенных автором, чей проект должен будет оценивается, в случае проверки без удаления проектов из списка оценивания для эксперта. В случае удаления, в массиве не происходит сдвига элементов, только замена текущего проекта в массиве на последний (или удаление единственного элемента массива) на последний в массиве, что позволяет получить фиксированную стоимость транзакции, следовательно порядок функции *checkReviewer* равен *О(Number of Sent Projects)*, где *Number of Sent Projects* – количество назначенных работ на экспертизу определенного автора, чей проект оценивается.

Стоимость функции добавления отзыва *addReview* зависит от функций проверки автора и проверки существования проекта, следовательно, количество gas, требующихся на выполнение данной функции зависит от количества работ, выложенных автором, чей проект оценивается. Данная транзакция доступна только владельцем кошельков, чьи адреса добавлены владельцем контракта в список экспертов.

#### **3.2.1.6. Умный контракт, оплачивающий работу экспертов**

Так как все умные контракты Ethereum являются наследниками типа адреса, то как и любой аккаунт Ethereum наследуют от него баланс. Для оплаты деятельности экспертов, администратору площадки необходимо поддерживать положительный баланс умного контракта. Для генерации такого типа умного контракта, необходимо установить лимит на авторов и экспертов, и назначить администрируемое назначение проектов к оцениванию, так как только при таких условиях у экспертов не будет возможности «выкачивать» криптовалюту из умного контракта. Ниже представлена часть варианта текста умного контракта, генерируемого подсистемой, который оплачивает работу экспертов. Контракту соответствует единственный критерии оценивания – целочисленное значение. Ниже не показаны функции, относящиеся к процессу добавления проектов и организации выборочной экспертизы, так как они аналогичны функциям, представленным в п.3.2.1.5.

// Назначенная сумма вознаграждения

uint reward;

// Пополнить баланс умного контракта

function () payable public returns (bool) {

if (msg.sender != owner) return false;

this.transfer(msg.value);

return true;

}

// Вывести деньги из умного контракта

function refund() payable {

if (msg.sender != owner) throw;

owner.transfer(this.balance);

}

// Изменить сумму вознаграждения

function changeReward(uint \_reward) public returns (bool){

if (msg.sender != owner) return false;

reward = \_reward;

return true;

}

// Конструктор умного контракта

function ExpertActivityOrganisationService()

{

owner = msg.sender;

reward = 5000000;

}

// Добавление отзыва

function addReview(int mark, address \_author, string \_projectName, bytes32 \_projectHash) payable public returns (bool)

{

if (this.balance < reward) return false;

if (!checkReviewer(msg.sender , \_author, \_projectName, \_projectHash, true)) throw;

if (checkProject(\_author, \_projectName, \_projectHash)){

msg.sender.transfer(reward);

reviewsByAuthor[\_author].push(Review(msg.sender, \_author, mark, \_projectHash, \_projectName));

reviewsByReviewer[msg.sender].push(Review(msg.sender, \_author, mark, \_projectHash, \_projectName));

return true;

}

return false;

}

Согласно приведенному выше тексту умного контракта, функции пополнения баланса умного контракта (функция без имени), функция снятия средств *refund* и изменения размера вознаграждения экспертов *changeReward* требуют фиксированного количества gas. Данные функции доступны исключительно владельцу умного контракта.

Перевод средств на счет эксперта не влияет на порядок функции отправки рецензии, следовательно, порядок функции *addReview* соответствует порядку этой функции в случае, описанном в п. 3.2.1.7

***3.2.1.7. Умные контракты с 2-мя и 3-мя ограничениями***

Подсистема создания умных контрактов сервиса организации экспертной деятельности позволяет создавать умные контракты с несколькими ограничениями, например, с ограничением на авторов и экспертов, авторов и наблюдателей и т.д. Так как методы умного контракта при различных ограничениях не влияют на работу друг друга, то порядок функций, отвечающих за реализацию соответствующего ограничения, соответствуют порядку функций с единственным соответствующим ограничением.

#### **3.2.1.8. Сравнение строк в умном контракте с помощью хэш-функции SHA-3**

В связи с тем, что в языке умных контрактов Ethereum Solidity не обладает средствами сравнения строк, а для различия работ нам необходимо сравнивать строковые типы данных, для этой цели используется хэш-функция SHA-3 использующей алгоритм Keccak-256[16][20]. Данная хэш-функция выбрана, так как она требует меньшее количество gas для выполнения согласно «Yellow Paper»[19]: 30 gas. Для сравнения, две другие доступные хэш-функции в умных контрактах являются более дорогими: SHA-256[21] – 60 gas, RIPEMD-160[22] – 600 gas, согласно Yellow Paper[19].

### **3.2.2. Компиляция сгенерированного умного контракта и его публикация в сеть Ethereum**

#### **3.2.2.1. Компиляция сгенерированного умного контракта**

В последнем критическом обновлении Geth, предоставляющим связь умных кошельков с сетью блокчейн и объект Web3, была исключена возможность компиляции умных контрактов, описанная в официальном Ethereum Javascript API[15]. В связи с чем, для компиляции сгенерированных умных контрактов используется компилятор Solc[16], рекомендуемый официальным порталом Solidity. Компилятор умного контракта представляет собой модуль Node.js сервера, который возвращает на клиентскую сторону приложения байт-код и интерфейс скомпилированного контракта для его публикации в блокчейн сеть Ethereum с помощью кошелька.

#### **3.2.2.2. Встраивание умного контракта в блокчейн сеть Ethereum**

Скомпилированные умные контракты встраиваются в сеть блокчейн Ethereum с помощью средств объекта web3, предоставляемого кошельком Ethereum. Для успешного встраивания умного контракта, необходима оценка предельного количества gas для текущего умного контракта. Данное оценивание происходит с помощью предоставляемого объекта и подсчета элементарных операций умного контракта web3, имеющее средства для оценки предельного количества gas. Для успешной оценки, данному методу нужно передать байт-код сгенерированного контракта, и после эмуляции функций контракта, данная функция возвращает оценку limit gas. Однако, при условии, что умный контракт будет хранить огромное количество работ и отзывов, владелец умного контракта может увеличить данное значение. Тогда для выполнения умного контракта, пользователям необходимо иметь на лицевом счете необходимое количество крипто-валюты для покупки максимального количества gas. Данное значение рассчитывается следующим образом:

Transaction Cost – цена выполнения транзакции,

Gas Limit – максимальное количества gas, необходимое для выполнения транзакции

Gas Price – цена 1 gas, определяется майнерами блокчейн сети Ethereum путем голосования

Таким образом, для выполнения транзакций пользователем, им необходимо иметь сумму на счету, большее или равное Transaction Cost. Весь неиспользованный в ходе выполнения функций gas возвращается пользователю.

После подтверждения выполнения транзакции и майнинга блока с транзакцией публикации умного контракта, смартконтрактом можно начинать пользоваться.

## **3.3. Описание и обоснование выбора метода организации входных и выходных данных**

### **3.3.1. Описание метода организации входных и выходных данных**

Входные данные вводятся администратором площадки оценивания вручную в соответствующие элементы ввода формы:

1. Для ввода ограничений администратор должен отметить необходимые пункты на сайте вею-сервиса.
2. Для ввода адресов авторов, экспертов и наблюдателей, которым предоставляется доступ к методам умного контракта, в случае ограничения, администратор должен добавлять адреса учетных записей Ethereum.
3. В случае создания умного контракта, оплачивающего труд экспертов, необходимо указать вознаграждение в wei в соответствующее поле.

Текст сгенерированного умного контракта демонстрируется пользователю в предназначенном для этого поле. После компиляции и встраивания в блокчейн сеть Ethereum пользователю демонстрируется хэш-транзакции и адрес встроенного умного контракта.

## **3.4. Описание и обоснование выбора состава технических и программных средств**

### **3.4.1. Состав технических и программных средств**

Для работы подсистемы необходим сервер Node.js версии 6.10.2 с системными требованиями:

1. Windows:
   1. Версия Windows 7 и позднее
   2. Процессор Intel Pentium 4 или старше с поддержкой SSE2
   3. 512 Мб RAM
2. Mac
3. OS X версии 10.9
4. Процессор Intel
5. 512 Мб RAM
6. Linux
7. Ubuntu 12.04 или поздняя версия, или Fedora 21, илиDebian 8
8. Процессор Intel Pentium 4 или старше с поддержкой SSE2
9. 512 Мб RAM

Для работы на клиентской стороне требуются:

1. Операционная система Windows 7, Windows 8, Windows 8.1, Windows 10 или более поздней версии, OS X Mavericks 10.9 или более поздней версии, Ubuntu 14.04 (64-разрядная версия) или старше, Debian 8 или старше, openSUSE 13.3 или старше, Fedora Linux 24 или старше.
2. Установленный браузер Google Chrome последней версии с установленным Ethereum кошельком Metamask, связанный с публичной или приватной блокчейн сетью Ethereum

## **3.4.2.** **Обоснование выбора технических и программных средств**

В качестве серверного пакета был выбран Node.js, так как он обладает большим количеством удобных библиотек, широкими возможностями работы с объектов web3.

Для клиентской стороны требование браузера Google Chrome, так как выбранный Ethereum кошелек Metamask является расширением для данного браузера. Metamask, в свою очередь, является одним из самых используемых кошельков, обеспечивающих работу Dapp.

# **Ожидаемые технико-экономические показатели**

## **4.1. Предполагаемая потребность**

Данная программный модуль используется сервисом организации экспертной деятельности. Создаваемые умные контракты являются частью создаваемых сервисом Dapp, которые могут быть использована высшими учебными заведениями или другими образовательными учреждениями в целях оценки работ учащихся, например, курсовых работ, проектов, а также другими организациями, которым необходимо провести прозрачную оценку чего-либо.

## **4.2. Ориентировочная экономическая эффективность**

Аналогов данного продукта выявлено не было. Ключевым достоинством создаваемых умных контрактов является прозрачностью проведения экспертной оценки, а технология блокчейн обеспечивает достоверность результатов экспертиз и надежно их хранит. Так же создаваемые контракты помогают автоматизировать оплату труда экспертов.

# **Список использованной литературы**

1. ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10. ГОСТ Р 7.02-2006 Консервация документов на компакт-дисках. Общие требования. – М.: ИПК Издательство стандартов, 2006
11. ГОСТ 18300-87 Спирт этиловый ректификованный технический. Технические условия. – М.:ИПК Издательство стандартов, 1997
12. ГОСТ 9805-84. Спирт изопропиловый. Технические условия. – М.: ИПК Издательство стандартов, 1984.
13. ГОСТ 19.602-78 Правила дублирования, учета и хранения программных документов, выполненных печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001
14. Google Chrome Системные требования // Google URL: https://support.google.com/chrome/answer/95346?co=GENIE.Platform%3DDesktop&hl=ru (дата обращения: 20.04.2017).
15. Ethereum JavaScript API // Ethereum URL: https://github.com/ethereum/wiki/wiki/JavaScript-API (дата обращения: 9.05.2017)
16. Solidity - Solidity.0.4.12 documentation // Solidity URL: http://solidity.readthedocs.io (дата обращения: 9.05.2017).
17. Docs Node.js // Node.js URL: https://nodejs.org/en/docs/ (дата обращения: 9.05.2017).
18. White Paper Ethereum // Ethereum URL: https://github.com/ethereum/wiki/wiki/White-Paper (дата обращения: 9.04.2017).
19. ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER // Yellow Paper URL: yellowpaper.io (дата обращения: 14.04.2017).
20. The Keccak sponge function family // The Keccak URL: http://keccak.noekeon.org/ (дата обращения: 9.05.2017).
21. Descriptions of SHA-256, SHA-384, and SHA-512 // NIST URL: http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf (дата обращения: 9.05.2017).
22. Алгоритм RIPEMD-160 // Solutionmes URL: http://solutionmes.wikidot.com/crypto-ripemd (дата обращения: 9.05.2017).

# **Приложение 1 Терминология**

**Блокчейн (Цепочка блоков транзакций) -**  выстроенная по определённым правилам цепочка из формируемых блоков транзакций.

**Блок транзакций -** специальная структура для записи группы транзакций в системе Биткойн и аналогичных ей.

**Автор проекта** – лицо, отправляющее проект в сервис организации экспертной деятельной для рецензирования экспертами.

**Dapp –** распределенное приложение сети Ethereum, представляющее собой систему умных контрактов и пользовательского интерфейса для работы с этими контрактами.

**Ether –** крипто-валюта блокчейн сети Etherеum.

**Wei –** минимальная кратная часть Ether, 1 Ether = 1018 Wei.

**Gas –** «топливо» для выполнения транзакций в сети Ethereum, покупается пользователями при совершении транзакций.

**Gas Limit –** максимальное количество gas, которое можно купить для выполнения транзакций.

**Эксперт** - лицо, производящее рецензирование проектов.

**Площадка –** место, в рамках которой оцениваются проекты, например, университет или фонд развития инновационных проектов.

**Администратор площадки** – работник информационного отдела площадки, владеющий доступом к информационным ресурсам площадки.

# **Приложение 2 Описание и функциональное назначение файлов**

Таблица 1.1

Описание и функциональное назначение файлов

|  |  |
| --- | --- |
| **Класс** | **Функциональное назначение** |
| contractGenerator.js | Данный файл представляет собой модуль Node.js сервера, отвечающего за генерацию умных контрактов |
| contractCompiler.js | Данный файл представляет собой модуль Node.js сервера, отвечающего за компиляцию сгенерированных умных контрактов |
| frontend.js | Файл содержит обработчики событий сайта сервиса, в том числе сбор параметров и встраивание умного контракта в сеть Ethereum |

# **Приложение 2 Описание и функциональное назначение переменных и методов**

Таблица 2.1

Описание переменных и методов файла contractGenerator.js

|  |  |
| --- | --- |
| **Поля** | |
| **Название** | **Назначение** |
| contractTemplate | Скелет умного контракта |
| contractOwnerDeclaration | Шаблон декларации владельца умного контракта |
| payableFunctionsTemplate | Шаблон функций оплачиваемого контракта |
| contractDesignerTemplate | Шаблон конструктора умного контракта |
| limitedAuthorsMapping | Шаблон декларации отображения авторов |
| checkAuthorFunctionTemplate | Шаблон метода проверки автора |
| checkProjectFunctionTemplate | Шаблон метода проверки проекта |
| addAuthorFunctionTemplate | Шаблон метода добавления автора |
| removeAuthorFunctionTemplate | Шаблон метода удаления автора |
| reviewersMappingTemplate | Шаблон декларации отображения экспертов |
| selectedReviewingMapping | Шаблон декларации отображение в проекты при выборочной экспертизе |
| checkReviewerFunctionTemplate | Шаблон метода проверки эксперта |
| selectedReviewingCheckReviewerArgs | Шаблоны параметров при проверке проектов при оценке при выборочном тестировании |
| selectedReviewingCheckingCondition | Шаблон проверки эксперта при выборочном тестировании |
| notSelectedReviewingCheckingCondition | Шаблон проверки эксперта при не выборочном тестировании |
| addReviewerFunctionTemplate | Шаблон метода добавление эксперта |
| removeReviewerFunctionTemplate | Шаблон метода удаление эксперта |
| addReviewFunctionTemplate | Шаблон метода добавление отзыва |
| checkReviewerContractPart | Шаблон метода проверки эксперта |
| sendRewardTemplate | Шаблон отправки вознаграждения эксперту |
| setProjectToReviewingFunctionTemplate | Шаблон метода назначение проекта для проверки эксперту |
| viewersMappingTemplate | Шаблон декларации отображения наблюдателей |
| checkViewerFunctionTemplate | Шаблон метода проверки наблюдателя |
| addViewerFunctionTemplate | Шаблон метода добавления наблюдателя |
| removeViewerFunctionTemplate | Шаблон метода удаления наблюдателя |
| showProjectsFunctionTemplate | Шаблон метода отзывов проектов |
| showReviewsFunctionsTemplates | Шаблон методов демонстрации отзывов |
| **Методы** | |
| **Название** | **Назначение** |
| app.post | Метод, обрабатывающий POST запросы |
| generateSmartContractCode | Метод, генерирующий умный контракт |
| getShowReviewsFunctions | Метод, генерирующий функции демонстрации отзывов |
| getAddReviewFunction | Метод генерирующий функцию добавления отзыва |
| getCheckReviewerFunction | Метод генерирующий функцию добавления эксперта |
| getReviewFields | Метод установки параметров оценивая |
| getContractDesigner | Метод генерирующий конструктор умного контракта |
| readFile | Метод чтения файлов шаблонов функций |

Таблица 2.2

Описание переменных и методов файла contractCompiler.js

|  |  |
| --- | --- |
| **Методы** | |
| **Название** | **Назначение** |
| app.post | Метод, обрабатывающий POST запросы |
| compileSolidity | Метод компиляции умного контракта |

Таблица 2.3

Описание переменных и методов файла frontend.js

В данной таблице указаны метод, необходимые для работы подсистемы создания умных контрактов.

|  |  |
| --- | --- |
| **Методы** | |
| **Название** | **Назначение** |
| checkAddresses | Проверка введенных адресов учетных записей Ethereum |
| generateSmartcontract | Отправка POST запроса для генерации умного контракта согласно введенным требованиям |
| deploySmartcontract | Компиляция и встраивания умного контракта в блокчейн сеть Ethereum |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ** | | | | | | | | | |
| Изм. | Номера листов (страниц) | | | | Всего  листов  (страниц)  в докум. | № докум. | Входящий № сопроводитель-ного документа и дата | Подпись | Дата |
| изме-ненных | заме-  ненных | новых | анну-  лиро-  ванных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |