**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  **«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  2017 г. |

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

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

**Текст программы**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.16 12 01-1-ЛУ**

|  |  |
| --- | --- |
|  | Исполнитель:  студент группы БПИ153  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** /А.А. Лукин/  **«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  2018 г |
|  |  |
|  |  |

УТВЕРЖДЕН

RU.17701729.04.16 12 01-1-ЛУ

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

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

**Текст программы**

**RU.17701729.04.16 12 01-1**

**Листов 59**

# **Аннотация**

В данном программном документе приведено техническое задание для «Программы обнаружения изменений землепользования по мультисенсорным спутниковым данным». Данная программа предназначена для обнаружения изменений землепользования по спутниковым снимкам.

Оформление программного документа произведено по требованиям ГОСТ 19.401-78 «Текст программы. Требования к содержанию и оформлению» [1] .

**Содержание**

[Аннотация 2](#_Toc482788585)

[Терминология 4](#_Toc482788586)

[1. Текст программы 5](#_Toc482788587)

[1.1. app.js 5](#_Toc482788588)

[1.2. contractGenerator.js 6](#_Toc482788589)

[1.3. contractCompiler.js 25](#_Toc482788590)

[1.4. frontend.js 26](#_Toc482788591)

[1.5. index.pug 49](#_Toc482788592)

[1.6. Шаблоны для генерации контрактов 52](#_Toc482788593)

[Список использованной литературы 59](#_Toc482788594)

# **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.

# **Текст программы**

Программа состоит из 4 файлов исходного кода, а также 20 файлов для генерации умного контракта.

## app.js

const express = require('express');

const path = require('path');

const favicon = require('serve-favicon');

const logger = require('morgan');

const cookieParser = require('cookie-parser');

const bodyParser = require('body-parser');

const cg = require('./ContractGenerator/contractsGenerator.js');

const comp = require('./ContractCompiler/contractCompiler.js');

const wwg = require('./WidgetGenerator/WebWidgetGenerator.js');

const service = require('./routes/index');

const app = express();

// view engine setup

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'pug');

// uncomment after placing your favicon in /public

//app.use(favicon(path.join(\_\_dirname, 'public', 'favicon.ico')));

app.use(logger('dev'));

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

app.use(cookieParser());

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use('/service', service);

app.use('/contractGenerator', cg);

app.use('/contractCompiler', comp);

app.use('/webwidgetgenerator', wwg);

// catch 404 and forward to error handler

app.use(function(req, res, next) {

const err = new Error('Not Found');

err.status = 404;

next(err);

});

// error handler

app.use(function(err, req, res, next) {

// set locals, only providing error in development

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

// render the error page

res.status(err.status || 500);

res.render('error');

});

module.exports = app;

## contractGenerator.js

/\*\*

\* Smartcontract's code generator

\*/

const express = require('express');

const app = express();

const fs = require('fs');

/\*\*

\* File reading function

\* @param path

\*/

function readFile(path) {

return fs.readFileSync(path).toString();

}

/\*\*

\* Contract template

\*/

const contractTemplate =

readFile('./ContractGenerator/Templates/ExpertActivityOrganisationServiceContractTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Contract owner declaration string

\* @type {string}

\*/

const contractOwnerDeclaration = 'address private owner;';

/\*\*

\* Payable functions template

\*/

const payableFunctionsTemplate = readFile('./ContractGenerator/Templates/DepositeTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Contract designer template

\*/

const contractDesignerTemplate =

readFile('./ContractGenerator/Templates/ContractDesignerTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Get contract designer

\* @param limitedAuthors array of authors addresses

\* @param limitedReviewers

\* @param limitedViewers

\* @param rewarding rewarding value

\* @return {\*}

\*/

function getContractDesigner(limitedAuthors, limitedReviewers, limitedViewers, rewarding) {

let contractPart = contractDesignerTemplate;

// Set up reward

if (rewarding !== null){

const part = 'reward = ' + rewarding.toString() + ';\n ';

contractPart = contractPart.replace('{{Set up reward}}', part);

} else {

contractPart = contractPart.replace('\n {{Set up reward}}', '');

}

// Add authors

if (limitedAuthors !== null){

let authors = '';

for (let i = 0; i < limitedAuthors.length; i++){

authors += 'authors[' + limitedAuthors[i] + '] = true;';

if (i !== limitedAuthors.length - 1){

authors += '\n ';

}

}

contractPart = contractPart.replace('{{Set up authors}}', authors);

} else {

contractPart = contractPart.replace('\n {{Set up authors}}', '');

}

// Add reviewers

if (limitedReviewers !== null){

let reviewers = '';

for (let i = 0; i < limitedReviewers.length; i++){

reviewers += 'reviewers[' + limitedReviewers[i] + '] = true;';

if (i !== limitedReviewers.length - 1){

reviewers += '\n ';

}

}

contractPart = contractPart.replace('{{Set up reviewers}}', reviewers);

} else {

contractPart = contractPart.replace('\n {{Set up reviewers}}', '');

}

// Add viewers

if (limitedViewers !== null){

let viewers = '';

for (let i = 0; i < limitedViewers.length; i++){

viewers += 'viewers[' + limitedViewers[i] + '] = true;';

if (i !== limitedViewers.length - 1){

viewers += '\n ';

}

}

contractPart = contractPart.replace('{{Set up viewers}}', viewers);

} else {

contractPart = contractPart.replace('\n {{Set up viewers}}', '');

}

return contractPart;

}

/\*\*

\* Limited authors array template

\* @type {string}

\*/

const limitedAuthorsMapping = 'mapping(address => bool) private authors;';

/\*\*

\* Checking for author function template

\*/

const checkAuthorFunctionTemplate = readFile('./ContractGenerator/Templates/CheckAuthorTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Check and get project function template

\*/

const checkProjectFunctionTemplate = readFile('./ContractGenerator/Templates/CheckProjectTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Adding for author function template

\*/

const addAuthorFunctionTemplate = readFile('./ContractGenerator/Templates/AddAuthorTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Removing of author function template

\*/

const removeAuthorFunctionTemplate = readFile('./ContractGenerator/Templates/RemoveAuthorTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Checking for project Sender Solidity function part

\* @type {string}

\*/

const checkForProjectSenderAccount = 'if (!checkAuthor(msg.sender)) throw;\n';

/\*\*

\* Get review structure params

\* @param reviewingParams reviewing params

\* @return {string}

\*/

function getReviewFields(reviewingParams) {

let contractPart = '';

for (let i = 0; i < reviewingParams.length; i++){

// Checking for values

if (reviewingParams[i].paramType !== 'string' && reviewingParams[i].paramType !== 'int'){

throw 'Error, reviewing param can be only \'int\' or \'string\'';

}

contractPart += reviewingParams[i].paramType + ' ' + reviewingParams[i].paramName + ';';

if (i !== reviewingParams.length - 1){

contractPart += '\n ';

}

}

return contractPart;

}

/\*\*

\* Reviewers Type Mapping Template

\* @type {string}

\*/

const reviewersMappingTemplate = 'mapping(address => bool) private reviewers;';

/\*\*

\* Selected reviewing mapping

\* @type {string}

\*/

const selectedReviewingMapping = 'mapping(address => mapping(address => Project[])) private selectedReviewingMap;';

/\*\*

\* Check reviewer function template

\*/

const checkReviewerFunctionTemplate =

readFile('./ContractGenerator/Templates/CheckReviewerTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Check review function argument when reviewing is selected

\* @type {string}

\*/

const selectedReviewingCheckReviewerArgs = ', address \_author, string \_projectName, bytes32 \_projectHash, bool flag';

/\*\*

\* Condition for checking when reviewing is selected

\*/

const selectedReviewingCheckingCondition =

readFile('./ContractGenerator/Templates/SelectedReviewingCheckingForReviewerTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Condition for checking when reviewing is not selected

\* @type {string}

\*/

const notSelectedReviewingCheckingCondition = 'return reviewers[\_reviewer];';

/\*\*

\* Get check reviewer function

\* @param selectedReviewing

\* @return {\*}

\*/

function getCheckReviewerFunction(selectedReviewing) {

let contractPart = checkReviewerFunctionTemplate;

if (selectedReviewing){

contractPart = contractPart.replace('{{Checking args}}', selectedReviewingCheckReviewerArgs);

contractPart = contractPart.replace('{{Reviewing condition}}', selectedReviewingCheckingCondition);

} else {

contractPart = contractPart.replace('{{Checking args}}', '');

contractPart = contractPart.replace('{{Reviewing condition}}', notSelectedReviewingCheckingCondition);

}

return contractPart;

}

/\*\*

\* Add reviewer function template

\* @type {string}

\*/

const addReviewerFunctionTemplate = readFile('./ContractGenerator/Templates/AddReviewerTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Removing reviewer function template

\*/

const removeReviewerFunctionTemplate = readFile('./ContractGenerator/Templates/RemoveReviewerTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Add review function template

\*/

const addReviewFunctionTemplate = readFile('./ContractGenerator/Templates/AddReviewTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Checking for reviewer in add review function

\* @type {string}

\*/

const checkReviewerContractPart = 'if (!checkReviewer(msg.sender {{Reviewer params}})) throw;';

/\*\*

\* Sending of rewarding template

\*/

const sendRewardTemplate = readFile('./ContractGenerator/Templates/RewardTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Getting the adding of review Solidity function

\* @param limitedReviewers reviewer's array

\* @param reviewingParams reviewing params types array

\* @param selectedReviewing flag to selecting which works should review reviewers

\* @param rewarding value

\* @return {\*}

\*/

function getAddReviewFunction(limitedReviewers, reviewingParams, selectedReviewing, rewarding){

let contractPart = addReviewFunctionTemplate;

// Set up add review function arguments and reviewing parameters

let reviewArgs = '';

let reviewingFields = '';

for (let i = 0; i < reviewingParams.length; i++){

//if (reviewingParams[i] !== 'int' && reviewingParams[i] !== 'string')

// throw 'Error, review params should have \'int\' or \'string\' value.';

let param = reviewingParams[i].paramName;

reviewArgs += reviewingParams[i].paramType + ' ' + param + ', ';

reviewingFields += param + ', ';

}

contractPart = contractPart.replace('{{Review params}}', reviewArgs);

contractPart = contractPart.replace(/{{Review fields}}/g, reviewingFields);

// Set up checking for reviewer part

if (limitedReviewers !== null) {

let checkingForReviewer = checkReviewerContractPart;

if (selectedReviewing){

checkingForReviewer = checkingForReviewer.replace('{{Reviewer params}}',

', \_author, \_projectName, \_projectHash, true');

if (rewarding !== null){

contractPart = contractPart.replace('{{Checking for balance}}', 'if (this.balance < reward) throw;');

contractPart = contractPart.replace('{{Send reward}}', sendRewardTemplate);

contractPart = contractPart.replace('{{Paymod}}', 'payable');

} else {

contractPart = contractPart.replace('{{Checking for balance}}', '');

contractPart = contractPart.replace('{{Send reward}}', '');

contractPart = contractPart.replace('{{Paymod}}', '');

}

} else {

contractPart = contractPart.replace('{{Checking for balance}}', '');

checkingForReviewer = checkingForReviewer.replace('{{Reviewer params}}', '');

contractPart = contractPart.replace('{{Send reward}}', '');

}

contractPart = contractPart.replace('{{Checking for reviewer}}', checkingForReviewer);

} else {

contractPart = contractPart.replace('{{Checking for balance}}', '');

contractPart = contractPart.replace('{{Checking for reviewer}}', '');

contractPart = contractPart.replace('{{Send reward}}', '');

contractPart = contractPart.replace('{{Paymod}}', '');

}

return contractPart;

}

/\*\*

\* Set project to reviewing function template

\* @type {string}

\*/

const setProjectToReviewingFunctionTemplate =

readFile('./ContractGenerator/Templates/SetProjectToReviewingTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Contract data viewers mapping template

\* @type {string}

\*/

const viewersMappingTemplate = 'mapping(address => bool) private viewers;';

/\*\*

\* Checking of viewers function template

\*/

const checkViewerFunctionTemplate = readFile('./ContractGenerator/Templates/CheckViewerTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Adding of viewer function template

\*/

const addViewerFunctionTemplate = readFile('./ContractGenerator/Templates/AddViewerTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Removing of viewer function template

\*/

const removeViewerFunctionTemplate = readFile('./ContractGenerator/Templates/RemoveViewerTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Show sent projects function template

\*/

const showProjectsFunctionTemplate = readFile('./ContractGenerator/Templates/ShowProjectsTemplate.sol').replace(/\r\n/g, '\n');

/\*\*

\* Show sent reviews by author address function template

\*/

const showReviewsFunctionsTemplates = readFile('./ContractGenerator/Templates/ShowReviewsFunctionsTemplates.sol').replace(/\r\n/g, '\n');

/\*\*

\* Get showing reviews function

\* @param reviewingParams reviewing params

\* @return {\*}

\*/

function getShowReviewsFunctions(reviewingParams) {

let contractPart = showReviewsFunctionsTemplates;

let tuple = '';

let res = '';

for (let i = 0; i < reviewingParams.length; i++) {

tuple += reviewingParams[i].paramType + ', ';

res += 'curReview.' + reviewingParams[i].paramName + ', ';

}

contractPart = contractPart.replace(/{{Review Types}}/g, tuple);

contractPart = contractPart.replace(/{{Review Fields}}/g, res);

return contractPart;

}

/\*\*

\* Contract generation

\* @param paramObj params

\* @return {\*}

\*/

function generateSmartContractCode(paramObj) {

// Parsing of limited authors

let limitedAuthors = paramObj.limitedAuthors;

if (limitedAuthors === undefined){

limitedAuthors = [];

}

if (!limitedAuthors){

limitedAuthors = null;

}

// Parsing of limited reviewers

let limitedReviewers = paramObj.limitedReviewers;

if (limitedReviewers === undefined){

limitedReviewers = [];

}

if (!limitedReviewers){

limitedReviewers = null;

}

// Parsing of limited viewers

let limitedViewers = paramObj.limitedViewers;

if (limitedViewers === undefined){

limitedViewers = [];

}

if (!limitedViewers){

limitedViewers = null;

}

// Parsing of reviewing parameters

const reviewingParams = paramObj.reviewingParams;

if (reviewingParams === undefined || reviewingParams === null)

throw "Error, reviewing parameters is not exist";

// Parsing of selected reviewing value

let selectedReviewing = (paramObj.selectedReviewing === 'true');

// Parsing of rewarding value

let rewarding = paramObj.rewarding;

if (!rewarding){

rewarding = null;

}

// Checking for reviewing params

if (reviewingParams === null || reviewingParams.length === 0)

throw 'Error, contract should have at list one reviewing parameter';

// Checking for reviewers' limit and selected reviewing param

if (limitedReviewers === null && selectedReviewing)

throw 'Error, selected reviewing is available only when exists limit for reviewers';

// Checking for rewarding

if (rewarding !== null && (limitedReviewers === null || limitedAuthors === null || !selectedReviewing))

throw 'Error, contract can pay reward only with limits and selected reviewing';

let contract = contractTemplate;

// Set up contract owner and designer

if (limitedAuthors !== null || limitedReviewers !== null || limitedViewers !== null){

contract = contract.replace('{{Contract owner declaration}}', contractOwnerDeclaration);

if (rewarding !== null){

contract = contract.replace('{{Payable functions}}', payableFunctionsTemplate);

} else {

contract = contract.replace('\n {{Payable functions}}\n', '');

}

contract = contract.replace('{{Contract designer}}',

getContractDesigner(limitedAuthors, limitedReviewers, limitedViewers, rewarding));

} else {

contract = contract.replace('\n {{Contract owner declaration}}\n', '');

contract = contract.replace('\n {{Payable functions}}\n', '');

contract = contract.replace('\n {{Contract designer}}\n', '');

}

// Set up viewing access

contract = contract.replace(/{{Viewing access modifier}}/g, limitedViewers !== null ? 'private' : 'public');

// Set up get project function

contract = contract.replace('{{Checking for project function}}', checkProjectFunctionTemplate);

// Set up limited authors

if (limitedAuthors !== null){

contract = contract.replace('{{Selected authors addresses mapping}}', limitedAuthorsMapping);

} else {

contract = contract.replace('\n {{Selected authors addresses mapping}}\n', '');

}

// Set up checking and adding of authors functions

if (limitedAuthors !== null)

{

contract = contract.replace('{{Checking of author function}}', checkAuthorFunctionTemplate);

contract = contract.replace('{{Adding of author function}}', addAuthorFunctionTemplate);

contract = contract.replace('{{Remove author function}}', removeAuthorFunctionTemplate);

} else {

contract = contract.replace('\n {{Checking of author function}}\n', '');

contract = contract.replace('\n {{Adding of author function}}\n', '');

contract = contract.replace('\n {{Remove author function}}\n', '');

}

// Set up adding of project function

if (limitedAuthors !== null){

contract = contract.replace('{{Adding of project author checking}}', checkForProjectSenderAccount);

} else {

contract = contract.replace('\n {{Adding of project author checking}}', '');

}

// Set up review params

contract = contract.replace('{{Review params}}', getReviewFields(reviewingParams));

// Set up reviewing functions

if (limitedReviewers !== null){

contract = contract.replace('{{Reviewers mapping}}', reviewersMappingTemplate);

if (selectedReviewing){

contract = contract.replace('{{Selected reviewing mapping}}', selectedReviewingMapping);

} else {

contract = contract.replace('\n {{Selected reviewing mapping}}\n','');

}

contract = contract.replace('{{Check reviewer function}}', getCheckReviewerFunction(selectedReviewing));

contract = contract.replace('{{Add reviewer function}}', addReviewerFunctionTemplate);

contract = contract.replace('{{Remove reviewer function}}', removeReviewerFunctionTemplate);

} else {

contract = contract.replace('\n {{Selected reviewing mapping}}\n','');

contract = contract.replace('\n {{Reviewers mapping}}\n', '');

contract = contract.replace('\n {{Check reviewer function}}\n', '');

contract = contract.replace('\n {{Add reviewer function}}\n', '');

contract = contract.replace('\n {{Remove reviewer function}}\n', '');

}

// Set up setting of project ot review function

if (selectedReviewing){

contract = contract.replace('{{Set up project to review function}}', setProjectToReviewingFunctionTemplate);

} else {

contract = contract.replace('\n {{Set up project to review function}}\n', '');

}

// Set up add review function

contract = contract.replace('{{Add review function}}',

getAddReviewFunction(limitedReviewers, reviewingParams, selectedReviewing, rewarding));

//Set up viewers params

if (limitedViewers !== null){

contract = contract.replace('{{Viewers array}}', viewersMappingTemplate);

contract = contract.replace('{{Check viewer function}}', checkViewerFunctionTemplate);

contract = contract.replace('{{Add viewer function}}', addViewerFunctionTemplate);

contract = contract.replace('{{Remove viewer function}}', removeViewerFunctionTemplate);

contract = contract.replace('{{Show projects function}}', showProjectsFunctionTemplate);

contract = contract.replace('{{Show reviews functions}}',

getShowReviewsFunctions(reviewingParams));

} else {

contract = contract.replace('\n {{Viewers array}}\n', '');

contract = contract.replace('\n {{Check viewer function}}\n', '');

contract = contract.replace('\n {{Add viewer function}}\n', '');

contract = contract.replace('\n {{Remove viewer function}}\n', '');

contract = contract.replace('\n {{Show projects function}}\n', '');

contract = contract.replace('\n {{Show reviews functions}}\n', '');

}

return contract;

}

/\*\*

\* Post method

\*/

app.post('/', function (req, res, next) {

res.send(generateSmartContractCode(req.body));

});

module.exports = app;

## contractCompiler.js

/\*\*

\* Created by Arthur on 09.05.2017.

\*/

const express = require('express');

const app = express();

const solc = require('solc');

function compileSolidity(paramObj) {

const contractCode = paramObj.code;

const compiledContract = solc.compile(contractCode, 1);

if (compiledContract.errors){

throw compiledContract.errors;

}

const abi = compiledContract.contracts[':ExpertActivityOrganisationService'].interface;

const byteCode = compiledContract.contracts[':ExpertActivityOrganisationService'].bytecode;

return {bytecode: byteCode, ABI: abi};

}

app.post('/', function (req, res, next) {

try {

res.send({cont: compileSolidity(req.body), error: ''});

} catch (error) {

res.send({cont: undefined, error: error});

}

});

module.exports = app;

## frontend.js

/\*\*

\* Created by pauldanilin on 21/04/2017.

\*/

"use strict";

const contractGenerator = new SmartcontractGenerator();

const contractCompiler = new SolidityCompiler();

const widgetGenerator = new WebWidgetGenerator();

const page = new PageInteractor(contractGenerator, contractCompiler, widgetGenerator);

function WebWidgetGenerator() {

this.generate = generateWebWidget;

/\*\*

\* Sends a request for widget generation to the server and downloads the response as a zip

\* @param parameters the parameters object of the widget. Properties: reviewingParam, abi, contractAddress

\* @param onReceive the function called on response receive (data) => {}

\*/

function generateWebWidget(parameters, onReceive) {

$.post('/webwidgetgenerator', {widgetParameters: JSON.stringify(parameters)}, onReceive);

}

}

function SmartcontractGenerator() {

this.generate = generateSmartContractCode;

function generateSmartContractCode(parameters, onReceive) {

const debugParameters = {

limitedAuthors: ['0xDd5dA50721Ba3C4e26Ada7AAA73cF9c6c26d9a86'],

limitedReviewers: null,

limitedViewers: null,

reviewingParams: [{paramType: 'int', paramName: 'mark'}],

selectedReviewing: false,

rewarding: null

};

console.log(parameters);

$.post('/contractGenerator', parameters, onReceive);

}

}

function SolidityCompiler() {

this.compile = compileSmartContract;

function compileSmartContract(parameter, onReceive) {

$.post('/contractCompiler', parameter, onReceive);

}

}

function PageInteractor(contractGenerator, contractCompiler, widgetGenerator) {

const semaphore = {

busy: false,

disabled: [],

startWaiting: () => {

document.getElementsByTagName('body')[0].className += ' loading';

const buttons = document.getElementsByTagName("button");

for (let i = 0; i < buttons.length; i++) {

if (buttons[i].disabled) {

semaphore.disabled.push(buttons[i]);

} else {

buttons[i].disabled = true;

}

}

const checks = document.getElementsByClassName("flag");

for (let i = 0; i < checks.length; i++) {

checks[i].disabled = true;

}

semaphore.busy = true;

},

stopWaiting: (unlock) => {

const className = document.getElementsByTagName('body')[0].className;

document.getElementsByTagName('body')[0].className = className.replace(' loading');

const buttons = document.getElementsByTagName("button");

for (let i = 0; i < buttons.length; i++) {

buttons[i].disabled = false;

}

for (let i = 0; i < semaphore.disabled.length; i++) {

semaphore.disabled[i].disabled = true;

}

semaphore.disabled = [];

if (unlock)

unlock.disabled = false;

this.busy = false;

}

};

const uncheckPayable = () => {

findElement('payableCheck').checked = false;

findElement('payable').setAttribute('hidden', 'hidden');

};

const uncheckPayableAndSelected = () => {

uncheckPayable();

findElement('selectedReviewingCheck').checked = false;

};

this.limitAuthors = hideOrShow('limitAuthors', null, uncheckPayableAndSelected);

this.addAuthorAccess = () => addParameter('restrictedAuthors', 'addedParameter restrictedAuthor');

this.limitReviewers = hideOrShow('limitReviewers', null, uncheckPayableAndSelected);

this.addReviewerAccess = () => addParameter('restrictedReviewers', 'addedParameter restrictedReviewer');

this.limitViewers = hideOrShow('limitViewers');

this.addViewerAccess = () => addParameter('restrictedViewers', 'addedParameter restrictedViewer');

this.addContractParameter = () => addContractParameter('reviewingParams',

'reviewingParam');

this.generateSmartcontract = generateSmartcontract;

this.deploySmartcontract = deploySmartcontract;

this.generateWebWidget = generateWidget;

this.debugGenWidget = debugGenerateWidget;

this.advancedClick = () => {

const contractCode = findElement('smartContractSource');

if (confirm('Are you sure you want to edit the smart contract?\nThe widget compatibility is not guaranteed!')) {

contractCode.setAttribute('contenteditable', '');

findElement('editButton').remove();

}

};

this.payable = hideOrShow('payable', (checkbox) => {

if (!findElement('limitAuthorsCheck').checked || !findElement('limitReviewersCheck').checked ||

!findElement('selectedReviewingCheck').checked) {

const message = 'You should check limit authors, limit reviewers and selected reviewing before checking "payable".';

alert(message);

checkbox.checked = false;

throw message;

}

});

this.selectedReviewing = (checkbox) => {

if (!findElement('limitAuthorsCheck').checked || !findElement('limitReviewersCheck').checked) {

alert('You should check limit authors, limit reviewers before checking "selected reviewing".');

checkbox.checked = false;

}

if (!checkbox.checked) {

uncheckPayable();

}

};

/\*\*

\* Adds an editable element to the parentID with the insert class class.

\* @param parentID the parent of the new node

\* @param insertClass the new node class

\*/

function addParameter(parentID, insertClass) {

const propertiesContainer = findElement(parentID);

const addedParameterCont = document.createElement("div");

addedParameterCont.setAttribute("class", "btn btn-default");

const addedParameter = document.createElement("div");

addedParameter.setAttribute("class", insertClass);

addedParameter.setAttribute("contenteditable", "true");

propertiesContainer.appendChild(addedParameterCont);

addedParameterCont.appendChild(addedParameter);

addedParameterCont.appendChild(deleteButton(addedParameterCont));

addedParameter.focus();

}

function deleteButton(container) {

const addedParameterDelete = document.createElement("button");

addedParameterDelete.innerText = "x";

addedParameterDelete.setAttribute("class", "btn btn-danger deleteButton addedParameter");

addedParameterDelete.onclick = () => {

container.remove();

};

return addedParameterDelete;

}

/\*\*

\* Adds a ui object for a new contract reviewing parameter.

\* @param parentID the id of the container element

\* @param insertClass the class of the newly added parameter

\*/

function addContractParameter(parentID, insertClass) {

const propertiesContainer = findElement(parentID);

const addedParameterContainer = document.createElement("div");

addedParameterContainer.setAttribute("class", "btn btn-default ".concat(insertClass));

const addedParameterName = document.createElement("div");

addedParameterName.setAttribute("class", "parameterName");

addedParameterName.setAttribute("contenteditable", "true");

propertiesContainer.appendChild(addedParameterContainer);

addedParameterContainer.appendChild(addedParameterName);

addedParameterContainer.appendChild(parameterTypeNodeSelector());

addedParameterContainer.appendChild(deleteButton(addedParameterContainer));

addedParameterName.focus();

}

/\*\*

\* Constructs an element for parameter type selection.

\* @returns {Element} the constructed element

\*/

function parameterTypeNodeSelector() {

const selector = document.createElement("select");

const intOption = document.createElement("option");

const stringOption = document.createElement("option");

selector.setAttribute("class", "parameterType");

intOption.setAttribute("class", "int");

stringOption.setAttribute("class", "string");

intOption.innerHTML = "Int";

stringOption.innerHTML = "String";

selector.appendChild(stringOption);

selector.appendChild(intOption);

return selector;

}

/\*\*

\* The generate smartcontract click handler

\*/

function generateSmartcontract() {

try {

semaphore.startWaiting();

const generationParameters = getSmartcontractParameters();

contractGenerator.generate(generationParameters, function (code) {

findElement("smartContractSource").innerHTML = code;

semaphore.stopWaiting(findElement("deploySmartcontract"));

findElement('generateSmartcontract').style.display = 'none';

findElement('deploySmartcontract').style.display = '';

const checks = document.getElementsByClassName("flag");

for (let i = 0; i < checks.length; i++) {

checks[i].disabled = true;

}

const addButtons = document.getElementsByClassName("btn-add");

for (let i = addButtons.length - 1; i >= 0; i--) {

const button = addButtons[i];

button.remove();

}

const delButtons = document.getElementsByClassName("deleteButton");

for (let i = delButtons.length - 1; i >= 0; i--) {

const button = delButtons[i];

button.remove();

}

const reviewingParams = document.getElementsByClassName("reviewingParam");

for (let i = 0; i < reviewingParams.length; i++) {

const param = reviewingParams[i];

param.children[0].removeAttribute('contenteditable');

param.children[1].disabled = true;

}

const otherParams = document.getElementsByClassName("addedParameter");

for (let i = 0; i < otherParams.length; i++) {

const param = otherParams[i];

param.removeAttribute('contenteditable');

param.disabled = true;

}

});

} catch (error) {

alert(error);

semaphore.stopWaiting();

const checks = document.getElementsByClassName("flag");

for (let i = 0; i < checks.length; i++) {

checks[i].disabled = true;

}

}

}

/\*\*

\* Collects the input from the webpage.

\* If an error occurs, it is thrown.

\* @returns {{}} the constructed object

\*/

function getSmartcontractParameters() {

const input = {};

input.reviewingParams = getReviewingParamsFromPage();

checkUniquenessEmptyness(input.reviewingParams.map(el => el.paramName));

if (input.reviewingParams.length === 0) {

throw("There has to be at least one assessment parameter!");

}

function checked(id) {

return findElement(id).checked;

}

if (checked('limitAuthorsCheck')) {

input.limitedAuthors = getElementsTextsFromPage("restrictedAuthor");

checkAddresses(input.limitedAuthors);

} else input.limitedAuthors = null;

if (checked('limitReviewersCheck')) {

input.limitedReviewers = getElementsTextsFromPage("restrictedReviewer");

checkAddresses(input.limitedReviewers);

} else input.limitedReviewers = null;

if (checked('payableCheck')) {

input.rewarding = document.getElementById("payableSum").value;

} else input.rewarding = null;

if (checked('selectedReviewingCheck')){

input.selectedReviewing = true;

} else input.selectedReviewing = false;

if (checked('limitViewersCheck')) {

input.limitedViewers = getElementsTextsFromPage("restrictedViewer");

checkAddresses(input.limitedViewers);

} else input.limitedViewers = null;

return input;

}

/\*\*

\* Gets the array of reviewing parameters from the web page and checks it.

\* @returns {Array} the array of reviewing parameters from the web page

\*/

function getReviewingParamsFromPage() {

const assessParameters = [];

const parametersNodes = document.getElementsByClassName('reviewingParam');

for (let i = 0; i < parametersNodes.length; ++i) {

const parNode = parametersNodes[i];

const name = parNode.getElementsByClassName('parameterName')[0].innerText;

const selector = parNode.getElementsByClassName('parameterType')[0];

const type = selector[selector.selectedIndex].className;

if (type !== 'int' && type !== 'string')

throw ("The type should be either 'string' or 'int'");

assessParameters.push({paramName: name, paramType: type});

}

// console.log(assessParameters.map(el => el.paramName));

// checkUniquenessEmptyness(assessParameters.map(el => el.paramName));

return assessParameters;

}

/\*\*

\* Collects the parameters from the web page and checks them.

\* @returns {Array} the array of strings of parameters for assessment

\*/

function getElementsTextsFromPage(parameterClass) {

const assessmentNodes = document.getElementsByClassName(parameterClass);

const parameters = [];

for (let i = 0; i < assessmentNodes.length; ++i) {

parameters.push(assessmentNodes[i].innerText);

}

return parameters;

}

/\*\*

\* Returns a function (checkbox) for hiding or showing the element by id on clicking the checkbox

\* @param id the id of the element to show/hide

\* @returns {function(\*)} the function that hides or shows the element by id on clicking the checkbox

\*/

function hideOrShow(id, check, onDisable) {

/\*\*

\* The checkbox click handler for limitation enabling and disabling

\* @param checkbox the checkbox for reaction

\* @returns {boolean} false (for the form)

\*/

return (checkbox) => {

if (check)

check(checkbox);

const limitAuthorsNode = findElement(id);

if (checkbox.checked === true) {

limitAuthorsNode.removeAttribute('hidden');

}

else {

limitAuthorsNode.setAttribute('hidden', 'hidden');

if (onDisable)

onDisable(checkbox);

}

}

}

/\*\*

\* Finds an element by id.

\* @param id the string id of the element

\* @returns {Element} the found element

\*/

function findElement(id) {

return document.getElementById(id);

}

/\*\*

\* Checks the parameters correctness.

\* Throws an exception if the parameters are not all unique and some parameter is ''.

\* @param parameters the parameters to check

\*/

function checkUniquenessEmptyness(parameters) {

/\*\*

\* Checks the parameter on correctness.

\* @param parameter the checked parameter

\* @returns {boolean} true if incorrect, false if correct

\*/

function checkCorrectness(parameter) {

return parameter !== null && parameter !== undefined && /^[\_a-zA-Z][\_a-zA-Z0-9]{0,30}$/.test(parameter);

}

if (parameters === null || parameters.length === 0)

return;

const sorted\_arr = parameters.slice().sort(); // copy and sort

for (let i = 0; i < parameters.length - 1; i++) {

if (!checkCorrectness(sorted\_arr[i])) {

throw ("Empty strings and not latin symbols are not allowed!");

}

if (sorted\_arr[i + 1] === sorted\_arr[i]) {

throw ("Duplicates are not allowed!");

}

}

if (!checkCorrectness(sorted\_arr[sorted\_arr.length - 1])) {

throw ("Empty strings and not latin symbols are not allowed!");

}

}

/\*\*

\* Checks the parameters correctness.

\* Throws an exception if the addresses invalid.

\* @param addresses

\*/

function checkAddresses(addresses) {

if (addresses === null)

return;

const sorter\_arr = addresses.slice().sort();

for (let i = 0; i < addresses.length - 1; i++){

if (!web3.isAddress(sorter\_arr[i])){

throw ('Account address should be 20 bytes value');

}

if (sorted\_arr[i + 1] === sorted\_arr[i]) {

throw ("Duplicates are not allowed!");

}

}

}

/\*\*

\* Compile and deploy generated smartcontract

\*/

function deploySmartcontract() {

semaphore.startWaiting();

let contractCode = findElement('smartContractSource').innerText;

window.compiled = contractCompiler.compile({code: contractCode}, (result, success) => {

if (!result.error){

alert('Compiled!');

const bytecode = result.cont.bytecode;

const abi = result.cont.ABI;

web3.eth.estimateGas({data:bytecode}, (err, res)=>{

if (!err){

const par = {

from: web3.eth.accounts[0],

data: bytecode,

gas: res

};

console.log('Gas estimated: ', res);

// alert(res);

const contract = web3.eth.contract(JSON.parse(abi));

contract.new(par, (e, r)=>{

if (!e){

if (!r.address){

console.log('Transaction hash: ', r.transactionHash);

const transactionHashElement = findElement('transactionHash');

transactionHashElement.hidden = false;

transactionHashElement.innerText = 'Transaction hash: ' + r.transactionHash;

} else {

const contAddressElement = findElement('contractAddress');

contAddressElement.hidden = false;

contAddressElement.innerText = 'Smartcontract address: ' + r.address;

console.log('Address: ' + r.address);

console.log('Abi: ' + JSON.stringify(abi));

widgetGenerator.abi = abi;

widgetGenerator.contractAddress = r.address;

findElement('generateWebWidget').style.display = '';

semaphore.stopWaiting(findElement('generateWebWidget'));

}

} else {

alert('Oops, an error occurred deploying: ' + e);

semaphore.stopWaiting();

}

});

} else {

alert("Oops, cannot to estimate gas: " + err);

semaphore.stopWaiting();

}

});

} else {

alert("Oops, an error occurred compiling: " + result.error);

semaphore.stopWaiting();

}

});

}

/\*\*

\* Collects the data on widget generation and generates it.

\*/

function generateWidget() {

semaphore.startWaiting();

let parameters = {};

try {

parameters = getSmartcontractParameters();

} catch (error) {

alert("Couldn't get parameters for widget generation: " + error);

return;

}

parameters.abi = widgetGenerator.abi;

if (!parameters.abi) {

alert("Couldn't find the contract's abi.\nMaybe you haven't created it yet?");

return;

}

parameters.contractAddress = widgetGenerator.contractAddress;

if (!parameters.contractAddress) {

alert("Couldn't find the contract's address.\nMaybe you haven't deployed it yet?");

return;

}

widgetGenerator.generate(parameters,

function (data) { // on response receive

// Download the zip archive with sources

let a = document.createElement('a');

a.download = 'archive.zip'; // Set the file name.

a.style.display = 'none';

a.href = data;

a.click();

semaphore.stopWaiting();

}

);

}

function getDebugParameters() {

const parameters = {};

parameters.reviewingParams = [{paramName: "Mark", paramType: "int"}];

parameters.abi = `[{

"constant": false,

"inputs": [{"name": "CommentOne", "type": "string"}, {"name": "Mark", "type": "int256"}, {

"name": "\_author",

"type": "address"

}, {"name": "\_projectName", "type": "string"}, {"name": "\_projectHash", "type": "bytes32"}],

"name": "addReview",

"outputs": [],

"payable": false,

"type": "function"

}, {

"constant": true,

"inputs": [{"name": "", "type": "address"}, {"name": "", "type": "uint256"}],

"name": "reviewsByReviewer",

"outputs": [{"name": "reviewer", "type": "address"}, {

"name": "author",

"type": "address"

}, {"name": "CommentOne", "type": "string"}, {"name": "Mark", "type": "int256"}, {

"name": "workHash",

"type": "bytes32"

}, {"name": "name", "type": "string"}],

"payable": false,

"type": "function"

}, {

"constant": true,

"inputs": [{"name": "", "type": "address"}, {"name": "", "type": "uint256"}],

"name": "reviewsByAuthor",

"outputs": [{"name": "reviewer", "type": "address"}, {

"name": "author",

"type": "address"

}, {"name": "CommentOne", "type": "string"}, {"name": "Mark", "type": "int256"}, {

"name": "workHash",

"type": "bytes32"

}, {"name": "name", "type": "string"}],

"payable": false,

"type": "function"

}, {

"constant": false,

"inputs": [{"name": "\_projectHash", "type": "bytes32"}, {"name": "\_projectName", "type": "string"}],

"name": "addProject",

"outputs": [],

"payable": false,

"type": "function"

}, {

"constant": true,

"inputs": [{"name": "", "type": "address"}, {"name": "", "type": "uint256"}],

"name": "projects",

"outputs": [{"name": "workHash", "type": "bytes32"}, {"name": "name", "type": "string"}],

"payable": false,

"type": "function"

}]`;

parameters.contractAddress = '0x30cd946010fc1cf480bf500c5bafbce8cd98cec6';

return parameters;

}

function debugGenerateWidget(limited) {

const parameters = getDebugParameters();

if (limited) {

parameters.limitedViewers = [];

parameters.limitedReviewers = [];

parameters.limitedAuthors = [];

} else {

parameters.limitedViewers = null;

parameters.limitedReviewers = [];

parameters.limitedAuthors = null;

}

widgetGenerator.generate(parameters,

function (data) { // on response receive

// Download the zip archive with sources

let a = document.createElement('a');

a.download = 'archive.zip'; // Set the file name.

a.style.display = 'none';

a.href = data;

a.click();

});

}

}

window.addEventListener('load', () => {

document.addEventListener('paste', (e) => {

e.preventDefault();

document.execCommand('insertText', false, e.clipboardData.getData('text'));

});

if (typeof web3 !== 'undefined') {

// Use Mist/MetaMask's provider

console.log('found web3');

window.web3 = new Web3(web3.currentProvider);

} else {

console.log('No web3? You should consider trying MetaMask!');

// fallback - use your fallback strategy (local node / hosted node + in-dapp id mgmt / fail)

// window.web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"));

alert("No Metamask!");

}

});

## index.pug

extends layout

block main

div#heading

h1 Smartcontract and Web Widget Generation Page

div#content

div.form-control

label.checkbox-inline

input.flag(type="checkbox" id="limitAuthorsCheck" onchange="page.limitAuthors(this);")

| Limit authors

label.checkbox-inline

input.flag(type="checkbox" id="limitReviewersCheck" onchange="page.limitReviewers(this);")

| Limit reviewers

label.checkbox-inline

input.flag(type="checkbox" id="limitViewersCheck" onchange="page.limitViewers(this);")

| Limit viewers

label.checkbox-inline

input.flag(type="checkbox" id="selectedReviewingCheck", onchange="page.selectedReviewing(this);")

| Selected Reviewing

label.checkbox-inline

input.flag(type="checkbox" id="payableCheck" onchange="page.payable(this)")

| Payable

div#parameters

h3 Assessment parameters:

button(

class="btn btn-default btn-add"

onclick="page.addContractParameter()"

) +

div(id="reviewingParams")

div(id='limitAuthors' hidden)

h3 Authors with access:

button(

class="btn btn-default btn-add"

onclick="page.addAuthorAccess()"

) +

div(id="restrictedAuthors")

div(id='limitReviewers' hidden)

h3 Reviewers with access:

button(

class="btn btn-default btn-add"

onclick="page.addReviewerAccess()"

) +

div(id="restrictedReviewers")

div(id='limitViewers' hidden)

h3 Viewers with access:

button(

class="btn btn-default btn-add"

onclick="page.addViewerAccess()"

) +

div(id="restrictedViewers")

div(id="payable" hidden)

h3 Reward

|Rewarding sum: &nbsp

input(type="number" value="0" min="0" id="payableSum")

| wei

button(

id="generateSmartcontract"

class="btn btn-primary"

onclick="page.generateSmartcontract()"

) Generate smartcontract

//button(class="btn btn-danger" id="editButton" onclick="page.advancedClick();") Edit smartcontract

pre(id="smartContractSource") Smartcontract code

br

p#transactionHash(hidden) Transaction hash:

p#contractAddress(hidden) Smartcontract address:

button(

id="deploySmartcontract"

class="btn btn-primary"

hidden

onclick="page.deploySmartcontract();"

style="display:none"

) Deploy smartcontract

br

button(id="generateWebWidget" class="btn btn-primary" onclick="page.generateWebWidget()" style="display:none") Generate Web Widget

div(id="webWidgetSources" hidden)

div.modal

## Шаблоны для генерации контрактов

1. // Adds new author

function addAuthor(address \_author)

{

if (msg.sender != owner)

throw;

authors[\_author] = true;

}

1. // Add new reviewer

function addReviewer(address \_reviewer)

{

// Checking for caller

if (msg.sender != owner)

throw;

reviewers[\_reviewer] = true;

}

1. // Add review

function addReview({{Review params}} address \_author, string \_projectName, bytes32 \_projectHash) {{Paymod}}

{

{{Checking for balance}}

{{Checking for reviewer}}

// Checking for project

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

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

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

{{Send reward}}

}

}

1. // Insertion of new contract data viewer

function addViewer(address \_viewer)

{

if (msg.sender != owner)

throw;

viewers[\_viewer] = true;

}

1. /// Check author's address

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

{

return authors[\_author];

}

1. // Check project

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;

}

1. // Checking for reviewer

function checkReviewer(address \_reviewer {{Checking args}}) private returns (bool)

{

{{Reviewing condition}}

}

1. // Viewers checking function

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

{

return viewers[\_viewer];

}

1. // Contract Designer

function ExpertActivityOrganisationService()

{

owner = msg.sender;

{{Set up reward}}

{{Set up authors}}

{{Set up reviewers}}

{{Set up viewers}}

}

1. // Reviewers reward

uint reward;

// Deposite

function () payable {

if (msg.sender != owner) throw;

}

// Refund

function refund() payable {

if (msg.sender != owner) throw;

owner.transfer(this.balance);

}

// Change reviewers reward

function changeReward(uint \_reward){

if (msg.sender != owner) throw;

reward = \_reward;

}

1. pragma solidity ^0.4.9;

/// The contract contains reviews to the projects.

contract ExpertActivityOrganisationService

{

{{Contract owner declaration}}

{{Payable functions}}

{{Contract designer}}

// Author's project struct

struct Project

{

bytes32 workHash;

string name;

}

// Mapping from author to projects

mapping(address => Project[]) {{Viewing access modifier}} projects;

{{Selected authors addresses mapping}}

{{Checking of author function}}

{{Adding of author function}}

{{Remove author function}}

{{Checking for project function}}

// Adds the project to the sender.

function addProject(bytes32 \_projectHash, string \_projectName)

{

{{Adding of project author checking}}

// Checking for resending of project

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

throw;

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

}

// Review's structure

struct Review

{

address reviewer;

address author;

// Review parameters

{{Review params}}

// Project params

bytes32 workHash;

string name;

}

// Mapping from project's author address to reviews

mapping (address => Review[]) {{Viewing access modifier}} reviewsByAuthor;

// Mapping from reviewer's address to reviews

mapping (address => Review[]) {{Viewing access modifier}} reviewsByReviewer;

{{Reviewers mapping}}

{{Selected reviewing mapping}}

{{Check reviewer function}}

{{Add reviewer function}}

{{Remove reviewer function}}

{{Set up project to review function}}

{{Add review function}}

{{Viewers array}}

{{Check viewer function}}

{{Add viewer function}}

{{Remove viewer function}}

{{Show projects function}}

{{Show reviews functions}}

}

1. // Removing of author

function removeAuthor(address \_author)

{

if (msg.sender != owner)

throw;

authors[\_author] = false;

}

1. // Removing of author

function removeAuthor(address \_author)

{

if (msg.sender != owner)

throw;

authors[\_author] = false;

}

1. // Removing of viewer

function removeViewer(address \_viewer)

{

if (msg.sender != owner)

throw;

viewers[\_viewer] = false;

}

1. // Checking for reviewer

if (!reviewers[\_reviewer])

return false;

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

// Checking for author

if (curProjects.length == 0)

return false;

// Checking for project

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;

}

// Function to set up project to reviewing

function setProjectToReview(address \_reviewer, address \_author, string \_projectName, bytes32 \_projectHash)

{

// Checking for caller

if (msg.sender != owner)

throw;

// Checking for project

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

throw;

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

throw;

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

}

return false;

1. // Show sent projects

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

{

if(!checkViewer(msg.sender))

throw;

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

}

1. // Show all reviews by author address

function showReviewsByAuthor(address \_author, uint i) returns (address, {{Review Types}}string, bytes32)

{

if(!checkViewer(msg.sender))

throw;

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

return (curReview.reviewer, {{Review Fields}}curReview.name, curReview.workHash);

}

1. // Show all reviews by reviewer address

function showReviewsByReviewer(address \_reviewer, uint i) returns (address, {{Review Types}}string, bytes32)

{

if (!checkViewer(msg.sender))

throw;

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

return (curReview.author, {{Review Fields}}curReview.name, curReview.workHash);

}

# **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 19.201-78 Техническое задание. 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).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |