Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**НИУ МЭИ**

**Лабораторная работа №4**

**по дисциплине**

**«Технология разработки программного обеспечения»**

**МОДЕЛИРОВАНИЕ РАБОТЫ КОМАНДЫ РАЗРАБОТЧИКОВ**

Выполнила: Таранова К. А.

Группа А-07м-16

Проверил:

Москва, 2017

1. **Постановка задачи**

Необходимо разработать приложение, упрощённо моделирующее жизнь команды разработчиков ПО в КБ. Существует возможность выбрать в качестве членов команды программистов, тестировщиков, технических писателей, веб-дизайнеров и менеджеров. Члены команды с утра приходят на работу, занимаются каждый своим делом, время от времени прерываются на чаепитие, беседы и совещания. Большинство ситуаций доведено до гротеска для упрощения модели и более наглядного отображения результатов работы.

1. **Описание программы.**

Программа написана на мультипарадигменном языке JavaScript, который поддерживает объектно-ориентированный и функциональный стили программирования.

В данной работе применялся объектно-ориентированный стиль программирования. Объект в JavaScript — это ассоциативный массив, который содержит в себе наборы пар ключ-значение. Одна из особенностей языка — реализация прототипного наследования. Идея наследования в JavaScript основана на клонировании методов объекта и дополнении его собственным поведением. Объект, который клонируется, называется прототипом.

Для моделирования работы команды разработчиков пользователь задаёт не более 20 членов команды, имеющих различную специализацию (программист, тестировщик, технический писатель, веб-дизайнер, менеджер), указывая их имя, пол и уровень опыта. Для программистов и веб-дизайнеров также существует возможность указать до 7 языков программирования, которыми они владеют (подразумевается, что веб-дизайнер не просто рисует или проектирует веб-интерфейсы, но и может запрограммировать то, что он задумал). Пример формы добавления члена команды показан на рис. 1.

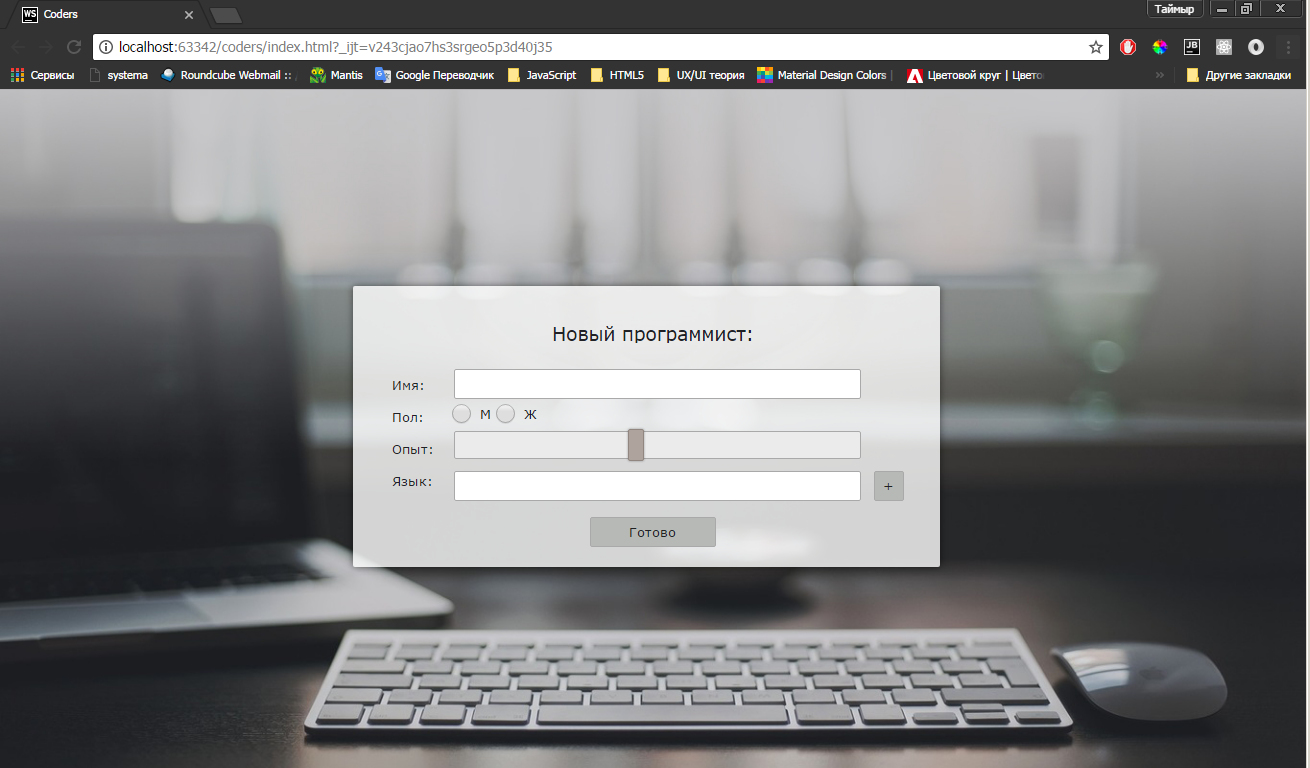


Рис. 1. Пример формы добавления члена команды.

Добавив желаемое число сотрудников, пользователь может настроить некоторые параметры проекта, такие как наличие аварийной ситуации (которая подразумевает, что члены команды будут вести себя серьёзнее и больше заниматься своими прямыми обязанностями, чем обычно), время, выделяемое на проект, и сложность проекта. Форма настройки параметров проекта показана на рис. 2.

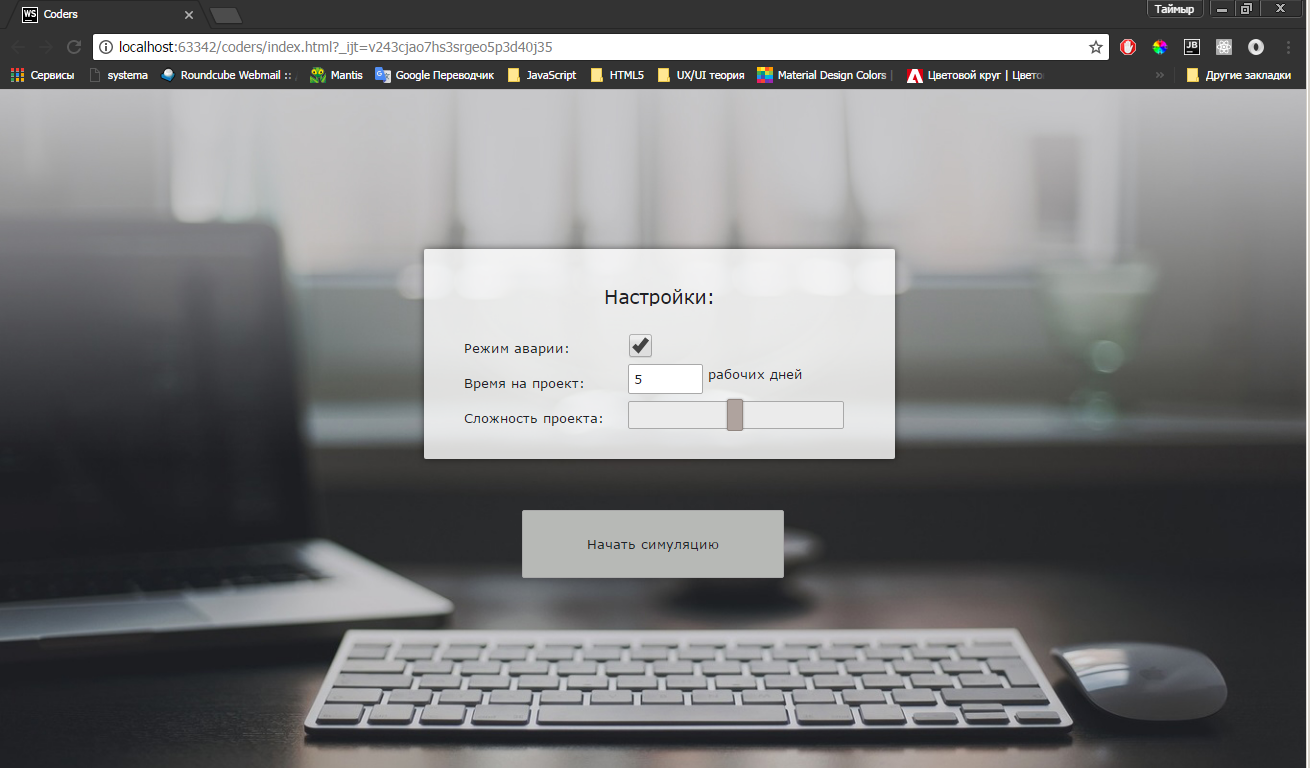


Рис. 2. Форма настройки параметров проекта.

Нажатие на кнопку «Начать симуляцию» запускает таймеры, отвечающие за циклическую смену дня и ночи и управляющие действиями сотрудников. Каждое текущее действие выбирается случайным образом (за исключением некоторых ситуаций), и может оказывать влияние на настроение и уровень опыта члена команды (они отображаются над схематическим изображением разработчика вместе с его именем и специализацией). Уровень опыта, в свою очередь, влияет на то, насколько эффективно человек будет справляться с поставленной задачей. Если работа над проектом закончена в срок или раньше, появляется поздравительное сообщение, если же команда не справилась с проектом за отведённое время, то появляется сообщение с приказом всех уволить. Работа программы в режиме симуляции показана на рис. 3.

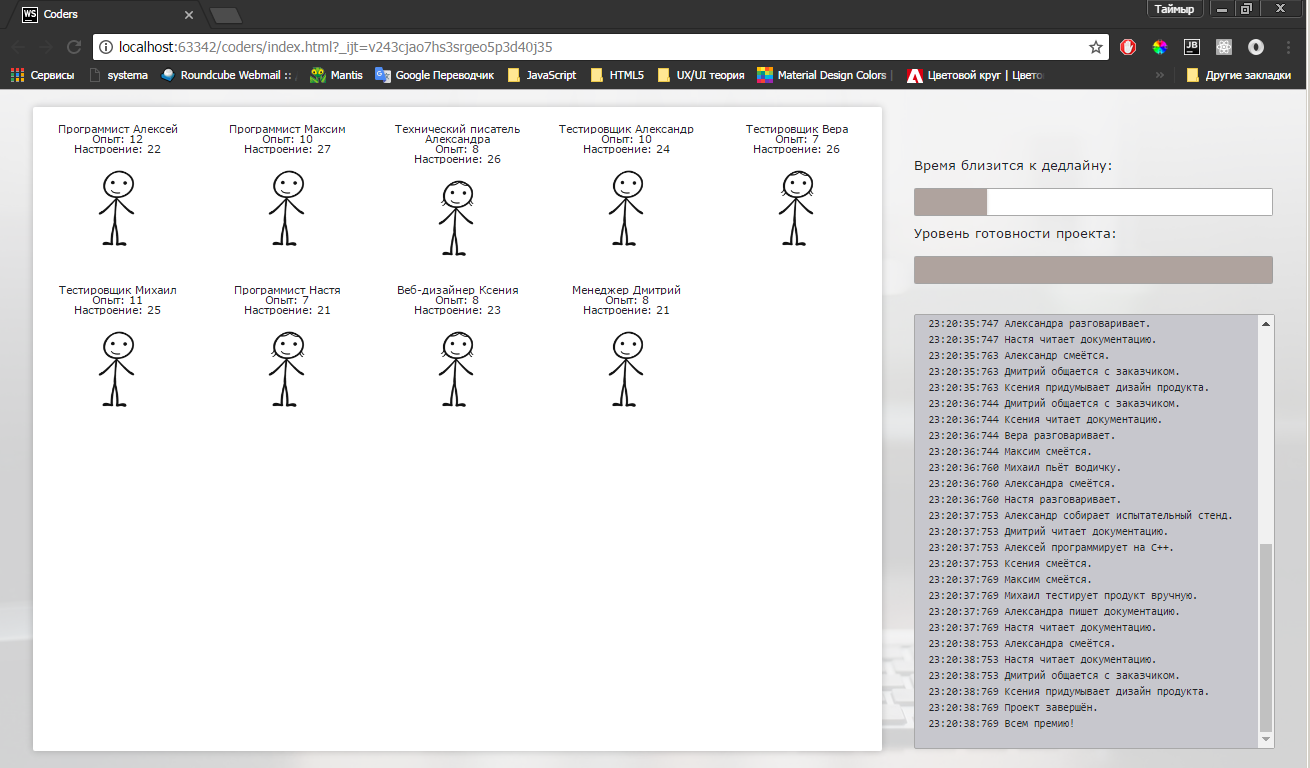


Рис. 3. Демонстрация работы программы в режиме симуляции.

1. **Классы**

В программе реализованы следующие классы:

1. Класс Project, описывающий текущий проект. Класс содержит следующие свойства:

* planned\_time – время, выделенное на проект
* difficulty – уровень сложности проекта
* emergency – флаг аварийной ситуации
* readiness – метрика готовности проекта

1. Класс Human, являющийся родительским для классов Coder, Tester, Techwriter, Designer и Manager, объединяющий общие черты вышеуказанных классов-потомков. Он имеет следующие свойства:

* name – имя сотрудника
* gender – пол сотрудника
* skill – уровень опыта
* specialty – специальность
* cheerfulness – уровень настроения
* start\_day – маркер начала рабочего дня
* work\_interval – идентификатор таймера управления работой для данного сотрудника
* place\_x – координата по горизонтали места сотрудника на изображении
* place\_y – координата по вертикали места сотрудника на изображении

Также класс Human реализует следующие методы:

* come\_to\_office() – прийти в офис
* go\_home() – уйти из офиса
* drink(drink) – выпить указанный в качестве параметра напиток
* laugh() – смеяться
* talk() – разговаривать (можно лишь при наличии в офисе кого-то ещё)
* read\_documentation() – читать документацию

1. Класс Coder, дочерний по отношению к Human, описывающий поведение программиста и являющийся родительским для класса Designer. Он имеет свойство language, хранящее массив языков, которыми владеет программист. Также класс имеет метод code() – программировать.
2. Класс Tester, дочерний по отношению к Human, описывающий поведение тестировщика. Он реализует следующие методы:

* test(how) – тестировать вручную или писать автотесты
* make\_testing\_method() – придумывать новую методику тестирования
* make\_testbench() – собирать испытательный стенд

1. Класс Techwriter, дочерний по отношению к Human, описывающий поведение технического писателя. Он имеет метод write\_documentation() – писать документацию.
2. Класс Designer, дочерний по отношению к Coder, описывающий поведение веб-дизайнера. Класс реализует следующие методы:

* create\_design() – придумывать дизайн проекта
* create\_UX() – проектировать взаимодействие с пользователем
* draw() – рисовать

1. Класс Manager, дочерний по отношению к Human, описывающий поведение менеджера. Класс реализует следующие методы:

* organize\_meeting() – организовать совещание
* communicate\_with\_customer()- общаться с заказчиком

Привычное понятие диаграммы классов неприменимо для прототипной модели, так как класс может измениться в любой момент. Условная диаграмма классов представлена на рис. 4.



Рис. 4. Диаграмма классов.

**Приложение А.**

**Техническое задание**

ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ №4

«МОДЕЛИРОВАНИЕ РАБОТЫ КОМАНДЫ РАЗРАБОТЧИКОВ»

Студент Таранова К. А. А-07м-16

С использованием методов объектно-ориентированного программирования разработать программу (на любом ЯП) моделирования выбранной предметной области. Желательно проиллюстрировать результаты работы (с помощью графиков, гистограмм, рисунков, анимации, таблиц и др. – по выбору).

**Тема:** Моделирование работы команды разработчиков ПО.

Необходимо разработать приложение, упрощённо моделирующее жизнь команды разработчиков ПО в КБ. Существует возможность выбрать в качестве членов команды программистов, тестировщиков, технических писателей, веб-дизайнеров и менеджеров. Члены команды с утра приходят на работу, занимаются каждый своим делом, время от времени прерываются на чаепитие, беседы и совещания. Большинство ситуаций доведено до гротеска для упрощения модели и более наглядного отображения результатов работы.

Программа будет реализована на мультипарадигменном языке JavaScript, который поддерживает объектно-ориентированный, функциональный и функциональный стили программирования.

Преподаватель: Раскатова М.В. Подпись Дата

Студент: Таранова К. А. Подпись Дата

**Приложение Б.**

**Листинг программы**

Файл main.js содержит глобальные переменные и основные константы:

'use strict';

let coders = [];

let testers = [];

let techwriters = [];

let designers = [];

let managers = [];

// Индекс каждого нового поля ввода для очередного языка программирования:

let coder\_lang\_index = 0;

let designer\_lang\_index = 0;

// Пол сотрудника, добавляемого в данный момент:

let gender\_val = "";

let workplace\_x = 0;

let workplace\_y = 0;

let current\_project = {};

// Время в миллисекундах, выделяемое на день и ночь:

const DAY\_PERIOD = 20000;

const NIGHT\_PERIOD = 5000;

Файл main.js содержит описание всех классов:

class Project {

constructor(time, difficulty, emergency) {

this.planned\_time = time;

this.difficulty = difficulty;

this.emergency = emergency;

this.readiness = 0;

}

}

class Human {

constructor(name, gender, skill) {

// Параметры упакованы в отдельный объект, чтобы с помощью Object.getOwnPropertyNames()

// было проще получать названия методов в виде массива, что необходимо для выполнения

// методов в случайном порядке.

// Таким образом, мы просто перечисляем свойства объекта не с нуля, что захватывало бы this.params,

// а с единицы. Также появляется возможность присвоить фиксированные порядковые номера методам,

// требующим "особого отношения" (таких, например, как this.come\_to\_office и this.go\_home).

this.params = {

name: name,

gender: gender,

skill: skill,

specialty: "",

cheerfulness: 25,

start\_day: true,

work\_interval: 0,

place\_x: 0,

place\_y: 0

}

// 1

this.come\_to\_office = function() {

if ("m" == this.params.gender) {

write\_log(this.params.name + " пришёл в офис.");

} else if ("f" == this.params.gender) {

write\_log(this.params.name + " пришла в офис.");

}

$('#simulation\_office\_screen').css("background", "white");

$('#workplace\_' + this.params.place\_y + '\_' + this.params.place\_x + ' .dude').removeClass("hide");

$('#workplace\_' + this.params.place\_y + '\_' + this.params.place\_x + ' .dude').addClass("expandOpen");

$('#workplace\_' + this.params.place\_y + '\_' + this.params.place\_x + ' .dude').css("visibility", "visible");

this.params.cheerfulness = 25;

}

// 2

this.go\_home = function() {

if ("m" == this.params.gender) {

write\_log(this.params.name + " ушёл домой.");

} else if ("f" == this.params.gender) {

write\_log(this.params.name + " ушла домой.");

}

$('#workplace\_' + this.params.place\_y + '\_' + this.params.place\_x + ' .dude').removeClass("expandOpen");

$('#workplace\_' + this.params.place\_y + '\_' + this.params.place\_x + ' .dude').addClass("hide");

$('#simulation\_office\_screen').css("background", "#101336");

/\* setTimeout(function () {

$('#simulation\_office\_screen').css("background", "URL('img/night.jpg')");

}, 1000); \*/

}

// 3

this.drink = function(drink) {

write\_log(this.params.name + " пьёт " + drink + ".");

change\_cheerfulness(this, 1);

}

// 4

this.laugh = function() {

if (!current\_project.emergency) {

write\_log(this.params.name + " смеётся.");

change\_cheerfulness(this, 1);

} else {

do\_something(this, 7);

}

}

// 5

// Этот метод будет перегружен: разговаривать можно, только если ты не один в офисе.

this.talk = function() {

write\_log(this.params.name + " разговаривает.");

}

// 6

this.read\_documentation = function() {

if (this.params.cheerfulness >= 1) {

write\_log(this.params.name + " читает документацию.");

change\_cheerfulness(this, -1);

}

else {

write\_log(this.params.name + " пытается читать документацию, но из-за сильной усталости не может. " +

"Надо сделать перерыв.");

do\_something(this, Math.round(Math.random() \* (5 - 3) + 3 ));

}

}

}

}

class Coder extends Human {

constructor(name, gender, skill, language) {

super(name, gender, skill);

this.params.language = language;

this.params.specialty = "Программист";

// 5

this.talk = function() {

if(coders.length >= 2 || testers.length >= 1 || techwriters.length >= 1 ||

designers.length >= 1 || managers.length >= 1) {

write\_log(this.params.name + " разговаривает.");

}

}

// 7

this.code = function() {

if (this.params.cheerfulness >= 1) {

let rand\_lang = Math.floor(Math.random() \* (this.params.language.length));

write\_log(this.params.name + " программирует на " + this.params.language[rand\_lang] + ".");

increase\_project\_readiness(this.params.skill);

increase\_skill(this);

change\_cheerfulness(this, -1);

} else {

write\_log(this.params.name + " пытается программировать, но из-за сильной усталости не может. " + "Надо сделать перерыв.");

do\_something(this, Math.round(Math.random() \* (5 - 3) + 3 ));

}

}

}

}

class Tester extends Human {

constructor(name, gender, skill) {

super(name, gender, skill);

this.params.specialty = "Тестировщик";

// 5

this.talk = function() {

if(coders.length >= 1 || testers.length >= 2 || techwriters.length >= 1 ||

designers.length >= 1 || managers.length >= 1) {

write\_log(this.params.name + " разговаривает.");

}

}

// 7

this.test = function(how) {

if (this.params.cheerfulness >= 1) {

if ("вручную" == how) {

write\_log(this.params.name + " тестирует продукт " + how + ".");

} else if ("автотесты" == how) {

write\_log(this.params.name + " пишет " + how + ".");

}

increase\_project\_readiness(this.params.skill);

increase\_skill(this);

change\_cheerfulness(this, -1);

} else {

write\_log(this.params.name + " пытается тестировать продукт, но из-за сильной усталости не может. " + "Надо сделать перерыв.");

do\_something(this, Math.round(Math.random() \* (5 - 3) + 3 ));

}

}

// 8

this.make\_testing\_method = function() {

if (this.params.cheerfulness >= 1) {

if (this.params.skill >= 200) {

write\_log(this.params.name + " разрабатывает новую методику тестирования.");

}

increase\_project\_readiness(this.params.skill);

increase\_skill(this);

change\_cheerfulness(this, -1);

} else {

write\_log(this.params.name + " пытается разработать новую методику тестирования, " +

"но из-за сильной усталости не может. Надо сделать перерыв.");

do\_something(this, Math.round(Math.random() \* (5 - 3) + 3 ));

}

}

// 9

this.make\_testbench = function () {

if (this.params.cheerfulness >= 1) {

write\_log(this.params.name + " собирает испытательный стенд.");

increase\_project\_readiness(this.params.skill);

increase\_skill(this);

change\_cheerfulness(this, -1);

} else {

write\_log(this.params.name + " пытается собрать стенд, " +

"но из-за сильной усталости не может. Надо сделать перерыв.");

do\_something(this, Math.round(Math.random() \* (5 - 3) + 3 ));

}

}

}

}

class Techwriter extends Human {

constructor(name, gender, skill) {

super(name, gender, skill);

this.params.specialty = "Технический писатель";

// 5

this.talk = function() {

if(coders.length >= 1 || testers.length >= 1 || techwriters.length >= 2 ||

designers.length >= 1 || managers.length >= 1) {

write\_log(this.params.name + " разговаривает.");

}

}

// 7

this.write\_documentation = function () {

if (this.params.cheerfulness >= 1) {

write\_log(this.params.name + " пишет документацию.");

increase\_project\_readiness(this.params.skill);

increase\_skill(this);

change\_cheerfulness(this, -1);

} else {

write\_log(this.params.name + " пытается писать документ, но из-за сильной усталости не может. " + "Надо сделать перерыв.");

do\_something(this, Math.round(Math.random() \* (5 - 3) + 3 ));

}

}

}

}

class Designer extends Coder {

constructor(name, gender, skill, language) {

super(name, gender, skill, language);

this.params.specialty = "Веб-дизайнер";

// 5

this.talk = function() {

if(coders.length >= 1 || testers.length >= 1 || techwriters.length >= 1 ||

designers.length >= 2 || managers.length >= 1) {

write\_log(this.params.name + " разговаривает.");

}

}

// 8

this.create\_design = function () {

if (this.params.cheerfulness >= 1) {

write\_log(this.params.name + " придумывает дизайн продукта.");

increase\_project\_readiness(this.params.skill);

increase\_skill(this);

change\_cheerfulness(this, -1);

} else {

write\_log(this.params.name + " пытается придумать дизайн, " +

"но из-за сильной усталости не может. Надо сделать перерыв.");

do\_something(this, Math.round(Math.random() \* (5 - 3) + 3 ));

}

}

// 9

this.create\_UX = function () {

if (this.params.cheerfulness >= 1) {

write\_log(this.params.name + " проектирует взаимодействие с пользователем.");

increase\_project\_readiness(this.params.skill);

increase\_skill(this);

change\_cheerfulness(this, -1);

} else {

write\_log(this.params.name + " пытается проектировать взаимодействие с пользователем, " + "но из-за сильной усталости не может. Надо сделать перерыв.");

do\_something(this, Math.round(Math.random() \* (5 - 3) + 3 ));

}

}

// 10

this.draw = function () {

if (this.params.cheerfulness >= 1) {

write\_log(this.params.name + " рисует.");

increase\_project\_readiness(this.params.skill);

increase\_skill(this);

change\_cheerfulness(this, -1);

} else {

write\_log(this.params.name + " пытается рисовать, " +

"но из-за сильной усталости не может. Надо сделать перерыв.");

do\_something(this, Math.round(Math.random() \* (5 - 3) + 3 ));

}

}

}

}

class Manager extends Human {

constructor(name, gender, skill) {

super(name, gender, skill);

this.params.specialty = "Менеджер";

// 5

this.talk = function() {

if(coders.length >= 1 || testers.length >= 1 || techwriters.length >= 1 ||

designers.length >= 1 || managers.length >= 2) {

write\_log(this.params.name + " разговаривает.");

}

}

// 7

this.organize\_meeting = function () {

write\_log(this.params.name + " организует совещание.");

increase\_project\_readiness(this.params.skill);

}

// 8

this.communicate\_with\_customer = function () {

if (this.params.cheerfulness >= 1) {

write\_log(this.params.name + " общается с заказчиком.");

increase\_project\_readiness(this.params.skill);

increase\_skill(this);

change\_cheerfulness(this, -1);

} else {

write\_log(this.params.name + " пытается общаться с заказчиком, " +

"но из-за сильной усталости срывается. Надо сделать перерыв.");

do\_something(this, Math.round(Math.random() \* (5 - 3) + 3 ));

}

}

}

}

Файл functions.js содержит описание всех функций:

function get\_time() {

let now = new Date();

let hour = now.getHours();

let minute = now.getMinutes();

let second = now.getSeconds();

let millisecond = now.getMilliseconds();

return hour + ":" + minute + ":" + second + ":" + millisecond;

}

function write\_log(something) {

$('#simulation\_log').append(get\_time() + " " + something + "<br>");

document.getElementById('simulation\_log').scrollTop = 9999;

}

function do\_something(who, what) {

switch (what) {

case 1: {

who.come\_to\_office();

break;

}

case 2: {

who.go\_home();

break;

}

case 3: {

let drinks = ["чай", "кофе", "какао", "водичку", "сок", "квас"];

let rand\_drink = Math.round(Math.random() \* (drinks.length - 1));

who.drink(drinks[rand\_drink]);

break;

}

case 4: {

who.laugh();

break;

}

case 5: {

who.read\_documentation();

break;

}

case 6: {

who.talk();

break;

}

case 7: {

if (who instanceof Coder || who instanceof Designer) {

who.code();

} else if (who instanceof Tester) {

let test\_methods = ["вручную", "автотесты"];

let rand\_method = Math.round(Math.random() \* (test\_methods.length - 1));

who.test(test\_methods[rand\_method]);

} else if (who instanceof Techwriter) {

who.write\_documentation();

} else if (who instanceof Manager) {

who.organize\_meeting();

}

break;

}

case 8: {

if (who instanceof Tester) {

who.make\_testing\_method();

} else if (who instanceof Designer) {

who.create\_design();

} else if (who instanceof Manager) {

who.communicate\_with\_customer();

}

break;

}

case 9: {

if (who instanceof Tester) {

who.make\_testbench();

} else if (who instanceof Designer) {

who.create\_UX();

}

break;

}

case 10: {

if (who instanceof Designer) {

who.draw();

}

break;

}

}

if (current\_project.readiness >= current\_project.difficulty) {

write\_log("Всем премию!");

alert("Команда закончила проект раньше срока!\nВсем премию!");

// Остановка всех таймаутов на странице:

var id = window.setTimeout(function() {}, 0);

while (id--) {

window.clearTimeout(id);

}

}

}

function start\_working\_day(who) {

// Идентификатор интервала для каждого конкретного человека хранится в params.work\_interval:

who.params.work\_interval = setInterval(function () {

if (who.params.start\_day) {

do\_something(who, 1);

who.params.start\_day = false;

}

let rand\_method = Math.round(Math.random() \* (Object.getOwnPropertyNames(who).length - 2) + 3);

do\_something(who, rand\_method);

}, 1000);

}

function start\_working\_day\_for(whom) {

if (whom.length) {

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

start\_working\_day(whom[i]);

}

}

}

function start\_working\_day\_all() {

start\_working\_day\_for(coders);

start\_working\_day\_for(testers);

start\_working\_day\_for(techwriters);

start\_working\_day\_for(designers);

start\_working\_day\_for(managers);

}

function initialize\_day() {

start\_working\_day\_all();

setTimeout(function () {

end\_working\_day\_all();

}, DAY\_PERIOD);

let day\_counter = 1;

$('#deadline\_progress\_bar').prop("value", day\_counter);

// Для обеспечения правильных интервалов между рабочими днями.

// Во всём виновата чёртова асинхронность.

for (let i = 1; i < current\_project.planned\_time; i++) {

setTimeout(function () {

start\_working\_day\_all();

}, (DAY\_PERIOD \* i + NIGHT\_PERIOD \* i));

setTimeout(function () {

end\_working\_day\_all();

day\_counter++;

$('#deadline\_progress\_bar').prop("value", day\_counter);

if (day\_counter == current\_project.planned\_time) {

if (current\_project.readiness < current\_project.difficulty) {

write\_log("Всех уволить!");

alert("Команда не справилась в срок!\nВсех уволить!");

} else {

write\_log("Закончили в срок!");

alert("Команда закончила ровно в срок!")

}

}

}, (DAY\_PERIOD \* (i + 1) + NIGHT\_PERIOD \* i));

}

}

function end\_working\_day(who) {

do\_something(who, 2);

clearInterval(who.params.work\_interval);

who.params.start\_day = true;

}

function end\_working\_day\_for(whom) {

if (whom.length) {

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

end\_working\_day(whom[i]);

}

}

}

function end\_working\_day\_all() {

end\_working\_day\_for(coders);

end\_working\_day\_for(testers);

end\_working\_day\_for(techwriters);

end\_working\_day\_for(designers);

end\_working\_day\_for(managers);

}

function count\_all\_people() {

return(coders.length + testers.length + techwriters.length + designers.length + managers.length);

}

function show\_all\_people() {

let coders\_number = "<p>Число программистов: " + coders.length + "</p>";

let testers\_number = "<p>Число тестировщиков: " + testers.length + "</p>";

let techwriters\_number = "<p>Число технических писателей: " + techwriters.length + "</p>";

let designers\_number = "<p>Число веб-дизайнеров: " + designers.length + "</p>";

let managers\_number = "<p>Число менеджеров: " + managers.length + "</p>";

$('#wanna\_add\_another\_human').html("<p>Добавить ещё сотрудника или перейти к настройкам?</p><br>" + coders\_number +

testers\_number + techwriters\_number + designers\_number + managers\_number);

}

function add\_human(who, where, name, gender, skill, language) {

let current\_type = "";

if ("coder" == who) {

where[where.length] = new Coder(name, gender, skill, language);

coders[coders.length - 1].params.place\_x = workplace\_x;

coders[coders.length - 1].params.place\_y = workplace\_y;

current\_type = coders[coders.length - 1].params.specialty;

} else if ("tester" == who) {

where[where.length] = new Tester(name, gender, skill);

testers[testers.length - 1].params.place\_x = workplace\_x;

testers[testers.length - 1].params.place\_y = workplace\_y;

current\_type = testers[testers.length - 1].params.specialty;

} else if ("techwriter" == who) {

where[where.length] = new Techwriter(name, gender, skill);

techwriters[techwriters.length - 1].params.place\_x = workplace\_x;

techwriters[techwriters.length - 1].params.place\_y = workplace\_y;

current\_type = techwriters[techwriters.length - 1].params.specialty;

} else if ("designer" == who) {

where[where.length] = new Designer(name, gender, skill, language);

designers[designers.length - 1].params.place\_x = workplace\_x;

designers[designers.length - 1].params.place\_y = workplace\_y;

current\_type = designers[designers.length - 1].params.specialty;

} else if ("manager" == who) {

where[where.length] = new Manager(name, gender, skill);

managers[managers.length - 1].params.place\_x = workplace\_x;

managers[managers.length - 1].params.place\_y = workplace\_y;

current\_type = managers[managers.length - 1].params.specialty;

}

let dude\_gender = "";

if ("m" == gender) {

dude\_gender = "<img src='img/dude.png'>";

} else if ("f" == gender) {

dude\_gender = "<img src='img/dude\_f.png'>";

}

$('#workplace\_' + workplace\_y + '\_' + workplace\_x).append("<div class='dude'>" +

"<p class='text\_small'>" + current\_type + " " + name + "</p>" +

"<p id='skill\_" + workplace\_y + "\_" + workplace\_x + "' class='text\_small'>Опыт: " + skill + "</p>" +

"<p id='cheerfulness\_" + workplace\_y + "\_" + workplace\_x + "' class='text\_small'>Настроение: " + 25 + "</p>" +

"<br>" + dude\_gender + "</div>");

if (workplace\_x < 4) {

workplace\_x++;

} else {

workplace\_x = 0;

if (workplace\_y < 3) {

workplace\_y++;

}

}

}

function get\_gender() {

gender\_val = "";

if ($('#' + $('#human\_selection').val() + '\_gender\_m').prop("checked")) {

gender\_val = "m";

} else if ($('#' + $('#human\_selection').val() + '\_gender\_f').prop("checked")) {

gender\_val = "f";

}

}

function increase\_skill(who) {

who.params.skill++;

$('#skill\_' + who.params.place\_y + '\_' + who.params.place\_x).text("Опыт: " + who.params.skill);

}

function change\_cheerfulness(who, how) {

who.params.cheerfulness += how;

$('#cheerfulness\_' + who.params.place\_y + '\_' + who.params.place\_x).text("Настроение: " + who.params.cheerfulness);

}

function increase\_project\_readiness(skill) {

let difficulty\_percent = current\_project.difficulty / 100;

let skill\_percent = skill;

let delta = skill\_percent \* difficulty\_percent

if (current\_project.readiness + delta <= current\_project.difficulty) {

current\_project.readiness += delta;

} else {

current\_project.readiness = current\_project.difficulty;

write\_log("Проект завершён.");

}

$('#project\_progress\_bar').prop("value", current\_project.readiness);

}

Файл index.html содержит разметку страницы и jQuery-обработчики, которые из-за ограничений области видимости невозможно переместить в другое место:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Coders</title>

<link rel="stylesheet" href="reset\_CSS.css">

<link rel="stylesheet" href="style.css">

<script src="jquery-3.2.1.min.js"></script>

<script src="main.js"></script>

<script src="functions.js"></script>

<script src="classes.js"></script>

</head>

<body>

<div id="choose\_human\_form" class="info\_fields">

<p class="text\_center">Добавьте сотрудника:</p>

<select id="human\_selection" class="field choose\_human\_form\_input">

<option value="coder" id="select\_coder">Программист</option>

<option value="tester" id="select\_tester">Тестировщик</option>

<option value="techwriter" id="select\_techwriter">Технический писатель</option>

<option value="designer" id="select\_designer">Веб-дизайнер</option>

<option value="manager" id="select\_manager">Менеджер</option>

</select>

<button id="submit\_choose\_human\_form" class="button button\_select">Выбрать</button>

</div>

<div id="add\_coder\_form" class="info\_fields">

<p class="text\_center">Новый программист:</p>

<div class="add\_form\_group\_span">

<span class="add\_form\_text">Имя:</span>

<span class="add\_form\_text">Пол:</span>

<span class="add\_form\_text">Опыт:</span>

<span class="add\_form\_text">Язык:</span>

</div>

<div class="add\_form\_group\_input">

<input type="text" id="coder\_name" name="name" class="add\_form\_input"/><br>

<input type="radio" class="radio" id="coder\_gender\_m" value="m" name="gender" />

<span class="add\_form\_text">М</span>

<input type="radio" class="radio" id="coder\_gender\_f" value="f" name="gender" />

<span class="add\_form\_text">Ж</span><br>

<input type="range" id="coder\_skill" min="1" max="10" step="1" value="5">

<input type="text" id="coder\_lang\_0" name="lang" class="add\_form\_input"/>

<button id="add\_coder\_lang" class="button">+</button>

</div>

<br>

<button id="submit\_add\_coder\_form" class="button button\_ok">Готово</button>

</div>

<div id="add\_tester\_form" class="info\_fields">

<p class="text\_center">Новый тестировщик:</p>

<div class="add\_form\_group\_span">

<span class="add\_form\_text">Имя:</span>

<span class="add\_form\_text">Пол:</span>

<span class="add\_form\_text">Опыт:</span>

</div>

<div class="add\_form\_group\_input">

<input type="text" id="tester\_name" name="name" class="add\_form\_input"/><br>

<input type="radio" class="radio" id="tester\_gender\_m" value="m" name="gender" />

<span class="add\_form\_text">М</span>

<input type="radio" class="radio" id="tester\_gender\_f" value="f" name="gender" />

<span class="add\_form\_text">Ж</span><br>

<input type="range" id="tester\_skill" min="1" max="10" step="1" value="5">

</div>

<br>

<button id="submit\_add\_tester\_form" class="button button\_ok">Готово</button>

</div>

<div id="add\_techwriter\_form" class="info\_fields">

<p class="text\_center">Новый технический писатель:</p>

<div class="add\_form\_group\_span">

<span class="add\_form\_text">Имя:</span>

<span class="add\_form\_text">Пол:</span>

<span class="add\_form\_text">Опыт:</span>

</div>

<div class="add\_form\_group\_input">

<input type="text" id="techwriter\_name" name="name" class="add\_form\_input"/><br>

<input type="radio" class="radio" id="techwriter\_gender\_m" value="m" name="gender" />

<span class="add\_form\_text">М</span>

<input type="radio" class="radio" id="techwriter\_gender\_f" value="f" name="gender" />

<span class="add\_form\_text">Ж</span><br>

<input type="range" id="techwriter\_skill" min="1" max="10" step="1" value="5">

</div>

<br>

<button id="submit\_add\_techwriter\_form" class="button button\_ok">Готово</button>

</div>

<div id="add\_designer\_form" class="info\_fields">

<p class="text\_center">Новый веб-дизайнер:</p>

<div class="add\_form\_group\_span">

<span class="add\_form\_text">Имя:</span>

<span class="add\_form\_text">Пол:</span>

<span class="add\_form\_text">Опыт:</span>

<span class="add\_form\_text">Язык:</span>

</div>

<div class="add\_form\_group\_input">

<input type="text" id="designer\_name" name="name" class="add\_form\_input"/><br>

<input type="radio" class="radio" id="designer\_gender\_m" value="m" name="gender" />

<span class="add\_form\_text">М</span>

<input type="radio" class="radio" id="designer\_gender\_f" value="f" name="gender" />

<span class="add\_form\_text">Ж</span><br>

<input type="range" id="designer\_skill" min="1" max="10" step="1" value="5">

<input type="text" id="designer\_lang\_0" name="lang" class="add\_form\_input"/>

<button id="add\_designer\_lang" class="button">+</button>

</div>

<br>

<button id="submit\_add\_designer\_form" class="button button\_ok">Готово</button>

</div>

<div id="add\_manager\_form" class="info\_fields">

<p class="text\_center">Новый менеджер:</p>

<div class="add\_form\_group\_span">

<span class="add\_form\_text">Имя:</span>

<span class="add\_form\_text">Пол:</span>

<span class="add\_form\_text">Опыт:</span>

</div>

<div class="add\_form\_group\_input">

<input type="text" id="manager\_name" name="name" class="add\_form\_input"/><br>

<input type="radio" class="radio" id="manager\_gender\_m" value="m" name="gender" />

<span class="add\_form\_text">М</span>

<input type="radio" class="radio" id="manager\_gender\_f" value="f" name="gender" />

<span class="add\_form\_text">Ж</span><br>

<input type="range" id="manager\_skill" min="1" max="10" step="1" value="5">

</div>

<br>

<button id="submit\_add\_manager\_form" class="button button\_ok">Готово</button>

</div>

<div id="wanna\_add\_another\_human" class="text label">Добавить ещё сотрудника или перейти к настройкам?</div>

<button id="button\_settings" class="button button\_big">Настройки</button>

<div id="settings\_form" class="settings\_fields">

<p class="text\_center">Настройки:</p>

<div class="settings\_form\_group\_span">

<span class="add\_form\_text">Режим аварии:</span>

<br>

<span class="add\_form\_text">Время на проект:</span>

<br>

<span class="add\_form\_text">Сложность проекта:</span>

</div>

<div class="settings\_form\_group\_input">

<input id="emergency\_checkbox" type="checkbox">

<br>

<input type="text" id="time\_for\_project\_input" class="settings\_form\_input" value="5"/>

<span class="add\_form\_text">рабочих дней</span>

<input type="range" id="project\_difficulty" min="100" max="10000" step="100" value="5000">

</div>

</div>

<button id="button\_simulation\_start" class="button button\_big">Начать симуляцию</button>

<div id="simulation\_screen">

<div id="simulation\_office\_screen">

<div id="workplace\_line\_0" class="workplace\_line">

<div id="workplace\_0\_0" class="workplace"></div>

<div id="workplace\_0\_1" class="workplace"></div>

<div id="workplace\_0\_2" class="workplace"></div>

<div id="workplace\_0\_3" class="workplace"></div>

<div id="workplace\_0\_4" class="workplace"></div>

</div>

<div id="workplace\_line\_1" class="workplace\_line">

<div id="workplace\_1\_0" class="workplace"></div>

<div id="workplace\_1\_1" class="workplace"></div>

<div id="workplace\_1\_2" class="workplace"></div>

<div id="workplace\_1\_3" class="workplace"></div>

<div id="workplace\_1\_4" class="workplace"></div>

</div>

<div id="workplace\_line\_2" class="workplace\_line">

<div id="workplace\_2\_0" class="workplace"></div>

<div id="workplace\_2\_1" class="workplace"></div>

<div id="workplace\_2\_2" class="workplace"></div>

<div id="workplace\_2\_3" class="workplace"></div>

<div id="workplace\_2\_4" class="workplace"></div>

</div>

<div id="workplace\_line\_3" class="workplace\_line">

<div id="workplace\_3\_0" class="workplace"></div>

<div id="workplace\_3\_1" class="workplace"></div>

<div id="workplace\_3\_2" class="workplace"></div>

<div id="workplace\_3\_3" class="workplace"></div>

<div id="workplace\_3\_4" class="workplace"></div>

</div>

</div>

<div id="deadline\_progress" class="progress">

<p class="add\_form\_text">Время близится к дедлайну:</p>

<br>

<progress id="deadline\_progress\_bar" class="progress\_bar" max="100" value="0"></progress>

</div>

<div id="project\_progress" class="progress">

<p class="add\_form\_text">Уровень готовности проекта:</p>

<br>

<progress id="project\_progress\_bar" class="progress\_bar" max="100" value="0"></progress>

</div>

<div id="simulation\_log"></div>

</div>

<script>

// Ограничения области видимости не позволяют перенести код в какой-то из модулей.

$('#submit\_choose\_human\_form').click(function() {

$('#choose\_human\_form').css("visibility", "hidden");

$('#wanna\_add\_another\_human').css("visibility", "hidden");

$('#button\_settings').css("visibility", "hidden");

$('#add\_' + $('#human\_selection').val() + '\_form').css("visibility", "visible");

});

$('#submit\_add\_coder\_form').click(function() {

get\_gender();

let coder\_lang\_arr = [];

for (let i = 0; i <= coder\_lang\_index; i++) {

coder\_lang\_arr[coder\_lang\_arr.length] = $('#coder\_lang\_' + i).val();

}

add\_human ("coder", coders, $('#coder\_name').val(), gender\_val, $('#coder\_skill').val(),

coder\_lang\_arr);

show\_all\_people();

if (count\_all\_people() < 20) {

$('#add\_' + $('#human\_selection').val() + '\_form').css("visibility", "hidden");

$('#wanna\_add\_another\_human').css("visibility", "visible");

$('#choose\_human\_form').css("visibility", "visible");

$('#button\_settings').css("visibility", "visible");

$('#coder\_name').val("");

$('#coder\_gender\_m').prop("checked", false);

$('#coder\_gender\_f').prop("checked", false);

$('#coder\_skill').val("");

$('#coder\_lang\_0').val("");

for (let i = 1; i <= coder\_lang\_index; i++) {

$('#coder\_lang\_' + i).remove();

}

coder\_lang\_index = 0;

} else {

$('#add\_' + $('#human\_selection').val() + '\_form').css("visibility", "hidden");

$('#button\_settings').css("visibility", "hidden");

$('#settings\_form').css("visibility", "visible");

$('#button\_simulation\_start').css("visibility", "visible");

}

});

$('#submit\_add\_tester\_form').click(function() {

get\_gender();

add\_human ("tester", testers, $('#tester\_name').val(), gender\_val, $('#tester\_skill').val());

show\_all\_people();

if (count\_all\_people() < 20) {

$('#add\_' + $('#human\_selection').val() + '\_form').css("visibility", "hidden");

$('#wanna\_add\_another\_human').css("visibility", "visible");

$('#choose\_human\_form').css("visibility", "visible");

$('#button\_settings').css("visibility", "visible");

$('#tester\_name').val("");

$('#tester\_gender\_m').prop("checked", false);

$('#tester\_gender\_f').prop("checked", false);

$('#tester\_skill').val("");

}

});

$('#submit\_add\_techwriter\_form').click(function() {

get\_gender();

add\_human ("techwriter", techwriters, $('#techwriter\_name').val(), gender\_val, $('#techwriter\_skill').val());

show\_all\_people();

if (count\_all\_people() < 20) {

$('#add\_' + $('#human\_selection').val() + '\_form').css("visibility", "hidden");

$('#wanna\_add\_another\_human').css("visibility", "visible");

$('#choose\_human\_form').css("visibility", "visible");

$('#button\_settings').css("visibility", "visible");

$('#techwriter\_name').val("");

$('#techwriter\_gender\_m').prop("checked", false);

$('#techwriter\_gender\_f').prop("checked", false);

$('#techwriter\_skill').val("");

}

});

$('#submit\_add\_designer\_form').click(function() {

get\_gender();

let designer\_lang\_arr = [];

for (let i = 0; i <= designer\_lang\_index; i++) {

designer\_lang\_arr[designer\_lang\_arr.length] = $('#designer\_lang\_' + i).val();

}

add\_human ("designer", designers, $('#designer\_name').val(), gender\_val, $('#designer\_skill').val(),

designer\_lang\_arr);

show\_all\_people();

if (count\_all\_people() < 20) {

$('#add\_' + $('#human\_selection').val() + '\_form').css("visibility", "hidden");

$('#wanna\_add\_another\_human').css("visibility", "visible");

$('#choose\_human\_form').css("visibility", "visible");

$('#button\_settings').css("visibility", "visible");

$('#designer\_name').val("");

$('#designer\_gender\_m').prop("checked", false);

$('#designer\_gender\_f').prop("checked", false);

$('#designer\_skill').val("");

$('#designer\_lang\_0').val("");

for (let i = 1; i <= designer\_lang\_index; i++) {

$('#designer\_lang\_' + i).remove();

}

designer\_lang\_index = 0;

}

});

$('#submit\_add\_manager\_form').click(function() {

get\_gender();

add\_human ("manager", managers, $('#manager\_name').val(), gender\_val, $('#manager\_skill').val());

show\_all\_people();

if (count\_all\_people() < 20) {

$('#add\_' + $('#human\_selection').val() + '\_form').css("visibility", "hidden");

$('#wanna\_add\_another\_human').css("visibility", "visible");

$('#choose\_human\_form').css("visibility", "visible");

$('#button\_settings').css("visibility", "visible");

$('#manager\_name').val("");

$('#manager\_gender\_m').prop("checked", false);

$('#manager\_gender\_f').prop("checked", false);

$('#manager\_skill').val("");

}

});

// TODO: Добавить возможность удалять пустые поля.

$('#add\_coder\_lang').click(function () {

if (coder\_lang\_index <= 6) {

$('#coder\_lang\_' + coder\_lang\_index).after("<input type=\"text\" id='coder\_lang\_" +

(coder\_lang\_index + 1) + "' name=\"lang'\" class=\"add\_form\_input\"/>");

coder\_lang\_index++;

if (coder\_lang\_index == 6) {

$('#add\_coder\_lang').css("visibility", "hidden");

}

}

});

$('#add\_designer\_lang').click(function () {

$('#designer\_lang\_' + designer\_lang\_index).after("<input type=\"text\" id='designer\_lang\_" +

(designer\_lang\_index + 1) + "' name=\"lang'\" class=\"add\_form\_input\"/>");

designer\_lang\_index++;

});

$('#button\_settings').click(function () {

$('#wanna\_add\_another\_human').css("visibility", "hidden");

$('#choose\_human\_form').css("visibility", "hidden");

$('#button\_settings').css("visibility", "hidden");

$('#settings\_form').css("visibility", "visible");

$('#button\_simulation\_start').css("visibility", "visible");

});

$('#button\_simulation\_start').click(function () {

$('#settings\_form').css("visibility", "hidden");

$('#button\_simulation\_start').css("visibility", "hidden");

$('#simulation\_screen').css("visibility", "visible");

let time\_for\_project = $('#time\_for\_project\_input').val();

let project\_difficulty = $('#project\_difficulty').val();

let project\_emergency = $('#emergency\_checkbox').prop("checked");

current\_project = new Project(time\_for\_project, project\_difficulty, project\_emergency);

$('#project\_progress\_bar').prop("max", current\_project.difficulty);

$('#deadline\_progress\_bar').prop("max", current\_project.planned\_time);

// alert($('#project\_progress\_bar').prop("max"));

initialize\_day();

// alert(JSON.stringify(current\_project));

});

$(document).ready(function () {

});

$(window).resize(function () {

alert("Не делай этого!");

});

</script>

</body>

</html>

Файлы каскадных таблиц стилей style.css и reset\_CSS.css, а также сторонняя библиотека jquery.js не приводятся в целях экономии денег на печать, а также потому, что не существенны для понимания работы программы.