

ПРИЛОЖЕНИЕ А

(обязательное)

Техническое задание

Листов 10

**Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой ИУ6

А.В. Пролетарский
« 21 » февраля 2020 г.

**ПРОГРАММНАЯ СИСТЕМА МОДЕЛИРОВАНИЯ ИСКУССТВЕННОЙ
ЖИЗНИ С ИСПОЛЬЗОВАНИЕМ ЦИФРОВЫХ АВТОМАТОВ**

Техническое задание

Листов 10

Студент

ИУ6-85Б
(Группа)



(Подпись, дата)

В.Д. Шульман

(И.О. Фамилия)

Руководитель



(Подпись, дата)

О.Ю. Ерёмин

(И.О. Фамилия)

1 ВВЕДЕНИЕ

Настоящее техническое задание распространяется на разработку программной системы «Программная система моделирования искусственной жизни с использованием цифровых автоматов», создаваемой для моделирования самоорганизующихся систем с использованием генетического алгоритма.

Актуальность программной системы обусловлена тем, что на данный момент весьма сильно стали популярны решения задач с использованием генетических алгоритмов и машинного обучения, которые позволяют решать задачи класса NP (недетерминированные полиномиальные). Оптимизация решения задач, не решаемые в виде какого-либо детерминированного алгоритма является одним из наиболее популярных направлений математики, информатики, криптографии и теории алгоритмов. Использование генетических алгоритмов позволяет избегать ситуации, когда решение задач требует экспоненциальное время работы в случае использования классических детерминированных алгоритмов.

Программная система позволяет моделировать эволюционирующие самоорганизующиеся системы на подобие биологических систем, запуская сеансы моделирования с различными параметрами, задавать критерии эффективности, отбора, формировать выходной поток данных для анализа результатов моделирования. Данная программная система ориентирована на пользователей, которые обладают общими знаниями в информатике, математике и программировании и имеющие представления о генетических алгоритмах и принципах имитационного моделирования.

2 ОСНОВАНИЯ ДЛЯ РАЗРАБОТКИ

Основанием для разработки программы является учебный план кафедры ИУ6 «Компьютерные системы и сети» факультета ИУ «Информатика и системы управления» МГТУ им. Баумана, утверждённого в установленном порядке.

3 НАЗНАЧЕНИЕ РАЗРАБОТКИ

Основным эксплуатационным назначением программной системы является представление пользователям сервиса для осуществления имитационного моделирования систем искусственной жизни с помощью задания множества параметров и получения результатов моделирование с помощью выходного потока данных, представленного в графическом формате (таблицы, гистограммы, графики и т.д.)

4 ИСХОДНЫЕ ДАННЫЕ, ЦЕЛИ И ЗАДАЧИ

4.1 Исходные данные

4.1.1 Исходными данными для разработки являются следующие материалы:

4.1.1.1 Перечень работ, содержащих исходные данные для разработки

– Эволюция и искусственная жизнь. <https://www.computer-museum.ru/histsoft/alife.htm>;

– Космики: моделирование эволюции многоклеточных организмов. Отчет за 7 лет. <https://habr.com/ru/post/458612>;

– Практика реализации генетических алгоритмов. <https://www.youtube.com/watch?v=OMkCW5NihA> и <https://www.youtube.com/watch?v=S1ADSNWyKwQ>;

– Создание «искусственной жизни» на компьютере. <https://www.pvsm.ru/programmirovaniye/287532>;

– Эволюционирующие клеточные автоматы. <https://habr.com/ru/post/455958/>;

– Симуляции колонии растений в замкнутой среде <https://vk.com/@evgbarish-istoriya-o-simulyacii-kolonii-rastenii-s-pechalnoi-razvyazko>;

– Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы / Под ред. В. М. Курейчика. — 2-е изд., исправл. и доп. — М.: ФИЗМАТЛИТ, 2010. — 368 с.

– Атлас клеточных автоматов Стивена Вольфрама. <http://atlas.wolfram.com>;

– Primer. Научный популяризатор эволюционных процессов и теории игр.
<https://www.patreon.com/primerlearning>.

4.1.1.2 Перечень прототипов

– CyberBiology. Симулятор жизни на компьютере, представляющие из себя десктоп приложение, написанное на языке программирования Java.
<https://github.com/CyberBiology/CyberBiology>;

– Программа Генезис проекта Кибербиология задумана для исследования образования и эволюции видов в условиях разделенных ареалов обитания.
<https://github.com/CyberBiology/Genesis>;

– Construct. Нативный симулятор эволюции цифровых организмов на основе JavaScript/ES6. Он используется для изучения эволюционной биологии самовоспроизводящихся и эволюционирующих компьютерных программ (цифровых организмов). <https://github.com/tmptrash/construct>;

– Areal. Программа для симуляции роста и размножения одноклеточных растения в замкнутой среде. Статья <https://vk.com/@evgbarish-istoriya-o-simulyacii-kolonii-rastanii-s-pechalnoi-razvyazko>, сам проект <https://yadi.sk/d/d0ZHqghsFVaM4w>;

– CellLife. Небольшая программа, симулирующая эволюцию клеток, написанная на языке программирования C#. <https://github.com/Elco-/CellLife>.

4.1.1.3 Техническое задание на выпускную квалификационную работу бакалавра

4.1.1.4 Курсовая научно-исследовательская работа студента (КНИРС)

4.2 Цель работы

Целью работы является прототип программной системы моделирования искусственной жизни с использованием цифровых автоматов для моделирования эволюционных процессов.

4.3 Решаемые задачи

4.3.1 Выбор модели жизненного цикла, архитектуры, подхода к разработке, технологии, методов, стандартов и средств разработки программной системы

4.3.2 Анализ требований технического задания с точки зрения выбранной технологии и уточнение требований к программной системе: техническим средствам, внешним интерфейсам.

4.3.3 Исследование предметной области – выбор моделей, описывающих предметную область, выбор генетического алгоритма и метода селекции.

4.3.4 Определение архитектуры программной системы: разработка ее структуры, определение набора необходимого оборудования, программного обеспечения и процессов обслуживания.

4.3.5 Анализ требований технического задания и разработка архитектуры программного обеспечения.

4.3.6 Разработка структуры программного обеспечения и определение программных компонентов системы.

4.3.7 Проектирование компонентов программного продукта: frontend-приложение, backend-сервер, база данных.

4.3.8 Обеспечение модульности и масштабируемости программной системы.

4.3.9 Реализация компонентов программной системы с использованием выбранных средств разработки.

4.3.10 Сборка программного обеспечения и проверка корректности взаимодействия между его компонентами.

4.3.11 Разработка методологии функционального тестирования, модульного тестирования и UX тестирования программной системы.

5 ТРЕБОВАНИЯ К ПРОГРАММНОЙ СИСТЕМЕ

5.1 Требования к функциональным характеристикам

5.1.1 Выполняемые функции

5.1.1.1 Для пользователя:

- Регистрация;
- Авторизация;
- Редактирование профиля;
- Создание и настройка нового сеанса моделирования;
- Запуск сеанса моделирования;
- Остановка сеанса моделирования;
- Возобновление сеанса моделирования;
- Завершение сеанса моделирования;
- Просмотр текущего состояния сеанса моделирования;
- Просмотр результатов завершенного сеанса моделирования;
- Просмотр списка сеансов моделирования.

5.2 Требования к надежности

5.2.1 Предусмотреть контроль вводимой информации.

5.2.2 Предусмотреть защиту от некорректных действий пользователя.

5.2.3 Предусмотреть защиту от взлома учетной записи или несанкционированного доступа к данным пользователей системы.

5.2.4 Обеспечить целостность информации в базе данных.

5.2.5 Предусмотреть восстановления системы в случае сбоя

5.3 Условия эксплуатации

5.3.1 Условия эксплуатации в соответствии с СанПиН 2.2.2/2.4.1340-03.

5.4 Требования к составу и параметрам технических средств

5.4.1 Программное обеспечение должно функционировать на кластере из IBM-совместимых компьютеров.

5.4.2 Минимальная конфигурация для каждого технического средства, входящего в состав кластера:

5.4.2.1 Тип процессора Intel Core.

5.4.2.2 Объем ОЗУ 4 Гб.

5.4.2.3 Объем ПЗУ 128 Гб.

5.5 Требования к информационной и программной совместимости

5.5.1 Программное обеспечение должно работать под управлением операционных систем семейства WIN64 (Windows 7, Windows 8, Windows 8.1, Windows 10) и операционных системах семейства Linux.

5.5.2 Входные данные должны быть представлены в следующем формате: текст, выбираемый из выпадающего списка, вводимый пользователем текст, контекстный выбор. Программа работает с кодировкой ANSI.

5.5.3 Результаты должны быть представлены в следующем формате: таблицы, списки, текст, графики.

5.5.4 Программное обеспечение должно поддерживать сетевые протоколы HTTP и GRPC для осуществления обмена данными между компонентами программной системы.

5.6 Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

5.7 Требования к транспортированию и хранению

Требования к транспортировке и хранению не предъявляются.

5.8 Специальные требования

Специальные требования не предъявляются.

6 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

6.1 Разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии.

6.2 Разрабатываемое программное обеспечение должно включать справочную систему.

6.3 В состав сопровождающей документации должны входить:

6.3.1 Расчетно-пояснительная записка на 65-75 листах формата А4 (без приложений).

6.3.2 Техническое задание (Приложение А).

6.3.3 Руководство пользователя (Приложение Б).

6.3.4 Исходный текст кода и конфигурационных файлов компонентов программной системы (Приложение В).

6.4 Графическая часть должна быть выполнена на 6 листах формата А1 (Приложение Г):

6.4.1 Концептуальная модель предметной области.

6.4.2 Схема структурная программной системы.

6.4.3 Диаграмма вариантов использования программной системы.

6.4.4 Граф состояний интерфейса.

6.4.5 Дatalogическая схема базы данных программной системы.

6.4.6 Диаграмма размещения программных компонентов системы.

7 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Выполнить технико-экономическое обоснование разработки.

8. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

№	Название этапа	Срок, даты, %	Отчетность
1.	Разработка технического задания	2.02.2020 - 29.02.2020 5 %	Утвержденное техническое задание и задание на выпускную квалификационную работу
2.	Анализ требований и уточнение спецификаций (эскизный проект)	01.03.2020- 12.03.2020 8 %	Спецификации программного обеспечения.
3.	Проектирование структуры программного обеспечения, проектирование компонентов (технический проект)	13.03.2020- 02.04.2020 35 %	Схема структурная системы и спецификации компонентов. Проектная документация: схемы, диаграммы и т.п.
4.	Реализация компонентов и автономное тестирование компонентов. Сборка и комплексное тестирование. Оценочное тестирование и рабочий проект.	13.03.2020- 18.04.2020 50 %	Тексты программных компонентов. Тесты, результаты тестирования.
5.	Разработка документации.	18.04.2020- 25.05.2020 80 %	Расчетно-пояснительная записка.

№	Название этапа	Срок, даты, %	Отчетность
6.	Прохождение нормоконтроля, проверка на антиплагиат, получение рецензии, подготовка доклада и предзащита.	25.05.2020- 7.06.2020 90 %	Иллюстративный материал, доклад, рецензия, справки о нормоконтроле и проценте плагиата.
7.	Защита выпускной квалификационной работы.	8.06.2020- 04.07.2020 100 %	

9 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ

9.1 Порядок контроля

Контроль выполнения осуществляется руководителем еженедельно.

9.2 Порядок защиты

Защита осуществляется перед государственной экзаменационной комиссией (ГЭК).

9.3 Срок защиты

Срок защиты определяется в соответствии с планом заседаний ГЭК.

10 ПРИМЕЧАНИЕ

В процессе выполнения работы возможно уточнение отдельных требований технического задания по взаимному согласованию руководителя и исполнителя.

ПРИЛОЖЕНИЕ Б

(обязательное)

Руководство пользователя

Листов 8

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего
образования
«Московский государственный технический университет имени Н.Э. Баумана»
(национальный исследовательский университет)
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»
Кафедра «Компьютерные системы и сети»

Программная система моделирования искусственной жизни с использованием цифровых автоматов

Руководство пользователя

Листов 7

Руководитель _____ О.Ю. Ерёмин

Студент ИУ6-85Б _____  В.Д. Шульман

2020 г.

СОДЕРЖАНИЕ

1	Общие сведения о программном продукте.....	85
2	Структура и настройка.....	85
3	Вход в систему и проверка работоспособности	85
4	Создание и просмотр своих сеансов моделирования.....	87
5	Взаимодействие с другими пользователями и просмотр чужих сеансов моделирования.....	89

1 Общие сведения о программном продукте

Программная система позволяет проводить сеансы моделирования эволюционных процессов, которые могут быть гибко настроены пользователем системы.

В рамках программной системы пользователь способен также просматривать сеансы моделирования других пользователей и взаимодействовать с ним с помощью текстовых сообщений, которыми можно обмениваться в рамках данной программной системы.

В рамках системы каждый пользователь обладает собственным профилем, в котором владелец может указать краткий перечень информации о себе и список контактов, по которым можно осуществлять с ним взаимодействие за рамками представленного программного продукта.

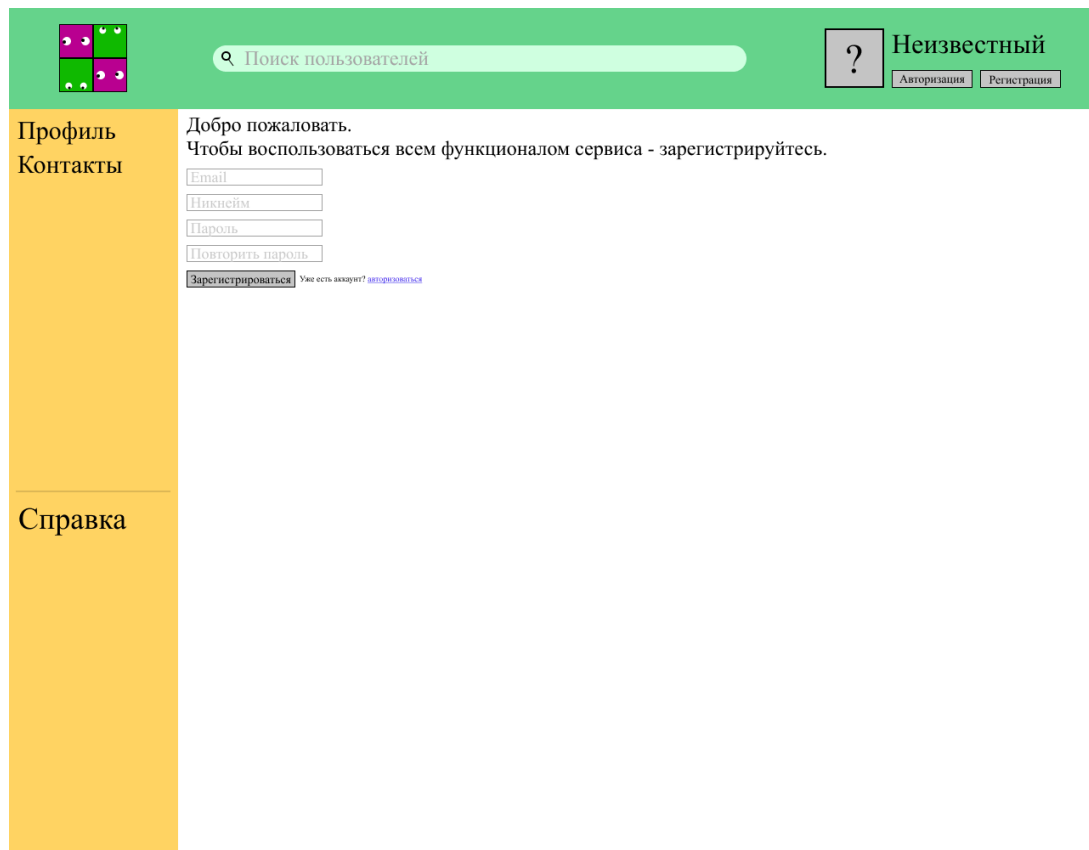
2 Структура и настройка

Программная система для использования не требует предварительной настройки. Для взаимодействия с системой пользователю необходимо иметь на персональном компьютере последнюю версию одного из популярных браузеров (Opera, Firefox, Internet Explorer, Google Chrome).

С точки зрения пользователя программная система является клиент-серверной. Большая часть бизнес-логики по обработке и хранению информации располагается на удаленном сервере, а пользователь располагает лишь частью приложения, отвечающей за графическое отображение.

3 Вход в систему и проверка работоспособности

Для проверки работоспособности системы необходимо зайти в браузер по ссылке, предоставленной разработчиком, системным администратором или любым другим третьим лицом и зарегистрироваться или авторизоваться в системе.

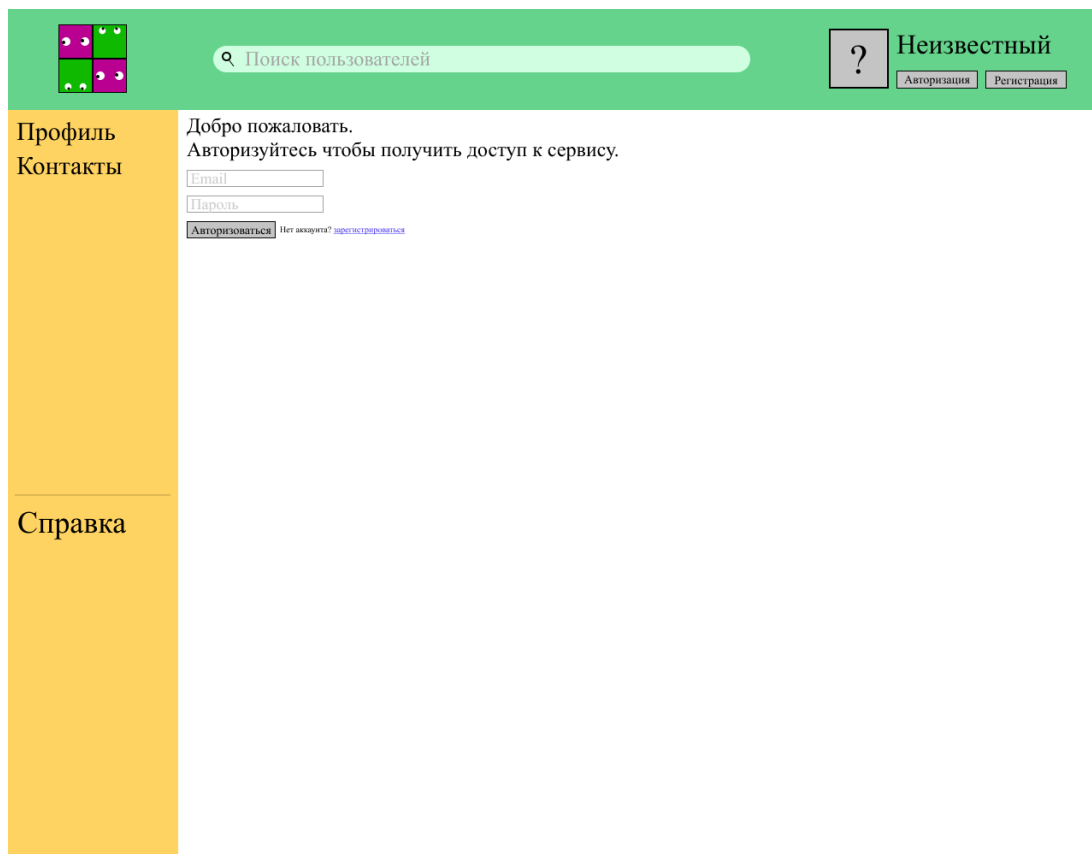


The registration screen features a green header bar. On the left is a logo with four green squares containing white eyes. In the center is a search bar with the text 'Поиск пользователей'. On the right is a user status area with a question mark icon, the text 'Неизвестный', and two buttons: 'Авторизация' and 'Регистрация'.

Below the header is a yellow sidebar on the left with two links: 'Профиль' and 'Контакты'. The main content area has a welcome message: 'Добро пожаловать. Чтобы воспользоваться всем функционалом сервиса - зарегистрируйтесь.' Below this are four input fields: 'Email', 'Никнейм', 'Пароль', and 'Повторить пароль'. At the bottom of the form is a 'Зарегистрироваться' button and a link: 'Уже есть аккаунт? [авторизоваться](#)'.

Below the main content area is another yellow sidebar with a link: 'Справка'.

Рисунок 1 – Экран регистрации



The authorization screen has the same green header bar as the registration screen. The user status area on the right now shows a question mark icon, the text 'Неизвестный', and two buttons: 'Авторизация' and 'Регистрация'.

The main content area has a welcome message: 'Добро пожаловать. Авторизуйтесь чтобы получить доступ к сервису.' Below this are two input fields: 'Email' and 'Пароль'. At the bottom of the form is an 'Авторизоваться' button and a link: 'Нет аккаунта? [зарегистрироваться](#)'.

The yellow sidebar on the left remains the same with links: 'Профиль', 'Контакты', and 'Справка'.

Рисунок 2 – Экран авторизации

В случае успешной регистрации или авторизации и успешного перенаправления на страницу профиля (рис. 3) можно утверждать, что система в данный момент исправна и готова к работе с пользователем.

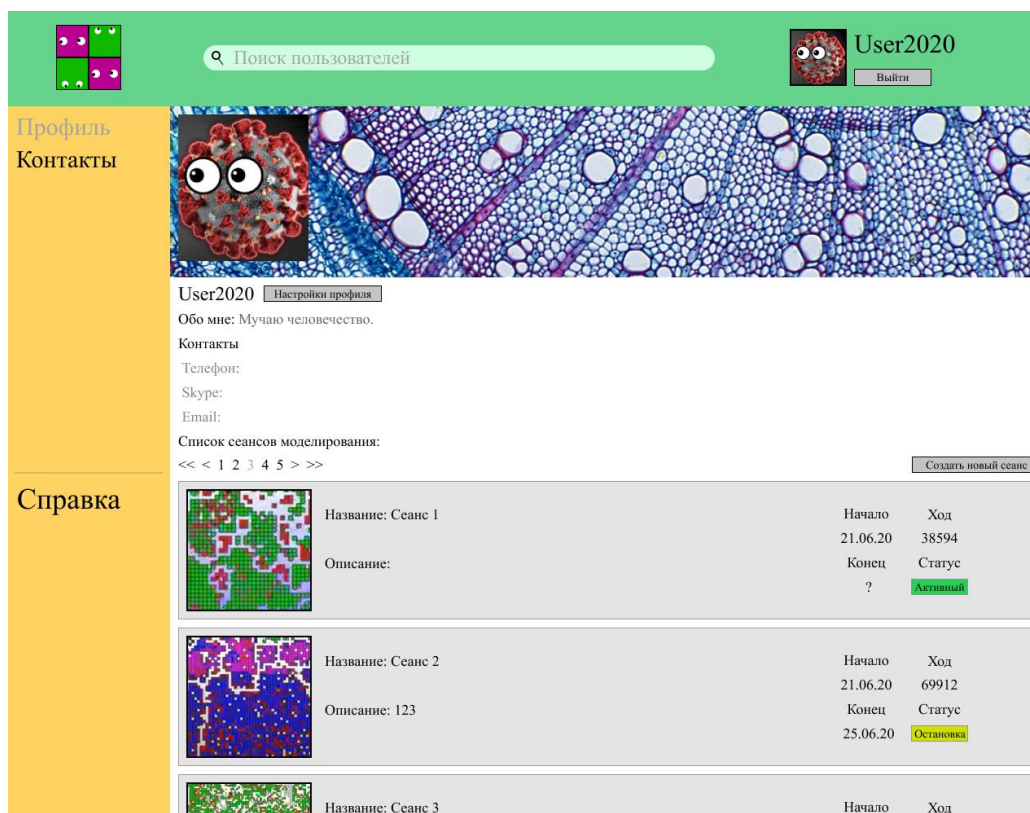


Рисунок 3 – Профиль пользователя

4 Создание и просмотр своих сеансов моделирования

Для просмотра своего сеанса моделирования необходимо перейти в профиль (рис. 3). Это можно сделать с помощью выбора соответствующего пункта из навигационного меню слева, либо посредством нажатия на своё имя или аватар в шапке.

Для перехода к просмотру какого-либо сеанса нужно выбрать соответствующий сеанс из списка сеансов в профиле пользователя.

Для создания нового сеанса моделирования необходимо в профиле нажать кнопку «Создать новый сеанс», после чего произойдет переход на экран создания нового сеанса моделирования (рис. 4).

Рисунок 4 – Экран создания сеанса моделирования

На экране создания сеанса моделирования можно настроить параметры моделирования, в том числе и форму пространства. Некоторые параметры недоступны и имеют серый оттенок ввиду отключения связанных с ними основных параметров.

После настройки и нажатия кнопки «Создать» происходит создание и запуск сеанса моделирования и автоматический переход к его просмотру.

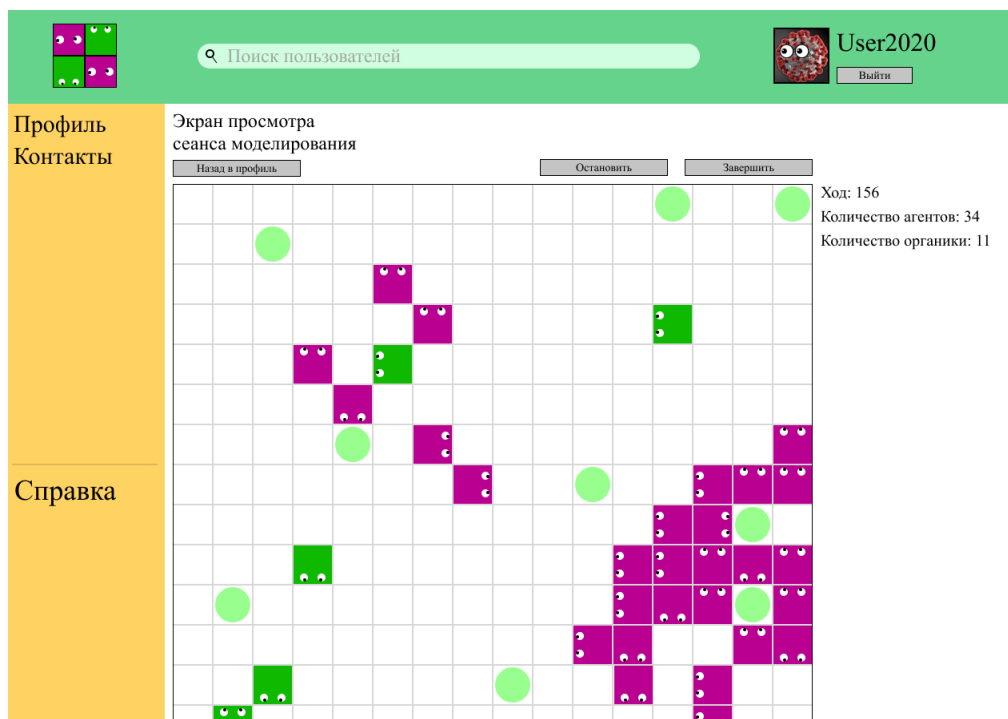


Рисунок 5 - Экран просмотра сеанса моделирования

5 Взаимодействие с другими пользователями и просмотр чужих сеансов моделирования

Для взаимодействия с другими пользователями их нужно для начала найти в системе. Поиск других пользователей может быть осуществлен поисковой строкой в шапке приложения. После ввода никнейма искомого пользователя выдаются подходящие результаты (рис. 6).

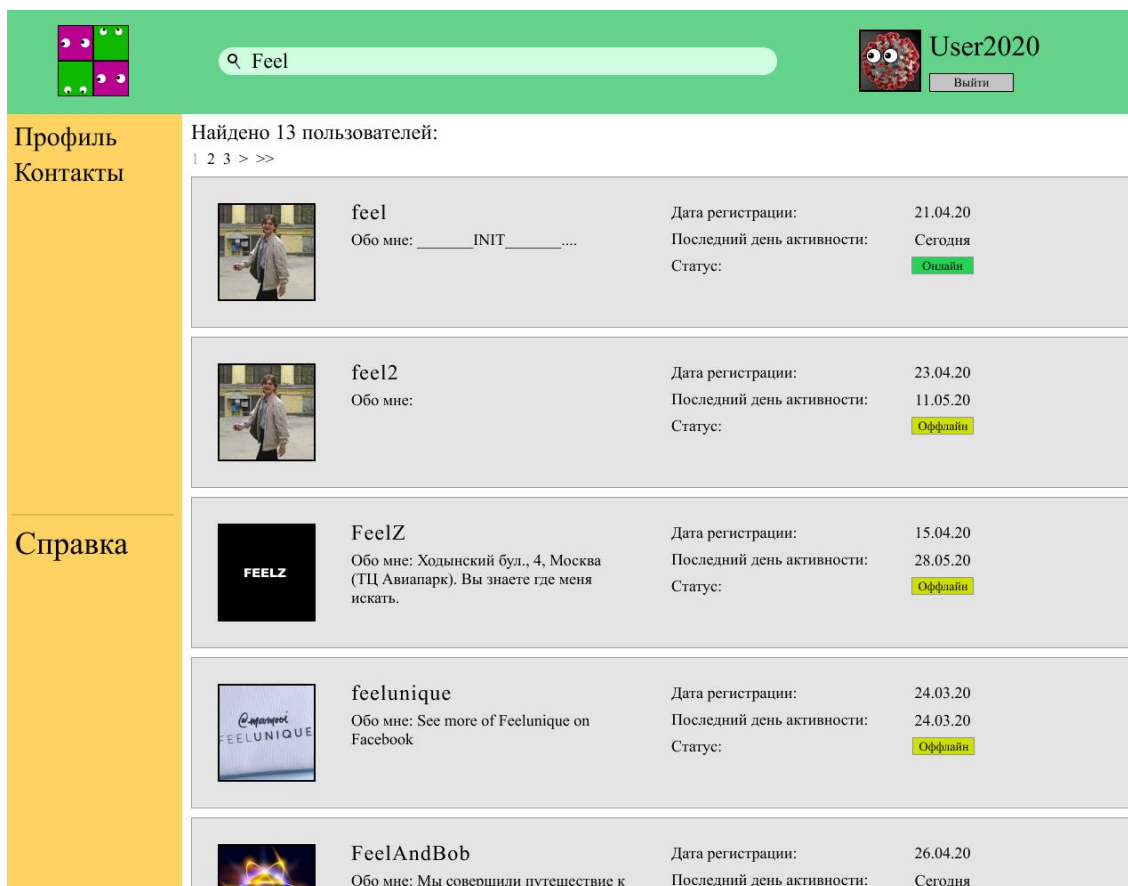


Рисунок 6 – Результат поиска пользователя

После нажатия на аватар или никнейм нужного пользователя происходит переход в его профиль, где пользователь может быть удален или добавлен в список контактов (рис. 7).

Также в профиле пользователя видны его сеансы моделирования, которые доступны для просмотра.

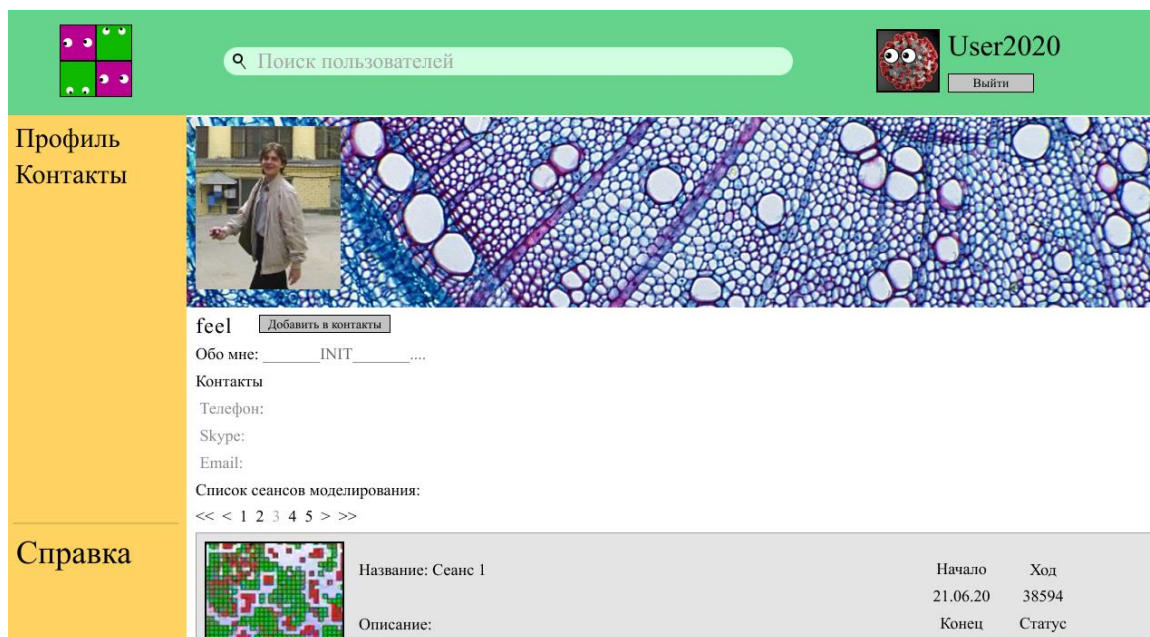


Рисунок 7 – Профиль другого пользователя

Для просмотра списка контактов необходимо нажать на соответствующий пункт из навигационного меню слева.

Для открытия диалога с интересующим пользователем необходимо щелкнуть на его аватар. Для отправки сообщения необходимо набрать текст в поле сообщения и нажать кнопку «Отправить».

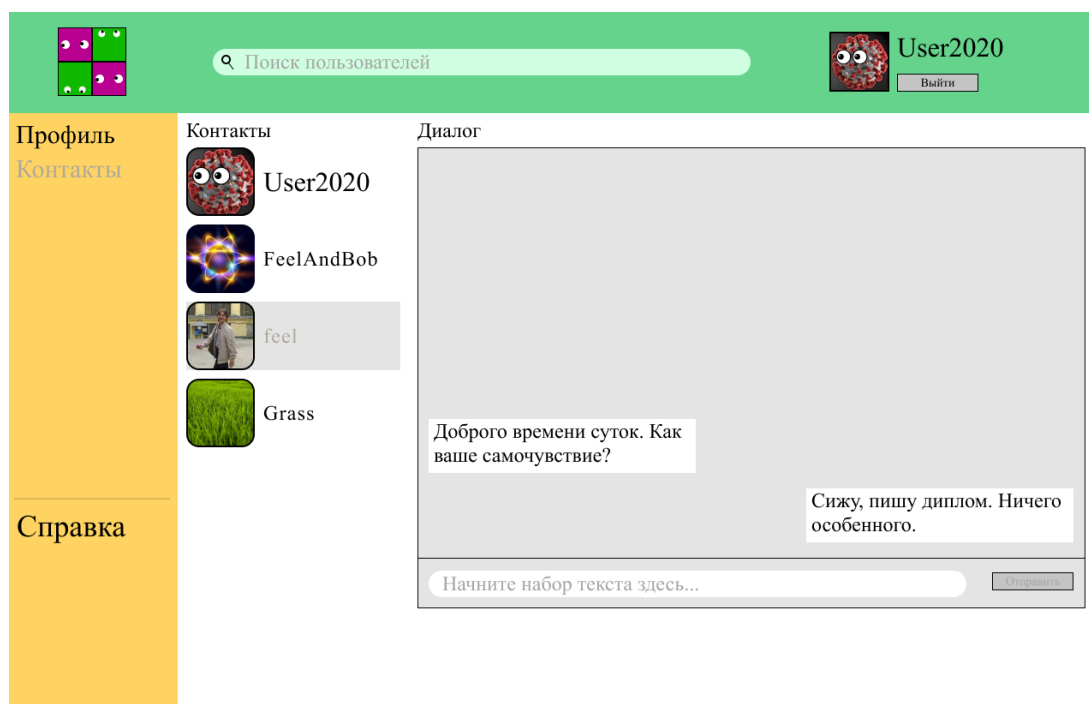


Рисунок 8 – Экран контактов и диалога

ПРИЛОЖЕНИЕ В

(рекомендуемое)

Фрагменты исходного кода системы

Листов 6

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего
образования
«Московский государственный технический университет имени Н.Э. Баумана»
(национальный исследовательский университет)
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»
Кафедра «Компьютерные системы и сети»

Программная система моделирования искусственной жизни с использованием цифровых автоматов

Фрагменты исходного кода программы

Листов 7

Руководитель _____ О.Ю. Ерёмин

Студент ИУ6-85Б _____  В.Д. Шульман

2020 г.

Листинг 1 – Корневой код клиентского приложения

```
import './index.css';
import * as serviceWorker from './serviceWorker';
import store from './redux/reduxStore';
import ReactDOM from "react-dom";
import {BrowserRouter} from "react-router-dom";
import App from "./App";
import React from "react";
import {Provider} from "react-redux";
ReactDOM.render(
  <BrowserRouter>
    <Provider store={store}>
      <App/>
    </Provider>
  </BrowserRouter>, document.getElementById('root'));
serviceWorker.unregister();
////////////////////////////////////
import React from 'react';
import './App.css';
import Navbar from './components/Navbar/Navbar.jsx';
import Settings from './components/Settings/Settings';
import SimsList from './components/SimsList/SimsList';
import Help from './components/Help/Help';
import {Route} from "react-router-dom";
import DialogsContainer from './components/Dialogs/DialogsContainer';
import UsersListContainer from
"./components/UsersList/UsersListContainer";
import ProfileContainer from './components/Profile/ProfileContainer';
import HeaderContainer from './components/Header/HeaderContainer';
import LoginContainer from './components/Login/LoginContainers';
import Reg from './components/Reg/Reg';

const App = (props) => {
  return (
    <div className='app-wrapper'>
      <HeaderContainer store={props.store}/>
      <Navbar/>
      <div className='app-wrapper-content'>
        <Route path='/bib' render={ () => <Reg/>} />
        <Route path="/login" render={ () => <LoginContainer/>} />
        <Route path="/profile/:id?" render={ () =>
<ProfileContainer/>} />
        <Route path="/settings" component={Settings} />
        <Route path="/sims" component={SimsList} />
        <Route path='/users' component={ () => <UsersListContainer/>}
/>
        <Route path="/dialogs" render={ () => <DialogsContainer/>} />
        <Route path="/help" component={Help} />
      </div>
    </div>
  );
};

export default App;
```


Листинг 2 – Корневой код сервера Golang

```
package main

import (
    "github.com/ValeryBMSTU/Diplom/Prog/Back/api"
    "github.com/ValeryBMSTU/Diplom/Prog/Back/bll"
    "github.com/ValeryBMSTU/Diplom/Prog/Back/consts"
    "github.com/ValeryBMSTU/Diplom/Prog/Back/db"
    "github.com/ValeryBMSTU/Diplom/Prog/Back/mv"
    "github.com/labstack/echo"
    "log"
)

func main() {
    e := echo.New()
    dataBase := db.NewDataBase()
    useCase := bll.NewUseCase(dataBase)
    handler := api.NewHandler(useCase)

    e.Use(mv.CORS)

    if err := api.SetApi(e, handler); err != nil {
        log.Fatal(err)
    }
    if err := e.Start(consts.Address); err != nil {
        log.Fatal(err)
    }

    e.Logger.Warnf("shutdown " + consts.Address)
}
```

Листинг 3 – Код api для пользовательских запросов сервера Golang

```
package api

import "github.com/labstack/echo"
import "github.com/ValeryBMSTU/Diplom/Prog/Back/bll"

type IHandler interface {
    /* status */
    Status(ctx echo.Context) error // Проверка статуса сервера

    /* auth */
    Login(ctx echo.Context) error // Авторизация
    Reg(ctx echo.Context) error   // Регистрация

    /* user */
    GetUsers(ctx echo.Context) error // Получение списка
пользователей
    GetUser(ctx echo.Context) error // Получение пользователя
    GetProfile(ctx echo.Context) error // Получение профиля
пользователя
    GetSettings(ctx echo.Context) error // Получение настроек профиля
текущего пользователя
}
```

```

        SetSettings(ctx echo.Context) error // Изменение настроек текущего
пользователя
        GetSims(ctx echo.Context) error    // Получение списка симуляций
текущего пользователя
        Subscribe(ctx echo.Context) error  // Подписка на пользователя

/* sim */
Start(ctx echo.Context) error // Запуск моделирования
Config(ctx echo.Context) error // Настройка моделирования
Stop(ctx echo.Context) error  // Остановка моделирования
End(ctx echo.Context) error   // Завершение моделирования

/* chat */
Send(ctx echo.Context) error // Отправка сообщения
List(ctx echo.Context) error // Получение списка сообщений

/* admin */
Block(ctx echo.Context) error // Блокировка пользователя
Unblock(ctx echo.Context) error // Разблокировка пользователя
Delete(ctx echo.Context) error // Удаление пользователя
Restore(ctx echo.Context) error // Восстановление пользователя
}

type Handler struct {
    bll bll.Bll
}

func NewHandler(useCase bll.Bll) IHandler {
    return &Handler{
        bll: useCase,
    }
}

func SetApi(e *echo.Echo, h IHandler) error {
    e.GET("/status", h.Status)

    e.POST("/users/login", h.Login)
    e.POST("/users/reg", h.Reg)

    e.GET("/users", h.GetUsers)
    e.GET("/users/:id", h.GetUser)
    e.GET("/users/:id/profile", h.GetProfile)
    e.GET("/users/:id/settings", h.GetSettings)
    e.PUT("/users/:id/settings", h.SetSettings)
    e.GET("/users/:id/sims", h.GetSims)
    e.POST("/users/:id/subscribe", h.Subscribe)

    e.POST("/sims/start", h.Start)
    e.PUT("/sims/config/:id", h.Config)
    e.POST("/sims/stop/:id", h.Stop)
    e.POST("/sims/end/:id", h.End)

    e.POST("/chat/send/:id", h.Send)
    e.GET("/chat/list", h.List)

    e.POST("/admin/block/:id", h.Block)

```

```

        e.POST("/admin/unblock/:id", h.Unblock)
        e.POST("/admin/delete/:id", h.Delete)
        e.POST("/admin/restore/:id", h.Restore)

    return nil
}

```

Листинг 4 – Код создания таблиц базы данных PostgreSQL

```

CREATE TABLE IF NOT EXISTS public.user (
    id SERIAL PRIMARY KEY,
    nickname VARCHAR UNIQUE NOT NULL,
    email VARCHAR UNIQUE NOT NULL,
    password VARCHAR NOT NULL,
    about TEXT,
    ava_path VARCHAR,
    role VARCHAR NOT NULL DEFAULT 'user',
    is_blocked BOOL NOT NULL DEFAULT false,
    is_deleted BOOL NOT NULL DEFAULT false
);

CREATE TABLE IF NOT EXISTS public.session (
    id SERIAL PRIMARY KEY,
    key VARCHAR NOT NULL,
    exp TIMESTAMP NOT NULL,
    user_id SERIAL NOT NULL
);

CREATE TABLE IF NOT EXISTS public.chat (
    id SERIAL PRIMARY KEY,
    message VARCHAR NOT NULL,
    time TIMESTAMP NOT NULL,
    sender_id SERIAL NOT NULL,
    receiver_id SERIAL NOT NULL
);

CREATE TABLE IF NOT EXISTS public.users_service (
    id SERIAL PRIMARY KEY,
    users INTEGER NOT NULL,
    blocked_users INTEGER NOT NULL,
    deleted_users INTEGER NOT NULL,
    active_users_per_day INTEGER NOT NULL,
    new_users_per_day INTEGER NOT NULL,
    date TIMESTAMP NOT NULL
);

CREATE TABLE IF NOT EXISTS public.modeling (
    id SERIAL PRIMARY KEY,
    title VARCHAR NOT NULL,
    start_time TIMESTAMP NOT NULL,
    end_time TIMESTAMP,
    user_id SERIAL NOT NULL
);

CREATE TABLE IF NOT EXISTS public.template (
    id SERIAL PRIMARY KEY,
    title VARCHAR NOT NULL,
    params TEXT NOT NULL,
    modeling_id SERIAL NOT NULL
);

CREATE TABLE IF NOT EXISTS public.snapshot (
    id SERIAL PRIMARY KEY,

```

```

        step INTEGER,
        time TIME,
        env_data TEXT NOT NULL,
        agents_data TEXT NOT NULL,
        modeling_id SERIAL NOT NULL
    );
CREATE TABLE IF NOT EXISTS public.agent (
    id SERIAL PRIMARY KEY,
    gen_code TEXT NOT NULL,
    modeling_id SERIAL NOT NULL
);
CREATE TABLE IF NOT EXISTS public.relation (
    id SERIAL PRIMARY KEY,
    parent_id SERIAL NOT NULL,
    child_id SERIAL NOT NULL
);
CREATE TABLE IF NOT EXISTS public.modelings_service (
    id SERIAL PRIMARY KEY,
    modelings INTEGER NOT NULL,
    active_modelings INTEGER NOT NULL,
    stopped_modelings INTEGER NOT NULL,
    ended_modelings INTEGER NOT NULL,
    date TIMESTAMP NOT NULL
);

ALTER TABLE public.session
ADD FOREIGN KEY (user_id)
REFERENCES public.user(id);
ALTER TABLE public.chat
ADD FOREIGN KEY (sender_id)
REFERENCES public.user(id);
ALTER TABLE public.chat
ADD FOREIGN KEY (receiver_id)
REFERENCES public.user(id);
ALTER TABLE public.modeling
ADD FOREIGN KEY (user_id)
REFERENCES public.user(id);
ALTER TABLE public.template
ADD FOREIGN KEY (modeling_id)
REFERENCES public.modeling(id);
ALTER TABLE public.snapshot
ADD FOREIGN KEY (modeling_id)
REFERENCES public.modeling(id);
ALTER TABLE public.agent
ADD FOREIGN KEY (modeling_id)
REFERENCES public.modeling(id);
ALTER TABLE public.relation
ADD FOREIGN KEY (parent_id)
REFERENCES public.agent(id);
ALTER TABLE public.relation
ADD FOREIGN KEY (child_id)
REFERENCES public.agent(id);

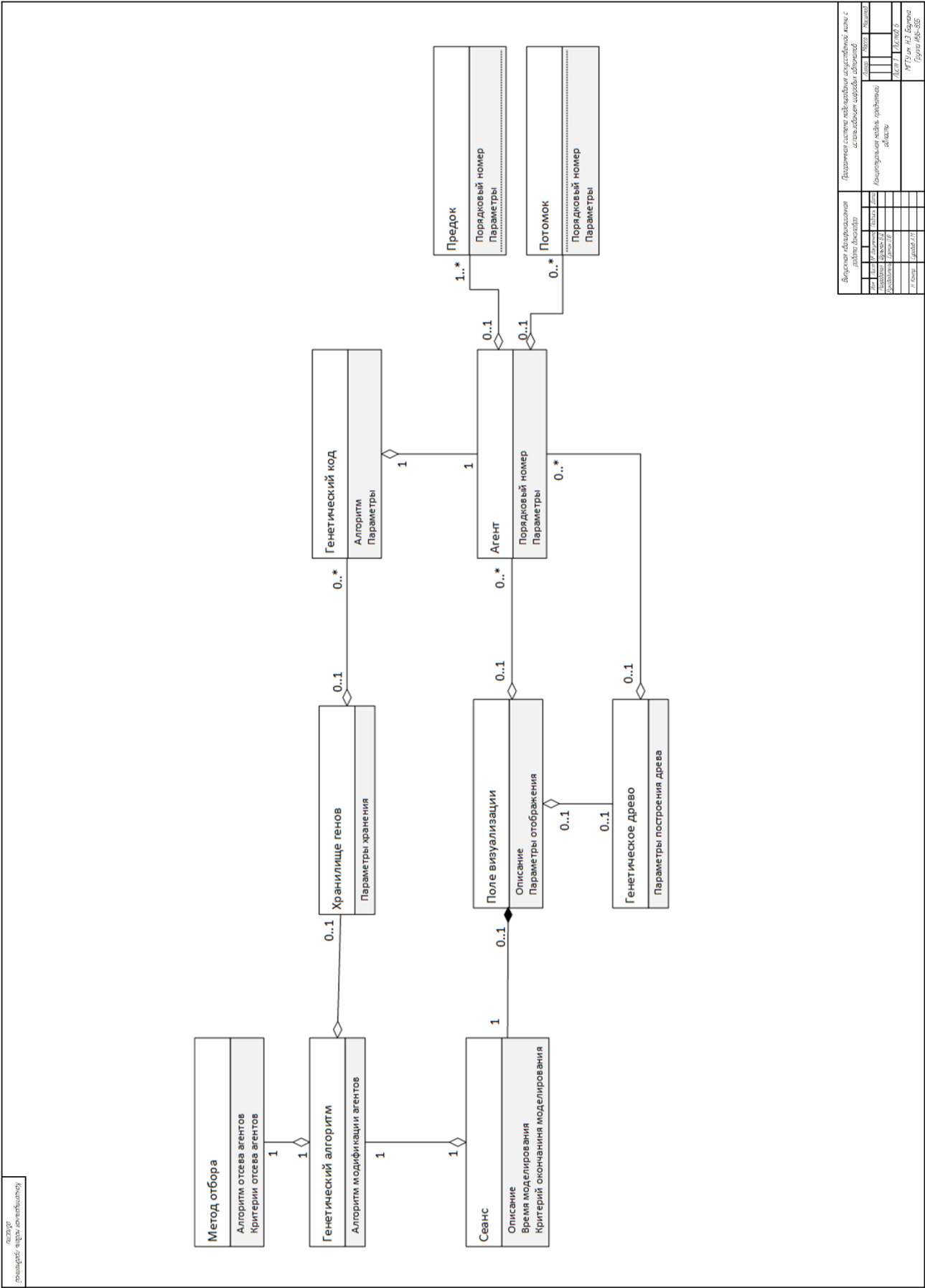
```

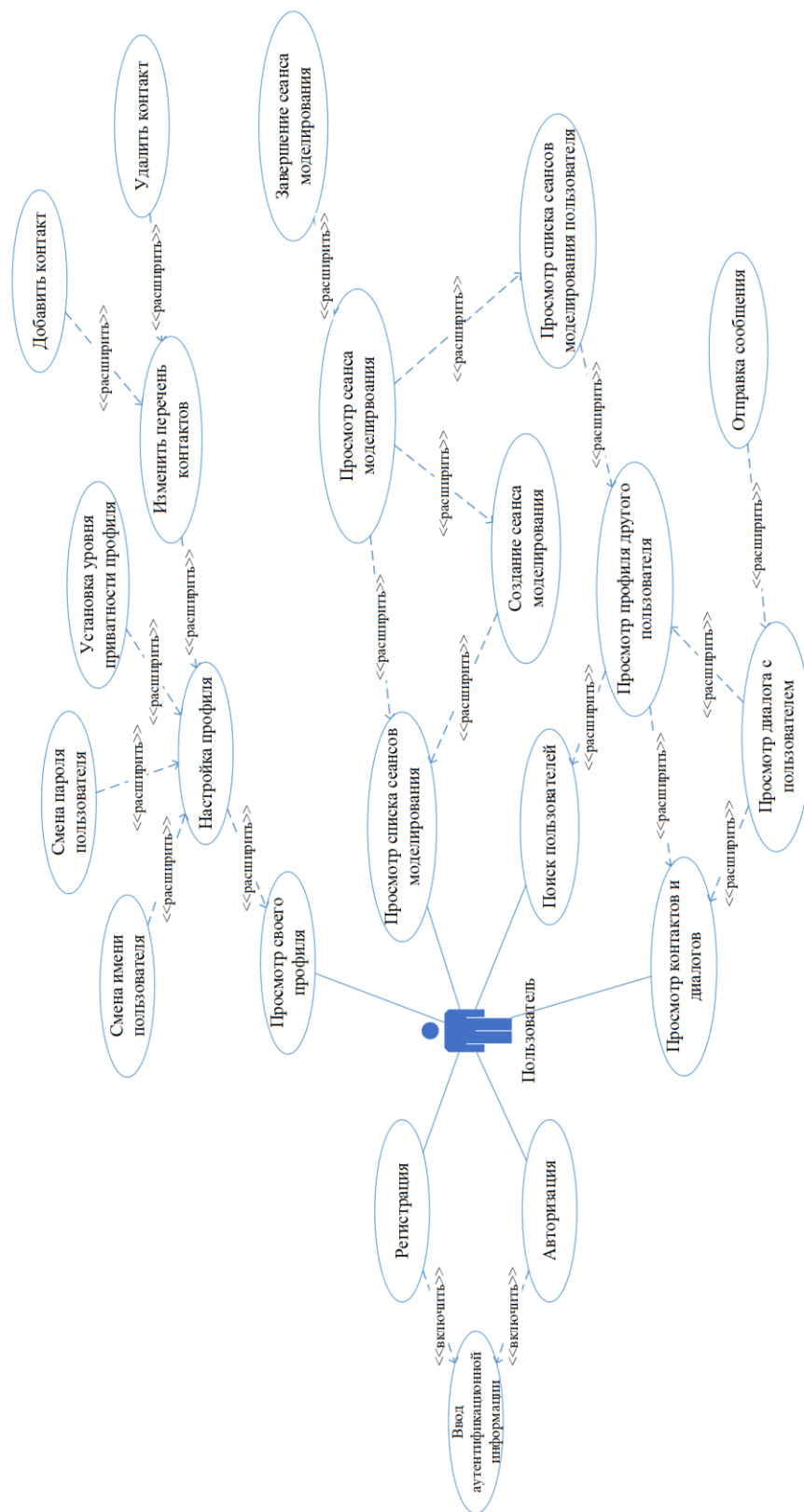
ПРИЛОЖЕНИЕ Г

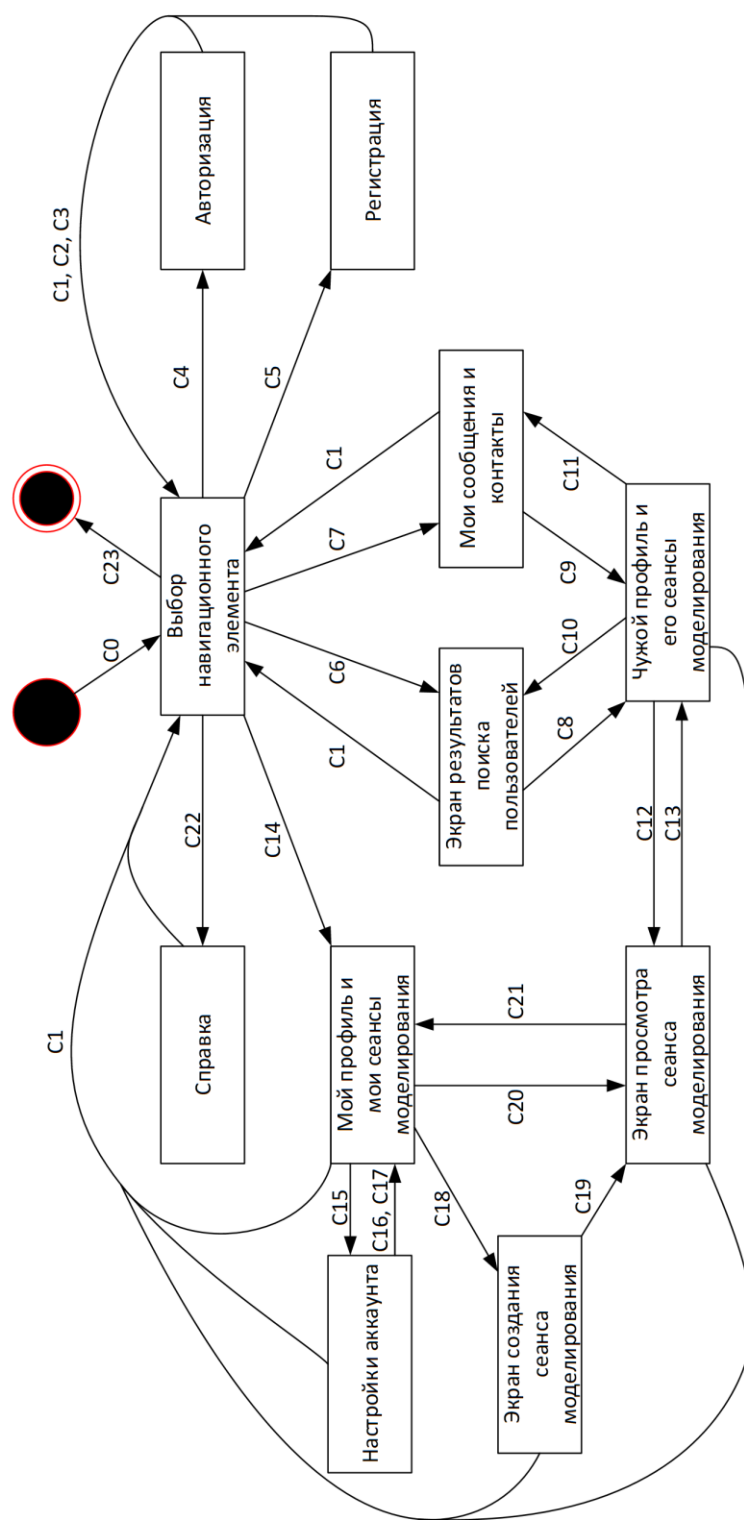
(обязательное)

Копии листов графической части

Листов 6



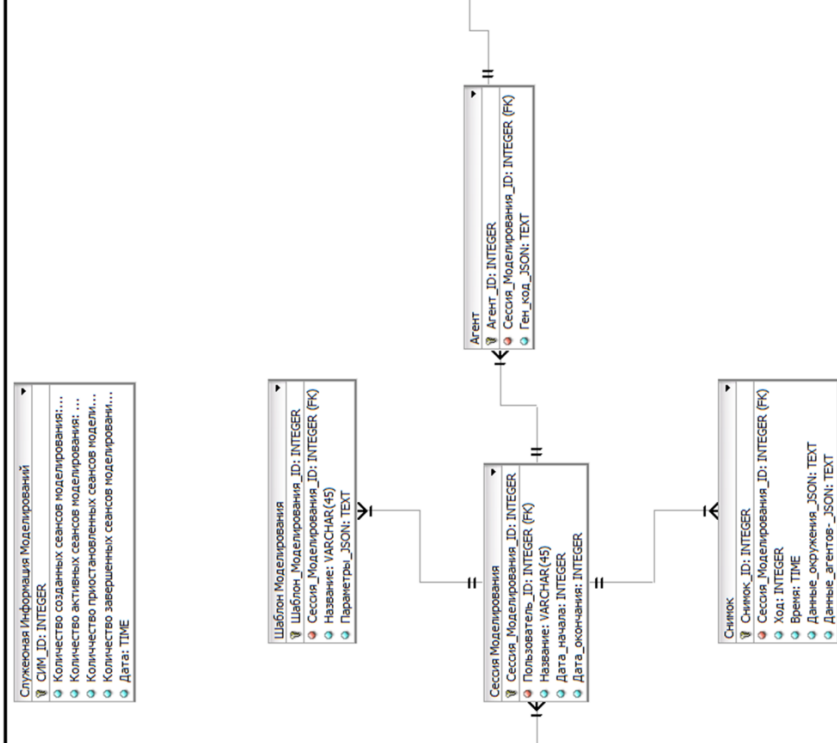
[illegible]



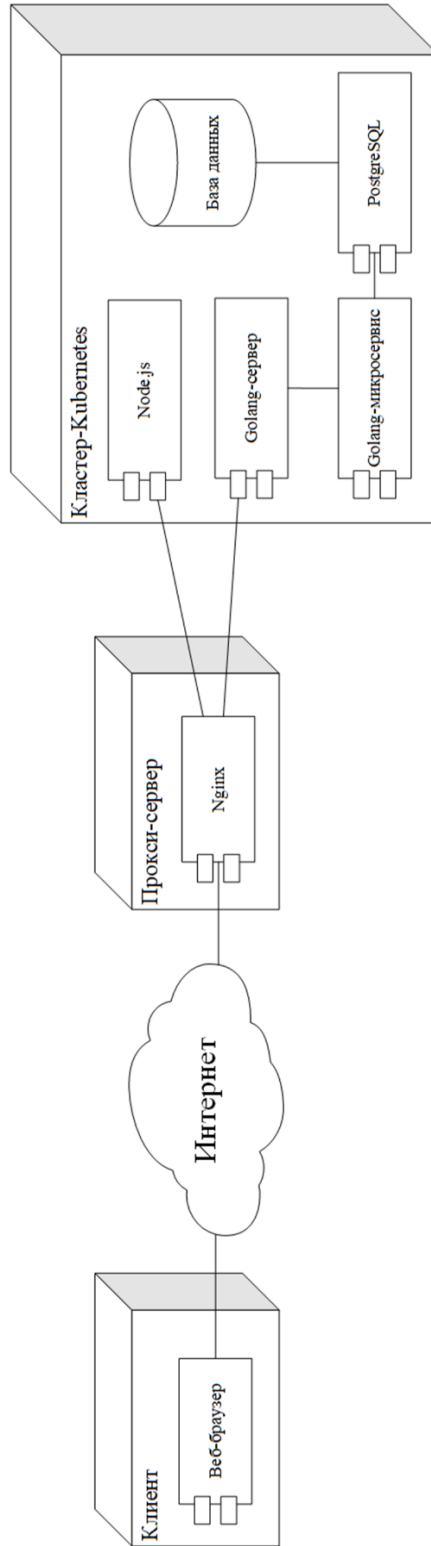
**Информация
пользователей**



Информация моделирования



Виды работ, выполняемых в соответствии с проектом	Информация об объекте		Информация о проекте		Информация о заказчике	
	№ п/п	Наименование работ	№ п/п	Наименование работ	№ п/п	Наименование работ
1	2	3	4	5	6	7
<p>Проектирование системы автоматического контроля за состоянием окружающей среды</p> <p>Дополнительные сведения: Водные ресурсы, природоохранные территории</p> <p>Итого: 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100</p>						



Выполнение задания		Проверка выполнения задания	
Этап	Оценки	Этап	Оценки
1. Анализ задачи	100%	2. Разработка архитектуры	100%
3. Разработка кода	100%	4. Тестирование	100%
5. Внедрение	100%	6. Поддержка	100%
Итого	500%	Итого	500%