Федеральное государственное автономное
образовательное учреждение
высшего профессионального образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.И. Легалов
подпись инициалы, фамилия
« _____» _____ 20 ___ г.

БАКАЛАВРСКАЯ РАБОТА

Редактор химических формул для поддержки процесса обучения
наименование темы,

09.03.01.09 – «Технологии разработки программного обеспечения»
код и наименование специальности (специализации), направления

Пояснительная записка

Научный руководитель/
руководитель _____ Доцент, к.т.н. А.В. Редькина
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ А.А. Винидиктов
подпись, дата инициалы, фамилия

Нормоконтролер _____ В.И. Иванов
подпись, дата инициалы, фамилия

Красноярск 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день интернет-технологии развиваются с невероятной скоростью. Интернет стал неотъемлемой частью нашей повседневной жизни, стал доступен практически каждому. Разного рода программное обеспечение, будь то игровой движок или текстовый редактор, стали переходить на веб-интерфейсы, при помощи которых пользователь взаимодействует с сайтом или любым другим приложением через браузер. Веб-интерфейсы получили широкое распространение в связи с ростом сети интернет, соответственно – повсеместного распространения браузеров. Согласно статистике крупнейшего веб-сервиса для хостинга IT-проектов GitHub.com, языки программирования, связанные с разработкой приложений для интернета, в настоящее время наиболее популярны [1].

Сегодня, когда веб-интерфейс стал универсальным, инструменты для редактирования непосредственно в браузере имеют большую ценность. Теперь, если вам нужно быстро отредактировать какой-либо текстовый документ или обработать фотографию, достаточно перейти по гиперссылке на нужный сайт-редактор, при этом имея те же возможности что и в обычном десктоп-приложении. Онлайн редакторы используются различных областях науки в том числе и химии, появились разнообразные инструменты для моделирования химических соединений и процессов, трехмерные визуализаторы и базы данных химических соединений.

Редактор химических формул – это специальная программа, позволяющая вводить, редактировать и выводить на экран информацию о структуре и составе молекул вещества. Существует множество молекулярных редакторов: платные и бесплатные, с открытым или закрытым исходным кодом, с различными графическими интерфейсами и с использованием различных технологий [2,3].

Основа веб-программирования это HTML – язык гипертекстовой разметки документов, поэтому появление HTML5 произвело настоящую революцию в

области программирования для интернета, появились новые возможности для отображения на странице браузера графики (двумерной, трехмерной) в том числе и создаваемой динамически. Одно из достоинств HTML5 – это улучшенный уровень поддержки мультимедиа-технологий, так как появились специфические теги для работы с аудио и видео, а также появилась поддержка формата SVG, для работы с векторной графикой. Все это в совокупности с CSS – языком описания внешнего вида HTML документа и языком программирования JavaScript, поддержку которого обеспечивают многие современные браузеры и который интегрирован с HTML и CSS, позволяет создавать интерактивные сайты.

Целью данной ВКР является – создание редактора химических формул для поддержки процесса обучения.

1 Анализ предметной области и постановка задачи

Онлайн редактор – это программа для создания и редактирования данных, таких как: текст, изображения, электронные таблицы, презентации, которая работает в браузере через интернет. Их преимущество в том, что они позволяют пользователю редактировать данные совместно с другими пользователями без необходимости пересылки данных. В настоящее время онлайн редакторы обеспечивают практически те же возможности, что и их аналоги для настольных компьютеров, которые так же стараются перейти на веб-интерфейс, например, MS Office, Photoshop.

1.1 Анализ предметной области

История химических графических инструментариев для интернета началась в 1998 году, кода появился плагин для браузера MDL Chime, который позволял визуализировать структуру молекул [4].

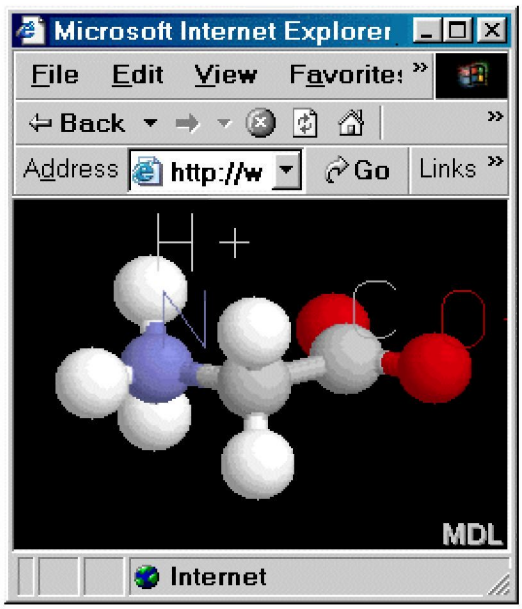

Рисунок 1 – Интерфейс MDL Chime

В настоящее время существует множество программных продуктов для построения, редактирования, моделирования химических соединений, рассмотрим несколько примеров таких программ, использующих различные технологий.

1.1.1 Редактор ChemDoodle Web Components

ChemDoodle Web Components – это один из первых химических инструментариев, построенных на основе стандарта технологий – JavaScript, CSS и HTML5, благодаря этому он работает во многих браузерах, включая Internet Explorer 9, Edge, Safari, Opera, Chrome и Firefox, а также с некоторыми мобильными версиями представленных браузеров. Стоит отметить что ChemDoodle Web Components является проектом с открытым исходным кодом, выпускается под лицензией GNU и сопровождается подробной документацией [5].

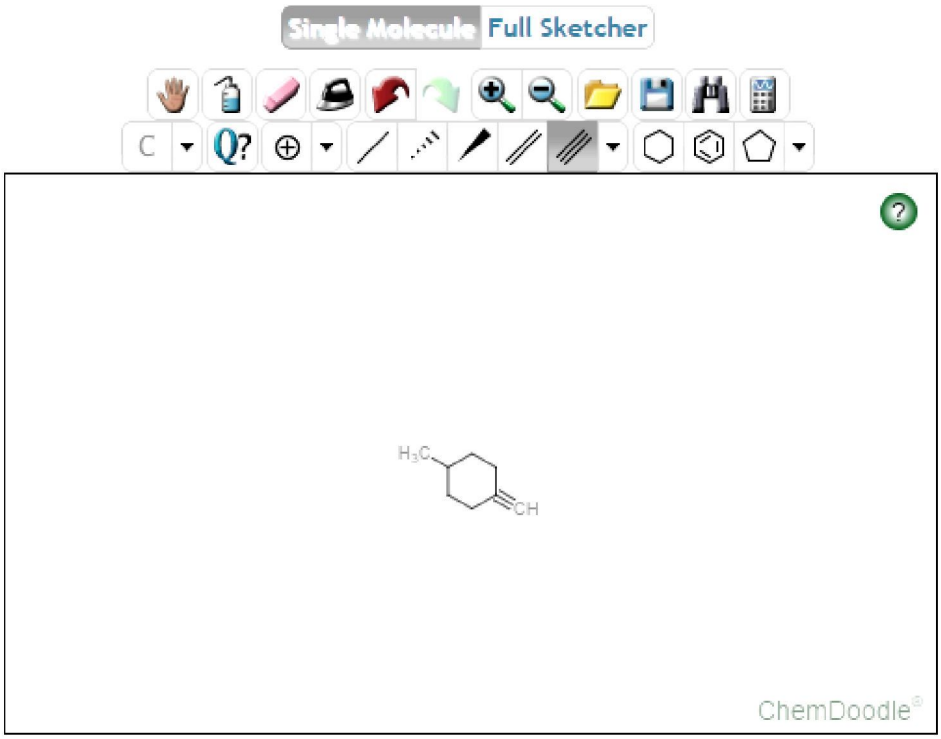

Рисунок 2 - Интерфейс редактора ChemDoodle Web Components

Данный редактор обладает двумя режимами работы: режим редактирования одной молекулы, в котором пользователю не разрешено создавать больше одной молекулы и полный режим, который позволяет создавать различные химические структуры от простых атомов до сложных соединений таких как кофеин($C_8H_{10}N_4O_2$). Взаимодействие с интерфейсом построено на работе с компьютерной мышью. Кнопки в верхней панели обеспечивают различные функциональные возможности: выделение областей, очистка поля, перемещение объектов, отмена последнего действия, масштабирование, элементы периодической таблицы и т.д.

Работа с ChemDoodle Web Components осуществляется в четыре этапа:

1. подключить ссылку на библиотеку в HTML-документе;
2. компоненты инстанцируются на странице в блоке кода JavaScript;
3. настройка внешнего вида компонента (не обязательно);
4. загрузка данных в компонент.

```
<script type="text/javascript">
  var viewer = new ChemDoodle.ViewerCanvas('viewer',100,100);
  viewer.specs.backgroundColor = 'blue';
  var caffeine = ChemDoodle.readMol(caffeine);
  viewer.loadMolecule(caffeine);
</script>
```

Рисунок 3 – Пример кода, реализующего построение молекулы кофеина

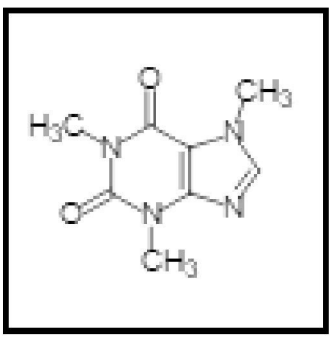

Рисунок 4 – Графическое представление молекулы кофеина

Особенности:

– поддержка всех современных браузеров;

– все компоненты обрабатываются встроенными функциями браузера, без использований сторонних расширений, таких как Java-апплеты и Flash плагины;

– поддержка мобильных устройств.

1.1.2 Редактор MolView

Редактор MolView – это приложение, которое помогает студентам и преподавателям визуализировать молекулярные структуры и просматривать их свойства. Данный редактор начинался как инструмент для преобразования двумерной химической структуры в трехмерную.

Данный программный продукт разработан с использованием JavaScript, HTML5 технологий, для трехмерной визуализации используется библиотека WebGL, которая в совокупности с Canvas обеспечивает API 3D графики без использования плагинов. Для построения пользовательского интерфейса сайта, использовался каркас Bootstrap.

Одно из достоинств рассматриваемого редактора – это возможность загружать химические структуры из больших баз данных, таких как PubChem и RCSB Protein Data Bank.

MolView состоит из двух главных частей, редактор формул и 3D визуализатор. Редактор формул окружен тремя панелями инструментов. После того как пользователь нарисует молекулу, он можете преобразовать молекулу к трехмерному виду [6].

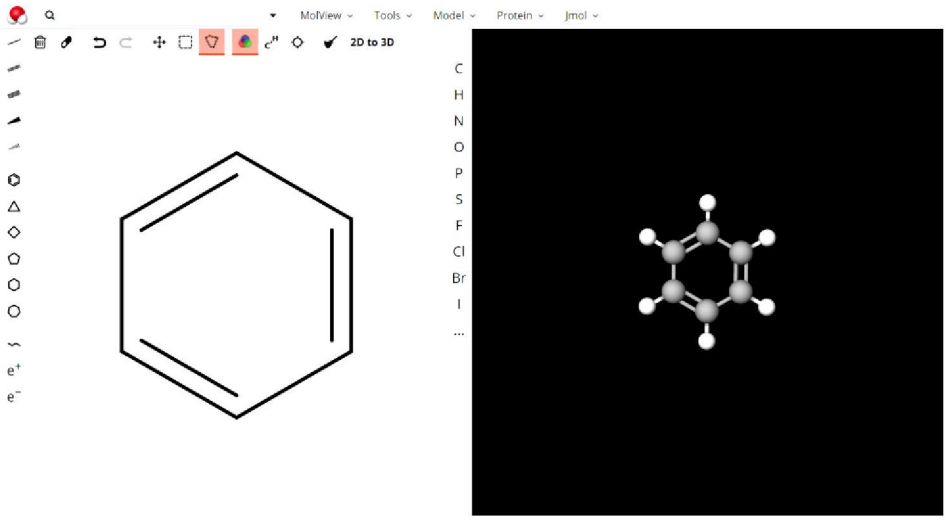

Рисунок 5 – Редактор MolView

Верхняя панель инструментов содержит инструмены для очистки поля, выделения, перемещения элементов, изменения цветовой гаммы, преобразование в 3D. Левая панель инструментов позволяет создавать один из типов связей (одинарную, двойную, тройную) или модифицировать выбранную связь, выбрать один из фрагментов (бензол, циклопропан и т.д.), сознание цепочки атомов углерода, увеличение и уменьшение заряда атомов. В правой панели содержится периодическая система химических элементов.

Рассматриваемый редактор успешно развивается в настоящее время, автор добавляет новый функционал, новые модули, благодаря удачно построенной архитектуре приложения [7].

Особенности:

- доступ к научным базам данных химических элементов;
- большой набор инструментов для построения соединений;
- трехмерная визуализация;
- открытый исходный код [8].

1.1.3 Редактор Avogadro

Avogadro – это многофункциональный молекулярный редактор, разработанный для использования в химии, молекулярного моделирования, биоинформатики, науки материалов и связанных областей [9].

Данный программный продукт написан на языке программирования C++, в качестве системы сборки используется CMake, которая делает процесс сборки относительно простой на всех поддерживаемых платформах.

При написании Avogadro использовалась парадигма модель-вид-контроллер (MVC) [10]. Поддерживается возможность расширяемости с помощью плагинов.

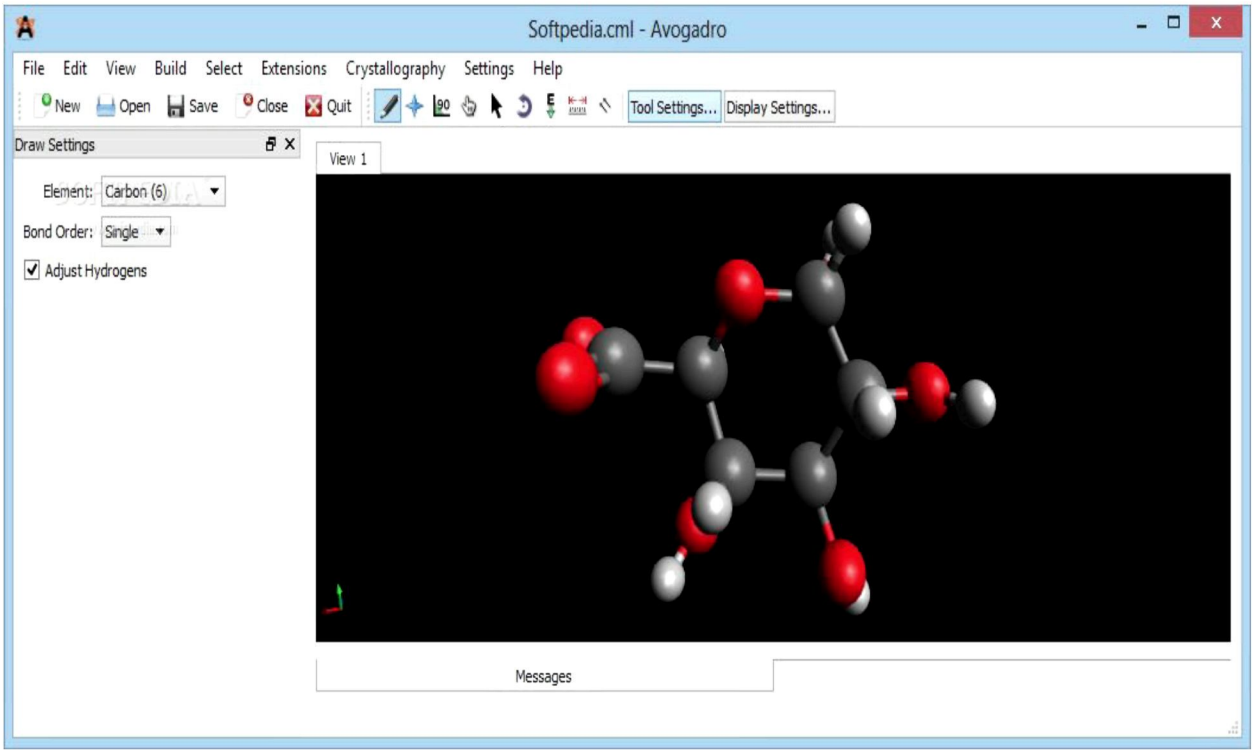

Рисунок 6 – Редактор Avogadro

Avogadro имеет тесные связи с другими свободными, кроссплатформенными проектами с открытым исходным кодом, например Qt, для обеспечения кроссплатформенного графического отображения, OpenBabel для поддержки различных форматов для файлового ввода/вывода, OpenGL для трехмерного отображения в реальном времени.

Общие химические элементы могут быть выбраны из выпадающего списка, отображающего периодическую таблицу. Существуют инструменты для выравнивания позиции атомов и еще множество разнообразных функций, касающихся как отрисовки элементов, различных анимаций, так и вычислений, связанных с этими элементами. Построить молекулу можно по атомам или с использованием методов описания химических формул, например, такого как SMILES.

Особенности:

- кроссплатформенность (Windows, Linux, OS X);
- исходный код распространяется под лицензией GNU;
- большим плюсом для разработчиков является то, что он может быть легко расширен с помощью архитектуры плагинов, для встраивания новых возможностей;
- использование унифицированных форматов файлов;
- некоммерческий программный продукт.

1.2 Постановка задачи

Одним из недостатков представленных выше редакторов, можно назвать отсутствие поддержки процесса обучения, поэтому разрабатываемый редактор должен его обеспечивать.

Согласно заданию, разрабатываемый программный редактор должен соответствовать следующим требованиям:

- иметь пользовательский веб-интерфейс;

– химическое соединение должно отображаться в виде графического рисунка;

– построение химического соединения должно производится в соответствии с законами химии;

– для построения химических соединений должны использоваться химические элементы из таблицы Менделеева;

– обеспечивать поддержку процесса обучения.

2 Выбор средств для реализации редактора

Зачастую, сетевые приложения создаются в различных вариантах архитектуры клиент-сервер. Для разрабатываемого редактора клиентом будет служить браузер, а сервером – веб-сервер.

Логика приложения распределена между сервером и клиентом, хранение, доступ к данным и обработку осуществляет сервер, клиент представляет эти данные в удобном для пользователя виде в окне браузера. Обмен информацией между клиентом и сервером происходит по сети, посредством запросов.

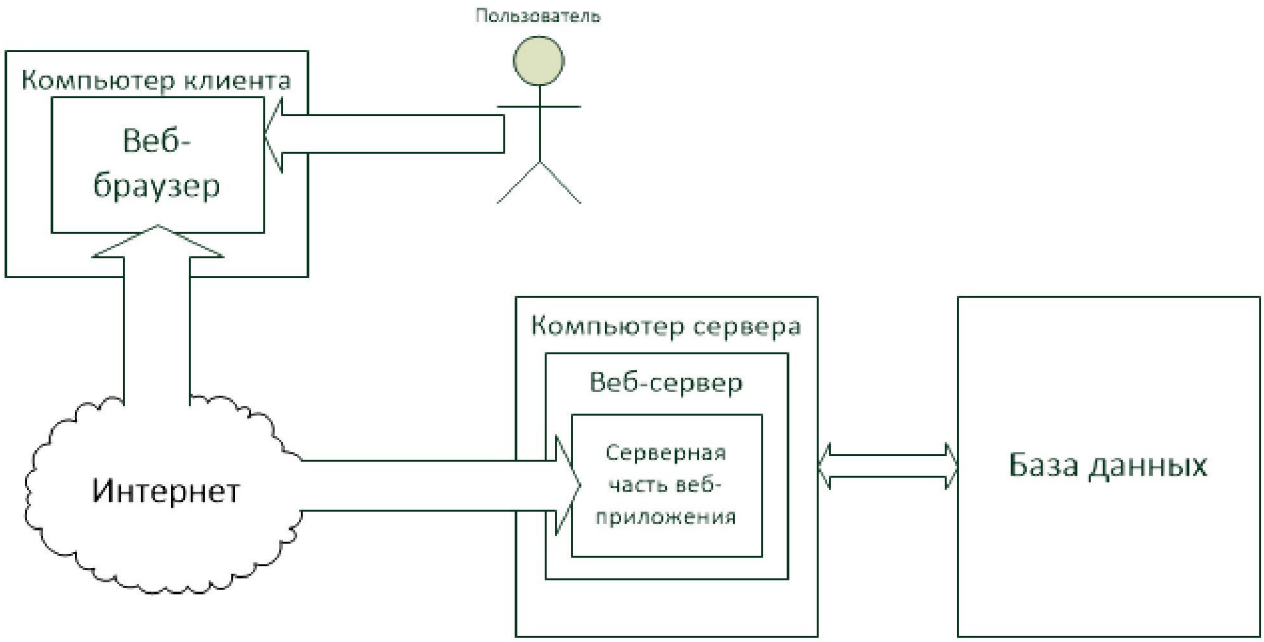

Рисунок 7 – Общая структура системы

Работу системы можно описать так:

1) пользователь запускает редактор в окне браузера;

2) браузер посылает http-запрос к серверу;

3) сервер обрабатывает запрос, если требуется обращается к базе данных;

4) сервер возвращает веб-страницу, которую отображает браузер.

Серверная часть приложения – это программа на сервере, которая обрабатывает запросы, присылаемые клиентом и возвращает на них веб-страницу, описанную в формате HTML, которая потом будет отображена в браузере.

Система управления базами данных – это программное обеспечение на сервере, которое обеспечивает хранение и выдачу данных. Клиент получает данные из базы данных через серверную часть приложения.

Клиентская часть приложения – это интерфейс, отображаемый в окне браузера.

Одним из преимуществ построения веб-приложения, с поддержкой стандартных функций браузера, заключается в том, что работа программы обеспечивается независимо от операционной системы данного клиента. То есть разработчику не нужно писать отдельные версии приложения для Microsoft Windows, Mac OS X, GNU/Linux и других операционных систем, приложение создается один раз для произвольно выбранной платформы и на ней разворачивается. Стоит учитывать, что существуют различные реализации HTML, CSS и других спецификаций в браузерах, которые могут вызвать проблемы при разработке приложения и его последующей поддержке.

2.1 Технологии для клиентской части

Согласно требования к разрабатываемому программному продукту, интерфейс пользователя должен включать в себя:

- меню для навигации по сайту;
- поле для отображения химических связей и элементов;
- инструменты редактирования поля (удаление, добавление элементов или связей);
- область с выводом задания на построение химических соединений;
- периодическую таблицу химических элементов.

Язык HTLM5 используется для разметки веб-станиц, язык CSS3 используется как средство описания, оформления вешнего вида станицы. Согласно стандарту организации World Wide Web Consortium, которая разрабатывает и внедряет технологические стандарты для Всемирной паутины, HTML и CSS в настоящее время являются одними из основных технологий для создания веб-страниц [12].

HTML описывает структуру веб-документа средствами дескрипторов (тегов) – именованных меток (обычно парных) таких как <head>, <title>, <body>, <article>, <section>, <p>, , <img>, <picture>.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
    </head>
    <body>
        <p>Содержимое</p>
    </body>
</html>
```

Рисунок 8 – Пример HTML разметки страницы

Дескрипторы имеют свойства, называемыми атрибутами, дающие дополнительные возможности, например атрибуты width и height тега <canvas> отвечают за его размеры, например: <canvas width="200" height="200"></canvas>, установят у поля canvas, высоту и ширину равную 200 пикселям.

Каскадные таблицы стилей CSS, используются для визуального представления внешнего вида HTML страницы (цвет, шрифты, расположение блоков, отступы). CSS применяется к HTML документу через внешний файл или через имеющийся у дескрипторов атрибут style.

JavaScript – это динамически типизированный язык сценариев, поддерживающий прототипное создание объектов. Базовый синтаксис

намеренно похож на Java и C++, чтобы уменьшить число новых концепций, необходимых для изучения языка. Такие языковые конструкции, как if, for, while, switch, try и catch похожи на конструкции этих языков. JavaScript наиболее широко применяется как язык сценариев веб-страниц. Стандартом языка JavaScript является ECMAScript. Все современные браузеры полностью поддерживают ECMAScript 5.1. Старые версии браузеров поддерживают по крайней мере - ECMAScript 3 [13]. Для подключения JavaScript к странице, используются тег <script> или ссылка на внешний файл. Чтобы передавать данные со стороны клиента на сервер без перезагрузки страницы, используется интерфейс программирования приложений (API) Fetch.

```
var showForm = document.getElementById('button');
showForm.addEventListener('click', function(event) {
    alert("Hello");
});
```

Рисунок 9 – Пример кода JavaScript

Технологии Adobe Flash, Silverlight и Java-апплеты тоже могут применяться при построении веб-приложения, в частности, для реализации пользовательского интерфейса, но для корректной работы такого приложения придется устанавливать специальное программное обеспечение – плагины. Их использование может в какой-то мере минимизировать проблемы с различными конфигурациями браузеров. В настоящее время производители популярных браузеров Google Chrome, Safari и некоторых других отказываются от поддержки Adobe Flash, компания Oracle отказывается от поддержки технологии java-апплетов, а Microsoft прекращает развитие Silverlight [11].

Веб-страница – это та часть приложения, с которой пользователь напрямую взаимодействует, и от скорости загрузки страницы, внешнего ее вида, удобства взаимодействия с ней, предоставляемых ею функциональных возможностей, будет многое зависеть. В настоящее время стек технологий

JavaScript, HTML и CSS является наиболее распространенным стандартом для разработки веб-приложений, с помощью него можно создавать полноценные приложения, предоставляя пользователю функциональные возможности серверной часть программы.

2.2 Технологии для серверной части

Язык программирования Python широко используется для разработки серверной части приложений. Данный язык имеет богатую стандартную библиотеку, и различные реализации каркасного подхода, в том числе для веб-разработки и для работы с базами данных.

PHP – это язык программирования, предназначенный для веб-разработки, который используется на стороне сервера. Код на языке PHP может быть встроен в HTML код.

Flask – это небольшая программная платформа для разработки веб-приложений на языке программирования Python. Данная платформа предоставляет набор базовых функций – ядро, но при этом ее функциональные возможности можно расширить. Например, некоторые расширения Flask осуществляют поддержку интеграции баз данных, различных открытых технологий аутентификации, форм проверки [14].

Особенности Flask:

- поддержка кодировки Unicode;
- имеет встроенный сервер и отладчик;
- расширяемость;
- использование шаблонов Jinja2;
- обширная документация.

```python
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

Рисунок 10 – Пример простого приложение с использованием Flask

Jinja2 – это библиотека для языка программирования Pyhton. Представляет собой полнофункциональный механизм шаблонов Python, имеет полную поддержку кодировки Unicode. Jinja2 используется для генерации текстовых файлов, в том числе с расширением .html, на основе одного или нескольких шаблонов. Позволяет писать это один из самых распространенных шаблонных движков для языка Python [15].

```
<title>{% block title %}{% endblock %}</title>
<ul>
{% for user in users %}
  <li><a href="{{ user.url }}">{{ user.username }}</a></li>
{% endfor %}
</ul>
```

Рисунок 11 – Пример Jinja2 шаблона

SQLAlchemy – это программная библиотека на языке Python для работы с реляционными СУБД с применением технологии объектно-реляционного отображения (ORM), предоставляет разработчикам возможность использовать язык SQL, но при этом существует возможность создавать базу данных и взаимодействовать с ней, не прибегая к SQL. SQLAlchemy предоставляет возможность работать с базами данных PostgreSQL, SQLite, Oracle и другими.

2.3 Системы управления базами данных

База данных – это организованная совокупность данных, которая представляет собой коллекции схем, таблиц, запросов, отчетов и других объектов.

Между физической базой данных (т.е. данными, которые реально хранятся на компьютере) и пользователями системы располагается уровень программного обеспечения, который можно называть по-разному: диспетчер базы данных (database manager), сервер базы данных (database server) или, что более привычно, система управления базами данных, СУБД (DataBase Management System – DBMS). Все запросы пользователей на получение доступа к базе данных обрабатываются СУБД. Все имеющиеся средства добавления файлов (или таблиц), выборки и обновления данных в этих файлах или таблицах также предоставляет СУБД. Основная задача СУБД – дать пользователю базы данных возможность работать с ней, не вникая во все подробности работы на уровне аппаратного обеспечения. (Пользователь СУБД более отстранен от этих подробностей, чем прикладной программист, применяющий языковую среду программирования.) Иными словами, СУБД позволяет конечному пользователю рассматривать базу данных как объект более высокого уровня по сравнению с аппаратным обеспечением, а также предоставляет в его распоряжение набор операций, выражаемых в терминах языка высокого уровня (например, набор операций, которые можно выполнять с помощью языка SQL [16].

Oracle – это ведущая коммерческая СУБД, способная поддерживать гигантские базы данных и невероятные количества транзакций. Она работает под многими операционными системами и на разных платформах аппаратно-технического обеспечения. Наконец, эта СУБД настолько сложна, что для поддержания ее в рабочем состоянии нужен высококвалифицированный администратор баз данных [17].

MySQL – это одна из ведущих (то есть наиболее распространенных, качественных и покупаемых в настоящее время) СУБД с открытым исходным кодом. Она отличается быстродействием, устойчивостью и возможностью поддерживать базы данных больших объемов. Но, несмотря на все ее достоинства, среди которых основным и наиболее известным является быстродействие, MySQL не поддерживает нескольких очень важных функций SQL. Немаловажно и то, что MySQL работает под многими популярными операционными системами и на разных аппаратных платформах, будучи при этом бесплатной [17].

PostgreSQL – одна из ведущих СУБД с открытым исходным кодом. Она отличается быстродействием, устойчивостью и возможностью поддерживать базы данных больших объемов с огромным числом транзакций, а также обширной функциональностью и высоким уровнем соответствия стандарту ANSI SQL. Немаловажно и то, что PostgreSQL работает под многими популярными операционными системами и на разных аппаратных платформах, будучи при этом бесплатной [17].

2.4 Обоснование выбора технологий

С учетом требований для разработки программного продукта решено было использовать следующие технологии для клиентской части:

- язык программирования JavaScript;
- язык разметки страниц HTML5;
- язык описание стилей HTML страницы CSS3.

Для серверной части программы:

- язык программирования Python версии 3.5;
- Flask 0.10 – программная платформа для создания сайтов на языке Python;

В качестве системы управления базами данных была выбрана свободная объектно-реляционная СУБД PostgreSQL.

Выбор обусловлен тем, что поддержка технологий для клиентской части, осуществляется любыми, достаточно современными браузерами (в которые, в свою очередь встроена поддержка интерпретации и отладки сценариев JavaScript).

Для реализации серверной части была выбрана программная платформа Flask, так она простая и достаточно небольшая, поддерживающая расширения, позволяет быстро развернуть веб-приложение.

Главным критерием при выборе СУБД, была то что PostgreSQL является бесплатной, обладает более полной документацией, а также данная СУБД соответствует более современным стандартам SQL [17].

3 Разработка редактора

Процесс разработки химического редактора для поддержки процесса обучения состоит из трех главных частей:

– разработка клиентской части;

– разработка серверной части;

– подключение системы управления базами данных.

Поддержка процесса обучения реализуется на всех вышеперечисленных уровнях, клиентская часть обеспечивает вывод задания на построение, сервер осуществляет выдачу задания и проверку результата, а СУБД хранит в себе таблицу с описанием заданий.

3.1 Разработка клиентской части

Клиентская часть приложения отвечает за пользовательский интерфейс приложения (отображение в окне браузера) и отправляет запросы к серверу. Пользовательский интерфейс редактора должен обеспечивать следующие возможности:

– создание и редактирование химических соединений;

– доступ к различным инструментам редактора;

– обеспечение графического представления упрощенного варианта периодической таблицы Менделеева;

– вывод задания, которое необходимо выполнить пользователю;

– доступ к методическому материалу.

3.1.1 Разработка поля для создания и редактирования химических соединений

Одной из главных частей редактора химических элементов, является поле, на котором ведется отображение и редактирование химических элементов и связей. В стандарт HTLM5, входят две технологии, наиболее подходящие для реализации данного элемента пользовательского интерфейса:

– язык разметки масштабируемой графики SVG;

– Canvas – поверхность для создания растровых изображений при помощи сценариев, обычно, на языке программирования JavaScript.

Особенности формата SVG:

– поддержка всеми современными браузерами;

– JavaScript имеет прямой доступ к объектам SVG используя SVG DOM API, к ним напрямую можно привязывать события нажатия кнопкой мыши, наведение курсором и т.д.;

– векторные изображения масштабируются лучше, чем растровые;

– к объектам SVG можно подключать таблицы стилей CSS.

Особенности Canvas:

– возможность манипуляции каждым пикселем;

– изменение ширины или высоты у объекта Canvas, создаст его заново, то есть удалит все что было отрисовано на нем ранее;

– имеет аппаратное ускорение;

– разработчик сам должен реализовывать управление состоянием графического объекта на Canvas (выбор объекта и т.д.);

– возможность сохранения полученного изображения в PNG или JPG файл.

```
<svg>
    <rect width="100" height="100"/>
</svg>
```

Рисунок 12 – Пример SVG кода

Пример HTML кода для вывода SVG графики, приведен на рисунке 12. Данный код выведет на HTML странице черный квадрат с длинами сторон 100 пикселей.

```
<canvas id="canvas" height="200px" width="200px"></canvas>
<script>
    var canvas = document.getElementById('canvas');
    var ctx = canvas.getContext('2d');
    ctx.fillRect(5,5,100,100);
</script>
```

Рисунок 13 – Пример кода canvas графики

Код для отрисовки квадрата при помощи canvas, представлен на рисунке 13.

SVG лучше подходит для создания статичных изображений, графиков, диаграмм, чертежей и схем с высокой точностью и детальностью. Для обработки и редактирования сложных изображений с большим числом элементов, лучше подойдет Canvas.

Для реализации поля для редактирования была выбрана технология Canvas, так как она в данный момент лучше поддерживается браузерами и работает быстрее за счет аппаратного ускорения, к тому же объекты на поверхности Canvas создаются при помощи сценариев JavaScript, что позволит сохранять данные объекты в формате JSON – подмножестве языка JavaScript.

Для создания веб-страниц используется HTML разметка и язык шаблонов Jinja2. Для верстки страниц используется спецификация CSS Flexible Box

Layout Module, позволяющая контролировать размер, порядок и выравнивание элементов веб-страницы по нескольким осям, распределять свободное место между элементами и другие возможности.

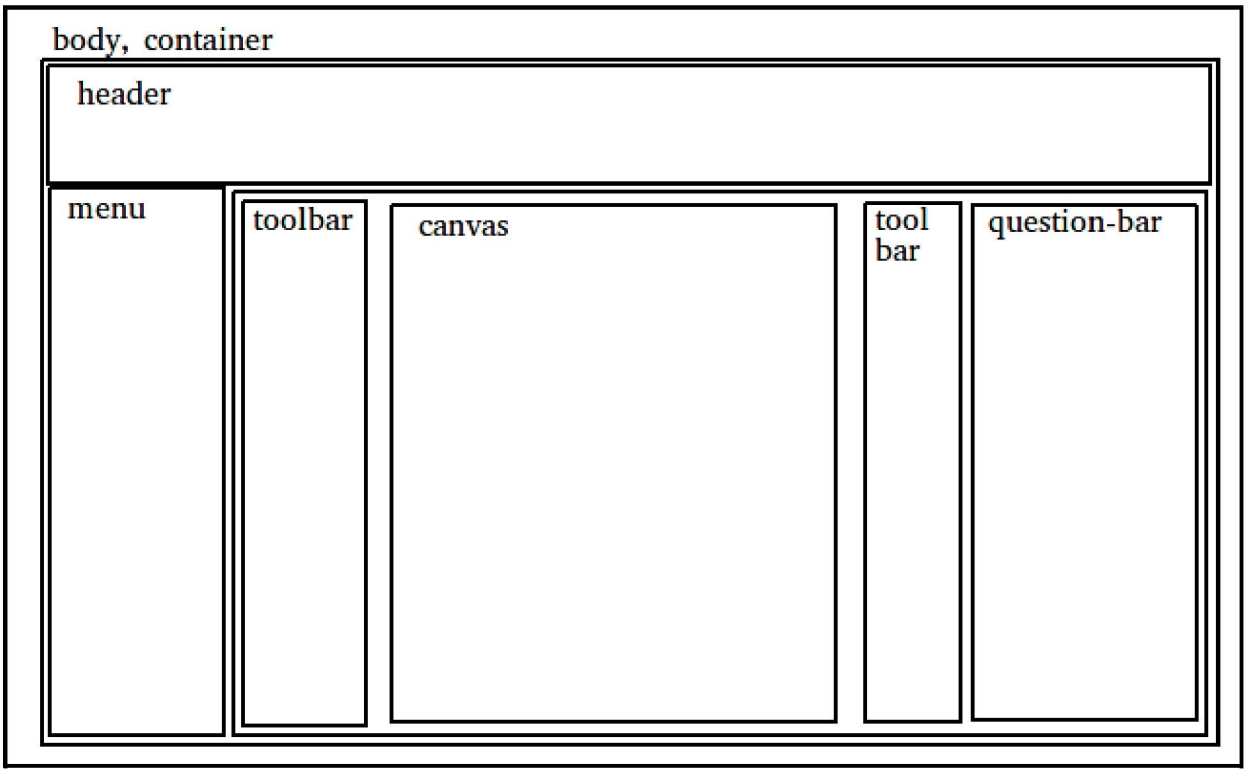

Рисунок 14 – Макет интерфейса для редактора химических формуд

Создание поля Canvas происходит путем добавления в HTML документ дескриптора <canvas>, а для работы с ним при помощи JavaScript требуется подключить внешний файл.

```
<canvas id="canvas" height="560px" width="560px"></canvas>
    <script src="{{ url_for('static', filename='js/script.js') }}">
</script>
```

Рисунок 15 – Пример кода для создания поля Canvas

В HTML коде, представленном на рисунке 15, в двойных фигурных скобках представлено выражение языка шаблонов Jinja2, которое подключает

файл со сценариями JavaScript из каталога на сервере /static/js.

```
function init() {
    canvas = document.getElementById('canvas');
    ctx = canvas.getContext('2d');
    WIDTH = canvas.width;
    HEIGHT = canvas.height;
}
```

Рисунок 16 – Функция инициализации поля Canvas

Функция инициализации поля Canvas, представленная на рисунке 16, получает доступ к элементу canvas HTML документа, получает его атрибуты width(ширина) и height(высота) для работы с позиционированием элементов на поле. Метод getContext, подготавливает элемент <canvas> к работе – создает контекст рисования, и связывает его с конкретным холстом.

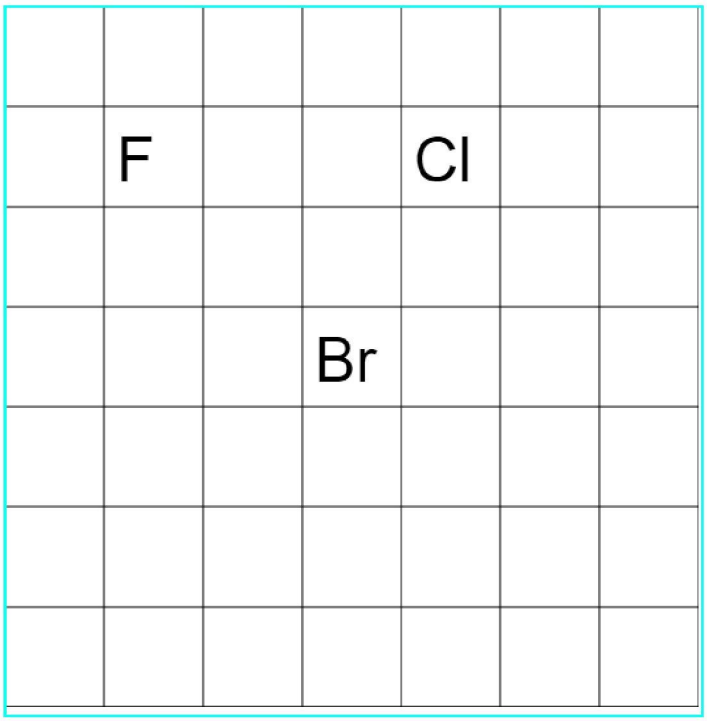

Рисунок 17 – Отображения холста в графическом редакторе

На пользовательском интерфейсе поле Canvas представлено в виде сетки (Рисунок 17). За хранение текущего состояния холста отвечает двумерный массив field, каждый элемент которого привязан к определенной ячейке сетки.

```
function Element() {
    this.name = ""; // name H,C ....
    this.bonds = 0; // number of bonds
    //position element on grid
    this.w = 0;
    this.h = 0;
}
```

Рисунок 18 – Конструктор объекта Element

Каждый элемент в двумерном массиве field – это объект Element (Рисунок 18), описывающий атом химического элемента, который характеризуется четырьмя параметрами:

- name – название элемента в периодической системе Менделеева;
- bonds – валентность атома;
- w – местоположение элемента на сетке, столбец;
- h – местоположение элемента на сетке, строка.

```
function drawGrid() {

    ctx.beginPath();
    ctx.lineWidth = 1;
    for (var i = 0; i <= WIDTH; i += interval) {
        ctx.moveTo(i, 0);
        ctx.lineTo(i, HEIGHT);
    }

    for (var j = 0; j <= HEIGHT; j += interval) {
        ctx.moveTo(0, j);
        ctx.lineTo(WIDTH, j);
    }
    ctx.globalAlpha = 1;
    ctx.closePath();
    ctx.stroke();
}
```

Рисунок 19 – Функция drawGrid

Функция drawGrid, представленная на рисунке 19, отображает сетку на поле Canvas, первый цикл for рисует горизонтальные линии, второй – вертикальные, свойство globalAlpha устанавливает значение прозрачности.

Для обработки реакции на действия пользователя, язык JavaScript осуществляет поддержку обработки событий. Событие – это сигнал от браузера о том, что произошло какое-то действие, например:

- click – происходит, при нажатии левой кнопки компьютерной мыши;
- mousemove – перемещение курсора мыши;
- submit – отправка формы <form>;
- focus – фокусировка на элементе.

```
canvas.addEventListener("mousemove", function (event) {
    if (event.type == "mousemove") {
        var mousePos = getMousePos(event, canvas);

        positionOnGrid = detectPosition(mousePos);
        drawCanvas();

    }
});
```

Рисунок 20 – Обработчик события mousemove

При наведении курсором на клетку в поле <canvas>, просчитывается положение курсора – функция getMousePos переводит значение координат по вертикальной и горизонтальной осям, в значение индексов двумерного массива, хранящего состояние холста. Далее выполняется функция drawCanvas, которая перерисовывает весь холст.

```
function drawCanvas(){
    function backLightCircles(pos){···
    }

    function drawElements() {···
    }
    ctx.clearRect(0,0, canvas.width, canvas.height); // clear canvas
    drawGrid();
    drawElements();
    drawBonds();
    backLightCircles(positionOnGrid);

}
```

Рисунок 21 – Функция drawCanvas

Функция drawCanvas реализует всю логику отрисовки на холсте <canvas>, очищает поле, рисует сетку, рисует элементы из массива field и линии химических связей из коллекции Bonds.

Функция backLightCircles подсвечивает выбранную клетку, но только если в ней находится атом химического элемента. Отрисовка кругов производит функция drawCircle, которая принимает в качестве параметров положение центра круга, радиус и цвет фоновой заливки.

```
function Bond (){
    this.x1 = 0;
    this.y1 = 0;
    this.x2 = 0;
    this.y2 = 0;
    this.type = ""; // single, double or triple

}
```

Рисунок 22 – Конструктор для объекта Bond

Для хранения химической связи реализован объект Bond (Рисунок 22), который характеризуется следующими свойствами:

– координаты начала линии химической связи;
– координаты окончания линии химической связи;

– тип ковалентной связи (одинарная, двойная, тройная).

Коллекция Bonds используется для хранения всех химических связей.

Функция drawElements проходит по всему двумерному массиву field, который хранит объекты Element, и выводит на экран в качестве текста, название атома химического элемента, находящегося в данной ячейке или, ничего не выводит, если ячейка пустая.

3.1.2 Создание панели с химическими элементами периодической таблицы

На сегодняшний день, в периодической таблице Менделеева насчитывается 126 химических элементов, такая большая таблица займет много места на пользовательском интерфейсе в окне браузера, поэтому было принято вывести на панель атомов, только некоторые элементы (C, H, N, O, P, S, F, Cl, Br, I), а полную периодическую таблицу выводить в модальном окне. Панель с химическими элементами располагается в столбец справа от поля редактирования.

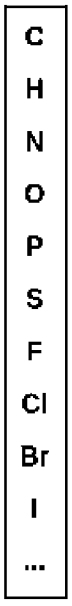

Рисунок 23 – Панель элементов

```
<div class="toolbar">
    <div id="element-tools">
        <div class="toolbar-inner">
            <div id="atom-c" title="Углерод" class="element-tool tool-button">
                C
            </div>
            <div id="atom-h" title="Водород" class="element-tool tool-button">
                H
            </div>
            ...
        </div>
    </div>
</div>
```

Рисунок 24 – HTML разметка панели элементов

Атрибут class, дескриптора указывает на то, что данный блок может повторятся в HTML документе, благодаря этому появляется возможность применять стили CSS сразу ко многим элементам. Атрибут id, дескриптора указывает на то, что данный элемент должен встречаться единственный раз, в HTML документе. Значение атрибута id для каждого химического элемента разное, так как каждый блок хранит в себе уникальное значение - текст с обозначением элемента.

```
.toolbar{
    margin-left: 5px;
    margin-right: 10px;
}

.toolbar-inner{
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    height: 560px;
    box-shadow: 0 3px 6px rgba(0,0,0,0.16), 0 3px 6px rgba(0,0,0,0.23);

}
```

Рисунок 25 – Стили для панели toolbar

Стили для класса toolbar, внутри которого определен блок toolbar-inner, устанавливают отступы. Для класса toolbar-inner устанавливаются: свойство display, определяет flex-контейнер, подключает flex для внутреннего содержимого класса toolbar; justify-content определяет выравнивание внутренних блоков; устанавливается высота блока toolbar-inner и тени вокруг него.

Выбор элемента на панели, обрабатывается событием click.

```
var atoms = document.getElementsByClassName('element-tool'); // from right panel

for (var i = 0; i < atoms.length; i++) {
    if (i != atoms.length - 1) { // atoms.length-1  -  call periodic table
        atoms[i].addEventListener('click', function () {
            store = this.innerText;
            state.name = "atom";
            //this.setAttribute('class', "element-tool-selected");
        });
    }

}
```

Рисунок 26 – Получение элементов из панели химических элементов, обработчик события click

Метод getElementsByClassName возвращает коллекцию дочерних элементов соответствующих, указанному в параметрах, имени класса. Цикл for проходит по коллекции и назначает каждому элементу, кроме последнего (кнопка вызова периодической таблицы, для которой существует отдельный обработчик), обработчик события click, при срабатывании которого содержимое блока (название элемента) записывается в переменную store.

3.1.3 Создание панели инструментов редактора

Панель инструментов редактора химических формул должна обеспечивать пользователя следующими возможностями для взаимодействия с полем Canvas:

1) отрисовка объектов. Для отрисовки химического соединения на поле элемента <canvas>, потребуется два типа объектов рисования:

   а) химический элемент периодической таблицы Менделеева;

   б) ковалентная связь (одинарная, двойная, тройная).

2) очистка поля <canvas>;

3) удаление элемента;

4) удаление связи;

5) перемещение элементов на поле;

6) отмена или повторение последнего выполненного действия.

Рисунок 27 – Представление панели инструментов на интерфейсе браузера

Так как каждый элемент панели элементов своя уникальная логика работы, то для блока задается атрибут id. Значение атрибута title выводится на экран при наведении на элемент курсором. Был использовал набор иконок (содержимое дескриптора <i>, class material-icons) и стилей, рекомендуемых корпорацией Google, эти же стили она использует для

внутренних разработок (например, Google Inbox). Фоновое изображения для элементов single-bond, double-bond и triple-bond, загружается из каталога на сервере static/img/bond. Стили для содержимого, класса tool-button, устанавливают стиль шрифта, выравнивает содержимое по центу блока и устанавливает форму курсора.

Обработка нажатия на панели инструментов происходит при помощи события click.

```
var clear = document.getElementById('clear');

clear.addEventListener('click', function () {
    initField();
    Bonds.splice(0, Bonds.length);
    drawCanvas();

});
```

Рисунок 28 – Функция отчистки экрана

Метод getElementById возвращает элемент, имеющий id атрибут равный значению, переданному в параметрах. При нажатии срабатывании события click на элементе clear, происходит очистка массива field и коллекции Bonds, после чего холст перерисовывается заново.

Благодаря тому, что состояние поле хранится в двух массивах, оно может быть легко преобразовано в объект JSON, являющимся подмножеством языка JavaScript. Появляется возможность загружать данные о состоянии поля из файла с расширением .json, и следовательно, появляется возможность сохранения состояния в файл. Текущее состояние поля характеризуется 4 свойствами:

- набор химических элементов, находящихся на холсте;
- набор химических связей;
- целочисленной значение, от которого зависит размер поля;

– размер шрифта, так как с изменением размера поля, меняется размер шрифта.

```
function JSONObject() {
    this.f =null;
    this.b = [];
    this.i = 0;
    this.fs = 0;
}
```

Рисунок 29 – Конструктор для объекта, хранящего поле

При вызове события click для элемента с атрибутом id равным file-save, формируется новый объект JSONObject, представленый на рисунке 29, метод JSON.stringify переносит объект JSONObject в строку JSON, а объект Blob создает из него файл.

Загрузка состояния поля из файла происходит следующим образом:

– происходит вызов модального окна с требованием выбрать файл, после этого идет отправка формы, срабатывает обработчик событие submit;

– происходит извлечение данных из файла и преобразование содержимого файла в объект JSONObject, из свойств которого заполняется состояние нового поля.

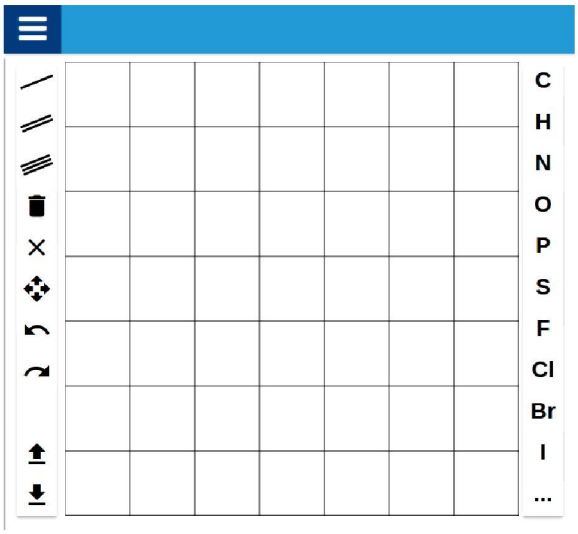

Рисунок 30 – Получившееся окно редактора

3.1.4 Создание панели вывода задания на построения химического соединения

Панель вывода задания обеспечивает вывод текста задания на пользовательский интерфейс, располагается в правой части экрана.

```
<div class="questionbar">
    <div id="task">
        <h1>{{ title_task }}</h1>
    </div>
    <div id="check-answer" class="">
        <i class="material-icons md-36">sync</i>
    </div>
    <div class="button-panel">
        <button id="back-button" class="btn" type="button"><span>Назад</span></button>
        <button id="next-button" class="btn" type="button"><span>Вперед</span></button>
    </div>
</div>
```

Рисунок 31 – HTML кода панели вывода задания

При загрузке страницы идет обращение к серверу, который извлекает из базы данных таблицу Tasks и возвращает значение поля, в котором хранится текст задания первой записи в таблице.

```
fetch('/tasks/'+taskIndex)
.then(function(response) {
    return response.json();
}).then(function(json) {
    //get name of task
    console.log('parsed json', json);
    title = json.task;
    console.log(title);
    taskTitle.innerHTML = "<h1>"+title+"</h1>";

}).catch(function(ex) {
    console.log('parsing failed', ex);
});
```

Рисунок 32 – Обращение к серверу, использование fetch

При отправке результата, вызывается функция convertToSMILES, которая последовательно обходит массив field, хранящий состояние поля. Встречая первую не пустую и не хранящую в себе атом водорода ячейку, вызывается рекурсивная функция checkBonds, которая, для с выбранного элемента, находит его связи с другими элементами, проверяет уже их связи, и так далее, попутно записывая полученные данные в строку SMILES.

SMILES – это спецификация для описания структуры химического соединения в виде строки обозначений в кодировке ASCII. Основные правила SMILES:

– все атомы записываются символами, водород (H) можно не обозначать;

– одинарные связи обозначаются символом «-» (можно не обозначать), двойные «=», тройные «#»;

– для обозначения разветвлений используются круглые скобки.

Полученная строка отправляется на сервер, там она проверяется, и сервер возвращает ответ, правильно построенная структура или нет.

```
function convertToSMILES() {
    function inSquareBrackets(str) {
        return "[" + str + "]";
    }

    function inRoundBrackets(str) {
        return "(" + str + ")";
    }


    function checkBonds(element) { // traversal of tree from "
    }

    SMILES = "";

    for (var i = 0; i < field.length; i++) {
        for (var j = 0; j < field[i].length; j++) {
            if (field[i][j].name && field[i][j].name != 'H') {
                SMILES += inSquareBrackets(field[i][j].name);

                console.log(counter);
                checkBonds(field[i][j]);
                console.log(SMILES);
                return SMILES;
            }
        }
    }

}
```

Рисунок 33 – Функция конвертирования химической структуры в формат SMILES

Блок-схемы алгоритма преобразования химической структуры в строку SMILES, можно увидеть в приложении А.

3.2 Разработка серверной части

Серверная часть приложения должна обеспечивать пользователя следующими возможностями:

1) обеспечивать доступ пользователя:
   а) к веб-страницам;
   б) к базе данных.
2) проверять правильность решения задания на построение химического соединения.

В программной платформе Flask, для обработки перехода по URL адресу существует декоратор route(), связывающий определенную функцию и URL.

```
@app.route('/SMILES-tutorial')
def SMILES_tutorial():
    return render_template('smiles-tutorial.html');
```

Рисунок. 34 – Обработка перехода по URL адресу

Код, приведенный на рисунке 34, возвращает HTML страницу smiles-tutorial.html при переходе по URL адресу http://127.0.0.1:5000/SMILES-tutorial.

При работе с библиотекой SQLAlchemy, для работы с реляционными СУБД, использовалась технология ORM, которая позволяет связывать базу данных с ее программным представлением, создавая «объектную базу данных», где каждая таблица в базе представлена в виде класса, в данном случае на языке Python.

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql://postgres:qwertyuiop@localhost/DiplomaDb'
db = SQLAlchemy(app)

from models import Task

tasks = Task.query.all()
```

Рисунок 35 – Настройка подключения к базе данных

Исходный код, представленный на рисунке 35, осуществляет подключение к базе данных DiplomaDB, подключает файл models.py в котором таблица Tasks представленная в виде класса, Task.query.all() возвращает коллекцию всех записей из таблицы Tasks.

```
class Task(db.Model):
    __tablename__ = 'tasks'

    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String())
    result = db.Column(db.String())

    def __init__(self, title, result):
        self.title = title
        self.result = result

    def __repr__(self):
        return '<id {}>'.format(self.id)
```

Рисунок 36 – Класс, описывающий таблицу Tasks

Программный код, представленный на рисунке 36, описывает таблицу Tasks.

Полностью, программный код клиентской и серверной частей, редактора, представлен в приложении Б.

Рисунок 37 – Представление панели вывода задания на пользовательском интерфейсе

3.3 СУБД PostgreSQL

База данных должна обеспечивать хранение информации о заданиях для построения химического соединения.

Таблица 1 – таблица Tasks

Название колонки	Тип хранимого значения
Id	int
Title	String
Solution	String

Столбец Id – первичный ключ, служит для уникальной идентификации записей таблицы. Title – хранит текст задания. Solution – хранит правильный ответ, строковое представление химического соединения в формате SMILES.

Диаграмма компонентов, разработанного химического редактора, представлена в приложении В.

4 Руководство пользователя

4.1 Требования для запуска приложения

Для корректной работы клиентской части приложения требуются последняя версия браузера Google Chrome или Mozilla Firefox и подключение к сети интернет. Для работы серверной части требуется установка пакетов Python3, Flask, PostgreSQL, Psycopg2, Flask-SQLAlchemy иац Flask-Migrate.

4.2 Развертывание приложения

После установки требуемых пакетов, через командную строку требуется перейти в корневой каталог приложения и выполнить команду «python manage.py db init» для инициализации миграций и создания базы данных (БД). Команда «python manage.py migrate» создаст миграцию базы данных, осонованую на модели из файла models.py. При выполнении команды «python manage.py upgrade», которая обноват базу данных до последней версии модели. Теперь база данных готова для использования приложением.

Команда «python3 DBcreation.py» заполнит БД тестовыми данными.

Команда «python3 manage.py runserver» запускает сервер приложения, перейдя в окне браузера по адресу «http://127.0.0.1:5000/», сервер выдаст стартовую страницу редактора.

4.3 Руководство к использованию редактора

Чтобы вызвать меню пользователю нужно нажать кнопку в левом верхнем углу страницы.

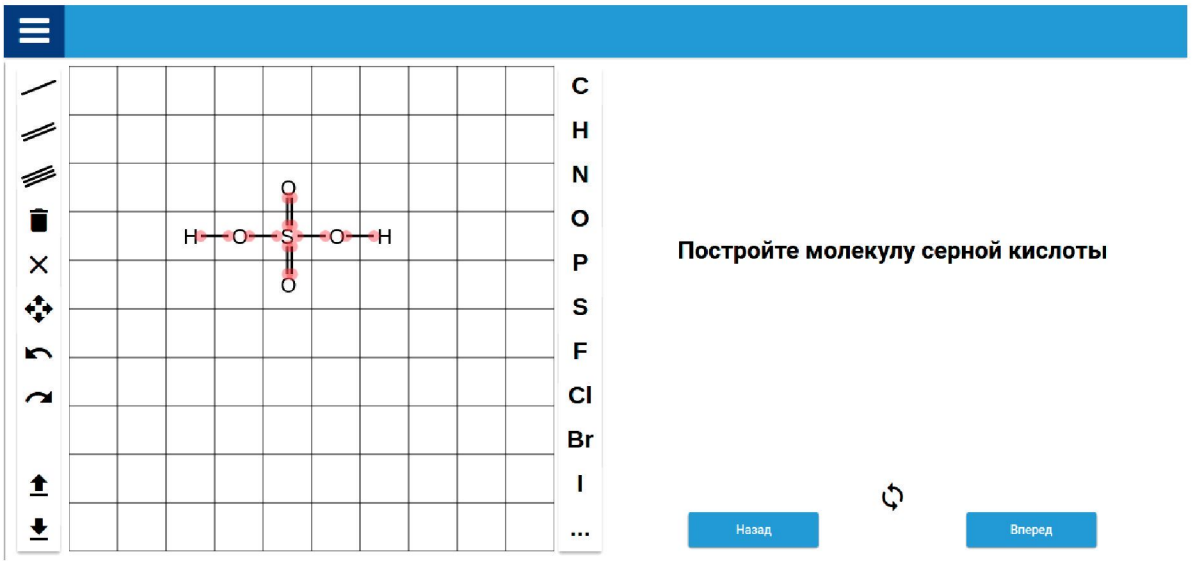

Рисунок 38 – Главная страница редактора

Процесс отображения атома химического элемента на экране происходит посредством выбора элемента на правой панели, после этого, при нажатии на клетку поля, в ней появится символ, состоящий из одной или пары букв – сокращенное название химического элемента. Чтобы добавить между двумя атомами ковалентную связь, следует выбрать на левой панели тип связи (одинарную, двойную, тройную), затем нажатием левой кнопки мыши сначала на одном элементе, а потом на другом, связь отобразится на экране, но только если соблюдены некоторые химические законы. Если пользователь выполняете задание, то после построения химического соединения, следует нажать на кнопку отправки ответа, которая находится под текстом задания, появится окно с результатом построения.

Следует отметить, что из-за недостаточной оптимизации алгоритма преобразования химической структуры в строку формата SMILES, при построении химического соединения, ковалентные связи следует добавлять последовательно, начиная от крайнего левого элемента, к крайнему правому.

Функциональная схема работы редактора представлена в приложении В.

ЗАКЛЮЧЕНИЕ

Результатом выполнения данной выпускной квалификационной работы стало создание редактора химических формул для поддержки процесса обучения. В процессе разработки использовались современные технологии разработки веб-приложений, были выполнены следующие требования:

– Химическое соединение должно отображаться в виде графического рисунка;

– Построение химического соединения должно производится в соответствии с законами химии;

– Для построения химических соединений должны использоваться химические элементы из таблицы Менделеева;

– Программа имеет пользовательский веб-интерфейс;

– Редактор обеспечивает поддержку обучения, предоставляет пользователю задания на построение химической формулы.

Одной из особенностей данной программы, для клиента, является то что пользователю не требуется установка дополнительного программного обеспечение, для работы с редактором требуется только современный браузер.

К недостаткам программного продукта можно отнести то, что поле для моделирования и редактирования химического соединения представлено в виде сетки, размер поля ограничен, кроме того, существуют некоторые проблемы при преобразовании химической структуры в строку SMILES.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. GitHub [электронный ресурс] : ресурс для хостинга проектов и их совместной разработки. / GitHub, Inc. Изд. 2015. – Режим доступа: https://github.com/blog/2047-language-trends-on-github свободный.

2. Molinspiration WebME Molecule Editor [электронный ресурс] : молекулярный онлайн-редактор. / Словакия Molinspiration Cheminformatics Nova ulica SK-900 26 Slovensky Grob Slovak Republic. – Режим доступа: http://www.molinspiration.com/docu/webme/index.html свободный.

3. XuMuK [электронный ресурс] : химический редактор бета-версия. – Режим доступа: http://www.xumuk.ru/rhf/ свободный.

4. Molecular Visualization Resources [электронный ресурс] : бесплатная программа для визуализации молекулярных структур. / Eric Martz. – Режим доступа: https://www.umass.edu/microbio/chime/abtchime.htm свободный.

5. ChemDoodle Web Components [электронный ресурс] : библиотека языка JavaScript. / 2008-2016 iChemLabs, LLC. All rights reserved. – Режим доступа: https://web.chemdoodle.com/demos/sketcher/ открытый.

6. MOLVIEW [электронный ресурс] : ресурс для моделирования химических элементов. / 2014, 2015 Herman Bergwerf, GitHub. – Режим доступа: http://molview.org/ свободный.

7. MOLVIEW3 [электронный ресурс] : блог создателя MolView. / 2014, 2015 Herman Bergwerf, GitHub. – Режим доступа: http://blog.molview.org/posts/2015/03/02/molview-3/ свободный.

8. Molview-1st-gen [электронный ресурс] : хранилище исходного кода проекта MolView. / GitHub, Inc. Изд. 2015. – Режим доступа: https://github.com/molview/molview-1st-gen свободный.

9. Avogadro [электронный ресурс] : сайт проекта Avogadro. / GNU Free Documentation License 1.2. – Режим доступа: http://avogadro.cc/wiki/Main_Page свободный.

10. SpringerOpen [электронный ресурс] : новости науки и техники. / Springer International Publishing Ltd unless otherwise stated. – Режим доступа: http://jcheminf.springeropen.com/articles/10.1186/1758-2946-4-17 свободный.

11. Java and Google Chrome Browser [электронный ресурс] : официальный сайт Java. / Oracle. – Режим доступа: https://java.com/en/download/faq/chrome.xml свободный.

12. W3C [электронный ресурс] : международное сообщество, разрабатывает web-стандарты. / 2016 W3C. – Режим доступа: https://www.w3.org/ свободный.

13. Mozilla Developer Network [электронный ресурс] : обучающая платформа для обучения веб-технологиям. /2005-2016 Mozilla Developer Network и отдельные соучастники. – Режим доступа: https://developer.mozilla.org/en-US/ свободный.

14. Welcome to Flask [электронный ресурс] : документация Flask. / 2013, Armin Ronacher. – Режим доступа: http://flask.pocoo.org/docs/0.10/ свободный.

15. Jinja [электронный ресурс] : документация Jinja2. / Copyright 2014 by Armin Ronacher. – Режим доступа: http://jinja.pocoo.org/ свободный.

16. Дейт, К. Дж. Введение в системы баз данных. : 8-е изд. – Москва: Вильямс, 2006. – 1328 с.

17. Фиайли, К. SQL : Руководство по изучению языка. – Москва: ДМК Пресс, 2013. – 456 с.

ПРИЛОЖЕНИЕ А

Блок-схемы

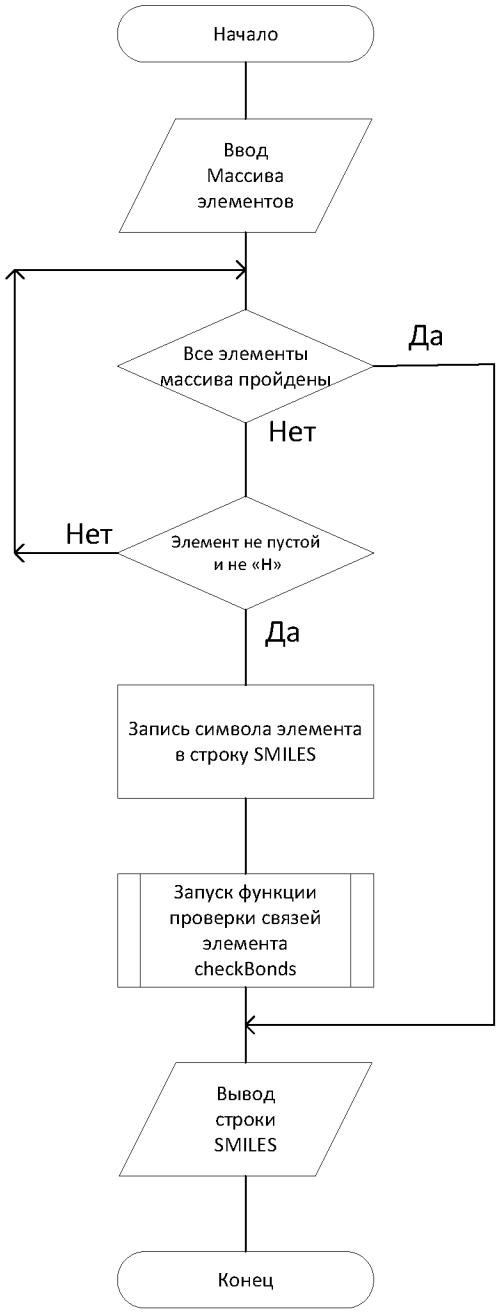

Рисунок А.1 – Блок-схема функции convertToSMILES

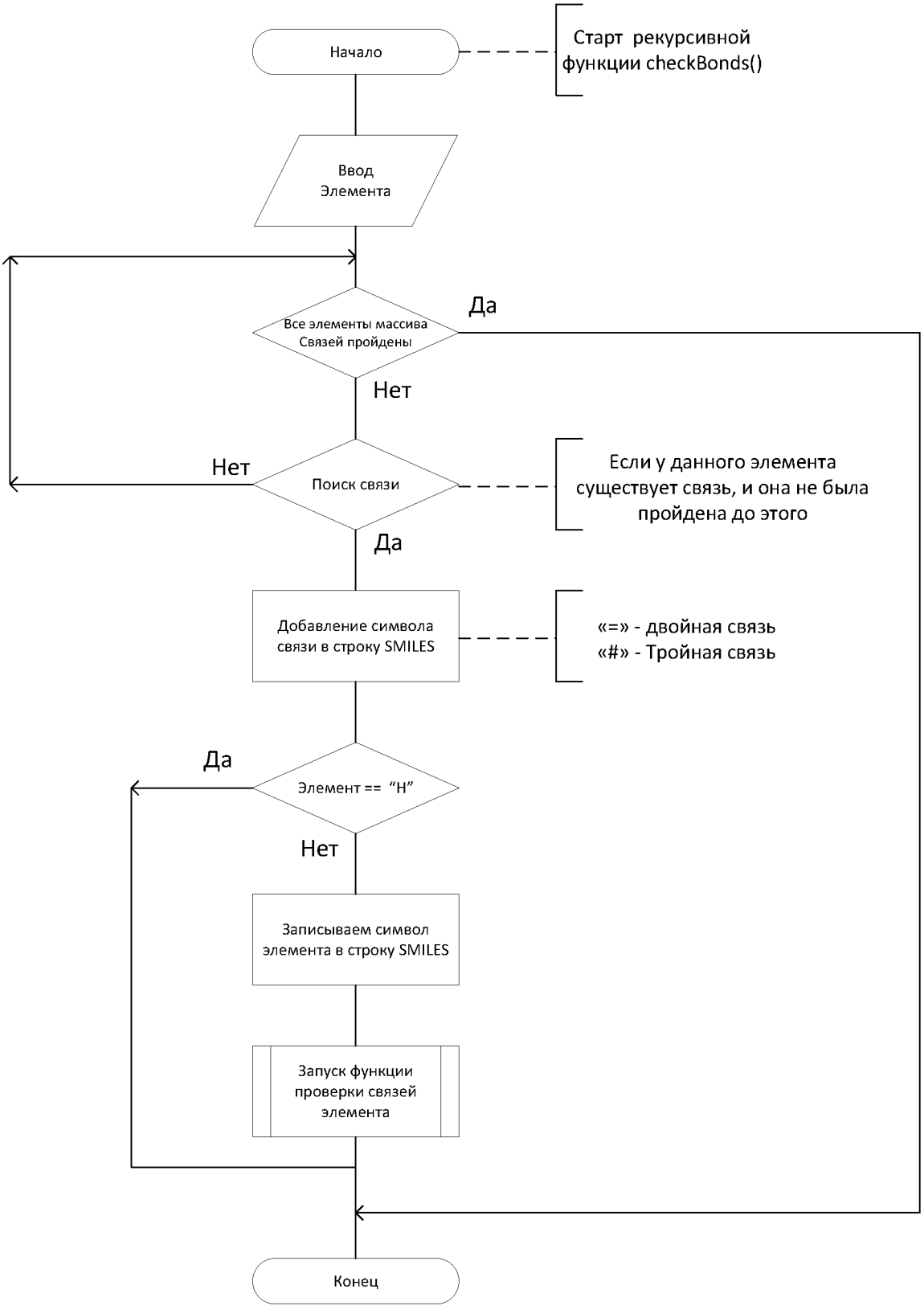

Рисунок А.2 – Блок-схема функции checkBonds

ПРИЛОЖЕНИЕ Б

Листинг файлов исходного кода программы

menu.js:

```
/**
  Side menu
*/
var menuWrapper = document.getElementsByClassName('ms-menu-body-wrapper')[0];

var msMenu = document.getElementsByClassName('ms-menu')[0];
//msMenu = document.querySelector('.ms-menu');

var modalMenu = document.getElementsByClassName('modal-menu')[0];

msMenu.addEventListener('click', function () {
    menuWrapper.classList.toggle('hide-menu');
    modalMenu.classList.toggle('modal-menu-show');
});

modalMenu.addEventListener('click', function () {
    menuWrapper.classList.toggle('hide-menu');
    modalMenu.classList.toggle('modal-menu-show');
});
```

bondLayout.js:

```
var Bonds = [];
var typeBond = 0;


function Bond (){
    this.x1 = 0;
    this.y1 = 0;
    this.x2 = 0;
    this.y2 = 0;
    this.type = 0; // single, double or triple
    this.backlightPadding = 5;

    this.isChecked = false;

}
```

```
        var singleBond = document.getElementById('single-bond');
        var doubleBond = document.getElementById('double-bond');
        var tripleBond = document.getElementById('triple-bond');

        /*
        for (var i = 0; i < toolButton.length; i++) {
            toolButton[i].addEventListener('click', clickOnToolButton);
        }
        function clickOnToolButton(){
            console.log(x);
        }
        */
        singleBond.addEventListener('click', clickOnBond);
        doubleBond.addEventListener('click', clickOnBond);
        tripleBond.addEventListener('click', clickOnBond);

        function clickOnBond(){
            state.name = "bond";
            switch (this.title) {
                case "Single bond":
                    typeBond = 1;
                    break;
                case "Double bond":
                    typeBond = 2;
                    break;
                case "Triple bond":
                    typeBond = 3;
                    break;
                default:
                    break;
            }

        }

        function addNewBond(x1, y1, x2, y2){
            function detectedElements(x1, y1, x2, y2) {
                function detectedNeigbor(x1, y1, x2, y2) {
                    if ((x1 === x2+1 && y1 === y2) || (x1 === x2 && y1 === y2 +1) || (x1
=== x2-1 && y1 === y2) ||
                     (x1 === x2 && y1 === y2 -1)) { // left top right bottom
                        return true;
                    }
                    return false;
                }

                if (field[x1][y1].name && field[x2][y2].name && detectedNeigbor(x1, y1,
x2, y2)) {
```

```
            return true;
        }
        return false;
    }

    function detectedCollisionBonds(bond){
        function collision(pointX1, pointX2, pointY1, pointY2) {
            if (pointX1 === pointX2 && pointY1 === pointY2) {
                return true;
            }
            return false;

        }

        for (var i = 0; i < Bonds.length; i++) {
            if ((collision(Bonds[i].x1, bond.x1, Bonds[i].y1, bond.y1) &&
             collision(Bonds[i].x2, bond.x2, Bonds[i].y2, bond.y2)) ||
             ( collision(Bonds[i].x1, bond.x2, Bonds[i].y1, bond.y2) &&
             collision(Bonds[i].x2, bond.x1, Bonds[i].y2, bond.y1))  ) {

                Bonds[i] = bond;
                return true;
            }
        }

        return false;
    }

    var bond = new Bond();
    bond.x1 = x1;
    bond.y1 = y1;
    bond.x2 = x2;
    bond.y2 = y2;
    bond.type = typeBond; // test

    if (field[x1][y1].name === "H" || field[x2][y2].name === "H") {
        bond.isChecked = true;
    }

    if (!detectedCollisionBonds(bond) && detectedElements(x1, y1, x2, y2)) {
        if (field[x1][y1].bonds >= typeBond && field[x2][y2].bonds >= typeBond) {
            field[x1][y1].bonds -= typeBond;
            field[x2][y2].bonds -= typeBond;
            Bonds.push(bond);
        }
        else {
            alert("incorrect bond");
```

```
            }
        }

    }

    var orientation = true; // true - horizontal or false - vertical (for course padding x or y )


    function drawBonds(){
        if (Bonds.length !== 0) {
            for (var i = 0; i < Bonds.length; i++) {

                var i1 = Bonds[i].x1, i2 = Bonds[i].x2, j1 = Bonds[i].y1, j2 = Bonds[i].y2;

                if (field[i1][j1].name && field[i2][j2].name) {
                    drawSingleOrDoubleOrTripleBond(Bonds[i]);
                }
                else {
                    Bonds.splice(i,1);
                }
            }
        }
    }

    function drawSingleOrDoubleOrTripleBond(bond) {
        function selectedTypeDrawLine() {
            switch (bond.type) {
                case 1:
                    drawSingleLine(this.x1, this.y1, this.x2, this.y2);
                    break;
                case 2:
                    drawDoubleLine(this.x1, this.y1, this.x2, this.y2);
                    break;
                case 3:
                    drawTripleLine(this.x1, this.y1, this.x2, this.y2);
                    break;
                default:
                    break;
            }
        }



        if (bond.x1 > bond.x2) { // it's left bond
            x1 = bond.x1 * interval + padding;
```

```
            y1 = bond.y1 * interval + interval / 2;

            x2 = bond.x2 * interval + interval - padding;
            y2 = y1;
            orientation = false;
            selectedTypeDrawLine();
            return;
        }

        if (bond.x1 < bond.x2) { // it's right bond
            x1 = bond.x1 * interval + interval - padding;
            y1 = bond.y1 * interval + interval / 2;

            x2 = bond.x2 * interval + padding;
            y2 = y1;

            orientation = false;
            selectedTypeDrawLine();

            return;
        }

        if (bond.y1 > bond.y2) { // it's top bond
            x1 = bond.x1 * interval + interval / 2;
            y1 = bond.y1 * interval + padding;

            x2 = x1;
            y2 = bond.y2 * interval + interval - padding;
            orientation = true;
            selectedTypeDrawLine();
            return;
        }

        if (bond.y1 < bond.y2) { // it's bottom bond
            x1 = bond.x1 * interval + interval / 2;
            y1 = bond.y1 * interval + interval - padding;

            x2 = x1;
            y2 = bond.y2 * interval + padding;
            orientation = true;
            selectedTypeDrawLine();
            return;
        }
    }

    function drawLine(x1, y1, x2, y2) {
        ctx.beginPath();
```

```
    ctx.lineWidth = 3;
    ctx.globalAlpha = 1;
    ctx.moveTo(x1,y1);
    ctx.lineTo(x2,y2);
    ctx.stroke();


    drawCirlce(x1, y1, '#F44336');
    drawCirlce(x2, y2, '#F44336');
}


function drawSingleLine(x1, y1, x2, y2){
    drawLine(x1, y1, x2, y2);


}

function drawDoubleLine(x1, y1, x2, y2) {
    var gap = interval / 10;
    if (!orientation) {
        drawLine(x1, y1, x2, y2);
        drawLine(x1, y1 + gap, x2, y2 + gap);
    }
    else {
        drawLine(x1, y1, x2, y2);
        drawLine(x1 + gap, y1, x2 + gap, y2);
    }


}

function drawTripleLine(x1, y1, x2, y2) {
    var gap = interval / 10;
    if (!orientation) {
        drawLine(x1, y1 - gap, x2, y2 - gap);
        drawLine(x1, y1, x2, y2);
        drawLine(x1, y1 + gap, x2, y2 + gap);


    }
    else {
        drawLine(x1 - gap, y1, x2 - gap, y2);
        drawLine(x1, y1, x2, y2);
        drawLine(x1 + gap, y1, x2 + gap, y2);

    }
```

```
}
```

instruments.js:

```
/* clear all */
var clear = document.getElementById('clear');

clear.addEventListener('click', function () {
    initField();
    Bonds.splice(0, Bonds.length);
    drawCanvas();
    counterElements = 0;

});

/*Erase
selected item
undo
redo*/

var json;
function JSONObject() {
    this.f =null;
    this.b = [];
    this.i = 0;
    this.fs = 0;
}

/*Export in json*/

var save = document.getElementById('file-export');

save.addEventListener('click', function () {
    function addNewJSONObject(f, b, i, fs){
        var object = new JSONObject();
        object.f = f;
        object.b = b;
        object.i = i;
        object.fs = fs;
        return object;
    }
    json = JSON.stringify(addNewJSONObject(field, Bonds, interval, fontSize),
null, '\t');
    var blob = new Blob([json], {type: "application/json"});
    var url  = URL.createObjectURL(blob);
```

```
            var a = document.createElement('a');
            a.style = "display: none";
            a.download = "field.json";
            a.href = url;
            a.textContent = "Download .json";
            a.click();
            window.URL.revokeObjectURL(url); //Call this method when you've finished using
a object URL

        });

        /*Import from json*/
        var modalForm = document.getElementById('modal-upload-form');
        var showForm = document.getElementById('file-import');
        var close = document.getElementsByClassName('close')[1];



        showForm.addEventListener('click', function(event) {
            modalForm.style.display = "block";
        });


        close.onclick = function() {
            modalForm.style.display = "none";
        };

        var form = document.querySelector('form');

        form.addEventListener('submit', function(event){
            event.preventDefault();

            var reader = new FileReader();
            var file = event.target.file.files[0];

            json = new JSONObject();
            if (file) {
                reader.onload = (function (theFile) {
                    return function (e) {
                        json = JSON.parse(e.target.result);
                        interval = json.i;
                        initField();
                        fontSize = json.fs;
                        field = json.f;
                        Bonds = json.b;
                        console.log(json);
                    };
```

```
        })(file);
        reader.readAsText(file);
    }


    close.click();


    /* for send file to server
    event.preventDefault();
    fetch('/json', {
      method: 'POST',
      body: new FormData(form)
    });
    */

});
```

questionbar.js:

```
var next = document.getElementById('next-button');
var back = document.getElementById('back-button');
var checkAnswer = document.getElementById('check-answer');


var taskTitle = document.getElementById('task');
console.log(taskTitle);


var taskIndex = 0;

function getTitleTask() {
    var title = "";

    fetch('/tasks/'+taskIndex)
    .then(function(response) {
        return response.json();
    }).then(function(json) {
        //get name of task
        console.log('parsed json', json);
        title = json.task;
        console.log(title);
        taskTitle.innerHTML = "<h1>"+title+"</h1>";

    }).catch(function(ex) {
        console.log('parsing failed', ex);
    });
```

```
}

next.addEventListener('click', function () {
    taskIndex = 1; //for test
    getTitleTask();
});

back.addEventListener('click', function () {
    taskIndex = 0; //for test
    getTitleTask();
});


checkAnswer.addEventListener('click', function () {
    switch (taskIndex) {
        case 0:
            testValue = convertToSMILES();
            if (testValue) {
                fetch("/checked/"+testValue)
                .then(function(response) {
                    return response.json();
                }).then(function(json) {
                    console.log('parsed json', json);
                    isFinded = json.isFinded;
                    console.log(isFinded);
                    if (isFinded === "true") {
                        alert("Правильно!");
                    }
                    else {
                        alert("Не правильно!");
                    }


                }).catch(function(ex) {
                    console.log('parsing failed', ex);
                });
            }
            else {
                alert("Поле пустое!");
            }
            break;
        case 1:
            testValue = convertToSMILES();
            if (testValue) {
                fetch("/checked/"+testValue)
                .then(function(response) {
                    return response.json();
```

```
                    }).then(function(json) {
                        console.log('parsed json', json);
                        isFinded = json.isFinded;
                        console.log(isFinded);
                        if (isFinded === "true") {
                            alert("Правильно!");
                        }
                        else {
                            alert("Не правильно!");
                        }


                    }).catch(function(ex) {
                        console.log('parsing failed', ex);
                    });
                }
                else {
                    alert("Поле пустое!");
                }
                break;
            default:
                break;

        }
    });

    var SMILES = "";


    var counter = 0;

    function convertToSMILES() {
        function inSquareBrackets(str) {
            return "[" + str + "]";
        }

        function inRoundBrackets(str) {
            return "(" + str + ")";
        }


        function checkBonds(element) { // traversal of tree from "element"
            for (var k = 0; k < Bonds.length; k++) {
                if ((Bonds[k].x1 === element.w && Bonds[k].y1 === element.h) ||
                    (Bonds[k].x2 === element.w && Bonds[k].y2 === element.h) &&
!Bonds[k].isChecked) {
                    Bonds[k].isChecked = true;
```

```
            var i = 0, j = 0;
            if (Bonds[k].type ===2) {
                SMILES += "=";
            }
            else {
                if (Bonds[k].type === 3) {
                    SMILES += "#";
                }
            }


            if (Bonds[k].x1 === element.w && Bonds[k].y1 === element.h) {
                i = Bonds[k].x2;
                j = Bonds[k].y2;
            }
            else {
                i = Bonds[k].x1;
                j = Bonds[k].y1;
            }

            if (field[i][j].name != "H") {
                SMILES += inSquareBrackets(field[i][j].name);
                checkBonds(field[i][j]);
                counter++;
                console.log(counter);
            }


        }
    }

}

SMILES = "";

for (var i = 0; i < field.length; i++) {
    for (var j = 0; j < field[i].length; j++) {
        if (field[i][j].name && field[i][j].name != 'H') {
            SMILES += inSquareBrackets(field[i][j].name);

            console.log(counter);
            checkBonds(field[i][j]);
            console.log(SMILES);
            return SMILES;
        }
    }
```

```
    }

}
```

modal.js:

```
/*modal
http://www.w3schools.com/howto/howto_css_modals.asp
*/
var modal = document.getElementById('modal-periodictable');
var showPerTable = document.getElementById("button-periodictable");

var span = document.getElementsByClassName("close")[0];

// When the user clicks on the button, open the modal
showPerTable.onclick = function() {
    modal.style.display = "block";
};

// When the user clicks on <span> (x), close the modal
span.onclick = function() {
    modal.style.display = "none";
};

// When the user clicks anywhere outside of the modal, close it
window.onclick = function() {
    if (event.target == modal) {
        modal.style.display = "none";
    }
    if (event.target == modalForm) {
        modalForm.style.display = "none";
    }

};
```

script.js:

```
var atoms = document.getElementsByClassName('element-tool'); // from right panel
var periodictable = document.getElementsByClassName('pt-element'); // from
periodictable modal


function State(){
    this.name ="";
    this.countClick = 0;
```

```
    this.first = 0;
    this.second = 0;

    this.Clear = function(){
        this.name = "";
        this.countClick = 0;
        this.first = 0;
        this.second = 0;
    };
    //state params
}

var state = new State();

var store = ""; // store name of selected element


for (var i = 0; i < periodictable.length; i++) {
    periodictable[i].addEventListener('click', function () {
        store = this.getElementsByTagName('h4')[0].innerText;
        modal.style.display = "none";
        state.name = "atom";
    });
}

for (var i = 0; i < atoms.length; i++) {
    if (i != atoms.length - 1) { // atoms.length-1  -  call periodic table
        atoms[i].addEventListener('click', function () {
            store = this.innerText;
            state.name = "atom";
            //this.setAttribute('class', "element-tool-selected");
        });
    }

}

var canvas = document.getElementById('canvas');
var ctx = canvas.getContext('2d');

var WIDTH;
var HEIGHT;
var interval = 80;
var padding = interval/5;

var positionOnGrid = 0;

var fontSize = 40;
```

```
var field; // elements on grid
var elements = [];

var counterElements = 0;

// it's bonds for test, "Br" and "I" is not fully
var covalentBonds = new Map([["H",1],["F",1],["Li",1],["Na",1],["K",1],
                    ["O",2],["Ca",2],["Mg",2],
                    ["B",3],["Al",3],
                    ["C",4],["Si",4],
                    ["N",5],["P",5],["Br",5],
                    ["S",6],
                    ["F",7],["Cl",7],["I",7]]);

function Element() {
    this.name = ""; // name H,C ....
    this.bonds = 0; // number of bonds

    //position element on grid
    this.w = 0;
    this.h = 0;
}

//Initialize a new Element, add it
function addNewElement(n, b, w, h, fill) {
    var element = new Element();
    element.name = n;
    element.bonds = b;
    element.w = w;
    element.h = h;
    field[w][h] = element;
    console.log(field[w][h]);
}



function init() {
    canvas = document.getElementById('canvas');
    ctx = canvas.getContext('2d');
    WIDTH = canvas.width;
    HEIGHT = canvas.height;

}

function matrixArray(columns, rows) {
    var arr = new Array(rows);
```

```
            for (var i = 0; i < arr.length; i++) {
                arr[i] = new Array(columns);
            }
            return arr;
        }

        function initField() {
            field = matrixArray(canvas.width / interval, canvas.height / interval);
            for (var i = 0; i < field.length; i++) {
                for (var j = 0; j < field[i].length; j++) {
                    field[i][j] = new Element();
                }
            }
        }

        function getMousePos(evt, canvas){
            return {
                x: evt.clientX + document.body.scrollLeft +
document.documentElement.scrollLeft - canvas.offsetLeft,
                y: evt.clientY + document.body.scrollTop +
document.documentElement.scrollTop - canvas.offsetTop
            };
        }

        canvas.addEventListener("click", function (event) {
            if (event.type == "click") {

                var mousePos = getMousePos(event, canvas);
                var positionOnGrid = detectPosition(mousePos);

                switch (state.name) {
                    case 'atom':
                        var bonds = covalentBonds.get(store);
                        addNewElement(store, bonds, positionOnGrid.x, positionOnGrid.y,
'#000000');
                        counterElements++;
                        drawCanvas();
                        break;
                    case 'bond':
                        state.countClick +=1;
                        if (state.countClick === 1) {
                            state.first = positionOnGrid;
                        } else {
                            state.second = positionOnGrid;
                            if (state.first === state.second) {
                                state.Clear();
                                break;
```

```
                        }
                        addNewBond(state.first.x,state.first.y, state.second.x,
state.second.y);
                        state.Clear();
                    }
                    break;
                default:
                    break;
            }

        }

    });

    canvas.addEventListener("mousemove", function (event) {
        if (event.type == "mousemove") {

            var mousePos = getMousePos(event, canvas);

            positionOnGrid = detectPosition(mousePos);

        }
    });

    canvas.addEventListener("wheel", function (event) {
        function zoomField(delta, newField){
            if (delta < 0) {
                for(var i = 0; i <field.length; i++){
                    for(var j =0; j<field[i].length; j++){
                        field[i][j] = newField[i][j];
                    }
                }
            }

            if (delta>0) {
                for( var k = 0; k <newField.length; k++){
                    for(l =0; l<newField[k].length; l++){
                        field[k][l] = newField[k][l];
                    }
                }
            }
        }

        if (event.type == "wheel") {

            var delta = event.deltaY;
```

```
        if ((delta <0 && interval === 56) || (delta >0 && interval === 140)) {
            interval = 80;
            fontSize = 40;
        }
        else {
            if (delta < 0) { // zoom +
                interval = 140;
                fontSize = 100;
            }
            if (delta > 0) { // zoom -
                interval = 56;
                fontSize = 24;
            }
        }
        var newField = field;
        initField();
        zoomField(delta, newField);
        drawCanvas();

    }
});
function drawCirlce(centerX, centerY, style){
    var radius = interval/8;
    ctx.beginPath();
    ctx.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);
    ctx.lineWidth = 0;
    ctx.fillStyle = style;
    ctx.globalAlpha = 0.5;
    ctx.fill();
}

function drawGrid() {

    ctx.beginPath();
    ctx.lineWidth = 1;
    for (var i = 0; i <= WIDTH; i += interval) {
        ctx.moveTo(i, 0);
        ctx.lineTo(i, HEIGHT);
    }

    for (var j = 0; j <= HEIGHT; j += interval) {
        ctx.moveTo(0, j);
        ctx.lineTo(WIDTH, j);
    }
    ctx.globalAlpha = 1;
    ctx.closePath();
    ctx.stroke();
```

```
        }
        function detectPosition(mousePos) {
            var moduloX = Math.floor(mousePos.x / interval);
            var moduloY = Math.floor(mousePos.y / interval);

            return {
                x: moduloX,
                y: moduloY
            };
        }
        function drawCanvas(){
            function backLightCircles(pos){
                //console.log(pos.x);
                //console.log(pos.y);
                if (field[pos.x][pos.y].name) {

                    var x = pos.x*interval;
                    var y= pos.y*interval;

                    drawCirlce(x+ padding, y+ interval/2, '#0D47A1');
                    drawCirlce(x+ interval/2, y+ padding);
                    drawCirlce(x+ interval - padding, y+ interval/2, '#0D47A1');
                    drawCirlce(x+ interval/2, y+ interval - padding, '#0D47A1');

                }
            }
            function drawElements() {
                for (var i = 0; i < WIDTH/interval; i++) {
                    for (var j = 0; j < HEIGHT/interval; j++) {
                        var marginTop = 20;
                        var marginLeft = 20;
                        ctx.textBaseline = "top";
                        ctx.font = fontSize + "px Arial";
                        ctx.fillStyle = "#000000";
                        ctx.fillText(field[i][j].name, field[i][j].w * interval +
marginLeft, field[i][j].h * interval + marginTop);
                        ctx.globalAlpha = 1;

                    }
                }
            }
            ctx.clearRect(0,0, canvas.width, canvas.height); // clear canvas
            drawGrid();
            drawElements();
            drawBonds();
            backLightCircles(positionOnGrid);
```

```
}

init();
initField();
drawGrid();
```

head-styles.css:

```
header{
    height: 60px;
    width: 100%;
    min-width: 1100px;
    background: #2196F3;
    box-shadow: 0 2px 5px rgba(0,0,0,0.26);
}
header .wrap{
    height: 60px;
    width: 100%;



}

.ms-menu-bar-wrapper {
    width: 60px;
    background: #0D47A1;
    padding: 5px;
    float: left;
    position: relative;
    z-index: 10;
}

.ms-menu-trigger {
    height: 50px;
    font-size: 40px;
    text-align: center;
    cursor: pointer;
    transition: color 0.6s;
}

.ms-menu-trigger:hover {
    color: #2196F3;
}

.fa-bars{
```

```
    margin-top: 4px;
}
```

menu-styles.css:

```
.ms-menu-body-wrapper {
    z-index: 2;
    background: white;
    box-shadow: 5px 0 5px -5px #333;
    position: absolute;
    left: 0;
    top: 60px;
    width: 300px;
    height: calc(100% - 60px);
    transition: transform 0.2s cubic-bezier(0.4, 0, 0.2, 1);
}

.ms-menu-body-wrapper.hide-menu {
    transform: translateX(-100%);
}
.ms-menu-body-wrapper ul{
    margin: 0;
    padding: 0;
    list-style: none;
}
.ms-menu-body-wrapper ul li {
    margin: 0;
}
.ms-menu-body-wrapper ul li a {
    padding: 15px 20px;
    font-family: 'Roboto', sans-serif;
    font-size: 15px;

    color: black;
    text-decoration: none;
    display: block;
}
.ms-menu-body-wrapper ul li:hover a {
    color: #0D47A1;
}

.modal-menu{
    height: calc(100% - 60px);
    z-index: 1;
    opacity: 0;
    position: absolute;
    bottom: 0;
```

```
    background-color: black;
    transition: opacity 0.2s cubic-bezier(0.4, 0, 0.2, 1);

}
.modal-menu-show{
    top: 60px;
    left: 0;
    bottom: 0;
    right: 0;
    opacity: 0.4;
    width: 100%;
    height: calc(100% - 60px);
    min-width: 1100px;
}
```

modal-styles.css:

```
.modal{
    display: none;
    position: fixed;
    z-index: 1;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    overflow: auto;
    background-color: rgb(0,0,0);
    background-color: rgba(0,0,0,0.4);
}
.modal-content, .modal-content-form  {
    background-color: #fefefe;
    margin: 15% auto;
    padding: 20px;
    border: 1px solid #888;
    width: 70%;
    height: auto;
}
.modal-content-form {
    margin: 15% auto;
    border: 1px solid #888;
    width: 350px;
}

.close:hover,
.close:focus {
    cursor: pointer;
```

```
}
```

pt-styles.css:

```
/*-------------------------------------------*/
/*periodic table*/
#periodictable{
    display: flex;
    flex-direction: column;;

}

.row{
    display: flex;
    flex-direction: row;

}
#periodictable >div{

}
.pt-element{
    cursor: pointer;
}

.pt-space, .pt-element{
    height: 40px;
    width: 50px;

}
```

styles.css:

```
*{
    margin: 0;
    padding: 0;
}
html,
body{
    width: 100%;
    height: 100%;
}
body{
    background-color: #ffffff;
    font-family: "Roboto","Helvetica","Arial",sans-serif;
    font-size: 14px;
```

```
}

ul, ol, li {
    list-style: none;
    color: white;
    margin: 0;
    padding: 0;
}

.container{
    min-height: 100%;
    min-width: 100%;
    min-width: 1100px;

}

.molpad{
    /*height: calc(100% - 60px);*/
    display: flex;
    flex-direction: row;
    align-items: stretch;
    align-content: stretch;
    padding: 5px;
    margin: 5px;

}
.canvas-content, .toolbar, .questionbar{
    /*box-shadow: 0 3px 6px rgba(0,0,0,0.16), 0 3px 6px rgba(0,0,0,0.23);*/
}

.toolbar{
    margin-left: 5px;
    margin-right: 10px;
}

.questionbar{
    margin-right: 10px;
    padding: 0 0 8px 5px;
    display: flex;
    flex-direction: column;
    justify-content:flex-end;
    align-items: stretch;
    flex: 1;
}
#task, .button{
    text-align: center;
```

```
}


#task{
    margin-bottom: 250px;
}

.button-panel{
    min-width: 350px;
    display: flex;
    flex-direction: row;
    justify-content:space-around;
    align-items: center;


}
.btn{
    /*
    margin-left: 8px;
    float: left;
    width: calc(50% - 8px);
    min-width: 150px;
    cursor: pointer;
    color: white;
    background-color: #2196F3;
    */
    position: relative;

    width: 150px;
    height: 40px;
    padding: 0;

    overflow: hidden;

    border-width: 0;
    outline: none;
    border-radius: 2px;
    box-shadow: 0 1px 4px rgba(0, 0, 0, .6);
    background-color: #2196F3;
    color: #ecf0f1;

    transition: background-color .3s;

}

btn:hover, .btn:focus {
    background-color: #1976D2;
```

```
}

.btn > * {
    position: relative;
}

#questionbar{

}

#check-answer{
    text-align: center;
    vertical-align: middle;
    cursor: pointer;
}

.btn span {
    display: block;
    padding: 12px 24px;
    font-family: "Roboto","Helvetica","Arial",sans-serif;
    font-size: 14px;
    font-weight: 500;
}

.btn:before {
    content: "";
    position: absolute;
    top: 50%;
    left: 50%;
    display: block;
    width: 0;
    padding-top: 0;

    border-radius: 100%;

    background-color: rgba(227, 242, 253, .3);

    -webkit-transform: translate(-50%, -50%);
    -moz-transform: translate(-50%, -50%);
    -ms-transform: translate(-50%, -50%);
    -o-transform: translate(-50%, -50%);
    transform: translate(-50%, -50%);
}

.btn:active:before {
    width: 120%;
    padding-top: 120%;
```

```
    transition: width .2s ease-out, padding-top .2s ease-out;
}


.material-icons{
    text-align: center;
    vertical-align: middle;

}

.material-icons.md-36{
     font-size: 36px;

 }

.toolbar-inner{
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    height: 560px;
    box-shadow: 0 3px 6px rgba(0,0,0,0.16), 0 3px 6px rgba(0,0,0,0.23);

}

.toolbar-inner >div, .empty-tool{
    line-height: 50px;
    width: 50px;
    height: 50px;
}
.tool-button{
    font-family: 'Roboto Black', sans-serif;
    font-weight: 700;
    font-size: 28px;
    text-align: center;
    cursor: pointer;
}
.toolbar-inner >div:active, #check-answer>i:active{
    background: #2196F3;

}

#single-bond{
    background-image: url('../img/bond/single.svg');
}

#double-bond{
    background-image: url('../img/bond/double.svg');
```

```
}

#triple-bond{
    background-image: url('../img/bond/triple.svg');
}

#single-bond, #double-bond, #triple-bond{
    background-repeat: no-repeat;
    background-position: center;
}
```

base.html:

```
<!DOCTYPE html>
<html>
    <head>
        {% block head %}
            <meta charset="utf-8">
            <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">
            <title>{% block title %}{% endblock %}</title>
            <link rel="stylesheet" href="{{ url_for('static',
filename='css/styles.css') }}">
            <link rel="stylesheet" href="{{ url_for('static', filename='css/menu-
styles.css') }}">
            <link rel="stylesheet" href="{{ url_for('static', filename='css/modal-
styles.css') }}">
            <link rel="stylesheet" href="{{ url_for('static', filename='css/pt-
styles.css') }}">
            <link rel="stylesheet" href="{{ url_for('static', filename='css/head-
styles.css') }}">

            <link rel="stylesheet" href="http://maxcdn.bootstrapcdn.com/font-
awesome/4.3.0/css/font-awesome.min.css">
            <link
href='https://fonts.googleapis.com/css?family=Roboto:400,400italic,500,500italic,700,700i
talic,900,900italic&subset=latin,cyrillic' rel='stylesheet' type='text/css'>
            <link
href="https://fonts.googleapis.com/icon?family=Material+Icons"rel="stylesheet">
        {% endblock %}
    </head>
    <body>
        <div class="container">
            <header>
                {% include 'header-template.html' %}
            </header>
```

```
            {% include 'menu-template.html' %}
            {% block molpad %}
            {% endblock %}
        </div>
        <script src="{{ url_for('static', filename='js/modal.js') }}">
        </script>
        <script src="{{ url_for('static', filename='js/menu.js') }}">
        </script>
        <script src="{{ url_for('static', filename='js/questionbar.js') }}">
        </script>
        <script src="{{ url_for('static', filename='js/script.js') }}">
        </script>
        <script src="{{ url_for('static', filename='js/bondLayout.js') }}">
        </script>
        <script src="{{ url_for('static', filename='js/instruments.js') }}">
        </script>
        <script src="{{ url_for('static', filename='js/drawFromSMILES.js') }}">
        </script>
    </body>
</html>
```

header-template.html:

```
<div class="wrap">
    <div class="ms-menu-bar-wrapper">
        <ul class="ms-menu">
            <li class="ms-menu-trigger"><i class="fa fa-bars"></i></li>
        </ul>
    </div>
</div>
```

index.html:

```
{% extends "base.html"%}

{% block title %}MolEditor{% endblock %}

{% block molpad %}
    <div class="molpad">
        <div class="toolbar">
            <div id="edit-tool">
                <div class="toolbar-inner">
                    <div id="single-bond" title="Single bond" class="tool-button">
                    </div>
                    <div id="double-bond" title="Double bond" class="tool-button">
                    </div>
```

```
                        <div id="triple-bond" title="Triple bond" class="tool-button">
                        </div>
                        <div id="clear" title="Clear all" class="tool-button">
                            <i class="material-icons md-36">delete</i>
                        </div>
                        <div id="erase" title="Erase" class="tool-button">
                            <i class="material-icons md-36">clear</i>
                        </div>
                        <div id="selected-item" title="selected item" class="tool-
button">
                            <i class="material-icons md-36">open_with</i>
                        </div>
                        <div id="undo" title="undo" class="tool-button">
                            <i class="material-icons md-36">undo</i>
                        </div>
                        <div id="redo" title="redo" class="tool-button">
                            <i class="material-icons md-36">redo</i>
                        </div>
                        <div class="empty-tool">
                        </div>
                        <div id="file-import" title="import .json" class="tool-
button">
                            <i class="material-icons md-36">file_upload</i>
                        </div>
                        <div id="file-export" title="export .json" class="tool-
button">
                            <i class="material-icons md-36">file_download</i>
                        </div>
                    </div>
                </div>
                <div class="canvas-content">
                    <canvas id="canvas" width="560" height="560"></canvas>
                </div>
                <div class="toolbar">
                    <div id="element-tools">
                        <div class="toolbar-inner">
                            <div id="atom-c" title="Углерод" class="element-tool tool-
button">
                                C
                            </div>
                            <div id="atom-h" title="Водород" class="element-tool tool-
button">
                                H
                            </div>
                            <div id="atom-n" title="Азот" class="element-tool tool-
button">
```

```
                            N
                        </div>
                        <div id="atom-o" title="Кислород" class="element-tool tool-
button">
                            O
                        </div>
                        <div id="atom-p" title="Фосфор" class="element-tool tool-
button">
                            P
                        </div>
                        <div id="atom-s" title="Сера" class="element-tool tool-
button">
                            S
                        </div>
                        <div id="atom-f" title="Фтор" class="element-tool tool-
button">
                            F
                        </div>
                        <div id="atom-cl" title="Хлор" class="element-tool tool-
button">
                            Cl
                        </div>
                        <div id="atom-br" title="Бром" class="element-tool tool-
button">
                            Br
                        </div>
                        <div id="atom-i" title="Йод" class="element-tool tool-button">
                            I
                        </div>
                        <div id="button-periodictable" title="Таблица Менделеева"
class="element-tool tool-button">
                            ...
                        </div>
                    </div>
                </div>
            </div>
            <div class="questionbar">
                <div id="task">
                    <h1>{{ title_task }}</h1>
                </div>
                <div id="check-answer" class="">
                    <i class="material-icons md-36">sync</i>
                </div>
                <div class="button-panel">
                    <button id="back-button" class="btn"
type="button"><span>Назад</span></button>
                    <button id="next-button" class="btn"
```

```
type="button"><span>Вперед</span></button>
                </div>
            </div>
        </div>
        {% include 'periodic-table.html' %}
        {% include 'upload-form.html' %}

    {% endblock %}
```

menu-template.html:

```
    <div class="ms-menu-body-wrapper hide-menu">
        <ul>
            <li><a href="/"><i class="material-icons">home</i>  Home</a></li>
            <li><a href="{{ url_for('SMILES_tutorial') }}"><i class="material-
icons">school</i>  SMILES tutorial</a></li>
            <li><a href="{{ url_for('about') }}"><i class="material-icons">help</i>
About</a></li>
        </ul>
    </div>
    <div class="modal-menu">

    </div>
```

periodic-table.html:

```
    <div id="modal-periodictable" class="modal">
        <div class="modal-content">
            <h2>Periodic Table</h2>
            <div id="periodictable">
                <div class="row">
                    <div class="pt-element" title="Водород">
                        <h3>1</h3>
                        <h4>H</h4>
                    </div>
                    <div class="pt-space"></div>
                    <div class="pt-space"></div>
                    <div class="pt-space"></div>
                    <div class="pt-space"></div>
                    <div class="pt-space"></div>
                    <div class="pt-space"></div>
                    <div class="pt-space"></div>
                    <div class="pt-space"></div>
                    <div class="pt-space"></div>
                    <div class="pt-space"></div>
                    <div class="pt-space"></div>
```

```
        <div class="pt-space"></div>
        <div class="pt-space"></div>
        <div class="pt-space"></div>
        <div class="pt-space"></div>
        <div class="pt-space"></div>
        <div class="pt-element" id="Гелий">
            <h3>2</h3>
            <h4>He</h4>
        </div>
    </div>
    <div class="row">
        <div class="pt-element" title="Литий">
            <h3>3</h3>
            <h4>Li</h4>
        </div>
        <div class="pt-element" title="Бериллий">
            <h3>4</h3>
            <h4>Be</h4>
        </div>
        <div class="pt-space"></div>
        <div class="pt-space"></div>
        <div class="pt-space"></div>
        <div class="pt-space"></div>
        <div class="pt-space"></div>
        <div class="pt-space"></div>
        <div class="pt-space"></div>
        <div class="pt-space"></div>
        <div class="pt-space"></div>
        <div class="pt-element" title="Бор">
            <h3>5</h3>
            <h4>B</h4>
        </div>
        <div class="pt-element" title="Углерод">
            <h3>5</h3>
            <h4>C</h4>
        </div>
        <div class="pt-element" title="Азот">
            <h3>7</h3>
            <h4>N</h4>
        </div>
        <div class="pt-element" title="Кислород">
            <h3>8</h3>
            <h4>O</h4>
        </div>
        <div class="pt-element" title="Фтор">
            <h3>9</h3>
```

```
                    <h4>F</h4>
                </div>
                <div class="pt-element" title="Неон">
                    <h3>10</h3>
                    <h4>Ne</h4>
                </div>
            </div>
        </div>
        <br/>
        <br/>
        <button class="close btn">close</button>
    </div>
</div>
```

upload-form.html:

```
<div id="modal-upload-form" class="modal">
    <div class="modal-content-form">
        <form>
            <input id="file" type="file" name="file"><br/><br/>
            <input id="submit" type="submit" value="Upload">
        </form>
        <br/>
        <br/>
        <button class="close btn">close</button>
    <div/>
</div>
```

DBcreation.py:

```
from app import db
db.create_all()

from app import Task
Water = Task('Постройте молекулу воды', '[O]')
SulfuricAcid = Task('Постройте молекулу серной кислоты', '[O][S]=[O]=[O][O]') #
TODO [O][S](=O)(=O)[O]

db.session.add(Water)
db.session.add(SulfuricAcid)

db.session.commit()
```

app.py:

```
#coding: utf-8

from flask import Flask
from flask import render_template, request, jsonify
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.debug = True
app.config['SQLALCHEMY_DATABASE_URI'] =
'postgresql://postgres:qwertyuiop@localhost/DiplomaDb'
db = SQLAlchemy(app)

from models import Task

tasks = Task.query.all()

@app.route('/')
def index():
    return render_template('index.html',title_task = tasks[0].title);

@app.route('/SMILES-tutorial')
def SMILES_tutorial():
    return render_template('smiles-tutorial.html');

@app.route('/about')
def about():
    return render_template('about.html');

index = 0

@app.route('/tasks/<task_id>')
def replace_task(task_id):
    if int(task_id) >=0 and int(task_id) <= len(tasks)-1:
        global index
        index = int(task_id)

    print(index)
    return jsonify(task = tasks[index].title)

@app.route('/checked/<result>')
def find_Elements(result):

    global index
    print(index)
    print (tasks[index].result+"  "+ result)
```

```
    if tasks[index].result == result:
        return jsonify(isFinded = "true")
    else:
        return jsonify(isFinded = "false")


if __name__ == '__main__':
    app.run()
```

manage.py:

```
import os
from flask.ext.script import Manager
from flask.ext.migrate import Migrate, MigrateCommand

from app import app, db

migrate = Migrate(app, db)
manager = Manager(app)

manager.add_command('db', MigrateCommand)

if __name__ == '__main__':
    manager.run()
```

models.py:

```
#coding: utf-8
from app import db

class Task(db.Model):
    __tablename__ = 'tasks'

    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String())
    result = db.Column(db.String())

    def __init__(self, title, result):
        self.title = title
        self.result = result

    def __repr__(self):
        return '<id {}>'.format(self.id)
```

ПРИЛОЖЕНИЕ В

UML-диаграммы и функциональная схема

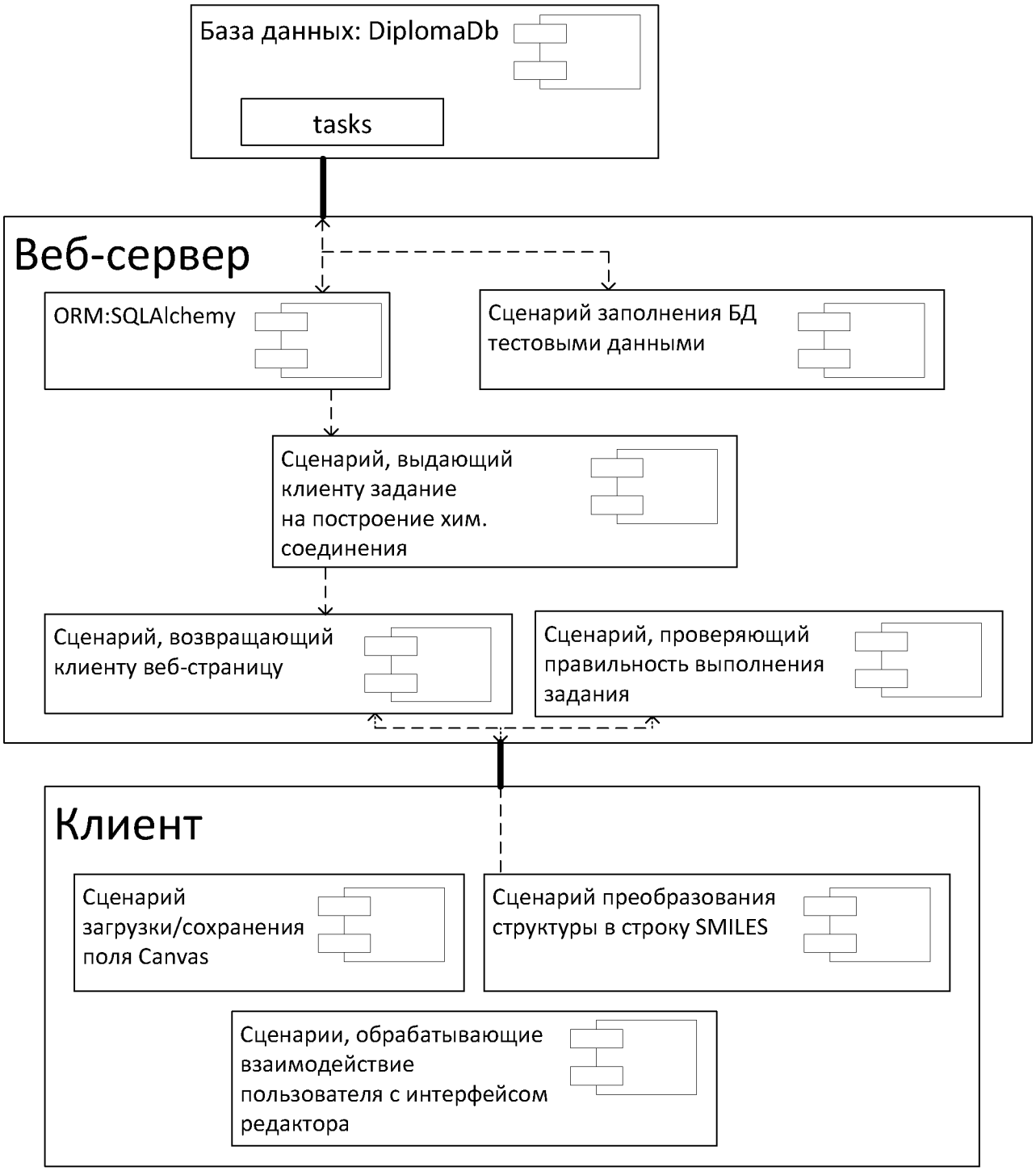

Рисунок В.1 – UML диаграмма компонентов

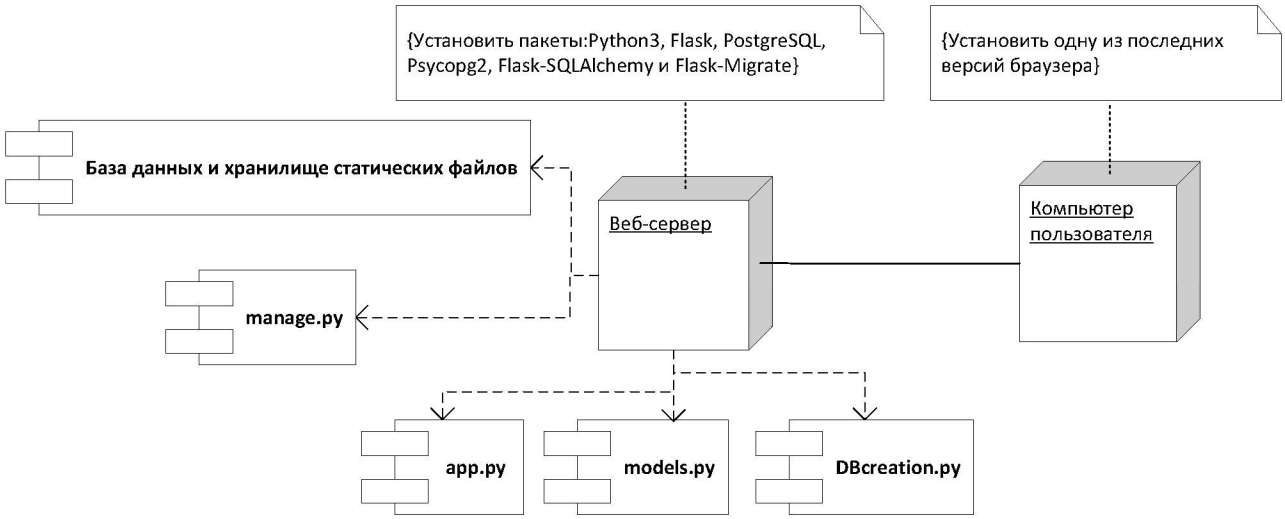

Рисунок В.2 – UML диаграмма развертывания

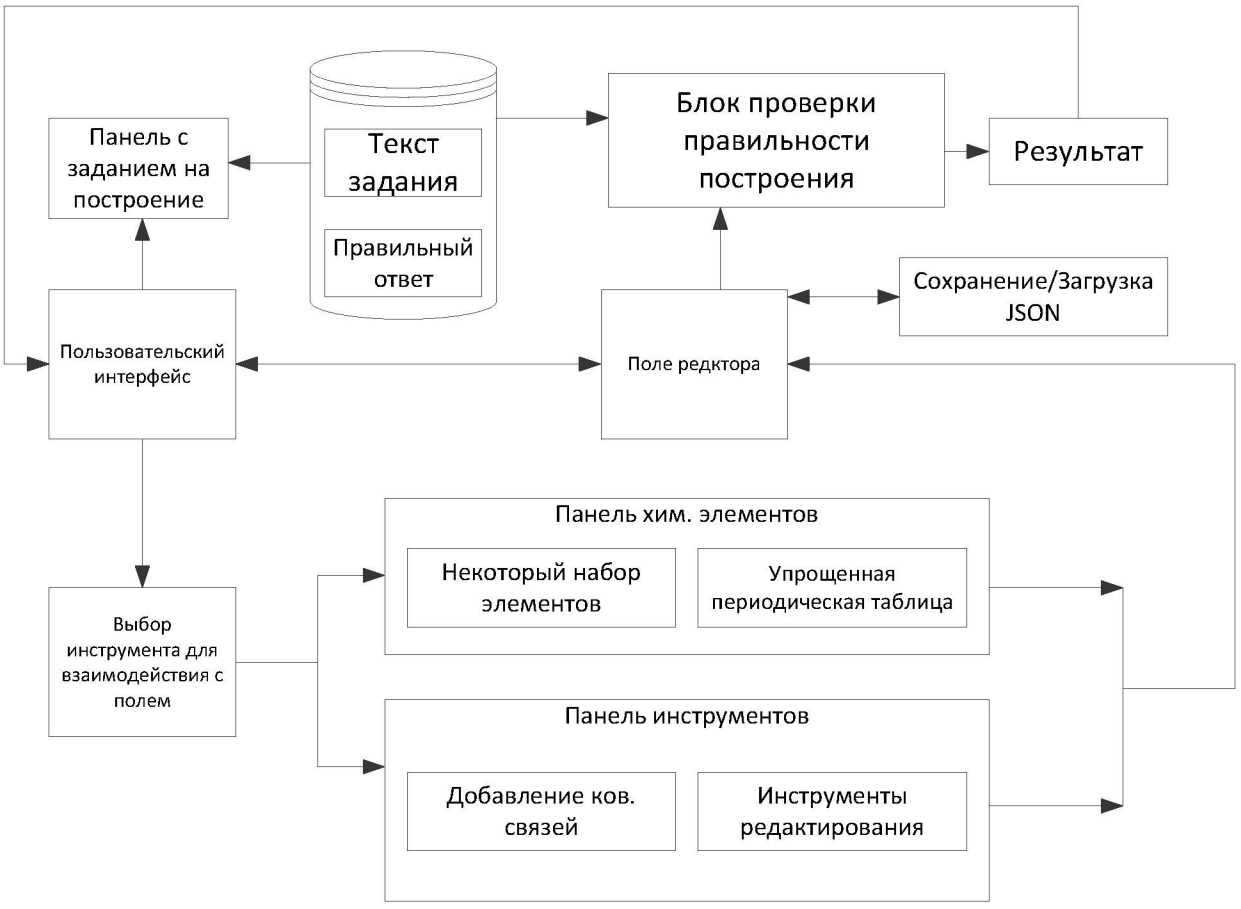

Рисунок В.3 – Функциональная схема редактора