МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
К.ф.-м.н., доцент кафедры
ВМИТ ЧелГУ
_____ А.Ю. Маковецкий
“___”_____ 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор
_____ Л.Б. Соколинский
“___”_____ 2020 г.

Разработка веб-сервиса для визуализации атомно-молекулярных структур

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2020.115-024.ВКР

Научный руководитель,
к.ф.-м.н., доцент кафедры СП
_____Т.Ю. Маковецкая

Автор работы,
студент группы КЭ-402
_____Д.Ю. Акулов

Ученый секретарь
(нормоконтролер)
_____И.Д. Володченко
“___”_____ 2020 г.

Челябинск-2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет (национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
09.02.2020

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-402
Акулову Денису Юрьевичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 24.04.2020 № 627)
Разработка веб-сервиса для визуализации атомно-молекулярных структур.
2. **Срок сдачи студентом законченной работы:** 05.06.2020.
3. **Исходные данные к работе**
Примеры файлов описания химических молекул
4. **Перечень подлежащих разработке вопросов**
4.1.Изучить структуру файла *.pdb
4.2.Рассмотреть программные структуры для сохранения данных
4.3.Изучить существующие инструменты визуализации химических соединений
4.4.Разработать веб-сервиса для визуализации атомно-молекулярных структур
5. **Дата выдачи задания:** 03.02.2020.

Научный руководитель,
к.ф.-м.н., доцент кафедры СП Т.Ю. Маковецкая

Задание принял к исполнению Д.Ю. Акулов

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

До последнего времени на свете не существовало прямых методов исследования внутренностей молекул. Поэтому точное местоположение атомов в большинстве молекул могло быть определено лишь косвенными методами или рассчитано теоретически. Такая нехватка экспериментальной информации о реальном расположении атомов служила препятствием в определении взаимосвязей между структурой молекулы и ее различными свойствами. Ранее исследователи уже пытались «влезть вглубь» молекул при помощи голографических методов, но все эти попытки увенчались лишь частичным успехом. Максимум чего удалось добиться ученым, это были изображения, на которых присутствовало до 10 атомов.

Одни из важнейших объектов, которыми оперирует химия, − атомы и молекулы − практически недоступны для прямого экспериментального наблюдения. В то же время, взаимное расположение атомов в веществе несет важнейшую информацию о его строении и возможных свойствах. В настоящее время для анализа строения вещества на атомно-молекулярном уровне активно применяется специализированное программное обеспечение [16].

Цель и задачи исследования

В данной работе рассматривается разработка веб-сервиса для визуализации атомно-молекулярных структур.

Для осуществления поставленной цели необходимо решить следующие задачи, перечисленные ниже.

1. Изучить структуру файла *.pdb.
2. Рассмотреть программные структуры для сохранения данных.
3. Изучить существующие инструменты визуализации химических соединений.
4. Разработать веб-сервис для визуализации атомно-молекулярных структур.

Структура и объем работы

Работа состоит из введения, четырех глав и заключения. Объем работы составляет 30 страниц, объем библиографии – 19 источников.

Содержание работы

В первой главе «Анализ предметной области» описана предметная область, заключающаяся в изучении структуры файла *.pdb, разборе структур хранения данных и обзоре инструментов визуализации.

Во второй главе «Реализация» описаны используемые технологии разработки и их использование.

В третьей главе «Тестирование» описаны виды тестирования и их результаты.

В заключении подводятся итоги работы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Большинство программ для визуализации химических соединений предоставляет широкие возможности для редактирования химических структур или создания их с нуля. С помощью ПО решаются следующие задачи:

- подготовка рисунков для публикаций;
- анализ параметров структуры (измерение расстояний между атомами, углов между связями, проверка наличия пустот и т.д.);
- подготовка входных данных для других программ, выполняющих анализ или моделирование (квантовохимическое, молекулярно-механическое, поиск по базам данных и т.д.);
- анализ результатов вычислений, выполненных другими программами.

1.1. Структура исходного файла

Данные о структурах молекулы хранятся в обычном текстовом файле, который называют «Молекулярным файлом». Такие файлы могут иметь расширения: *.pdb, *.p2n, *.mol2. Все перечисленные виды молекулярных файлов имеют одинаковую фиксированную структуру.

Молекулярные файлы содержат различные сведения: функции, название, дату публикации, трехбуквенный уникальный код, число цепей, источник, из которого выделена молекула, авторов публикации, ссылка на статью, где была опубликована структура, метод исследования, разрешение структуры и другие. Рассмотрим файл с расширением *.pdb.

Файл содержит несколько секций различной информации: «ATOM», «HETATM», «CONECT».

В секции «ATOM» перечислены данные об атомах, входящих в структуру:

ATOM 1 N GLY A 11 146.193 32.659 18.836

В секции «HETATM» указаны неполимерные или другие «нестандартные» химические координаты, такие как молекулы воды. Также могут предоставлять коэффициент занятости и температуры для каждого атома:

HETATM 5241 H2 HOH A1080   144.552 22.513 -13.700

В секции «CONECT» определяют связь между атомами, для которых предоставляются координаты. Связь описывается с использованием серийного номера атома:

CONECT   64   63   71   65   67

Считывая файл *.pdb, мы будем рассматривать только секцию «ATOM». Необходимая нам информация – наименование химического элемента с координатами в виде числа с плавающей точкой с 3 знаками после запятой (0.000).

В таблице 1 описан формат записи в секции «ATOM».

Табл. 1. «ATOM»

Данные	**Пример**
запись «ATOM»	ATOM
серийный номер атома	1
наименование химического элемента	N
имя аминокислоты	GLY
идентификатор цепи	A
номер аминокислоты	11
координаты (X Y Z)	146.193 32.659 18.836

Секция «ATOM» может содержать до десятков тысяч записей. В молекулярном файле записи каждой секции упорядочиваются по серийному номеру атома. Для выполнений сложных вычислений над элементами химического соединения могут потребоваться легко алгоритмизируемые структуры хранения данных.

1.2. Структуры хранения данных

Для вычислений различных больших объемов информации в пространстве могут потребоваться структуры хранения данных. Рассмотрим существующие структуры [9] разбиения пространства такие, как:

- диаграммы Вороного;
- KD-деревья;
- BSP-деревья;
- VP-деревья;
- R-деревья.

Диаграмма Вороного

Диаграмма Вороного обладает свойствами как векторных, так и растровых пространственных типов моделей: с её помощью может быть представлен каждый отдельный объект, а в целом массив ячеек вполне способен определить смежность не связанных объектов (каждая точка порождает одну ячейку, и эта ячейка имеет несколько соседей). Каждой ячейки Вороного может быть присвоен определенный набор атрибутов различного типа.

Диаграмма имеет следующие преимущества перед другими типами моделей.

1. Это адаптивный метод – размер ячеек зависит от густоты распределения точек данных.

2. Это автоматический метод, который не требует задания пользователем каких-то параметров для построения.

3. Исходные данные сохраняются в диаграмме, а не теряются бесследно, как это происходит при растровом представлении и в блочном моделировании.

4. При использовании двойственного графа диаграммы Вороного, тетраэдрализации Делоне, рендеринг оптимизирован, поскольку

треугольные элементы являются основными примитивами выбора в большинстве графических пакетов и видеокарт.

5. Имеется возможность локального обновления, не затрагивающего полностью всю диаграмму.

Известно, что диаграмма Вороного (VD) и тетраэдрализация Делоне (DT) [8] являются геометрически эквивалентными структурами. Имея одну из этих структур, вторая может быть построена по ней. В виду того, что легче построить по рассеянным точкам сеть тетраэдров, то в большинстве алгоритмов строится в начале сеть тетраэдров DT, а затем по ней строится диаграмма полиэдров VD. Таким образом, мы можем конструировать, сохранять и модифицировать некоторую VD работая только с её двойственным графом в виде DT. Извлечение VD из DT занимает время O(n), где n равно числу точек данных в множестве.

KD-деревья

KD-дерево [3] (K-мерное дерево), специальная «геометрическая» структура данных, которая позволяет разбить K-мерное пространство на «меньшие части», посредством сечения этого самого пространства гиперплоскостями (K > 3), плоскостями (K = 3), прямыми (K = 2) и в случае одномерного пространства − точкой.

Такое разбиение обычно используют для сужения диапазона поиска в K-мерном пространстве. Например, поиск ближнего объекта (вершины, сферы, треугольника и т.д.) к точке, проецирование точек на 3D сетку, трассировка лучей и т.п. При этом объекты пространства помещаются в специальные параллелепипеды, которые описывают исходное множество объектов или сам объект, если мы строим параллелепипеды лишь для него.

Прежде чем использовать KD-Дерево [7], нужно его построить. Все объекты помещаются в один большой параллелепипед, описывающий объекты исходного множества (при этом каждый объект ограничен своим параллелепипедом), который затем разбивается (делится) плоскостью, параллельной одной из его сторон, на два. В дерево добавляются два

новых узла. Таким же образом каждый из полученных параллелепипедов разбивается на два и т.д. Процесс завершается либо по специальному критерию, либо при достижении определенной глубины дерева, либо же при достижении определенного числа элементов внутри узла дерева. Некоторые элементы могут одновременно входить как в левый, так и в правый узлы (например, когда в качестве элементов дерева рассматриваются треугольники).

BSP-деревья

Множество программных продуктов активно использует графику. При этом очень важна скорость и точность вывода на экран различных объектов. Представим, что необходимо вывести на экран серию перекрывающих друг друга фигур. Тогда фигура, прорисованная последней, будет частично перекрывать некоторые из предыдущих. Эта задача весьма легко и элегантно решается с помощью BSP-дерева [1]. Другая задача, при решении которой BSP-дерево трудно заменить, это ускорение вывода на экран графической информации, что достигается путем радикального сокращения объема выводимых данных.

В дословном переводе Binary Space Partitioning Tree − это "двоичное дерево для разбиения пространства (разбивающее пространство)". BSP-дерево представляет собой структуру, показывающую рекурсивное, иерархическое деление n-мерного пространства на выпуклые подпространства с помощью гиперплоскостей.

BSP-дерево − это гибкий и мощный инструмент, который использует множество программ. Его применение позволяет резко сократить необходимый объем вычислений, следовательно, и затраты времени.

VP-деревья

VP-деревья, от слов Vantage point – опорная точка, основываются на выборе опорной точки и разбиении множества точек на 2 поддерева по принципу среднего расстояния от этой опорной точки до остальных точек множества. Каждый узел дерева имеет центральную точку и радиус, точки

находящиеся в котором принадлежат узлу. Работа с точками происходит только с помощью функции расстояния, алгоритм с отдельными координатами точек не имеет дела, поэтому размерность пространства и его замкнутость не имеют значения. Сложность алгоритма построения дерева – $O(n \bullet \log n)$.

Построение VP-дерева является рекурсивной процедурой. На входе имеется множество точек, для которого необходимо построить дерево. На первой итерации алгоритма все множество разбивается на два поддерева и для каждого поддерева повторяется аналогичная операция.

На каждой итерации построения поддерева необходимо выбрать опорную точку и вычислить среднее расстояние от неё до всех остальных точек. Входное множество точек разбивается на две части, относя точку в множество точек внутреннего (или левого) поддерева, если расстояние от неё до опорной точки меньше среднего, и в множество точек внешнего (или правого) в противном случае. Для каждого множества выполняется рекурсивная операция построения дерева до тех пор, пока размеры множеств не достигнут заданного предела – максимального размера листа.

R-деревья

R-дерево – (R-Tree) это индексная структура для доступа к пространственным данным, предложенная А. Гуттманом в 1984 году. R-дерево допускает произвольное выполнение операций добавления, удаления и поиска данных без периодической переиндексации. При этом дерево получается сбалансированным, что является одним из важных свойств любой иерархической структуры данных.

R-дерево – это сбалансированное по высоте дерево, листовые узлы которого содержат ссылки на конечные объекты. Если индексная структура находится на жестком диске, то каждый узел соответствует дисковой странице. Структура разработана так, чтобы для пространственного поиска требовалось посещение как можно меньшего числа узлов. Индексная структура полностью динамическая – добавление

и удаление может выполняться одновременно с поиском, и никакой периодической реорганизации структуры производить не нужно. Для организации такой индексной структуры используют пространственную базу данных, состоящую из набора записей, каждой из которых соответствует некоторый уникальный идентификатор. Этот идентификатор используют как средство ссылки на запись из индекса. В качестве идентификатора может выступать некоторое уникальное число или номер записи в файле (второй вариант предпочтительнее, так как работает быстрее, однако для него присущи некоторые недостатки, связанные с удалением записей из файла).

R-дерево должно удовлетворять следующим требованиям.

1. Корень, если он не является листом, содержит как минимум двух потомков.

2. Для каждой индексной записи листового узла n-мерного прямоугольника является минимальным прямоугольником, который полностью вмещает в себя пространственный объект, на который ссылается запись.

3. Для каждой индексной записи внутреннего узла дерева n-мерного прямоугольника является минимальным прямоугольником, охватывающим все n-мерные прямоугольники дочерних узлов.

4. Все листовые узлы дерева расположены на одном уровне (дерево является сбалансированным).

5. Каждый объект упоминается в дереве ровно один раз.

1.3. Инструменты визуализации химический соединений

Рассмотрим существующие аналоги среди программ для визуализации химических соединений.

Программа MolRed на рисунке 1 – молекулярный редактор, представляющий собой программу с графическим интерфейсом, который предоставляет пользователю возможности визуального восприятия 3-х

мерных изображений молекул и молекулярных структур, редактирования молекул, создания новых молекулярных соединений. Программа работает с файлами, описывающими молекулярные соединения (*.hin,*.mrk,*.pdb), производит чтение и сохранение молекулярных структур в соответствии со стандартами этих файлов. Программа работает под управлением операционной системы WinXP, в дальнейшем планируется распространение и на Linux системы. Для отображения 3D сцен используется аппаратное ускорение OpenGL.

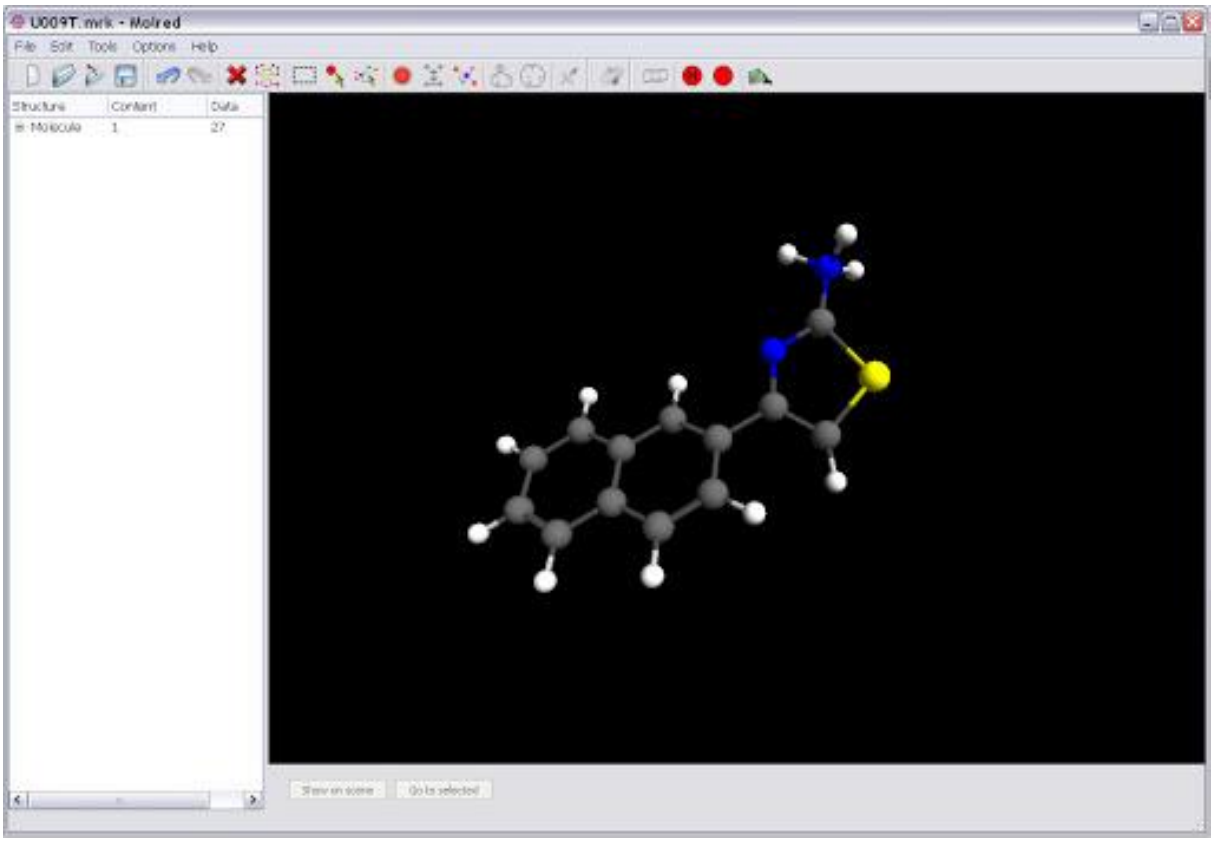

Рис. 1. MolRed

ChemCraft на рисунке 2 – квантово-химический визуализатор, который понимает множество форматов химических данных, выводы большинства популярных программ; имеет большое количество отображаемых свойств. Множество стилей отображения молекул позволяют получить изображения нужного качества. ChemCraft позволяет удобно редактировать геометрию молекул и готовить входные файла. Очень полезной является возможность экспорта анимированных gif-ов.

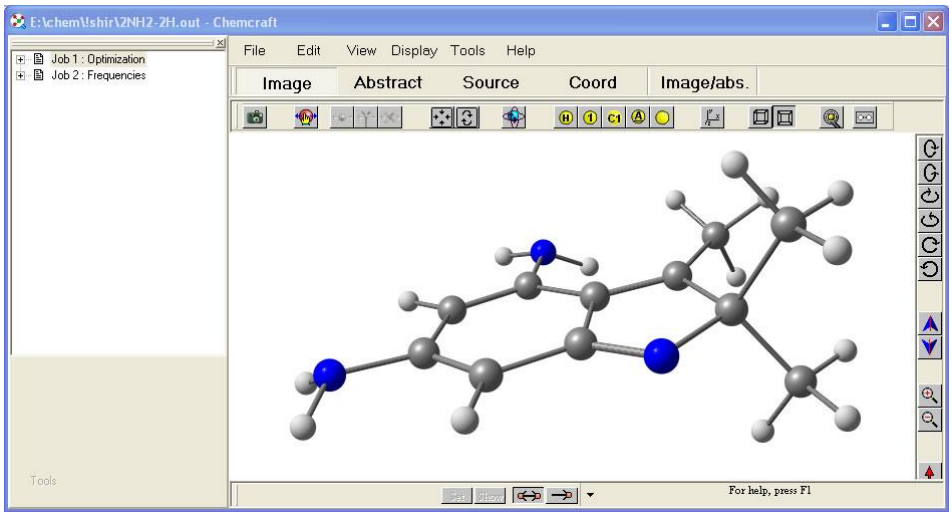

Рис. 2. ChemCraft

Rasmol на рисунке 3– Программа функционирует в режиме двух окон − графического и текстового. В графическом окне происходит визуализация структур макромолекул, в текстовое окно вводится управляющая информация, которая позволяет изменять масштаб рисунка, цвет, представление молекул, выделять группы атомов, остатков, белковых цепей. Управляющие данные вводятся в виде текстовых команд, описание которых приведено на странице помощи. Популярность этой программы и доступность ее кодов привела к тому, что на ее основе разработаны новые более функциональные средства визуализации структур.

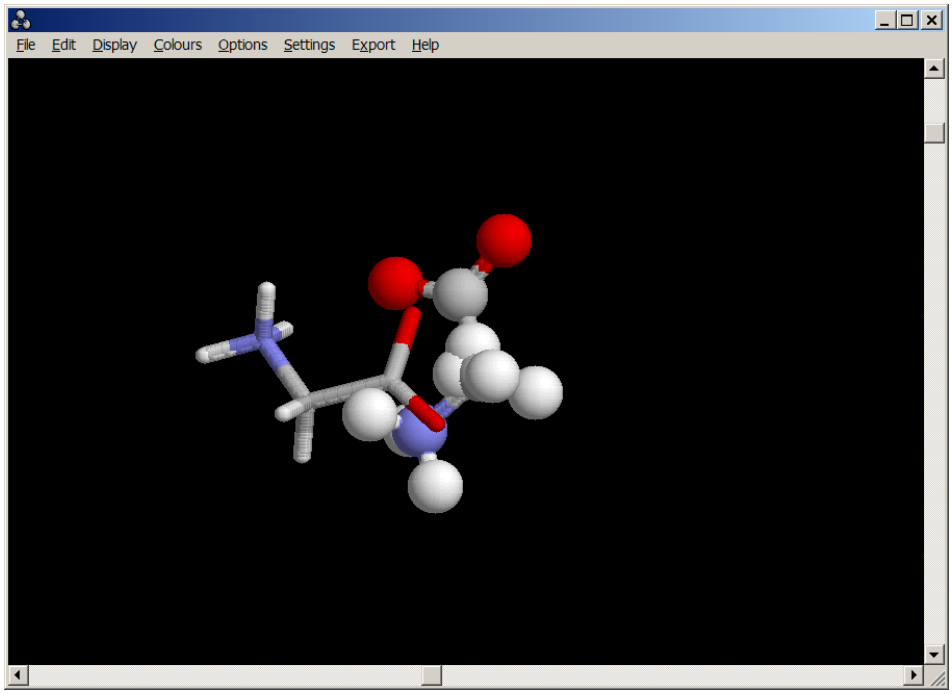

Рис. 3. Rasmol

ViewerLite на рисунке 4 – программа имеет полноценный Windows интерфейс, управление режимом отображения осуществляется полностью на основе оконного интерфейса. Данная программа имеет более совершенную графику при рисовке макромолекул. Одной из удобных особенностей является наличие окна, которое отображает структуру макромолекул в виде списков атомов, остатков, цепей.

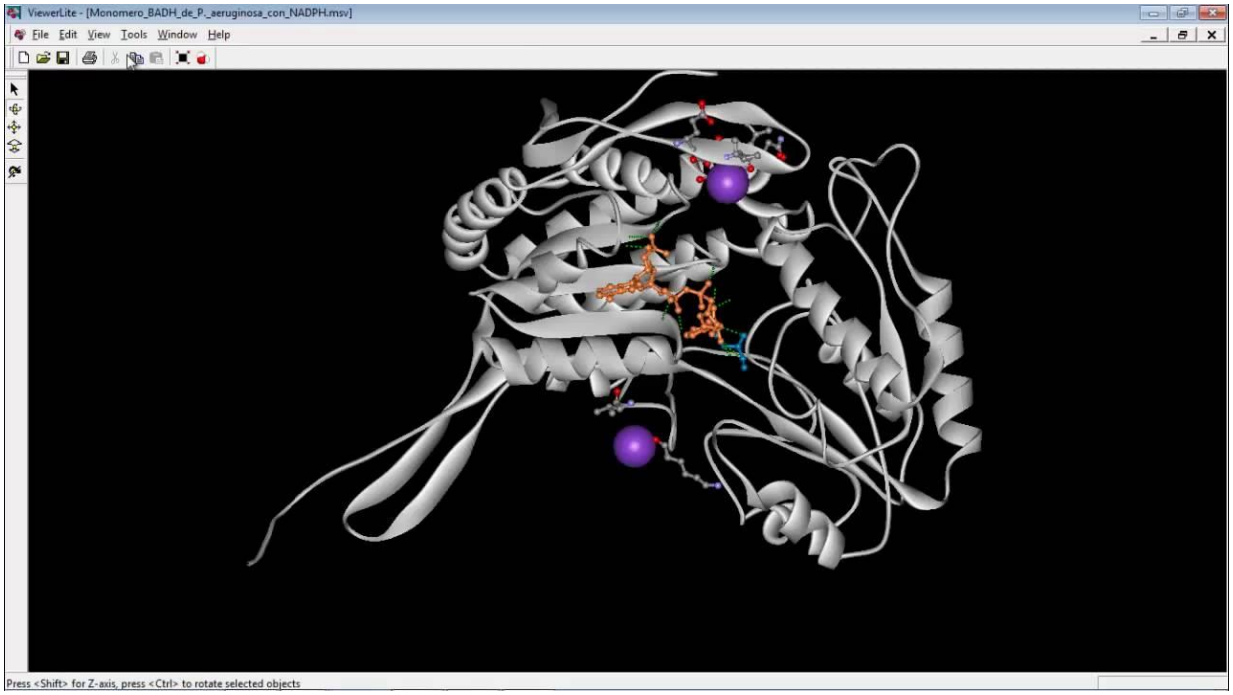

Рис. 4. ViewerLite

XtalDraw на рисунке 5 − программа для визуализации кристаллических и молекулярных структур в виде шаростержневых моделей, многогранников и объемных моделей. Программа содержит большой набор файлов данных, поддерживает не только файлы данных XtalDraw, но и американские минералогические базы данных (American Mineralogist Crystal Structure Database). XtalDraw с большой достоверностью показывает, что большинство неорганических веществ имеет немолекулярное строение.

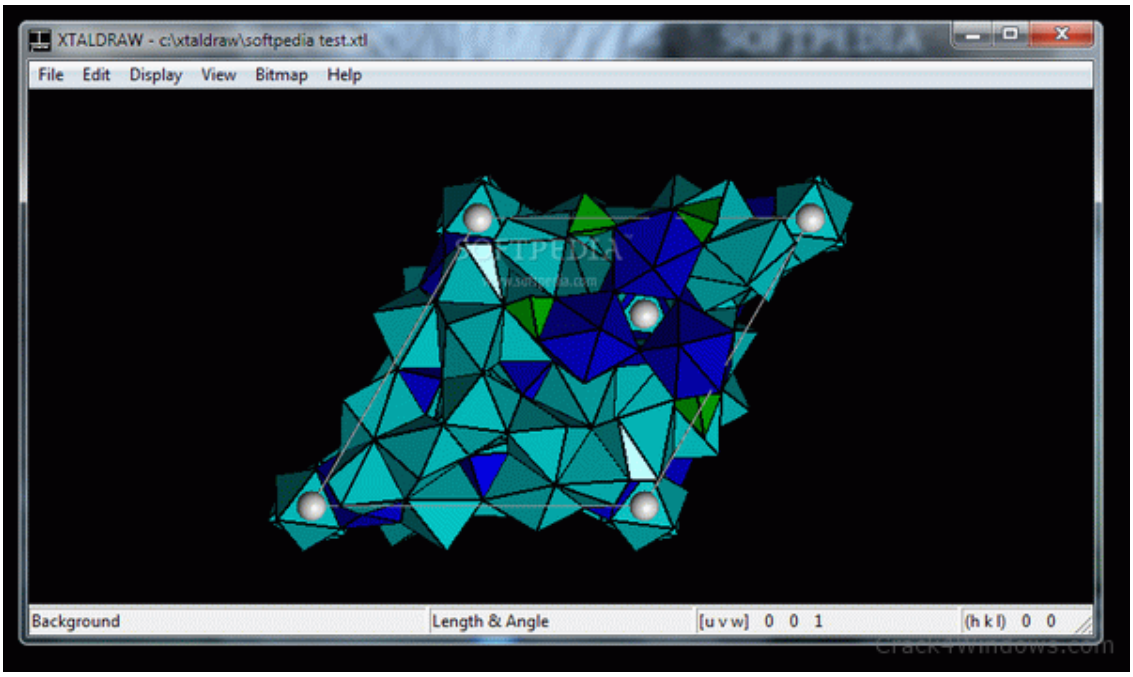

Рис. 5. XtalDraw

Минусом ПО − необходимость установки его на новое устройство или перенос готового к работе устройства. Существуют также веб-инструменты визуализации химических соединений. Их минусом является отсутствие возможности загружать исходный файл с описанием соединения.

1.4. Постановка задач

В разрабатываемом веб-сервисе для визуализации атомно-молекулярных структур необходимо реализовать следующие функции:

1. загрузка исходного молекулярного файла;
2. визуализация каждого атома из загруженного молекулярного файла;
3. вращения молекулы во всех направлениях;
4. перемещение экрана в любую сторону;
5. изменение дистанции молекулы от экрана;
6. отображение таблицы визуализированных атомов.

2. РЕАЛИЗАЦИЯ

Для реализации веб-сервиса использовались C#, javascript и CSS. Язык C# использовался для парсинга исходного файла. На языке javascript, используя библиотеку Three.js технологии WebGL, была реализована визуализация атомов на веб-сервисе. CSS использован для написания и применения стилей на веб-странице. Опишем используемые технологии разработки.

2.1. Используемые технологии разработки

C# − объектно-ориентированный язык программирования. Разработан в 1998−2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. на нём можно делать всё, что нужно в 2020-м: от игр и приложений до веб-сервисов.

Все Web-страницы пишутся на языке HTML (HyperText Markup Language – язык гипертекстовой разметки). HTML описывает представление Web-страницы на экране, которое программа Web-обозревателя должна предоставить пользователю. В большинстве случаев, это представление статично. Чтобы добавить в Web-страницу динамические элементы, необходимо описать, как сделать их динамичными. То есть, нужен еще один язык программирования. Одним из таких языков стал Javascript [11].

Язык Javascript [2] был разработан Бренданом Айком из корпорации Netscape Communications Corporation в 1995 году. Его первая разработка была весьма скороспелой, и критики Javascript по большей части порицали недостатки предварительного планирования во время его разработки. Однако у Брендан Айк был серьезный опыт в информатике, и он заложил в Javascript сложные и передовые идеи. Так или иначе, он опередил время, и

потребовалось 15 лет, чтобы этот замечательный язык завоевал популярность у ведущих разработчиков.

Javascript позволяет придать веб-страницам дополнительную интерактивность [10], так как предоставляет доступ к их контенту и возможность изменять его, а также разметку на веб-странице, когда она открыта в браузере. Язык позволяет выделить на HTML-странице любой элемент, атрибут или текст. Можно использовать для добавления на страницу (или для удаления с нее) элементов, атрибутов и текста. Позволяет указать последовательность операций, которые должен выполнить браузер (это похоже на кулинарный рецепт). Данная последовательность обеспечивает доступ к контенту страницы либо дает возможность изменять его. Дает возможность создать сценарий, запускающийся после конкретного события. Вот несколько примеров таких событий. В JavaScript действуют многие общеизвестные правила программирования [19]. Этот язык помогает сделать страницу интерактивной, реагирующей на действия пользователя.

CSS (Cascading Style Sheets) − это код, который вы используете для стилизации вашей веб-страницы. CSS − формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц.

WebGL (Web-based Graphics Library) – программная библиотека, предназначенная для создания интерактивной трехмерной графики в веб-браузерах с использованием ресурсов графического процессора.

Благодаря технологии WebGL можно обрабатывать и показывать реальные 3D модели непосредственно на страницах сайта, как можно увидеть на рисунке 6. При помощи мыши можно поворачивать, перемещать и масштабировать модель, чтобы рассмотреть все ракурсы и интересующие детали. Не нужно устанавливать дополнительных

приложений, достаточно открыть в браузере нужную ссылку и любоваться результатом.

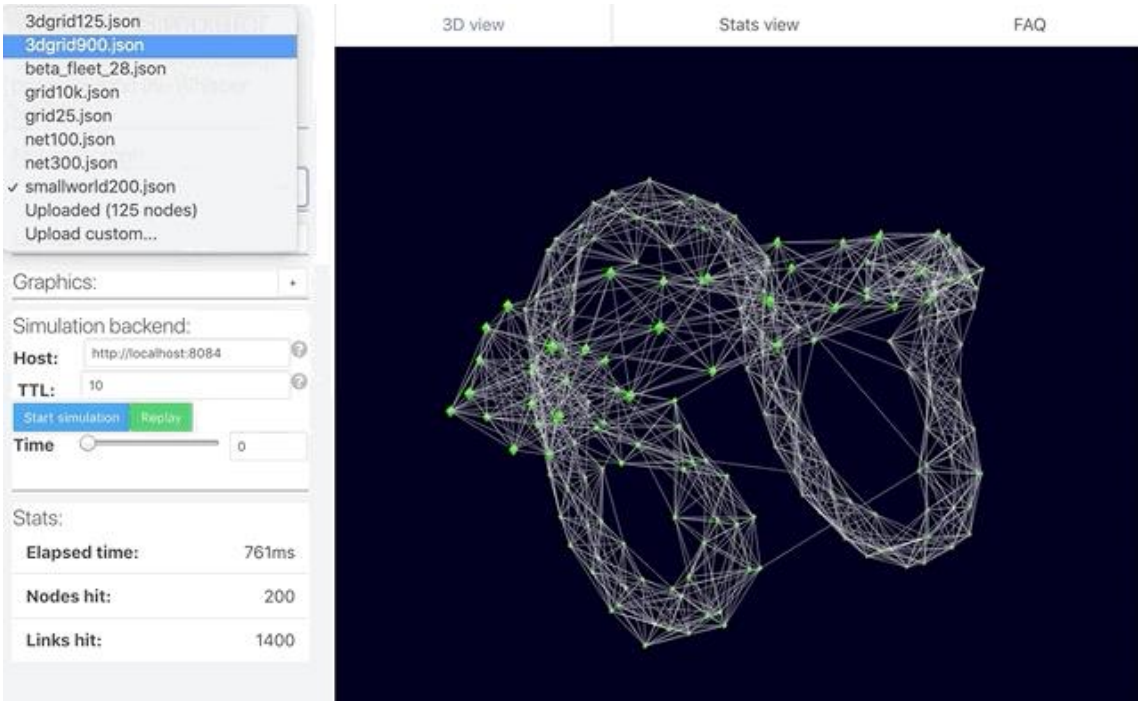

Рис. 6. 3D-приложение на основе WebGL

Для упрощения разработки WebGL-приложений существуют различные библиотеки. Первой общедоступной стала библиотека WebGLU. Среди других библиотек для WebGL − Babylon.js, Three.js.

Babylon.js − библиотека JavaScript, используемая для отображения 2D и 3D-графики в браузере без использования каких-либо сторонних плагинов и дополнений.

Three.js [5] − библиотека Javascript, используемая для создания и отображения анимированной компьютерной 3D графики. Three.js это мощный инструмент. Он помогает использовать 3D дизайн в браузере с приемлемой производительностью.

В нашей работе будем использовать библиотеку Three.js.

Моделирование графики с использованием Three.js можно сравнить со съемочной площадкой, так как есть возможность оперировать такими понятиями как сцена, свет, камера, объекты и их материалы.

2.1. Общий алгоритм

В ходе выполнения работы, была разработана диаграмма деятельности для веб-сервиса, которая изображена на рисунке 7.

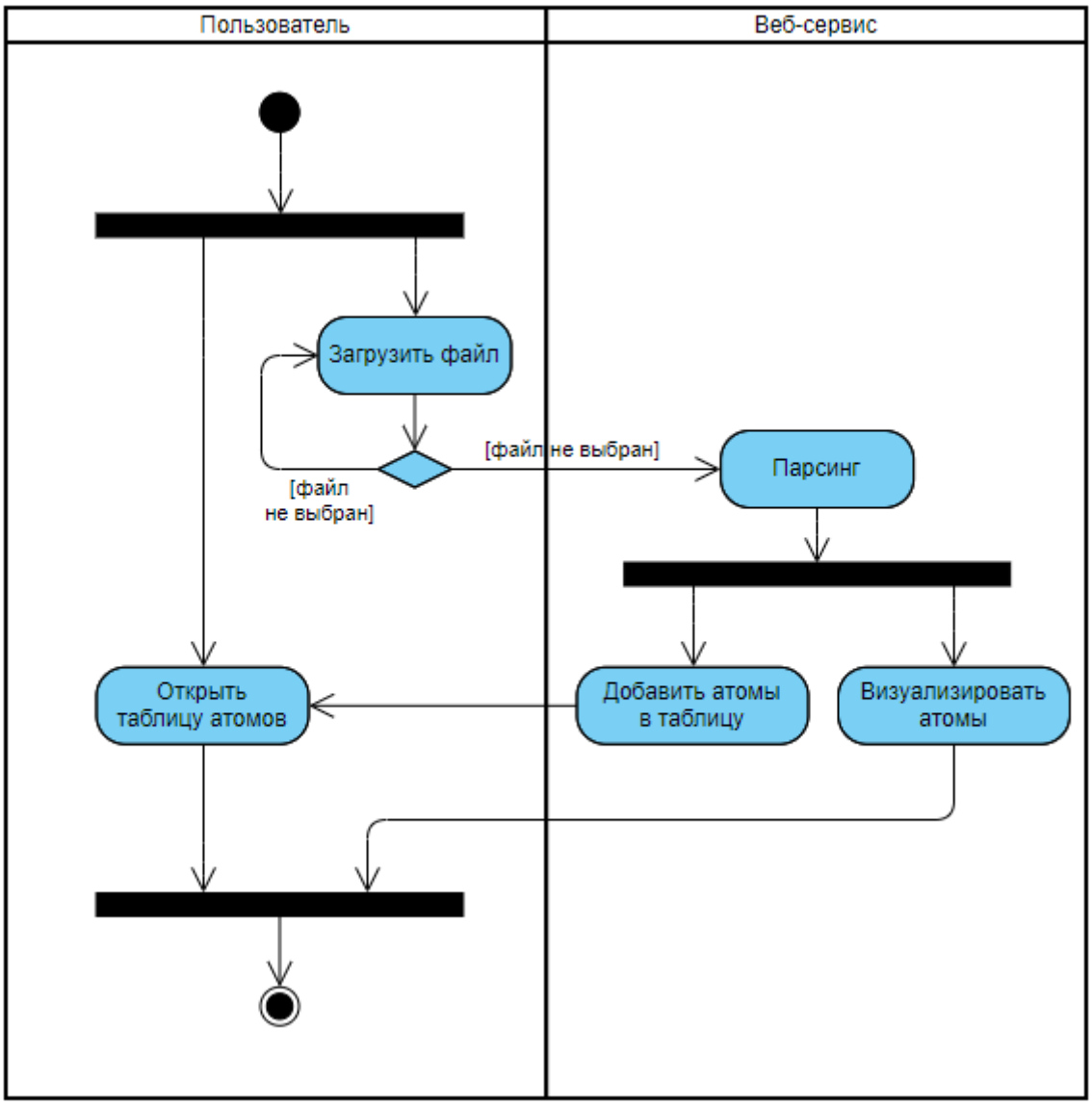

Рис. 7. Диаграмма деятельности.

2.2. Обработка исходного файла

К парсингу прибегают, когда предстоит обработать большой массив информации, с которым сложно справиться вручную. Программа, которая производит сбор и синтаксический анализ, – это парсер

Существуют этапы парсинга, которые указаны ниже.

- Поиск данных – программу-парсер разбивает весь исходный текст на лексемы, выделяя необходимую информацию.

- Извлечение информации – поиск данных происходит благодаря определенному набору знаков, описывающих цель поиска. Этот набор также называется регулярными выражениями. Они позволяют выделить из всего массива только интересующие фрагменты.

- Сохранение данных. После получения информация сохраняется в виде таблиц или вносится в базу данных.

Исходный молекулярный файл разбивается на поля, как показано на листинге 1.

Листинг 1. Atom_list

```
  List<Atom> atom_list = new List<Atom>();
  Stream stream = file;
  using (var streamReader = new StreamReader(stream,
System.Text.Encoding.UTF8, true, 256))
  {
    String line;
    String[] line_world = { };
    streamReader.ReadLine();
    while ((line = streamReader.ReadLine()) != null )
    {
      line_world = line.Split(new char[] { ' ' },
StringSplitOptions.RemoveEmptyEntries);
      if(line_world[0]=="ATOM")
      atom_list.Add(new Atom(line_world[1], line_world[2], line_world[3],
line_world[4], line_world[5], line_world[6], line_world[7],
line_world[8]));
      else
      {
        break;
      }
    }
  }
```

Файл проходит обработку, где каждое поле атома получает свое значение, на листинге 2.

Листинг 2. Atom

```
  Atom(string init_index, string init_name, string init_node, string
init_node2,
       string init_node_index, string init_x, string init_y, string init_z)
  {
     name = init_name;
    node = init_node + " " + init_node2;

    index = Int32.Parse(init_index);
```

```
    node_index = Int32.Parse(init_node_index);

    x = double.Parse(init_x, CultureInfo.InvariantCulture);
    y = double.Parse(init_y, CultureInfo.InvariantCulture);
    z = double.Parse(init_z, CultureInfo.InvariantCulture);  }
```

Каждая запись из молекулярного файла может пройти обработку, чтобы получить информацию об атомах. Необходимо создать функцию «добавления атома» для визуализации атома на экране.

2.3. Создание 3D графики

Three.js содержит объекты:

- Scene − платформа, где размещаются все объекты, которые мы создаем;
- Camera – «глаз», который будет направлен на сцену. Камера снимает и отображает объекты, которые расположены на сцене;
- Renderer − визуализатор, который позволяет показывать сцену, снятую камерой.
- Light − Без освещения на сцене, будет складываться впечатление, что вы находитесь в темной комнате. Помимо этого, с помощью освещения сцене можно придать большую реалистичность.
- Geometry − описывает форму (положения вершин, грани, радиус и т.д).
- Material − описывает внешний вид объектов (цвет, текстура, прозрачность и т.д.).

На листинге 3 для визуализации химических соединений, используемая библиотека three.js, позволяет создать необходимые объекты: визуализатор, камера, сцена.

Листинг 3. Объекты.

```
    const stage = document.querySelector("#stage"),
        renderer = new THREE.WebGLRenderer({canvas:stage, alpha:true}),
        camera = new THREE.PerspectiveCamera(camera_param.fov,
camera_param.aspect, camera_param.near, camera_param.far),
        scene = new THREE.Scene(),
        controls = new OrbitControls( camera, stage ),
```

```
        hemisphereLight = new THREE.HemisphereLight(0xfbc2eb,0xa6c1ee, .7),
        shadowLight = new THREE.DirectionalLight(0xaaaaaa, .7);
```

Для разрабатываемого веб-сервиса необходимо перемещение камеры по сцене, как на листинге 4.

Листинг 4. Камера

```
    document.addEventListener("mousedown", (event)=>
    {
        document.addEventListener("mousemove", mousemove);
        document.addEventListener("mouseup", mouseup);
    });

    stage.addEventListener("wheel",()=>{camer_pos();});

    function mousemove(event)
    {
        camer_pos();
    }

    function mouseup(event)
    {
        document.removeEventListener('mousemove', mousemove);
        document.removeEventListener('mouseup', mouseup);
```

На листинге 5 создается освещение, чтобы при перемещении камеры отсутствовали тени.

Листинг 5. Освещение

```
        shadowLight.position.set(150, 350, 350);

        shadowLight.castShadow = true;

        shadowLight.shadow.camera.left = -400;
        shadowLight.shadow.camera.right = 400;
        shadowLight.shadow.camera.top = 400;
        shadowLight.shadow.camera.bottom = -400;
        shadowLight.shadow.camera.near = 1;
        shadowLight.shadow.camera.far = 5000;

        shadowLight.shadow.mapSize.width = 1024;
        shadowLight.shadow.mapSize.height = 1024;

        scene.add(hemisphereLight);
        scene.add(shadowLight);
```

На листинге 6 описано создание размеров и материала сферы – атома.

Листинг 6. Сфера

```
    let geometry = new THREE.SphereGeometry(0.6, 7, 7);
    let material = new THREE.MeshLambertMaterial ( {color:colors.blue});
    let sphere = new THREE.Mesh( geometry, material );
```

На листинге 7 написана функция добавление атомов на сцену, из подготовленной информации об атомах молекулярного.

Листинг 7. Добавить объект

```
        fetch('/atoms', {
            method: 'POST',
            headers: {"Content-Type": "text/plain"},
            body: file
        }).then(success => {
            console.log(success);
            for ( let i = 0, length = success.length; i <= length; i++ ) {
                let object = new THREE.Mesh(geometry, new
THREE.MeshLambertMaterial({color: Math.random() * 0xffffff}));

                object.position.x = success[i].x;
                object.position.y = success[i].y;
                object.position.z = success[i].z;

                scene.add(object);
            }

        }
```

В результате разработан веб-сервис, с помощью которого можно визуализировать химические соединения, описанные в исходном файле. Веб-сервис можно развивать дальше, добавляя больше информации из молекулярного файла. Также на веб-сервисе можно добавлять различные функции для удобства просмотра молекулы:

- Фильтр по названию химического элемента − в таблице выбирать определенные атомы, которые будут отображаться, остальные будут скрыты.
- Связи между атомами – в молекулярном файле также существует информация о номерах атомов в молекуле, которые имеют связь между собой.

- Вычисления – на веб-сервисе реализовать функции расчетов (например расстояние между опеределенными атомами), необходимые для пользователей.

3. ТЕСТИРОВАНИЕ

Тестирование является процессом поиска ошибок в реализации программы [14]. Нужно организовать тестирование таким образом, чтобы выявить все возможные ошибки и дефекты в программе. Но чтобы утверждать, что программа полностью свободна от дефектов необходимо подготовить все возможные наборы входных данных, выполнить программу на всех возможных перестановках и вариациях входных данных, проанализировать все выходные данные и установить, что каждый тестовый выходной набор соответствует правильному. Устранить все возможные ошибки, с которыми может столкнуться пользователь, просто невозможно. Тестируя, мы снижаем риск того, что пользователь с ними столкнется, или снижаем серьезность последствий такого столкновения.

Для тестирования разработанного веб-сервиса были выбраны 2 вида тестирования:

- функциональное – для проверки корректности функций: вращение молекулы, движении экрана, приблизить−отдалить, отобразить таблицу молекулы, загрузить файл;
- конфигурационное – для проверки работы веб-сервиса в различных браузерах и платформах.

3.1. Функциональное тестирование

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, т.е. способности программного обеспечения в определенных условиях решать задачи, нужные пользователям. При функциональном тестировании имитируется фактическое использование системы, что позволяет обнаружить ошибки до того, как их обнаружит пользователь.

Основные этапы функционального тестирования:

1. Подготовка – проводится анализ исходных документов о системе: функциональные и бизнес-требования, техническое задание, паспорт

проекта. Также происходят разработка плана тестирования, тест-кейсов, числа итераций, оценка возможных рисков.

2. Проведение – функциональное тестирование ведется вручную по подготовленным заранее тестовым сценариям с занесением всех найденных ошибок в багтрекинговую систему.

3. Отчет – происходит разработка и согласование отчетов о проведенном тестировании со списком обнаруженных отклонений.

В таблице 2 показаны набор тестов для функционального тестирования и его результаты.

Табл. 2. Функциональное тестирование

Объект	**Функция**	**Результат**
камера	перемещение камеры относительно точки	+
	движение экрана в стороны	+
	указание координат для камеры	- (в координатах не ставится точка)
кнопки	загрузка файла	+
	открытие таблицы	+

С помощью функционального тестирования проверена корректность работы функций сайта. В результате тестирование обнаружена ошибка в задании координат для камеры.

3.2. Конфигурационное тестирование

Конфигурационное тестирование [13] − специальный вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т.д.).

Исходя из определенного вида проекта, тестирование предусматривает выполнение одной из двух целей:

1) определение наиболее эффективной конфигурации оборудования, обеспечивающей нужные параметры производительности;

2) проверка объекта на совместимость с указанным в спецификации оборудованием и другими программными.

Веб-сервис тестируется с позиции рабочего окружения конечного пользователя.

Было проведено кроссплатформенное тестирование на Windows 7/8/11. Также веб-сервис был протестирован в различных браузерах, в таких как: Opera, Mozilla Firefox, Google Chrome, Microsoft Edge.

При конфигурационном тестировании веб-сервиса ошибок не обнаружено.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы были выполнены следующие задачи.

1. Изучена структуру файла *.pdb.

2. Рассмотрены программные структуры для сохранения данных.

3. Изучены существующие инструменты визуализации химических соединений.

4. Разработан веб-сервис для визуализации атомно-молекулярных структур.

ЛИТЕРАТУРА

1. BSP-деревья. URL: https://www.kv.by/archive/index1997301201.htm (дата обращения 15.04.2020).

2. JavaScript [Электронный ресурс] URL: https://developer.mozilla.org/en-US/docs/Learn (дата обращения 17.05.2020).

3. KD-деревья и R-деревья. URL: https://fat-crocodile.livejournal.com/156564.html (дата обращения 15.04.2020).

4. Protein Data Bank. URL: https://www.rcsb.org/ (дата обращения 28.04.2020).

5. Three.js [Электронный ресурс] URL: https://threejs.org/ (дата обращения 28.04.2020).

6. Анатомия KD-Деревьев. URL: https://itnan.ru/post.php?c=1&p=312882 (дата обращения 15.04.2020).

7. Байнев В.В. Применение kd-деревьев для оптимизации трассировки лучей в оптической системе. 2019.

8. Васильев П.В., Ледоукс Х. Применение 3D триангуляции Делоне и диаграммы Вороного в ГИС недропользования.

9. Гулаков В.К., Трубаков А.О. Многомерные структуры данных. 2010.

10. Джон Дакетт. Javascript и jQuery. Интерактивная веб-разработка. 2017.

11. Информационные технологии. Разработка web-сайта на основе html с использованием javascript [Электронный ресурс] URL: http://elib.belstu.by/bitstream/123456789/3313/1/informacionnye-texnologii_zhilyak.pdf (дата обращения 17.05.2020).

12. Константин Азарский. Тестирование. Легкий старт. – 2014.

13. Конфигурационное тестирование [Электронный ресурс] URL: http://www.protesting.ru/testing/types/configuration.html (дата обращения 24.05.2020).

14. Лихицкий, А. С. Исследование стратегий тестирования программного обеспечения. 2016.

15. Международная база данных структуры белков (Protein Data Base, PDB) URL:http://fbm.msu.ru/education/lectures/biophys/pdf/Владимиров_PDB%20RasTop.pdf (дата обращения 28.04.2020).

16. Пронкин П.Г., Сорокина О.Н., Ботнарь Ю.В. Эволюция средств и методов информатики в практической и фундаментальной химии. 2009.

17. Сэм Канер, Джек Фолк, Енг Кек Нгуен. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. 2001.

18. Ученые впервые получили трехмерные голографические изображения сложных молекул. URL: https://enginclub.ru/uchenye-vpervye-poluchili-trekhmernye-g/ (дата обращения 12.03.2020).

19. Этан Браун. Изучаем JavaScript. Руководство по созданию современных веб-сайтов. 2017.