МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий
Кафедра параллельных вычислений

Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль): Программная инженерия и компьютерные науки

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Юракова Никиты Дмитриевича

Тема работы:

РАЗРАБОТКА ФОРМАТА И ПОДСИСТЕМЫ ИНТЕРАКТИВНОГО ПРЕДСТАВЛЕНИЯ БАЗ АКТИВНЫХ ЗНАНИЙ В СИСТЕМЕ LUNA

«К защите допущена»
Заведующий кафедрой,
д.т.н., профессор
Малышкин В. Э. /……..
(ФИО) / (подпись)
«31» мая 2025г.

Руководитель ВКР
к.т.н.,
доц. каф. ПВ ФИТ НГУ
Перепёлкин В. А. /……...
(ФИО) / (подпись)
«……»……2025г.

Соруководитель ВКР
ст. преп. каф. ПВ ФИТ НГУ
Матвеев А. С. /……..
(ФИ О) / (подпись)
«……»……2025г.

Новосибирск, 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)
Факультет информационных технологий
Кафедра параллельных вычислений
Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль): Программная инженерия и компьютерные науки

УТВЕРЖДАЮ

Зав. кафедрой Малышкин В. Э.
(фамилия, И., О.)
…….
(подпись)
«17» января 2025г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

Студент*у(ке)* Юракову Никите Дмитриевичу, группы 21210
(фамилия, имя, отчество, номер группы)

Тема Разработка формата и подсистемы интерактивного представления баз активных знаний в системе LuNA
(полное название темы выпускной квалификационной работы)

утверждена распоряжением проректора по учебной работе от 21.10.2024 №0377, скорректирована распоряжением проректора по учебной работе от 17.01.2025 №0020

Срок сдачи студентом готовой работы 20 мая 2025 г.

Исходные данные (или цель работы):

разработка формата и подсистемы интерактивного представления баз активных знаний в системе LuNA

Структурные части работы:

обзор литературы, постановка задачи, формулирование требований к системе, разработка и реализация системы, тестирование

Руководитель ВКР
доц. каф. ПВ ФИТ НГУ,
к.т.н.
Перепёлкин В. А. /……..
(ФИ О) / (подпись)
«17» января 2025г.

Задание принял к исполнению
Юраков Н. Д. /……..
(ФИО студента) / (подпись)
«17» января 2025г.

Соруководитель ВКР
ст. преп. каф. ПВ ФИТ НГУ,
Матвеев А. С. /……..
(ФИ О) / (подпись)
«17» января 2025г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время происходит стремительный рост знаний, обусловленный постоянным развитием общества. Данный рост провоцирует увеличение вариативности применения методов решения конкретных задач и способов их реализации в сфере вычислений. Однако этот рост сопровождается сложностями в систематизации информации. Постоянное увеличение объемов знаний требует новых подходов к их сбору, структурированию и представлению, чтобы обеспечить простоту и эффективность их использования.

При этом существует проблема различных форматов представления информации. Знания, представленные в человеко-ориентированной форме, интуитивно понятны пользователям, но трудно воспринимаются и обрабатываются компьютерами. В то же время есть машинно-ориентированные форматы, с которыми отлично работают вычислительные системы, но они сложны для восприятия человеком. Эта разница между форматами создаёт множество трудностей при взаимодействии человека и компьютера.

Одним из современных решений проблемы систематизации знаний является использование баз активных знаний [1]. Они представляют собой машинно-ориентированное хранилище информации, например, формул, функций, констант. Такие базы используются для написания и конструирования программ путем выявления подходящего модуля и его применения через систему программирования, например LuNA [2]. Базы активных знаний решают проблему организации большого количества информации, однако они всё ещё остаются в машинно-ориентированном формате, что делает их трудными для восприятия человеком.

На данный момент наблюдается проблема в отсутствии средств отображения, которые делали бы работу с машинно-ориентированными данными, такими как базы активных знаний, удобной, понятной и интерактивной. Применение даже самых продвинутых баз активных знаний затруднено, если пользовательских интерфейс взаимодействия с ними сложен и непонятен. Это приводит к снижению их полезности и ограничивает

возможности их применения. Возникает необходимость создания интерфейса, который будет представлять базы активных знаний в удобной для человека форме, при этом сохраняя их машинно-ориентированный формат. Актуальность работы заключается в том, что подобный интерфейс должен обеспечить наглядное и интерактивное отображение базы, сохранить возможность работы с ней для системы LuNA и предоставить простой и удобный способ взаимодействия для пользователей.

Очевидно, что интерактивная система, отображающая базу активных знаний и осуществляющая взаимодействие с системой LuNA, облегчит использование и ускорит процесс разработки новых программ.

Цель работы – разработка формата и подсистемы интерактивного представления баз активных знаний в системе LuNA

Для достижения этой цели были поставлены следующие задачи:

- Провести анализ существующих решений по отображению машинно-ориентированной информации.
- Разработать собственный способ представления базы активных знаний для дальнейшего отображения.
- Разработать веб-интерфейс для отображения представления базы активных знаний и осуществления взаимодействия с пользователем.
- Провести тестирование разработанного способа представления баз активных знаний и веб-интерфейса на нескольких практически значимых предметных областях.

Практическая ценность работы состоит в разработке средств и интерактивной системы отображения баз активных знаний для системы LuNA, которая обеспечит простоту и удобство пользования, а также для демонстрации возможностей данной системы с целью популяризации.

Научная новизна данной работы состоит в создании новой модели описания визуального представления базы активных знаний для отображения с возможностью взаимодействия.

В соответствии с намеченной целью и поставленными задачами определены следующие методы исследования.

1. Анализ и сравнение существующих решений по отображению машинно-ориентированной информации.
2. Синтез, абстрагирование базы активных знаний при разработке собственного способа представления.
3. Синтез веб-интерфейса с отображением представления базы активных знаний.

Структура и объем работы. Работа состоит из введения, 3 глав и заключения. В первой главе дается определение предметной области, делается анализ существующих подходов к визуализации и организации отображения данных. Во второй главе даётся описание предлагаемых решений: собственного способа представления баз активных знаний и концепции визуального отображения на основе разработанного способа представления. Третья глава посвящена освещению деталей реализации и тестирования. В заключении сформулированы основные выводы работы.

1 Анализ предметной области

При рассмотрении подходов к визуализации и организации отображения машинно-ориентированной информации важными аспектами являются наличие дополнительного описания, необходимого для визуализации, выбор формата его хранения и метода отображения. Для задачи создания интерактивного представления баз активных знаний ключевым моментом является способ расположения элементов, который также необходимо учесть.

В ходе дальнейшего анализа существующие подходы и решения будут рассматриваться на основе следующих критериев:

1. Используется ли дополнительное описание для визуализации? Если да, то в каком формате хранится данная информация?
2. Каким способом располагаются и отображаются элементы машинно-ориентированных данных?

Данный анализ поможет сформировать требования к разрабатываемым решениям и учесть существующий опыт в области визуализации машинно-ориентированных данных, что позволит сформировать лучший подход для создания формата и подсистемы интерактивного представления баз активных знаний в системе LuNA.

1.1 Обзор подходов к визуализации и организации отображения данных

1.1.1 Графовое представление данных

В области визуализации машинно-ориентированных данных графы играют важную роль, поскольку многие типы данных можно представить в виде графов. Существует множество форматов и инструментов для работы с такими данными, и у каждого есть свои особенности. Рассмотрим несколько форматов для описания графов и соответствующие инструменты для их визуализации.

Одним из наиболее популярных форматов для представления графов является GraphML [3, 4]. Это формат, основанный на XML, который позволяет описывать как ориентированные, так и неориентированные графы. В GraphML можно хранить атрибуты, которые могут быть добавлены к узлам и рёберам,

например, для хранения весов рёбер или меток узлов. Кроме того, можно указать цвет, форму или размер узла, стиль рёбер, а также другие визуальные параметры, которые могут помочь отобразить граф более наглядно. Важно, что сами координаты для размещения элементов GraphML не задаёт напрямую. При этом, несмотря на свою гибкость в описании, GraphML может быть менее удобным для ручного редактирования, чем текстовые форматы, такие как DOT [3, 5].

DOT — это текстовый формат, который в основном используется с инструментом Graphviz для описания графов. Он отличается своей простотой в описании и понимании, а также лёгкостью добавления атрибутов. В DOT-файлах также можно добавлять стили и атрибуты, которые влияют на внешний вид графа при его визуализации. Эти атрибуты позволяют описывать цвет, форму, размер, название вершины и другие визуальные характеристики. Обычно в формате DOT у элементов не указываются абсолютные координаты, но за счёт использования атрибутов можно задать необходимое расположение.

Другой формат для представления графов, GML (Graph Modelling Language) [3, 6], был разработан с целью предоставления простого и понятного способа обмена графовыми данными между различными приложениями, а также для хранения атрибутов графа. В отличие от других рассмотренных форматов, GML представляет графы в виде иерархических списков ключ-значение, где каждый элемент (вершина, ребро) может иметь дополнительные атрибуты. Для задания расположения элементов графа в формате GML используется ключ graphics, которая описывает графические атрибуты, координаты и размеры объектов, используемые для их отображения. Через graphics.x и graphics.y указываются координаты центра объекта в 2D пространстве, а graphics.z добавляет координаты для 3D пространства. Кроме того, можно задать размеры объекта через graphics.w — ширину, graphics.h — высоту, а также graphics.d — глубину для 3D объектов. Эти атрибуты позволяют точно настроить положение и размеры элементов графа.

Для визуального представления графов широкое распространение получил инструмент с открытым исходным кодом Graphviz [7, 8], разработанный в AT&T

Labs. Здесь визуализация строится на основе текстового файла в формате DOT, где описываются узлы, ребра и их свойства. В большинстве случаев Graphviz сам определяет расположение элементов на основе выбранного алгоритма расстановки, не требуя задания координат вручную, но при необходимости можно явно указать координаты узлов с помощью атрибута pos. Проект также включает в себя инструменты командной строки (gml2gv и gv2gml), которые позволяют преобразовывать графы формата GML в формат DOT и обратно.

В отличие от Graphviz, uDraw [9] позволяет пользователю работать с графом через графический интерфейс, что делает процесс более наглядным и удобным. uDraw больше ориентирована на динамическое редактирование и визуализацию графа, используя собственные формат файлов *.udg*, который содержит информацию о структуре графа и визуальные атрибуты. Для сохранения информации о конкретных координатах узлов и рёбер, а также текущих настройках пользовательского интерфейса, используется файл состояния *.status*. Эти файлы могут быть написаны пользователем или созданы другими приложениями. При автоматическом расположении элементов программа использует алгоритмы для их размещения. Однако, если пользователь вручную перемещает узлы, их новые координаты будут сохранены в файл состояния, и при следующем открытии программа будет восстанавливать граф с его текущим расположением, не используя алгоритмы расстановки.

Из современных решений можно выделить yEd [10] — мощный редактор от компании yWorks, предназначенный для создания и визуализации различных типов графов и диаграмм. Он предоставляет понятный и функциональный интерфейс с возможностью автоматического и ручного расположения элементов. Кроме того, yEd поддерживает импорт и экспорт данных в различных форматах: GraphML, GML, Excel, JSON и др.

Также существуют такие инструменты, как VisualGraph [11] и VCG [12], которые поддерживают форматы GraphML, GML и др. Оба инструмента используют алгоритмы автоматической расстановки для размещения элементов графа. В отличие от VCG, предназначенного исключительно для визуализации,

VisualGraph предоставляет возможность взаимодействия с графом и его редактирования.

Данное решение не подходит, поскольку отображение информации в виде графов огрианиченно в своей наглядности. Не любую информацию можно наглядно и понятно представить в форме графа, в некоторых случаях такая визуализация может быть трудной для восприятия. Этот подход может не всегда подходить для отображения более сложных баз активных знаний. Однако способ задания конкретных координат для элементов является универсальным и может быть использован для описания и визуального отображения баз активных знаний.

1.1.2 Представление баз данных

Для удобной работы с базами данных, информация в которых хранится в машинно-ориентированном формате, необходимы системы, которые позволяют не только просматривать, но и редактировать данные в наглядном формате. Чаще всего подобные системы поддерживают два основных способа отображения данных. Первый — в виде графа или диаграммы, где показываются связи между таблицами, данный формат уже был рассмотрен в предыдущем разделе. Второй способ — в виде таблицы с реальными значениями, такой формат необходимо рассмотреть с точки зрения расположения элементов на экране и работы с ними. Важно отметить, что в случае данного способа представления информации дополнительное описание для визуализации не используется.

DBeaver [13, 14] — это универсальная система с открытым исходным кодом, поддерживающая работу с множеством СУБД. Здесь данные отображаются в основном в виде таблицы, то есть машинно-ориентированная информация представляется в ячейках таблицы, которые знакомы и понятны пользователю, что позволяет работать с актуальными значениями, получая наглядное представление о содержимом баз данных. Это делает инструмент удобным для анализа и работы с данными без необходимости погружаться в диаграммы или графа.

Ещё один инструмент для работы с базами данных, DataGrip [15] от JetBrains, поддерживает более 20 СУБД и предлагает мощные функции для написания SQL-запросов. В DataGrip основным форматом работы с данными также являются таблицы с реальными значениями, которые дают разработчикам баз данных удобный способ анализа и работы с данными.

Кроме того, другие системы для работы с базами данных, такие как DbVisualizer [16], Navicat [17], также используют формат представления информации в ячейках таблицы, что делает отображение данных более наглядными и структурированным и упрощает процесс работы с базами.

Данное решение не подходит для нашей задачи, поскольку способ представления информации в формате таблицы ограничивает гибкость системы. Такой подход недостаточно универсален, что позволяет описать и визуализировать только ограниченный набор баз активных знаний.

1.1.3 HTML и CSS

HTML [18] (HyperText Markup Language) и CSS [19] (Cascading Style Sheets) — это основные технологии для разработки веб-страниц, они позволяют задавать структуру и визуальное представление материала страницы. HTML отвечает за структуру страницы, за её содержание, а CSS используется для изменения внешнего вида и расположения объектов на этой странице. При работе вместе они образуют систему, способную отображать информацию в совершенно разных образах и форматах.

В своей основе HTML является языком разметки, который используется для определения и организации содержания страницы в виде различных элементов: заголовков, кнопок, списков, изображений и других. Также к элементам HTML можно добавить атрибуты, которые могут влиять на их поведение или внешний вид. Однако сам по себе HTML не имеет большого набора возможностей для описания сложных визуальных данных.

При использовании HTML браузер будет размещать элементы на экране в том порядке, в котором они указаны в HTML-разметке. То есть по умолчанию элементы располагаются в потоке документа [20] один за другим, сверху вниз.

Существуют блочные элементы, такие как , они занимают всю доступную ширину и располагаются вертикально, при этом строчные элементы, например , выстраиваются в одну строку друг за другом [21]. Также в HTML есть вложенность, которая позволяет помещать один элемент внутрь другого, создавая тем самым контейнеры, которые могут влиять на расположение вложенных объектов. Например, если элемент содержит несколько <p>, то их размещение и внешний вид будет зависеть от характеристик родительского контейнера.

В отличие от HTML, CSS ориентирован на управление визуальными аспектами данных. Он позволяет задавать стили для каждого элемента страницы, включаешь шрифты, цвета, отступы, размеры и многое другое. Кроме того, он предоставляет возможности для позиционирования элементов на странице, что особенно важно в случае отображения данных.

CSS обладает различными методами для изменения стандартного расположения элементов. Одним из таких методов является свойство position [22], которое помогает задавать точное расположение элементов на странице. С помощью значений relative, absolute, fixed и sticky можно управлять позиционированием элементов относительно их исходного положения. Например, при использовании значения absolute элемент выводится из нормального потока документа и позиционируется относительно его ближайшего родителя. Вместе с этим для более точного указания позиции элемента используются свойства top, right, bottom, left, которые указывают размер отступа. Также существуют механизмы Flexbox и Grid Layout для более гибкого позиционирования. Flexbox позволяет создавать адаптивные макеты, где элементы могут растягиваться, сжиматься и выравниваться относительно друг друга, что удобно для разных размеров экранов. Grid Layout же представляет собой двумерную систему для расположения элементов в строках и столбцах, позволяя создавать сложные сетки, полезные для визуализации данных и отображения таблиц, схожий метод был рассмотрен в предыдущем разделе.

Таким образом, при совместной работе HTML и CSS образуют систему визуального отображения машинно-ориентированных данных, которая даёт мощные инструменты для создания веб-страниц различных форматов. В этой системе данные отделены от описания визаульного представления, HTML служит основой, которая хранит в себе ключевую информацию, на которой всё построено, а CSS хранит информацию, необходимую для визуального представления и позиционирования элементов из HTML.

Данное решение не подходит для нашей задачи, поскольку HTML и CSS не поддерживают такие концепции, как база активных знаний, вычислительная модель, её операции и переменные. Внедрение этих элементов в текущую структуру системы оказалось бы слишком трудозатратным, что делает эту систему не подходящей для решения данной задачи. Однако вариант разграничения основной информации и визуального представления может быть полезен и будет учтён при решении поставленной задачи.

1.1.4 Model-View-Controller

Одним из наиболее устойчивых и широко применяемых архитектурных шаблонов при построении пользовательских интерфейсов является MVC (Model-View-Controller) [23, 24]. Его основная идея заключается в разделении логики приложения на три взаимосвязанных компонента: модель (Model), представление (View) и контроллер (Controller). Такой подход повышает модульность, повторность использования компонентов, даёт возможность замены одного модуля на другой и упрощает поддержку программного кода.

Модель (Model) представляет собой центральный компонент архитектуры, отвечающий за управление данными и реализацию бизнес-логики приложения. Она определяет поведение системы, выступая в роли абстракции, через которую осуществляется доступ к данным и их модификации. Модель не зависит от способов визуального отображения информации и, как правило, не содержит сведений о пользовательском интерфейсе, что делает её независимым компонентом.

Представление (View) отвечает за отображение информации, содержащейся в модели. Оно формирует визуальный интерфейс, представляя данные и способы взаимодействия пользователя с системой в необходимом для этого визуальном формате. Представление может быть как статическим, так и динамическим, обновляясь автоматически при изменении состояния модели. При этом оно не занимается логикой обработки данных, а служит лишь средством их визуального отображения. Поэтому модуль представления может быть заменён на другой модуль представления с подходящими параметрами для системы и с отличным визуальным представлением.

Контроллер (Controller) выполняет функцию посредника между пользователем и системой. Он принимает ввод от пользователя (например, действия с интерфейсом), интерпретирует его и преобразует в команды, направленные на изменение состояния модели или обновление представления. Контроллер также может заниматься логикой навигации, маршрутизации и управления потоком взаимодействия в приложении.

В архитектуре MVC вся информация, связанная с визуализацией, такая как структура интерфейса, способы отображения данных, стили, относятся исключительно к модулю View. Этот компонент отвечает за формирование визуального представления и делает это независимо от других частей системы. Благодаря такому разграничению, View не содержит логики обработки данных и не зависит напрямую от модели или контроллера, что позволяет легко изменить или заменить визуальную часть без изменения других модулей. Поскольку размещение элементов на экране также является частью визуального представления, то этим занимается модуль View. При этом архитектура MVC не накладывает жёстких требований на реализацию представления, что позволяет использовать как абсолютное позиционирование с указанием координат, так и любые другие методы размещения элементов. Конкретные способы расположения объектов определяются задачами проекта и используемыми технологиями.

Реализация разделённой архитектуры, как MVC, потребует значительных трудозатрат, связанных с разработкой и поддержанием трёх независимых компонентов. Это может существенно усложнить реализацию даже относительно простых интерфейсных решений. В то же время базовая идея архитектуры MVC — отделение визуального представления от логики и данных, а также возможность гибкой замены интерфейса без изменения остальной системы — представляется полезной и будет учтена при решении задач.

1.2 Вывод

Все рассмотренные решения не обладают необходимой функциональностью для визуального представления баз активных знаний или являются слишком сложными для реализации. Тем не менее некоторые подходы, использованные в анализируемых работах, могут оказаться полезными и будут учтены. В связи с этим возникает необходимость создания собственного формата и подсистемы визуального интерактивного представления баз активных знаний, которые смогут обеспечить их наглядное и удобное отображение.

2 Разработка визуального представления баз активных знаний

2.1 Необходимые определения

Для дальнейшего понимания задачи, необходимо ввести следующие определения:

LuNA (Language for Numerical Algorithms) [2] — это система активных знаний, предназначенная для параллельного выполнения численных алгоритмов, в которой программа представляется как набор переменных и операций с зависимостями между ними. Такая модель позволяет эффективно управлять выполнением, автоматически распределять ресурсы и динамически оптимизировать порядок операций, что упрощает разработку и обеспечивает высокую производительность.

База активных знаний [1] — это машинно-ориентированное хранилище формализованных прикладных модулей, вычислительных моделей, данных и расширений, предназначенных для автоматического решения прикладных задач. Модули в базе снабжены спецификациями, описывающими их функциональные и нефункциональные свойства, что позволяет системе активных знаний автоматически подбирать, комбинировать и применять их для построения оптимальных решений без участия пользователя.

Вычислительная модель [25] — это двудольный ориентированный граф, где вершины в одной доле соответствуют операциям, а в другой — переменным. Дуга графа, соединяющая операцию с переменной, определяет переменную как результат этой операции, а противоположно направленная дуга определяет переменную как входной аргумент. Переменные в модели соответствуют каким-либо величинам описываемой предметной области, а операциям на этапе исполнения сопоставляется некоторый программный модуль. Входным и выходным параметрам модуля при этом ставятся в соответствие переменные вычислительной модели (см. пример на рис. 1).

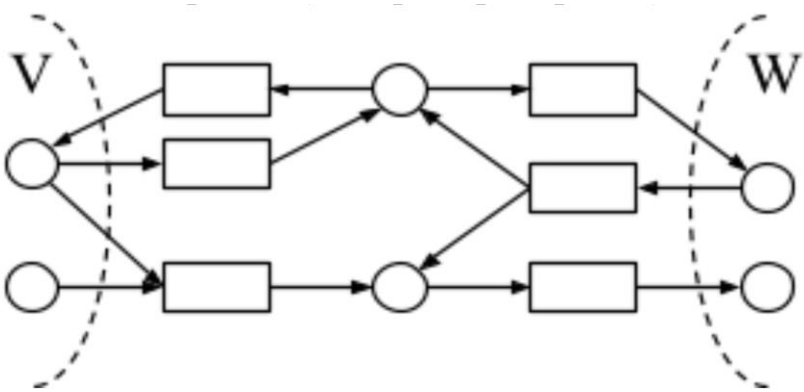

Рисунок 1 — Пример вычислительной модели. Круги — переменные, прямоугольники — операции

JSON [26] — это текстовое представление структурированных данных, основанное на парах «ключ-значение» и упорядоченных списках. JSON-формат широко применяется для передачи информации между веб-клиентом и веб-сервером, в том числе в мобильных приложениях и веб-сайтах. Хотя JSON является производным от JavaScript, он поддерживается либо изначально, либо через библиотеки на большинстве основных языков программирования.

JAZ — это специализированный формат представления данных, основанный на стандарте JSON, разработанный в рамках системы LuNA. В структуре JAZ могут храниться вычислительные модели, фрагменты кода, константы и другие дополнительные модули.

2.2 Постановка задачи

В рамках системы LuNA используются базы активных знаний, которые включают в себя вычислительные модели. Эти модели представляются в виде графов, где узлы и ребра отражают основные компоненты модели и их взаимосвязи. Такое представление позволяет специалистам, знакомым с предметной областью, легко ориентироваться в структуре модели. Тем не менее графовое представление вычислительной модели имеет свои ограничения в наглядности и всё ещё представляет собой формат, который может быть сложен для восприятия, особенно для пользователей, не имеющих опыта работы с вычислительными моделями или базами активных знаний.

Для того чтобы сделать вычислительные модели более понятными и доступными для пользователей, требуется разработать модель описания её визуального представления, которое заменит переменные и операции на понятные наглядные образы. Эти образы должны отражать ключевые компоненты вычислительной модели и помогать в понимании логики взаимодействия элементов, они должны быть понятны и привычны людям, разбирающимся в предметной области данной вычислительной модели.

Такое представление должно быть не статичной иллюстрацией, а основой для интерактивного взаимодействия. Под интерактивностью в данном случае понимается наличие связи между интерфейсом и вычислительной моделью с визуальным представлением. Модель визуального представления должна позволить интерфейсу не только отображать текущую структуру модели, но и предоставлять возможность реализации механизмом взаимодействия с ней — например, выбор переменных и операций, изменение выбранных значений.

В данной работе необходимо разработать модель визуального представления и формат описания, который реализует эту модель. Также требуется создать концепцию визуального отображения модели и интерфейс, который реализует данную концепцию. Таким образом, под разработкой формата и интерфейса интерактивного представления баз активных знаний подразумевается разработка модели и формата описания визуального представления, концепции отображения, а также интерфейса.

2.3 Требования, предъявляемые к решению

1. Модель описания визуального представления должна давать достаточное описание для визуального представления баз активных знаний. Под достаточностью подразумевается способность модели охватить несколько практически значимых предметных областей.
2. Концепция визуального отображения должна поддерживать совместимость как для веб-приложений, так и для приложений, предназначенных для компьютеров и мобильных устройств.

3. Интерфейс должен иметь достаточную функциональность для возможности работы с практически значимыми предметными областями.
4. Интерфейс должен быть адаптивен для отображения и функционирования на устройствах с узким экраном, включая мобильные устройства, с сохранением той же функциональности, что и на компьютерах.

2.4 Описание предлагаемых решений

2.4.1 Концепция визуального отображения

Концепция визуального отображения представляет собой идею, определяющую, какие элементы системы и каким образом должны быть представлены пользователю в визуальной форме. Она описывает функциональность интерфейса, а также предназначение содержащихся компонентов.

В данной концепции предполагается визуальная секция, основанная на двумерной плоскости. В этой области переменные отображаются в графической форме в виде образов, что позволит пользователю интуитивно воспринимать структуру задачи и взаимодействовать с её компонентами.

Для повышения наглядности и удобства работы предусмотрено отображение фона плоскости, например, изображений, отражающих предметную область или схему процесса. Также предполагается наличие системы навигации, позволяющей перемещаться по различным частям рабочей области.

Концепция предполагает, что каждая отображаемая переменная может быть выбрана пользователем для задания значений или просмотра уже установленных. Важно также предусмотреть возможность обозначения роли переменной — входной или выходной, поскольку это имеет принципиальное значение при составлении задачи.

Для возможности пользователя видеть текущее состояние переменных предусмотрена отдельная область, предназначенная для отображения и изменения их значений. Она позволяет пользователю в любой момент контролировать актуальные значения параметров задачи.

Кроме того, в ряде задач присутствует набор переменных, значения которых являются обязательными для задания перед запуском вычислений. Для повышения удобства и снижения риска пропуска таких данных в концепции предусмотрена отдельная зона, где перечисляются переменные, требующие обязательного заполнения.

Также предусмотрена секция, предназначенная для вывода дополнительных данных, которые могут быть необходимы для решения различных типов задач. В данной области отображаются такие элементы, как формульные решения, графики, а также текстовые ответы, которые могут служить важным дополнением к визуальному представлению переменных. Это позволяет пользователю получить более полную картину произошедших вычислений и облегчить понимание результатов задач.

На рисунке 2 приведён пример схемы интерфейса, который разделён на три визуальных блока. Так как концепция отображения не даёт конкретного описания расположения элементов, то наличие, размер и расположение подобных блоков может быть любым. Например, в случае реализации интерфейса в форме подобных блоков, возможна их вертикальная организация, что является более удобным решением для устройств с узким экраном, таких как смартфон.

Рисунок 2 — Пример схемы интерфейса

2.4.2 Модель описания визуального представления

Главной идеей предлагаемого решения является расширение базы активных знаний, за счёт внедрения дополнительной прослойки, выполняющей функцию описания визуального интерактивного представления модели базы активных знаний. Целью данного подхода является обеспечение отделения визуального представления от данных вычислительной модели, за счёт добавления дополнительного раздела с описанием, а не вписывания представления в данные модели, что повышает адаптивность, масштабируемость и гибкость решения.

Предлагаемая прослойка представляет собой структурированное описание, содержащее информацию, достаточную для отображения элементов модели в визуальном интерфейсе. Она включает в себя следующие ключевые элементы:

1. Пространственное размещение переменных — здесь задаются координаты и взаимное расположение отображаемых переменных и объектов на экране.

2. Визуальное оформление — в описание могут включаться такие параметры, как изображение заднего фона, цветовая схема, а также стили отображения переменных.
3. Перспективы — реализуется возможность задания различных точек обзора и ракурсов визуализации, это позволяет гибко адаптировать отображение под различные задачи, в которых сложно уместить всю необходимую информацию в ограниченном экранном пространстве.
4. Фиксированные переменные ввода — выделяется набор переменных, значения которых должны быть обязательно заданы пользователем. Для них описывается название переменной, её тип и краткое описание.

2.5 Анализ предлагаемых решений

Одним из основных предъявляемых требований являлось создание модели, обеспечивающей описание, достаточное для охвата практически значимых предметных областей. Разработанная модель описания визуального представления удовлетворяет данному критерию, обеспечивая при этом чёткую структуру и функциональность, что проявляется в следующих аспектах:

Во-первых, в модели предлагается описание пространственного размещения переменных, что позволяет для каждой вычислительной модели задать наглядное расположение элементов на экране. Гибкость размещения делает визуализацию адаптируемой под конкретные особенности каждой отдельной модели, что расширяет область применения решения.

Во-вторых, модель содержит описание визуального оформления, включая параметры фона и цветовые схемы. Это позволяет адаптировать интерфейс под специфические требования конкретной вычислительной модели. Также визуальные параметры могут использоваться для передачи дополнительной смысловой информации.

В-третьих, наличие перспектив отображения — возможность создания сцен с различными наборами отображаемых переменных и визуальными оформлениями, обеспечивая при этом удобную навигацию. Это актуально в условиях работы с вычислительными моделями, содержащими большое

количество параметров, где трудно эффективно отобразить всю структуру на одной сцене.

В-четвёртых, модель поддерживает фиксированные переменные ввода, позволяющие указать параметры, значения которых должны быть заданы пользователем до запуска вычислений. Эта возможность разграничения обязательных входных параметров от прочих делает решение более гибким к применению на различных вычислительных моделях.

В результате, все описанные компоненты модели визуального представления обеспечивают её способность охватывать несколько практически значимых предметных областей.

Также реализованным требованием предлагаемой концепции визуального представления является её переносимость. Концепция не зависит от конкретной реализации интерфейса или среды выполнения, что делает её универсальной и пригодной для применения на различных платформах.

Таким образом, разработанные решения полностью соответствуют предъявляемым требованиям.

3 Программная реализация и тестирование

3.1 Программная реализация

В данном параграфе будут использоваться два вида переменных. Первый вид включает в себя поля, находящиеся в JSON-файле с форматом описания. Такие переменные обозначаются в тексте с использованием двойных кавычек, например, “id”. Второй вид представляет собой глобальные переменные, определённые в коде веб-приложения. Они обозначаются с использованием курсива, например, *perspective*.

3.1.1 Реализация формата описания

Формат визуального представления был реализован в JSON-формате, а именно в специализированном формате JAZ. JAZ предусматривает хранение не только описания вычислительных моделей, но и фрагментов кода, констант и других вспомогательных данных, которые могут быть использованы интерфейсом для визуализации.

Выбор JSON и его подмножества JAZ для хранения формата визуального представления был основан на нескольких важных факторах:

- Единый формат. В системе LuNA уже используется JAZ для хранения баз активных знаний, и для упрощения архитектуры было принято решение хранить описание визуальных представлений в том же формате. Это позволяет избежать необходимости работы с множеством различных форматов данных и уменьшает сложность интеграции различных частей системы.
- Гибкость и масштабируемость формата. JSON является лёгким для обработки форматом, поддерживаемым практически всеми распространёнными языками программирования и библиотеками, что упрощает разработку и дальнейшую поддержку системы. Также он позволяет легко масштабировать формат, то есть добавлять новые поля и данные.

Сам формат описания визуального представления, пример приведён в листинге 1, был спроектирован с учётом удобства работы с данными и содержит следующие поля:

- “cm”: Это поле указывает на связь с вычислительной моделью, к которой относится данное визуальное представление. Подробнее об этом будет сказано в разделе 3.1.2.
- “perspectives”: Это массив объектов, каждый из которых описывает одну из точек обзора. В каждой перспективе описываются её свойства, задний фон и расположение переменных на экране.

  “id”: Уникальный идентификатор перспективы.

  “title”: Название перспективы, которое используется для отображения пользователю, чтобы он мог необходимую по названию.
- “images”: В этом объекте указаны параметры изображения, которое будет использоваться в качестве фона.
  “path”: Путь к изображению.
  “width” и “height”: Размеры изображения в пикселях.
- “variables”: Это массив объектов, каждый из которых представляет конкретную переменную, которую необходимо отразить на плоскости.
  “name”: Имя переменной, которое совпадает с именем этой переменной в вычислительной модели. Необходимо для связи отображаемых переменных с переменными вычислительной модели.

  “type”: Тип переменной, который определяет, как она может быть визуально представлена в интерфейсе. На основе этого типа выбирается подходящий визуальный компонент, который будет отображён на месте переменной.

  “coordX” и “coordY”: Координаты переменной относительно центра верхней стороны фонового изображения.
- “required_inputs”: Массив фиксированных переменных ввода.

```
"view": {
    "cm": "local://cm/seismic",
    "perspectives": [
        {
            "id": 1,
            "title": "Задний фон",
            "image": {
                "path": "/assets/background.svg",
                "width": "544",
                "height": "470"
            },
            "variables": [
                {
                    "name": "Trace 1",
                     "type": "chart",
                    "coordX": -136,
                    "coordY": 237
                }
            ]
        }
    ],
    "required_inputs": [
        {
            "name": "cuda_operation_count",
            "label": "Количество операций на CUDA",
            "type": "int"
        }
    ]
},
```

Листинг 1 — Пример описания визуального представления

Этот формат обеспечивает структурированное описание визуальных представлений, где каждый элемент имеет чётко определённое место и роль. Разделение на перспективы и их переменные позволяют легко масштабировать формат и адаптировать его под различные типы моделей и задачи.

3.1.2 Особенности формата описания

Одной из ключевых особенностей реализации формата визуального представления является встроенный механизм связи с вычислительной моделью. Поскольку в системе может одновременно существовать множество моделей и соответствующих им визуальных описаний, необходимо обеспечить привязку каждого представления к конкретной модели. Для этого в структуре формата было добавлено специальное поле “cm”, содержащее ссылку на вычислительную модель, к которой относится данное представление. Такая конструкция позволяет системе избегать конфликтов и однозначно определять соответствие между моделью и её визуализацией.

Ещё одной важной особенностью является явное указание размеров фонового изображения, используемого в визуальном представлении. Эти данные позволяют правильно вычислять масштаб отображаемых переменных на

фоновом изображении. При изменении масштаба интерфейса важно, чтобы все переменные правильно позиционировались относительно фона, и размеры изображения обеспечивают возможность этих вычислений. Подробнее об этом будет сказано в разделе 3.1.4.

3.1.3 Реализация веб-интерфейса

Пользовательский интерфейс был реализован в формате веб-приложения с использованием библиотеки React в сочетании с библиотекой готовых компонентов Material UI. Выбор React обусловлен его широкой распространённостью, высокой скоростью разработки, а также обилием готовых решений и библиотек. Использование Material UI позволило значительно сократить время на реализацию стандартных элементов управления благодаря наличию адаптивных и визуально единообразных готовых компонентов.

Визуальный интерфейс был организован в виде трёх отдельных секций, каждая из которых выполняет конкретную функциональную задачу:

- Левая секция служит точкой ввода и отображения параметров модели. В этой области пользователь может просматривать список выбранных входных и выходных переменных, список обязательных параметров, а также осуществлять ввод значений. Дополнительно в нижней части этой секции располагаются кнопки запуска расчёта и генерации программы, активирующие формирование запроса, содержащего все выбранные пользователем переменные, а также введённые значения, и отправляет его на сервер. Полученные результаты отображаются в соответствующих полях выходных переменных.
- Центральная секция представляет собой основную рабочую область, на которой визуализированы переменные в виде графических объектов, размещённых на плоскости. Здесь реализована возможность взаимодействия с визуальными элементами: выбор переменных, переключение между режимами (входные/выходные) и сброс выбранных переменных. Центральная область также включает фон, представляющий собой изображение, служащее визуальным дополнением.

- Правая секция предназначена для отображения дополнительных результатов, полученных во время вычислений: это могут быть текстовые описания, формульные решения, графики или прочая сопутствующая информация, дополняющая полученные результаты.

Примеры работы реализованного интерфейса представлены на рисунках 4-6 в параграфе 3.2. Тестирование.

Функциональность переключения режима роли переменной реализована в виде двух кнопок, расположенных в верхней части центральной секции. Каждая из кнопок имеет текст “Входные” и “Выходные” соответственно. При активации одной из кнопок она начинает визуально выделяться, в то время как ранее выбранная кнопка теряет выделение. Обработчик события нажатия кнопки меняет значение переменной состояния *stateInOutMode* в соответствии с выбранным режимом.

Работа интерфейса тесно связана с форматом описания, информация из которого используется для генерации всех элементов отображения. При инициализации страницы программа извлекает из JSON-файла поле “view” и ищет в нём объект перспективы с идентификатором id = 1, присваивая его переменной *perspective*. На основе этой переменной строится всё дальнейшее отображение, из массива *perspective.variables* извлекается информация о переменных, необходимая для визуализации, включая их наименования и координаты размещения. Путь к фоновому изображению и его наименование получаются из поля *perspective.image.path*. Кроме того, из поля “required_inputs” объекта “view” извлекается информация для отображения полей о переменных, обязательных для заполнения пользователем. Таким образом, визуальный интерфейс формируется динамически в зависимости от описания, поступающего из формата, что обеспечивает универсальность и независимость интерфейса от конкретной вычислительной модели.

В правой секции пользовательского интерфейса предусмотрена возможность отображения дополнительных результатов в виде графиков и формул. Для реализации данной функциональности были использованы готовые

программные библиотеки. Визуализация графиков осуществляется с помощью библиотеки Recharts, разработанной для React. Для отображения формульных выражений в формате LaTeX применяется библиотека react-latex-next, также разработанная для React. Выбор данных библиотек не принципиален, в случае необходимости они могут быть заменены на другие. Сам реализованный веб-интерфейс хранится в репозитории, к которому можно запросить доступ (https://gitlab.ssd.sscc.ru/n.yurakov/interactive-display-system-for-active-knowledge-base). Также для реализованной подсистемы были написаны Руководство программиста (см. Приложение А) и Описание программы (см. Приложение Б).

На схеме, представленной на рисунке 3, отображено взаимодействие клиентского веб-интерфейса с сервером базы активных знаний. Веб-приложение, выступающее в роли клиента, формирует запрос на вычисление сформированной пользователем задачи и отправляет его на сервер, который может быть расположен как локально, так и на удалённом веб-сервере. После обработки запроса сервер возвращает результат, который отображается в пользовательском интерфейсе. В дальнейшем планируется поддержка перехода на внешний инженерный веб-интерфейс с возможностью возврата в данный интерфейс.

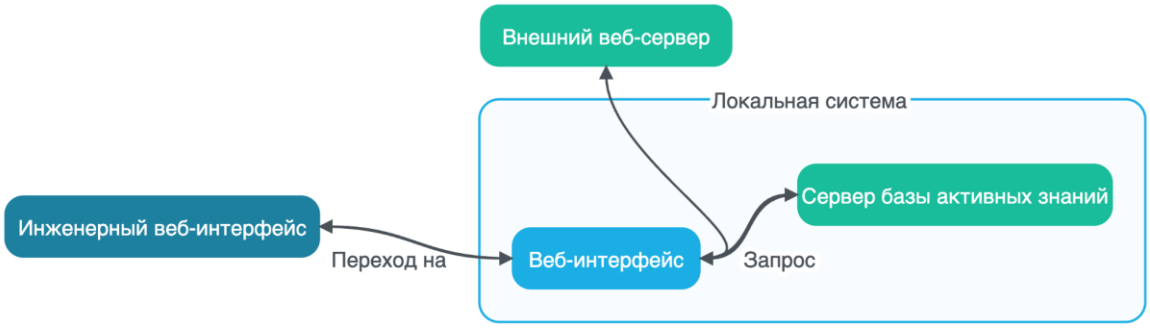

Рисунок 3 — Схема взаимодействия клиентского веб-интерфейса с сервером базы активных знаний

Реализованный интерфейс соответствует предъявляемым требованиям, обеспечивая работу на устройствах с узким экраном с сохранением функциональности. Это было получено за счёт того, что при реализации выбирались и использовались только те элементы интерфейса, которые

поддерживали бы функциональность и удобство использования как на компьютере, так и на смартфоне. Кроме того, в случае использования устройства с узким экраном применяется альтернативная схема расстановки секций, вместо горизонтального размещения блоки выстраиваются в вертикальном порядке. В верхней части отображается блок для ввода значений, за ним блок, предназначенный для выбора переменных, и в конце следует блок, содержащий дополнительную информацию.

В рамках веб-интерфейса реализован следующий функционал:

- извлечение и обработка данных из формата визуального представления
- отображение переменных и заднего фона на плоскости
- выбор переменных в зависимости от режима (вход/выход)
- ввод значений для выбранных параметров
- отправка запроса на запуск вычислений
- отображение дополнительной информации

Во многих предметных областях при решении задач необходимо работать с набором переменных. Разработанный веб-интерфейс обеспечивает отображение таких переменных, а также предоставляет средства для задания их значений. Возможны случаи, когда простого отображения переменных недостаточно, тогда для повышения интуитивной понятности и наглядности предусмотрена возможность представления переменных в различных образах и использование фонового изображения с расположением переменных относительно этого изображения. Для настройки параметров, необходимых в большинстве задач, интерфейс предусматривает механизм выбора переменных и задания их значения в специальной секции. В тех случаях, когда численного ответа недостаточно и требуется визуализация дополнительных данных, предусмотрена соответствующая секция для отображения такого рода информации. Таким образом, реализованного функционала достаточно для покрытия широкого спектра возможных задач и обеспечения применимости интерфейса для отображения баз активных знаний различных предметных областей.

Следует отметить, что несмотря на то, что формат представления поддерживает описание различных перспектив отображения, механизм переключения между ними пока не реализован. Тем не менее архитектура системы позволяет установить данную функциональность без необходимости масштабной переработки.

3.1.4 Обеспечение масштабируемости по размеру экрана

Для обеспечения правильного отображения интерфейса при изменении размеров окна браузера была реализована система масштабируемого позиционирования элементов на основе относительных координат. Так как визуальные переменные размещаются на плоскости относительно фонового изображения, то важно сохранять их точное положение относительно фонового изображения вне зависимости от текущего размера окна.

Для этого в формате визуального представления были добавлены поля с исходными размерами фонового изображения, а координаты переменных указываются относительно этих размеров. При каждом изменении размеров окна программа автоматически вычисляет новые размеры фонового изображения, и на основе этой информации вычисляются новые координаты переменных, используя соотношение между текущим размером окна и базовым размером фона. Алгоритм основан на следующей формуле:

*new_coord = (new_image_size / base_image_size) * base_coord*

new_coord - новые координаты элемента

new_image_size - текущий размер фонового изображения

base_image_size - размер изображения, указанный в формате описания

base_coord - координаты этого элемента указанные в формате описания

Такой подход позволяет переменным оставаться привязанными к заданным областям изображения независимо от разрешения экрана, обеспечивая адаптивность интерфейса.

3.1.5 Алгоритм выбора параметров задачи

Алгоритм выбора параметров, был реализован на языке программирования JavaScript с использованием библиотеки React. Основной задачей алгоритма

является динамическое формирование двух списков переменных на основе действий пользователя: входных и выходных. Выбор роли переменной был описан ранее и хранит своё значение в переменной *stateInOutMode*. Так как эти списки переменных должны отображаться динамически, то было принято решение использовать переменные состояния (useState), обеспечивающие необходимую реактивность. Таким образом, входные и выходные параметры, хранящиеся в состояниях *inputVars* и *outputVars*, позволяют интерфейсу отображать любые изменения, внесённые пользователем.

Реализованная функция *addNewVars* принимает на вход объект из массива *perspective.variables*, соответствующий выбранной пользователем переменной, и по полю name ищет в вычислительной модели соответствующую переменную. На основе полученной информации определяется её тип, после чего формируется объект переменной для списка. Далее переменная добавляется в соответствующий массив в зависимости от выбранной пользователем роли.

Также алгоритм предусматривает защиту от дублирования: если выбранная переменная уже присутствует в целевом массиве, то повторное добавление не будет произведено. В случае, если переменная уже содержится в противоположном массиве, то она предварительно из него удаляется перед добавлением в новый. Таким образом, переменная не может одновременно выступать как входной, так и выходной, что исключает логические конфликты в рамках одной вычислительной модели.

3.2 Тестирование

Для проверки заявленного свойства разработанной модели описания визуального представления баз активных знаний, а именно её достаточности для охватить практически значимых предметных областей, было проведено тестирование. Тестирование проводилось практическим методом, данный способ был выбран в связи с тем, что в данном случае свойство достаточности можно проверить только на практике.

Для тестирования были выбраны несколько предметных областей:

1. Тригонометрия — классическая математическая область, хорошо подходящая для описания визуального представления и проверки интерфейса на отображение графических элементов и взаимодействие. Выбор данной предметной области для создания визуального представления такой вычислительной модели обусловлен её практической значимостью, которая заключается в необходимости создания демонстрационного на понятном для пользователя примере возможностей системы LuNA, в том числе в образовательных целях.
2. Многоканальная сейсмосвёртка [27] — более сложная, практически значимая область геофизических исследований, связанная с обработкой сейсмических сигналов, поступающих с множества каналов. Суть задачи заключается в применении сейсмосвёртки — процедуры сопоставления записанных сигналов с заранее известным опорным сигналом для определения момента прихода сейсмических волн.

   Само тестирование проводилось следующим образом:

   Этап 1. Описание визуального представления

Вначале был проведён анализ предметной области, в ходе которого были выявлены ключевые объекты и их взаимосвязи. Для тригонометрии это стороны, углы, периметр и площадь треугольника. Для задачи многоканальной сейсмосвёртки это сейсмические сигналы и опорный сигнал, которые задаются в вычислительной модели. Кроме того, пользователь должен иметь возможность выбора определённых сейсмических сигналов для обработки, а также настройки параметров вычислений, таких как количество операций, выполняемых на графическом процессоре.

На основе полученных данных было сформировано представление того, как должно выглядеть отображение, расположение переменных и фоновое оформление. После чего это визуальное представление было описано с использованием разработанного формата описания, что позволило проверить применимость модели и формата для описания предметных областей подобного типа.

Этап 2. Тестирование описания в интерфейсе

Полученные файлы описания визуального представления были загружены в разработанный веб-интерфейс, после чего была проведена проверка соответствия отображения и ожидаемого результата. Результат отображения описанных визуальных представлений в веб-интерфейсе представлены на рисунках 4-6. Кроме того, на данном этапе проверяется правильность взаимодействия интерфейса с описанием визуального представления. Если определённые элементы не отображались должным образом, визуализация или функциональность оказывались недостаточными, то это свидетельствовало о недостатке функциональности либо в модели, либо в формате, либо в интерфейсе. Таким образом, данный этап позволял не только подтвердить работоспособность системы, но и выявить направления, требующие доработки.

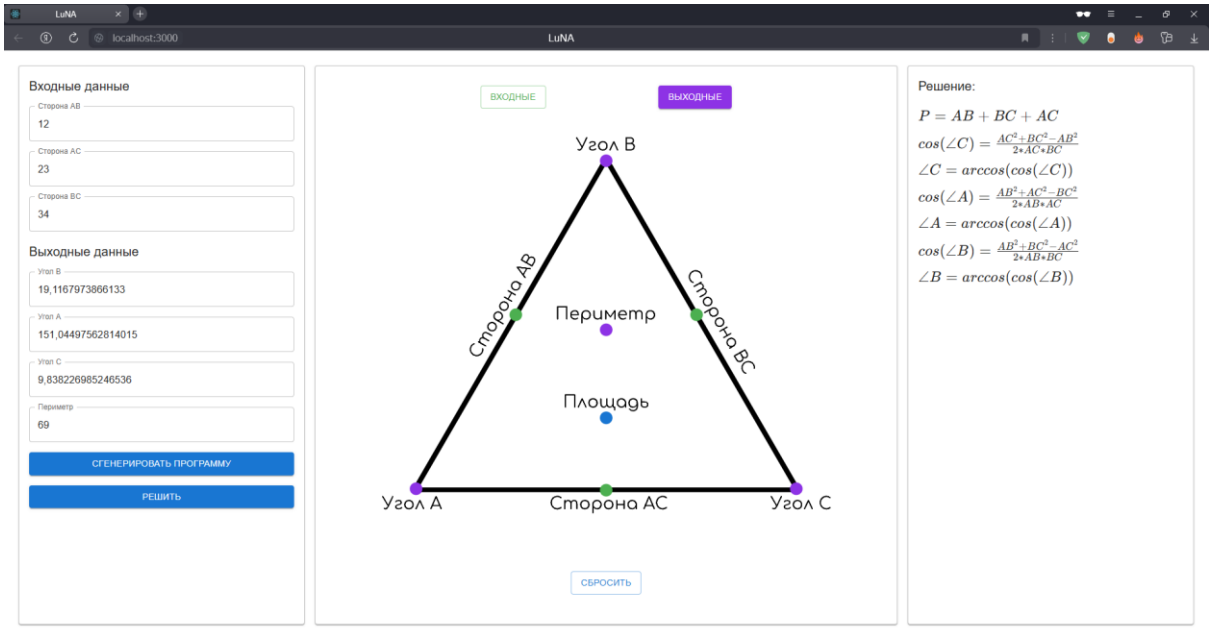

Рисунок 4 — Веб-интерфейс, отображающий визуальное представление задачи тригонометрии

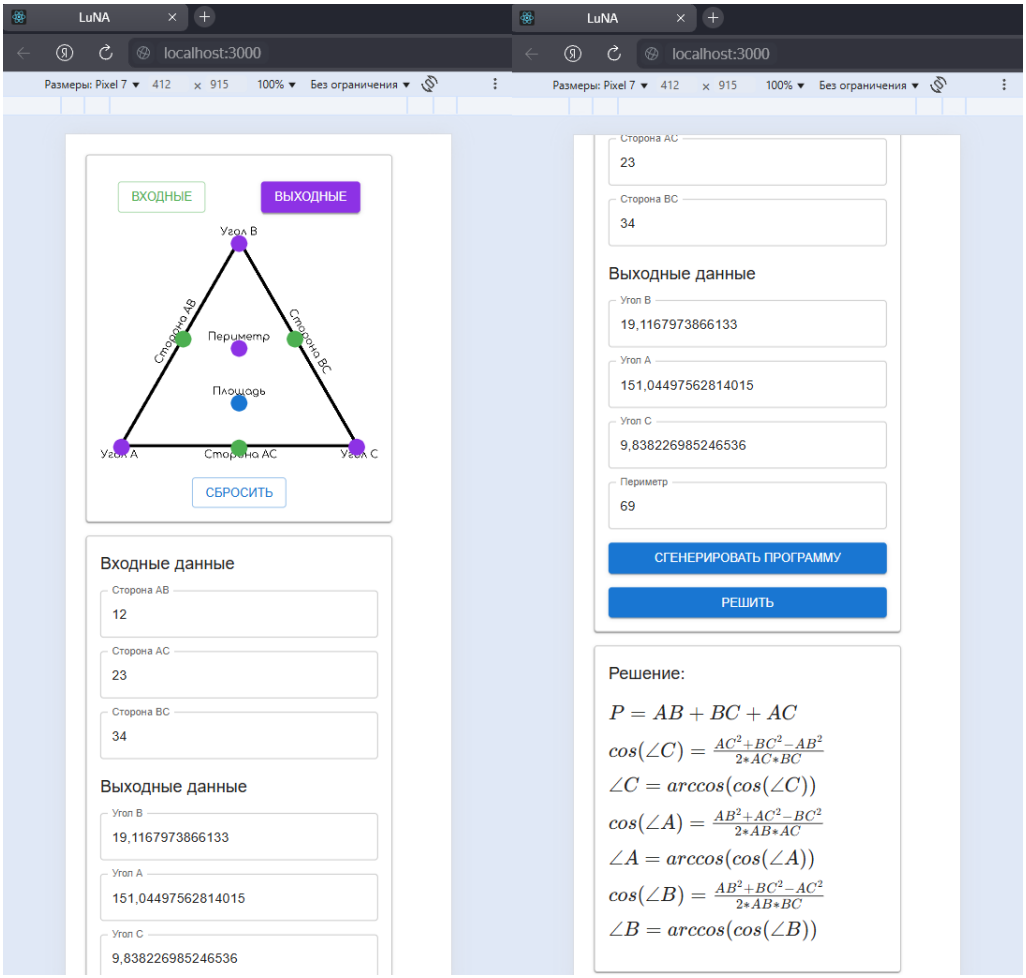

Рисунок 5 — Веб-интерфейс, отображающий визуальное представление задачи тригонометрии в мобильном формате

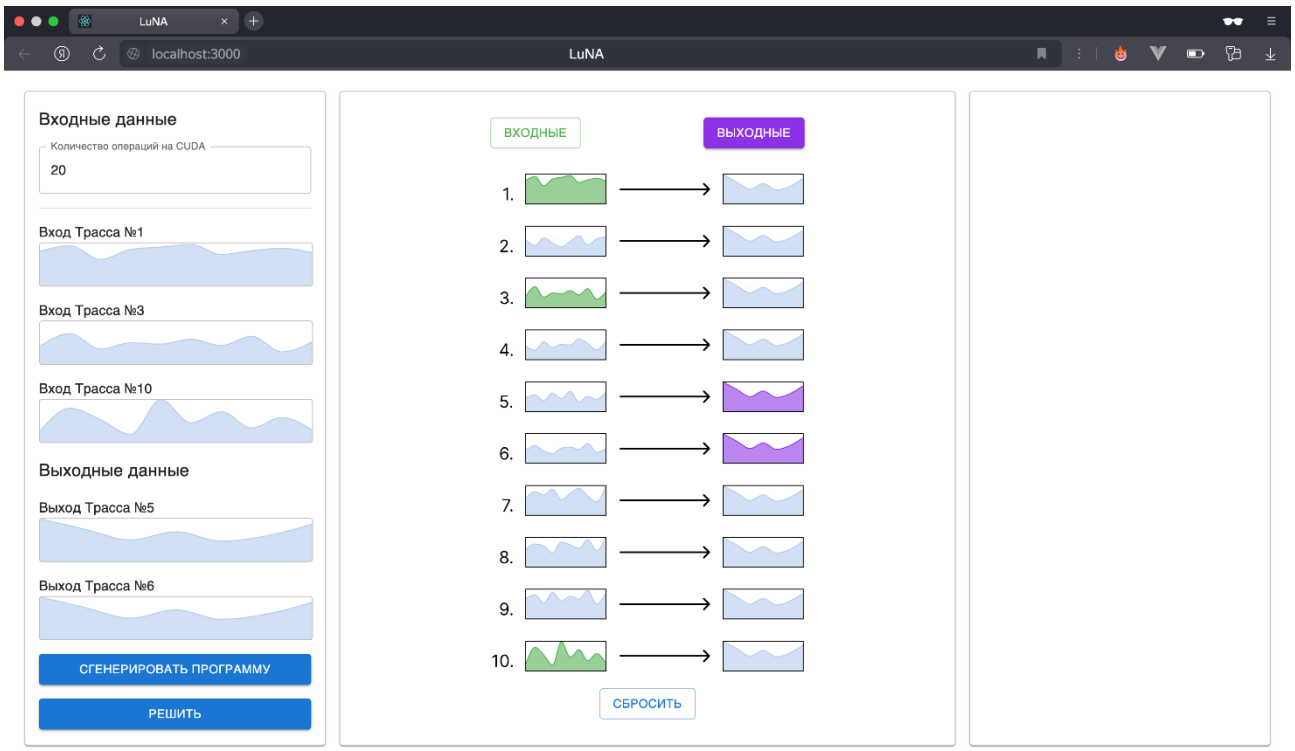

Рисунок 6 - Веб-интерфейс, отображающий визуальное представление задачи многоканальной сейсмосвёртки

Проведённое тестирование подтвердило достаточность разработанной модели описания визуального представления для описания практически значимых предметных областей. Кроме того, тестирование позволило выявить потенциальные направления дальнейшего развития как модели с форматом, так и интерфейса.

Таким образом, экспериментальные тесты не только подтвердили заявленное свойство разработанного решения, но и продемонстрировали его практическую применимость и потенциальную масштабируемость.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была разработана модель описания визуального представления, а также реализованы формат и подсистема интерактивного представления баз активных знаний в системе LuNA. Разработанный формат и подсистема, которые обеспечивают вычислительные модели наглядной и интерактивной визуализацией, были встроены в систему активных знаний LuNA в качестве модуля визуального отображения.

Было проведено тестирование разработанного формата и подсистемы интерактивного представления, которое подтверждает, что формат и подсистема обеспечивают достаточное описание и функциональность для того, чтобы охватить несколько практически значимых предметных областей.

Результаты работы были представлены на 63-ей Международной научной студенческой конференции, г. Новосибирск, 2025 г. [28]

Защищаемые положения:

1. Разработана концепция визуального отображения пользовательского интерфейса.
2. Разработана модель описания визуального представления баз активных знаний.
3. Разработанные модель описания и концепция визуального отображения были реализованы в виде веб-интерфейса с форматом описания и интегрированы в систему LuNA.

В дальнейшем планируется продолжить работу над интерактивной визуализацией баз активных знаний в системе активных знаний LuNA. Возможные направления развития:

1. Усовершенствование формата и подсистемы визуального представления для охвата большего количества практически значимых предметных областей.
2. Обеспечение совместимости и интеграции с другими компонентами системы LuNA.

Выпускная квалификационная работа выполнена мной самостоятельно и с соблюдением правил профессиональной этики. Все использованные в работе материалы и заимствованные принципиальные положения (концепции) из опубликованной научной литературы и других источников имеют ссылки на них. Я несу ответственность за приведенные данные и сделанные выводы.

Я ознакомлен с программой государственной итоговой аттестации, согласно которой обнаружение плагиата, фальсификации данных и ложного цитирования является основанием для не допуска к защите выпускной квалификационной работы и выставления оценки «неудовлетворительно».

_____ _____

ФИО студента *Подпись студента*

« ____ »_____20 __г.
(заполняется от руки)

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Малышкин В.Э., Перепёлкин В.А. Построение баз активных знаний для автоматического конструирования решений прикладных задач на основе системы LuNA // Параллельные вычислительные технологии – XVIII всероссийская научная конференция с международным участием, ПаВТ’2024, г. Челябинск, 2–4 апреля 2024 г. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2024. с. 57-68.
2. Malyshkin V. E., Perepelkin V. A. LuNA fragmented programming system, main functions and peculiarities of run-time subsystem //International Conference on Parallel Computing Technologies. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2011. – С. 53-61.
3. Messina A. Overview of standard graph file formats. – Technical report, RT-ICAR-PA-2018-06, 2018.
4. Касьянов В. Н., Касьянова Е. В. Представление графов и графовых моделей: базовые средства языка GraphML //Проблемы информатики. – 2018. – №. 1 (38). – С. 4-19.
5. Дюндюков В. С. Ресурсно-целевые сети //Программные продукты и системы. – 2014. – №. 2 (106). – С. 93-99.
6. Himsolt M. GML: A portable graph file format. – Technical report, Universitat Passau, 1997.
7. Патаракин Е. Д. Карты и диаграммы связей для совместного конструирования и исследования //Школьные технологии. – 2010. – №. 2. – С. 84-91.
8. Graphviz. Официальная документация Graphviz [Электронный ресурс]. – Режим доступа: https://graphviz.org/ (дата обращения 22.02.2025).
9. uDraw(Graph) - The powerful solution for graph visualization. Официальная документация uDraw [Электронный ресурс]. – Режим доступа: https://www.informatik.uni-bremen.de/uDrawGraph/en/uDrawGraph/uDrawGraph.html (дата обращения 22.02.2025).

10. yEd Graph Editor Manual. Официальная документация yEd [Электронный ресурс]. – Режим доступа: https://yed.yworks.com/support/manual/index.html (дата обращения 23.02.2025)
11. Золотухин Т. А. Визуализация графов при помощи программного средства Visual Graph //Информатика в науке и образовании. – 2012. – С. 135-148.
12. Sander G. VCG-visualization of compiler graphs. – 1995.
13. DBeaver Community. Официальная документация DBeaver [Электронный ресурс]. – Режим доступа: https://dbeaver.io/ (дата обращения 05.03.2025)
14. Фролов Ю. В. Изучение языка SQL с применением кроссплатформенного менеджера в процессе подготовки бизнес-информатиков //Актуальные проблемы теории и практики обучения физико-математическим и техническим дисциплинам в современном образовательном пространстве. – 2022. – С. 206-211.
15. DataGrip Documentation. Официальная документация DataGrip [Электронный ресурс]. – Режим доступа: https://www.jetbrains.com/help/datagrip/getting-started.html (дата обращения 06.03.2025)
16. DbVisualizer User Guide. Официальная документация DbVisualizer [Электронный ресурс]. – Режим доступа: https://www.dbvis.com/docs/ug/ (дата обращения 06.03.2025)
17. Fathulloh A. H., Adauwiyah H. I. Perbandingan tingkat efisiensi waktu query select pada database interface navicat dan SQLYog di MySQL DBMS //Appl. Inf. Syst. Manag. – 2021. – Т. 4. – №. 2. – С. 101-105.
18. Цыганкова Д. С., Козлов В. В. Основы HTML: изучаем основной язык веб-разработки //Форум молодых ученых. – 2023. – №. 10 (86). – С. 189-193.
19. Ахмеджанова З., Гафурова П. Применение html и css для создания интерактивных Веб сайтов //Евразийский Союз Ученых. – 2019. – №. 4-3 (61). – С. 7-10.
20. Поток документа - HTML - Дока [Электронный ресурс]. – Режим доступа: https://doka.guide/html/flow/ (дата обращения 09.03.2025)

21. Block and inline layout in normal flow. Официальная документация MDN [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_display/Block_and_inline_layout_in_normal_flow (дата обращения 09.03.2025)
22. In flow and out of flow. Официальная документация MDN [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_display/In_flow_and_out_of_flow (дата обращения 09.03.2025)
23. Бахтин И. В. Главные принципы MVC и смысл использования в разработке программных продуктов //Форум молодых ученых. – 2020. – №. 1 (41). – С. 63-65.
24. Кедрин В. С., Кузьмин О. В. Применение программной архитектуры Model-View-Controller для построения системы визуального проектирования и моделирования алгоритмов //Информатика, телекоммуникации и управление. – 2011. – №. 4 (128). – С. 36-42.
25. Артюхов А. А. Прототип базы активных знаний на основе вычислительных моделей //Проблемы информатики. – 2021. – №. 4 (53). – С. 55-66.
26. Коптева А. В., Князев И. В. Анализ проблемы преобразования данных формата JSON в строго типизированных языках программирования на примере Golang //Проблемы науки. – 2021. – №. 7 (66). – С. 5-10.
27. Выродов А. Ю. и др. Принципы организации программно-аналитической системы для параллельной обработки сейсмических данных //Вестник СибГУТИ. – 2024. – Т. 18. – №. 2. – С. 57-68.
28. Юраков, Н. Д. Разработка формата и подсистемы интерактивного представления баз активных знаний в системе LuNA / Н. Д. Юраков // Информационные технологии. Программная инженерия и инженерия знаний: Материалы 63-й Междунар. науч. студ. конф. 16–22 апреля 2025 г. /Новосиб. гос. ун-т. (в печати)

ПРИЛОЖЕНИЕ А

ПОДСИСТЕМА ВИЗУАЛЬНОГО ОТОБРАЖЕНИЯ БАЗ АКТИВНЫХ ЗНАНИЙ ДЛЯ СИСТЕМЫ “LuNA”
РУКОВОДСТВО ПРОГРАММИСТА

Листов 8
Новосибирск 2025

СОДЕРЖАНИЕ

АННОТАЦИЯ

В данном документе приведено руководство программиста для подсистемы визуального отображения баз активных знаний для системы активных знаний LuNA.

Функцией подсистемы является интерактивное и адаптивное отображение визуального представления баз активных знаний, заданного в формате описания.

Исходными языками программирования являются JavaScript, JSX, HTML и CSS. Рекомендуемое средство разработки – среда разработки Visual Studio Code от компании Microsoft.

Оформление программного документа «Руководство оператора» произведено по требованиям ГОСТ 19.504-79 «ЕСПД. Руководство программиста» и ГОСТ 19.105-78 «Единая система программной документации (ЕСПД). Общие требования к программным документам (с Изменением N 1)».

1 НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

1.1 Назначение программы

Обеспечение наглядного и интерактивного представления баз активных знаний системы LuNA с возможность взаимодействия.

1.2 Функции, выполняемые программой

Программа выполняет визуализацию баз активных знаний, предоставляет пользователю интерфейс для выбора входных и выходных параметров, позволяет вводить значения параметров, отправлять запрос на вычисление задачи на сервер базы активных знаний и отображать полученные результаты, включая текстовые данные, графики и формульные выражения.

1.3 Условия, необходимые для выполнения программы

Для выполнения программы необходим веб-браузер с поддержкой JavaScript (например, Google Chrome, Mozilla Firefox), подключение к серверу базы активных знаний системы LuNA и наличие файла описания визуального представления в формате JAZ. Также для запуска программы требуется установленная среда Node.js (рекомендуемая версия от 18.x) и менеджер пакетов npm.

2 ХАРАКТЕРИСТИКА ПРОГРАММЫ

Разработанный веб-интерфейс является независимым модулем визуального отображения, предназначенным для отображения и взаимодействия с базами активных знаний в системе LuNA. Программа реализована в виде одностраничного приложения на базе библиотеки React и запускается в большинстве современных браузеров. Исходный код интерфейса расположен отдельно от серверной части системы LuNA и взаимодействует с ней посредством стандартных HTTP-запросов.

Программа не требует жёсткой интеграции с другими модулями системы и построена по принципу слабо связанной архитектуры. Основное требование к системе LuNA — наличие API-интерфейса, обеспечивающего приём задач и возврат вычисленных результатов. Все визуальные компоненты и логика взаимодействия с пользователем реализуются на стороне клиента.

Визуализация осуществляется на основе файла описания в формате JAZ, содержащего визуальное описание, координаты и параметры переменных вычислительной модели. Программа обеспечивает отображение элементов, выбор входных и выходных параметров, ввод значений, отправку задачи на вычисление и получение результата в текстовом, графическом и формульном виде.

Для обработки пользовательских действий и генерации интерфейса используются механизмы состояния React и событийные обработчики. Масштабируемость интерфейса обеспечивается системой адаптивного позиционирования визуальных элементов в зависимости от размеров экрана.

3 ОБРАЩЕНИЕ К ПРОГРАММЕ

Запуск визуальной подсистемы осуществляется с помощью командой npm start. Загрузка визуального представления производится автоматически при инициализации компонента MainPage, при наличии корректного файла в формате JAZ. Привязка к вычислительной модели обеспечивается через поле “cm” JSON-файла.

Активация вычислений происходит при вызове функции handelSolve. На основании выбранных пользователем переменных формируются структуры входных данных (inputObjToSend) и список выходных переменных (outputToSend). Запрос на выполнение вычислений отправляется функцией createNewJob, результатом чего является job_id.

Интерфейс опрашивает состояние задачи с использованием getInfoJob(job_id) до получения ответа с кодом 200, из полученного ответа хранятся результаты вычислений. По завершении вычислений происходит обновление значений выходных переменных и отображение последовательности операций в виде LaTeX-формул или графов. Очистка задачи выполняется через deleteJob.

При изменении размеров окна вызывается функция updateImageSize, которая вычисляет текущие высоту (height) и ширину (width) фонового изображения и обновляет значения размеров картинки, используя команду setImageSize({width, height}).

4 ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

4.1 Организация используемой входной информации

Входными данными для подсистемы визуального представления являются:

- Файл визуального описания в формате JAZ, содержащий информацию о вычислительной модели, её переменных, перспективах отображения, координатах, типах данных и дополнительных параметрах визуализации. Данный файл загружается автоматически при инициализации интерфейса и используется для построения интерактивной визуальной области.
- Данные, вводимые пользователем — значения переменных, выбранные входные и выходные переменные. Ввод осуществляется через соответствующие поля в левой панели интерфейса, тип которых определяется автоматически на основе описания переменной в модели.

4.2 Организация используемой выходной информации

Выходными данными являются:

- Визуальное отображение модели, сформированное на основе координат и параметров, заданных в JAZ-файле. Веб-интерфейс отображает переменные и фоновые изображения.
- Результаты вычислений, полученные от сервера базы активных знаний. После завершения расчётов значения выходных переменных отображаются в интерфейсе в виде значений переменных и дополнительной информации в виде графиков или формульных выражений в формате LaTeX.

5 СООБЩЕНИЯ

В процессе работы подсистемы интерактивного представления баз активных знаний вывод сообщений осуществляется через консоль браузера. Сообщения используются преимущественно для отладки и информирования разработчика о ходе выполнения программы и возможных ошибках при взаимодействии с сервером.

В случае успешного создания задания выводится содержимое ответа сервера с job_id (“Ответ сервера: {“job_id”: ...}:)

При получении результатов задачи отображается полный JSON-ответ с данными переменных и операций (“Ответ сервера: { variables: ..., operations: ... }”)

В случае, если ответ сервера не удаётся разобрать, в консоль выводится ошибка и ответ сервера (“Ошибка при парсинге JSON”, “ Ответ сервера: <текстовое содержимое>”)

ПРИЛОЖЕНИЕ Б

ПОДСИСТЕМА ВИЗУАЛЬНОГО ОТОБРАЖЕНИЯ БАЗ АКТИВНЫХ ЗНАНИЙ ДЛЯ СИСТЕМЫ “LuNA”
ОПИСАНИЕ ПРОГРАММЫ

Листов 12
Новосибирск 2025

СОДЕРЖАНИЕ

АННОТАЦИЯ

В данном документе приведено описание подсистемы визуального отображения баз активных знаний для системы активных знаний LuNA.

Функцией подсистемы является интерактивное и адаптивное отображение визуального представления баз активных знаний, заданного в формате описания.

Исходными языками программирования являются JavaScript, JSX, HTML и CSS. Рекомендуемое средство разработки – среда разработки Visual Studio Code от компании Microsoft.

Оформление программного документа «Описание программы» произведено по требованиям ГОСТ 19.402-78 «ЕСПД. Описание программы» и ГОСТ 19.105-78 «Единая система программной документации (ЕСПД). Общие требования к программным документам (с Изменением N 1)»

1 ОБЩИЕ СВЕДЕНИЯ

1.1 Обозначение и наименование программ

Полное название программы - подсистема визуального отображения баз активных знаний для системы “LuNA”

1.2 Программное обеспечение, необходимое для функционирования программы

Для функционирования программы необходимы веб-браузер с поддержкой JavaScript (например, Google Chrome, Mozilla Firefox), подключение к серверу базы активных знаний системы LuNA, наличие файла описания визуального представления в формате JAZ, установленная среда Node.js (рекомендуемая версия от 18.x) и менеджер пакетов npm.

1.3 Языки программирования

Исходными языками программирования являются JavaScript, JSX, HTML и CSS. Для разработки использовался редактор Visual Studio Code от компании Microsoft.

2 ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

2.1 Назначение программы

Программа предназначена для обеспечение наглядного и интерактивного представления баз активных знаний системы LuNA с возможность взаимодействия.

2.2 Сведения о функциональных ограничениях на применение

Программа не предназначена для запуска на устройствах, на которых отсутствует веб-браузер с поддержкой JavaScript (например, Google Chrome, Mozilla Firefox), подключение к серверу базы активных знаний системы LuNA, файл описания визуального представления в формате JAZ, среда Node.js (рекомендуемая версия от 18.x) и менеджер пакетов npm.

3 ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1 Алгоритм программы

Программа загружает визуальное описание вычислительной модели в формате JAZ и на его основе формирует интерактивное представление в виде двумерной схемы с визуальными переменными. Пользователь выбирает входные и выходные параметры, после чего программа формирует и отправляет запрос на выполнение вычислений на сервере базы активных знаний. Далее программа циклически опрашивает сервер на предмет завершения задачи. По завершении вычислений интерфейс получает значения результат вычислений, после чего обновляет отображение и выводит результаты пользователю.

3.2 Используемые методы

Программа принимает JAZ-файл и на основе него формирует визуальное отображение, где переменные отображаются в виде интерактивных элементов. Для взаимодействия с системой LuNA используются методы createNewJob, getInfoJob и deleteJob, реализующие передачу задачи на выполнение, мониторинг её статуса и удаление после завершения. Эти методы обеспечивают асинхронный обмен данными с сервером базы активных знаний.

Масштабирование визуальных элементов при изменении размера окна реализуется через функцию updateImageSize, которая пересчитывает размеры изображения и на основе этой инфомации программа меняет позиционирование переменных в соответствии с новыми пропорциями.

3.3 Структура программы

Программа состоит из нескольких основных файлов, первая часть из которых относится к реализации веб-интерфейса с использованием библиотеки React, это такие файлы как MainPage.jsx, App.js, index.js, ко второй части относится файл с логикой работы с сервером базы активных знаний, он отвечает за отправку и принятие запросов. Общий набор функций приведён в таблице 1.

Таблица 1 – Набор функций программы

Наименование	Описание

Обновление размеров фонового изображения	Функция updateImageSize() обновляет текущие размеры фонового изображения, чтобы изменить положение визуальных переменных при изменении размера окна
Задержка	Функция sleep(ms) вспомогательная функция задержки на заданное количество миллисекунд
Вычисление задачи	Функция handleSolve() формирует и отправляет запрос к серверу с выбранными входными и выходными переменными, после чего опрашивает её статус и сохраняет полученные результаты
Добавление новой переменной в задачу	Функция addNewVars(varName) добавляет полученную переменную в список входных или выходных переменных
Обработка ввода пользователя	Функция handleChange(e, mode) обновляет значение входных переменных при вводе пользователем
Сброс состояний интерфейса	Функция reset() сбрасывает состояния интерфейса: выбранные переменные, их значения и полученные результаты
Отправка задачи	Функция createNewJob(model, inputs, outputs) формирует и отправляет запрос на вычисление задачи на сервер
Обновление статуса задачи	Функция getInfoJob(jobId) запрашивает у сервера состояние

	задачи, и в случае завершения вычислений возвращает результат
Удаление задачи	Функция deleteJob(jodId) отправляет серверу запрос на удаление задачи из памяти

3.4 Связи между составными частями программы

Связь между файлами программы осуществляется при помощи вызова функций и методов объектов и файлов.

3.5 Связи программы с другими программами

Для возможности связи с сервером базы активных знаний, необходимо указать адрес для доступа как к локальному, так и к удалённому серверу.

4 ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Режим работы — клиентское веб-приложение, запускаемое в браузере. Для работы требуется устройство с поддержкой веб-технологий, наличие установленного Node.js (версии 18 и выше) и доступ к серверу базы активных знаний. Рекомендуется использовать систему с оперативной памятью от 4 ГБ и браузером (Chrome, Firefox).

5 ВЫЗОВ И ЗАГРУЗКА

Запуск программы осуществляется путём набора команды в терминале npm start.

6 ВХОДНЫЕ ДАННЫЕ

Входными данными для подсистемы визуального представления являются:

- Файл визуального описания в формате JAZ, содержащий информацию о вычислительной модели, её переменных, перспективах отображения, координатах, типах данных и дополнительных параметрах визуализации. Данный файл загружается автоматически при инициализации интерфейса и используется для построения интерактивной визуальной области.
- Данные, вводимые пользователем — значения переменных, выбранные входные и выходные переменные. Ввод осуществляется через соответствующие поля в левой панели интерфейса, тип которых определяется автоматически на основе описания переменной в модели.

7 ВЫХОДНЫЕ ДАННЫЕ

Выходными данными являются:

- Визуальное отображение модели, сформированное на основе координат и параметров, заданных в JAZ-файле. Веб-интерфейс отображает переменные и фоновые изображения.
- Результаты вычислений, полученные от сервера базы активных знаний. После завершения расчётов значения выходных переменных отображаются в интерфейсе в виде значений переменных и дополнительной информации в виде графиков или формульных выражений в формате LaTeX.