Федеральное агентство по рыболовству
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Астраханский государственный технический университет»
Система менеджмента качества в области образования, воспитания, науки и инноваций сертифицирована DQS по международному стандарту ISO 9001:2015

Институт Информационных технологий и коммуникаций

Направление подготовки 09.03.04 Программная инженерия

Профиль Разработка программно-информационных систем

Кафедра Автоматизированные системы обработки информации и управления

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Интегрированная среда для обучения алгоритмизации DRAKON IDE

(название темы)

Работа выполнена обучающимся группы ДИПРб-41
Самарским Владиславом Валерьевичем
(Фамилия Имя Отчество)

Руководитель ВКР
к.т.н. доцент Лаптев Валерий Викторович
(ученая степень, ученое звание, Фамилия Имя Отчество)

Консультант по проектной документации ст. преп. Куркурин Н.Д.
(название раздела, ученая степень, ученое звание, Фамилия И.О., подпись)

Нормоконтролер ст. преп. Толасова В.В.
(ученая степень, ученое звание, Фамилия И.О., подпись)

Допущена к защите «___» _____ 20 ___г.

Заведующий кафедрой _____Хоменко Т.В.

Астрахань 2021

Федеральное агентство по рыболовству
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Астраханский государственный технический университет»
Система менеджмента качества в области образования, воспитания, науки и инноваций сертифицирована DQS по международному стандарту ISO 9001:2015

УТВЕРЖДАЮ
Заведующий кафедрой
Автоматизированные системы
обработки информации и управления
д.т.н. профессор Т.В. Хоменко

_____
(подпись)
«__»_____20 ___г.

З А Д А Н И Е
на выполнение выпускной квалификационной работы

Обучающемуся
учебной группы ДИПРб-41 института Информационных технологий и коммуникаций
(институт/факультет)
Самарскому Владиславу Валерьевичу
(фамилия, имя, отчество - полностью)

ТЕМА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
Интегрированная среда для обучения алгоритмизации DRAKON IDE

Тема ВКР сформулирована в соответствии с запросом ИиТИК АСОИУ АГТУ
(стратегический партнер, работодатель, подразделение (службы) АГТУ и т.д.)

Утверждена распоряжением директора института Информационных технологий и коммуникаций (распоряжение от « 07 » декабря 2020 г. № 49)

РУКОВОДИТЕЛЬ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
кандидат технических наук, доцент Лаптев Валерий Викторович
(фамилия, имя, отчество – полностью, ученая степень, ученое звание)

Утвержден распоряжением директора института Информационных технологий и коммуникаций (распоряжение от « 07 » декабря 2020 г. № 49)

Представление выпускной
квалификационной работы на кафедру «___» _____ 20___ г.

Дата защиты «___» _____20 ___ г.

Целевая установка и исходные данные:
Разработать информационную систему, позволяющую пользователю разрабатывать алгоритмы на языке ДРАКОН. Обеспечить автоматизацию процесса обучения основам построения алгоритмов. Предоставить пользователю возможность транслировать ДРАКОН-схемы в некоторый современный и популярный текстовый язык программирования.
Информационная система должна быть независимой от ОС и поддерживать множество пользователей.

№ п/п	Перечень чертежей, подлежащих разработке	Формат, количество
1	Модель икон языка ДРАКОН	А4, 1
2	Модель данных ДРАКОН-схемы	А4, 1
3	Роли в системе и варианты использования	А4, 1
4	Представление сущностей в системе	А4, 1
5	Физическая модель данных	А4, 1
6	Диаграмма компонентов системы	А4, 1

Руководитель выпускной квалификационной работы: _____
(подпись)

№ п/п	Содержание расчетно-пояснительной записки (перечень вопросов, подлежащих разработке)	Консультанты
1	Разработка технической документации	Толасова В.В.
2	Изучение механизмов графического представления данных в компьютере	Куркурин Н.Д.
3	Проектирование компонентов системы	Филоненко А.В.
4	Реализация клиент-серверной архитектуры	Морозов А.В.
5	Разработка программного продукта	Морозов А.В.
6	Тестирование программного продукта	Филоненко А.В.
7	Проектная документация	Филоненко А.В.

Руководитель выпускной квалификационной работы: _____
(подпись)

Основная рекомендуемая литература

1. Пароджанов В.Д. Язык ДРАКОН, краткое описание. 4-е изд. – М.: Дело, 2002 – 124 с.
2. Пароджанов В.Д. Алгоритмы и жизнеритмы на языке ДРАКОН. Разработка алгоритмов. Безошибочные алгоритмы. — М., 2019. — 374 с.
3. Леоненков, А.В. Самоучитель UML. – 2-е изд., перераб. и доп. – СПб., БХВ-Петербург, 2004. – 432 с.

Задание принял к исполнению «____» _____ 20 ___г.

Обучающийся _____
(подпись)

УТВЕРЖДАЮ

Заведующий кафедрой

д.т.н., профессор Т.В. Хоменко

_____
(подпись)

«___»_____20 ___г.

К заданию на выпускную квалификационную работу

К А Л Е Н Д А Р Н Ы Й Г Р А Ф И К

подготовки выпускной квалификационной работы

№ п/п	Разделы, темы и их содержание, графический материал	По плану		По факту		Отметка руководителя о выполнении
		Дата	Объем в %	Дата	Объем в %	
1	Составление и согласование технического задания на выпускную квалификационную работу	1.12.2020	3			
2	Обследование предметной области	01.02.2021	20			
3	Разработка технического проекта	01.03.2021	30			
4	Получение задания на преддипломную практику	30.04.2021	32			
5	Защита отчёта по преддипломной практике	31.05.2021	60			
6	Подготовка к защите ВКР	17.06.2021 – 28.06.2021	100			
7	Предварительная защита ВКР	14.06.2021– 18.06.2021	100			
8	Защита ВКР	28.06.2021 – 03.07.2021	100			

Руководитель выпускной квалификационной работы: _____
(подпись)

Обучающийся: _____
(подпись)

«____» _____ 20 ___г

РЕФЕРАТ

с. 89, рис. 47, табл. 33, прил. 4

ДРАКОН-схема, обучение, преподаватель, трансляция в текстовый язык программирования, платформонезависимость, онлайн система.

Проведено изучение языка ДРАКОН, способов его представления и написания алгоритмов на нём. Проанализированы существующие реализации систем по созданию алгоритмов на языке ДРАКОН, выявлены плюсы и недостатки существующих решений.

Выделены классы пользователей системы и целевая группа, для которой система разрабатывается. Выявлены и учтены требования безопасности к многопользовательской системе.

Проведен анализ информационной ценности алгоритмов на языке ДРАКОН. Учтены требования повышения практической пользы ДРАКОН-схем.

Разработана информационная многопользовательская онлайн система с распределением ролей для возможности обучения алгоритмизации через проектирование алгоритмов с помощью сильно структурированного языка ДРАКОН.

Система рассчитана на использование в учебных заведениях с численностью до 10000 человек. Система обеспечивает эффективное взаимодействие между преподавателями и учениками.

Система проходит внутренний этап тестирования, к 1 сентября 2021 планируется начать этап закрытого тестирования на студентах 1 курса ИиТИК АГТУ.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Кафедра «АСОИУ» Астраханского государственного технического университета производит подготовку студентов IT-специальностей. Абитуриенты, ежегодно поступающие в университет, обладают разным уровнем подготовки к моменту началу обучения. Задача первого года обучения - выровнять знания учеников. Для достижения данной цели используется среда Semantic IDE, которая обучает студентов правилам хорошего форматирования кода, позволяет плавно войти в программу обучения и подготовиться к более сложным языкам программирования.

Данный программный комплекс не позволяет решить одну из главных проблем современного процесса обучения программированию – обучение правильному выстраиванию логики поведения программы. Молодые программисты часто совершают логические ошибки, которые нарушают правильное поведение системы. На выходе получается продукт без синтаксических ошибок, но с практической пользой равной нулю.

Мышление человека работает с помощью представления образов. Графическую информацию мозг воспринимает в несколько раз быстрее текстовой, что позволяет потребить и проанализировать больше информации за единицу времени. Соответственно возникает резонный вопрос, а что произойдёт, если заменить текстовое программирование визуальным? Попыткой ответить на этот вопрос стало представление языка ДРАКОН – Дружелюбного Русского Алгоритмического Языка, Который Обеспечивает Наглядность. Суть его в представлении программ в виде блок-схем с определенной дисциплиной построений, в которых вся функциональная часть спрятана «под капотом», внутри функциональных блоков. Снаружи представлено лишь описание процессов формальным языком. Это позволяет существенно упростить процесс понимания алгоритма работы того или иного процесса.

Программно-аппаратный комплекс, позволяющий разрабатывать программные продукты с помощью языка ДРАКОН, позволит студентам правильно подходить к проектированию алгоритмов, что сократит время, затрачиваемое на разработку и тестирование.

Конвертация в высокоуровневые языки программирования позволит существенно популяризировать данное решение из-за возрастания практической

пользы программно-аппаратного комплекса и избавит пользователей системы от необходимости переносить алгоритм на другой язык программирования вручную.

Существует ряд редакторов, в том числе WEB, такие, как «Dragon.tech», «ИС ДРАКОН», «DRAKON Editor». Все они имеют как преимущества, так и недостатки. Ни один из вышеперечисленных не обладает одновременно следующими важными качествами:

- платформонезависимость. Программный продукт должен быть кроссплатформенным;
- нетребовательность к ресурсам. Программный продукт должен рационально использовать ресурсу компьютера;
- конвертация в язык программирования высокого уровня. редакторы с открытым исходным кодом не обладает таким качеством.

Разработка и внедрение подобной информационной системы с возможностью создавать ДРАКОН-схемы, с последующей конвертацией в высокоуровневый язык программирования, на кафедру «АСОИУ» позволит существенно снизить нагрузку на преподавателя путём автоматизации процесса обучения студентов основам алгоритмизации. Снизит время, затрачиваемое на проектирование, и положительно скажется на поиске ошибок в логике поведения разрабатываемых программ.

Целью данной выпускной квалификационной работы является разработка информационной системы для автоматизации процесса обучения основам алгоритмизации с помощью языка ДРАКОН с возможностью конвертации в текстовый язык программирования.

Назначение: автоматизация процесса обучение пользователей основам алгоритмизации и автоматизации процессов, снижение нагрузки на преподавателей на кафедре «АСОИУ», повышение практической пользы языка ДРАКОН как инструментария автоматизации процессов.

1 ТЕХНИЧЕСКИЙ ПРОЕКТ

1.1 Анализ предметной области

1.1.1 Язык ДРАКОН

Язык ДРАКОН – дружелюбный российский алгоритмический язык, который обеспечивает наглядность. Дракон схема – диаграмма, состоящая из специальных икон и соединений между ними, которые формируются по определенным правилам, которые, в совокупности, формируют конечный алгоритм.

В первую очередь, язык ДРАКОН ориентирован на лиц, не владеющих языком программирования, но нуждающихся в программном обеспечении для удовлетворения задач в их предметной области.

Основа языка ДРАКОН – иконы и макроиконы. Икона – минимальная единица в алгоритме, представляет какое-либо действие, событие. Объединяясь в группы иконы формируют макроиконы. Иконы также часто называют операторами, далее допускаются оба варианта наименования.

Любая дракон-схема имеет начало и конец. Они представлены в виде специальных икон, которые отличаются от всех остальных. Икона «Заголовок», изображенная на рисунке 1.1, указывает на начало алгоритма, и содержит в себе его название.

Рисунок 1.1 – Икона «Заголовок»

Икона «Конец», показанная на рисунке 1.2, обозначает точку завершения алгоритма. Иконы «Заголовок» и «Конец» не могут существовать в разрыве друг от друга и представляют собой неделимую основу для будущей последовательности действий.

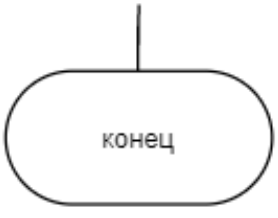

Рисунок 1.2 – Икона «Конец»

Икона «Действие» содержит в себе некую операцию, проводимую над доступными данными, которая производится безусловно. Представление такой иконы показано на рисунке 1.3.

Рисунок 1.3 – Икона «Действие»

Икона «Вопрос» представляет из себя условную конструкции, в которой есть два варианта исхода: положительный и отрицательный, в зависимости от результата проверки условия, хранящегося в иконе «Вопрос». Результаты проверки условия можно сопоставить с ответами на вопрос «Да» и «Нет». В языке ДРАКОН такая икона обозначается шестиугольником, пример показан на рисунке 1.4.

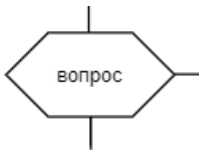

Рисунок 1.4 – Икона «Вопрос»

Для иконы «Вопрос» важную роль играет направление альтернативной ветки. Если она указывает на ту часть алгоритма, которая находится ниже иконы с условием, то для алгоритма она является обычным условным оператором. А если указывает на часть алгоритма выше её, то икона «Вопрос» представляет из себя цикл.

Для графического представления этих отличий для циклов указывается стрелка - направление, пример представлен на рисунке 1.5. Такие конструкции называют макроиконами, так-как образованы объединением икон в группы для решения одной общей задачи.

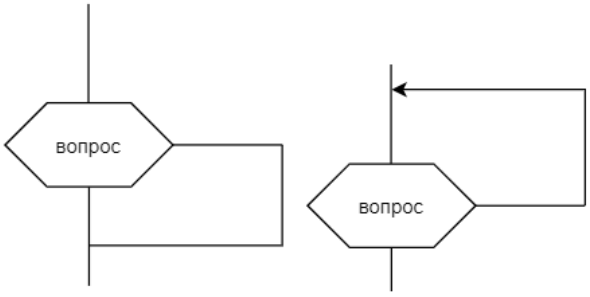

Рисунок 1.5 – Макроиконы «Вопрос» и «Цикл»

Другой важной иконой является «Вариант». Её представление показано на рисунке 1.6. Она хранит схожее с иконой «Вопрос» условие, однако решение принимается исходя не из положительности проверки условия, а из точного соответствия конкретному значению.

Рисунок 1.6 – Икона «Вариант»

Каждое такое значение должно быть учтено в схеме, для этого в языке ДРАКОН есть специальная икона «Выбор», в которой описывается точное состояние значения для иконы «Вариант». Графическое представление этой иконы показано на рисунке 1.7.

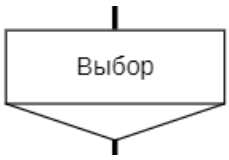

Рисунок 1.7 – Икона «Выбор»

Икон «Выбор» может быть несколько, в зависимости от потребностей разработчика алгоритма, ограничений на их количество условно нет.

Каждый «Вариант» формирует макроикону «Ветку» выполнения, каждая ветка выполняется независимо друг от друга и не может выполняться одновременно с другой. Таким образом разработчик может описывать последовательность действий для каждого состояния системы. Графическое представление макроиконы «Ветка» показано на рисунке 1.8.

Рисунок 1.8 – Макроикона «Ветка»

Все макроиконы «Ветки», содержащие иконы «Выбор», вместе с привязанной к ним иконой «Вариант», образуют макроикону «Варианта». Конец всех «Веток»

соединяется в единый поток, и управление переходит следующей иконе. Графическое представление макроиконы «Вариант» показано на рисунке 1.9.

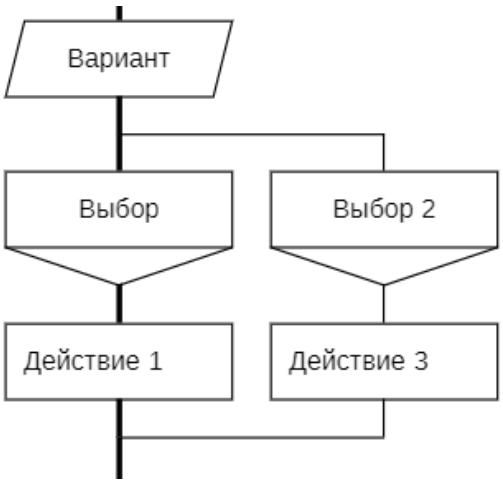

Рисунок 1.9 – Макроикона «Вариант»

Икона «Ввод» отвечает за получение входных данных от устройств ввода. Икона «Вывод», соответственно, за вывод информации в консоль или на экран, в зависимости от реализации системы. Обе иконы представлены на рисунке 1.10.

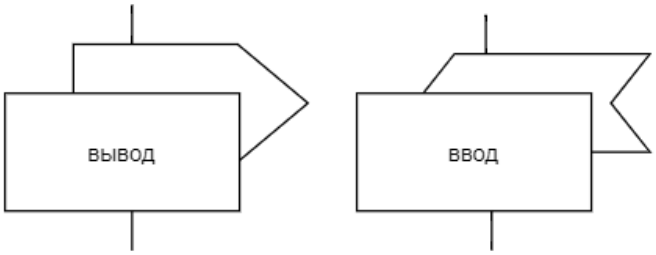

Рисунок 1.10 – Иконы «Вывод» и «Ввод»

Икона «Комментарий» не выполняет функциональной роли, только благоприятствует пониманию алгоритма. В ней пользователь описывает состояние системы, данных в ней или любую иную произвольную информацию, которую пользователь считает необходимой. Вид иконы «Комментарий» представлен на рисунке 1.11.

Рисунок 1.11 – Икона «Комментарий»

Икона «Пауза», как следует из её названия, отвечает за установку задержки перед выполнением следующей инструкции. Как правило задержку указывают в миллисекундах. Вид данной иконы представлен на рисунке 1.12.

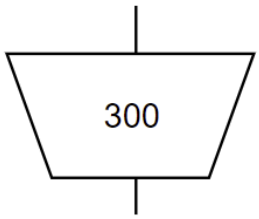

Рисунок 1.12 – Икона «Пауза»

Особой иконой является «Имя ветки». Она содержит произвольное количество икон внутри себя, которые описывают весь или часть какого-либо алгоритма. В некоторых случаях, например в макроиконе «Вопрос» и «Вариант», ветки не обозначаются графически, однако содержатся в каждом «Выборе».

Но в отдельных случаях, которые описаны в пункте 1.1.2, у иконы «Ветка» есть графическое представление, оно также представлено на рисунке 1.13. Внутри данная икона содержит формальное описание содержимого в ней или просто название алгоритма внутри себя. Важно, чтобы это название было уникальным для недопущения разночтения.

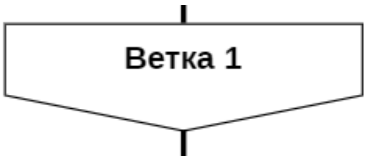

Рисунок 1.13 – Икона «Имя ветки»

Каждая икона «Имя ветки» явно или неявно сопровождается иконой «Адрес», которая содержит название ветки, в который переходит процесс управления. Явно данная икона отображается в случае, если указывает на другую ветку, в таком случае она отображается перевернутой иконой «Имя ветки», её представление показано на рисунке 1.14.

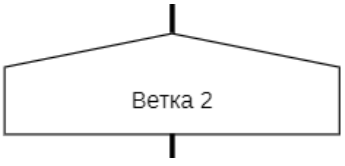

Рисунок 1.14 – Икона «Адрес»

Если «Адрес» указывает на «Конец» ДРАКОН-схемы, то в таком случае икона явно не отображается. А икона «Имя ветки» графически указывает на конец схемы, как это показано на рисунке 1.15.

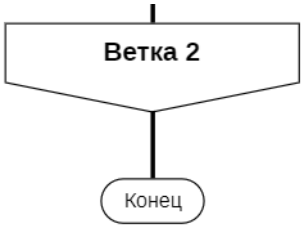

Рисунок 1.15 – Неявная икона «Адрес» в ветке

В уникальном представлении каждой иконы и заключается принцип наглядности: каждый тип иконы отличается друг от друга. На вопрос об отличии дракон-схемы от обычного графического представления алгоритма, однозначно можно ответить следующим образом: язык ДРАКОН дополняет стандартное графическое представление алгоритмов и устанавливает строгие правила взаимодействия икон друг с другом, что позволяет упростить написание программ, улучшить наглядность алгоритмов, а также даёт возможность реализовать транслятор удовлетворимой сложности для представления дракон-схемы в виде алгоритма на языке программирования высокого уровня.

1.1.2 Правила построения ДРАКОН-схем

Самым простом способом представления ДРАКОН-схемы является «Примитив» - простая конструкция языка ДРАКОН, которая позволяет описать алгоритм в виде последовательности икон. Пример «Примитива» представлен на рисунке 1.16.

Рисунок 1.16 – Пример «Примитива»

Каждая икона в «Примитиве» следует одна за другой, формируя линию. Макроиконы, такие как «Вопрос» или «Вариант», могут формировать ответвления вправо, однако все они соединяются с главной вертикалью в конце своих веток.

Вторым способом представления ДРАКОН-схем является конструкция «Силуэт». Её графическое представление показано на рисунке 1.17. Такая конструкция представляет из себя набор «Примитивов», каждый из которых содержит ссылку на следующий для выполнения набор инструкций, или на конец алгоритма. Каждый «Примитив» в «Силуэте» выполняется слева направо, если одна из веток выполнения ссылается на другую, то её необходимо размещать левее той, управление к которой перейдёт после.

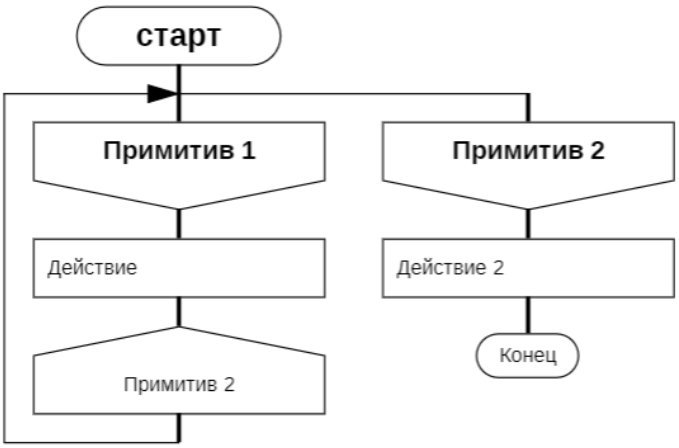

Рисунок 1.17 – Пример «Силуэта»

С точки зрения поведения «Силуэт» ничем не отличается от «Примитива». Главная задача силуэта - упростить представление последовательного алгоритма. Каждый «Примитив» можно преобразовать к «Силуэту», справедливо и обратное утверждение.

Одним из главных правил построения схем в языке ДРАКОН является «Шампур». Оно гласит: ДРАКОН-схема состоит из «Шампур-блоков», имеющие один вход сверху и один выход снизу, которые содержат одну или несколько икон. Причём точки входа и выхода располагаются на одной вертикали, ровно как и иконы внутри.

Данное правило должно реализовываться программно. Последовательность икон визуально должна ощущаться таковой, для этого их необходимо размещать в одной вертикальной плоскости.

Чтобы исключить возможность для пользователя нарушать основные принципы построения ДРАКОН-схем.

Пример нарушения правила «Шампур» представлен на рисунке 1.18.

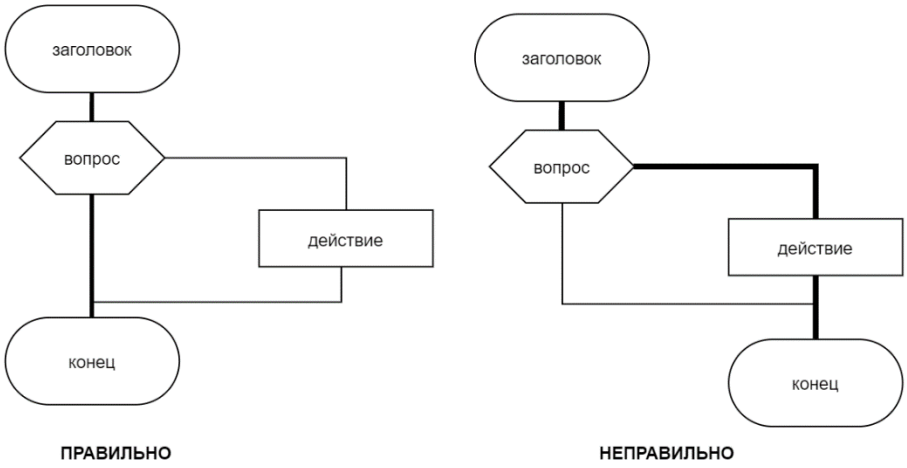

Рисунок 1.18 – Графическое представление правила «Шампура»

Второе важное правило, которое вытекает из первого – главный маршрут алгоритма всегда должен идти по «Шампуру». Применение данного правила на практике представлено на рисунке 1.19.

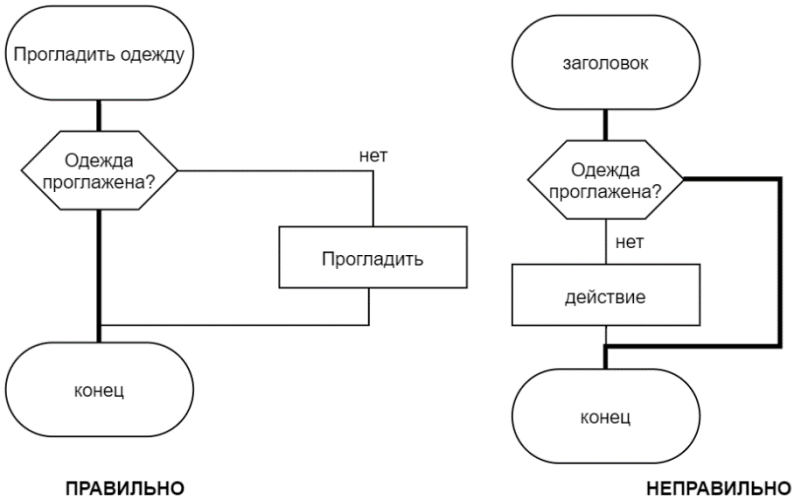

Рисунок 1.19 – Правило главный маршрут по «Шампуру»

Под главным маршрутом, в данном случае, подразумевается путь, который отражает наиболее позитивный исход, достижение результата по наиболее благоприятному пути. Это правило регулируется привязкой координат начала и конца ДРАКОН схемы. Но построение правильной бизнес-логикой остаётся за пользователем.

Третье правило: побочные маршруты необходимо рисовать всегда справа от основного. Чем более неблагоприятное развитие события, тем правее оно описано.

Это необходимо для улучшения читаемости алгоритма. Также его можно описать другими словами: «Чем правее, тем хуже». Пример реализации такого правила представлен на рисунке 1.20.

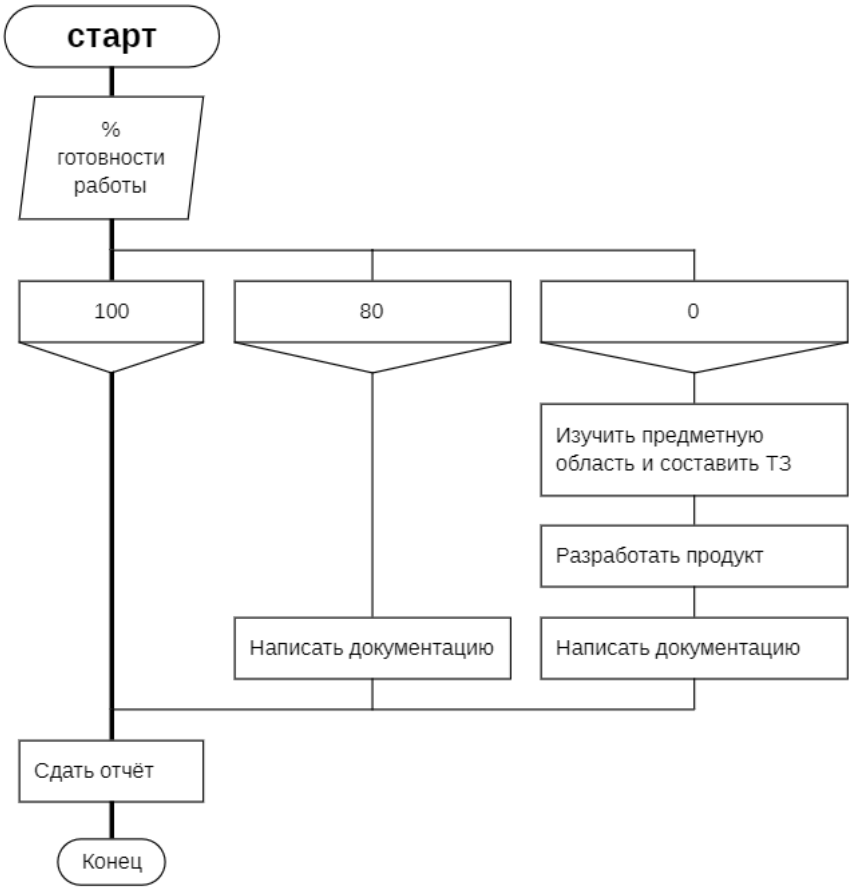

Рисунок 1.20 – Графическое представление правила: побочные маршруты справа

Программно данное правило реализуется тем, что альтернативные ветки для икон вопроса отрисовываются справа от основного маршрута. Но это не означает, что этого правила необходимо строго придерживаться. Иногда оно принципиально неприменимо из-за совершенно иной логики программы, в котором отсутствует понятие «хуже». Тогда следует изменить логику работы программы, чтобы она удовлетворяла иному условию, по выбору пользователя.

Самое главное правило дракон-схем - пересечение линий запрещены. Любые две иконы должны быть соединены таким образом, чтобы не пересекать любые другие, соединяющие иконки. Для этого существует целый ряд правил для отрисовки элементов дракон-схем, которые хоть и не всегда способны защитить пользователя от нарушения этого правила, но свести такие ситуации к минимуму.

Стоит отметить, что пересечение линий всегда можно преобразовать к виду, в котором они пересекаться не будут.

1.2 Технология обработки информации

Разрабатываемая система представляет собой многопользовательское онлайн приложение. В системе требуется реализация идентификации пользователя, сбора, хранения и предоставления ДРАКОН-схем. В информационной системе выделяются следующие роли:

- пользователь – получает доступ к своим схемам с целью разработки алгоритмов на языке ДРАКОН;
- куратор – получает доступ к схемам курируемых пользователей с целью осуществления контроля;
- администратор – осуществляет контроль над учётными записями пользователей и их ролями.

Диаграмма вариантов использования представлена на рисунке 1.21.

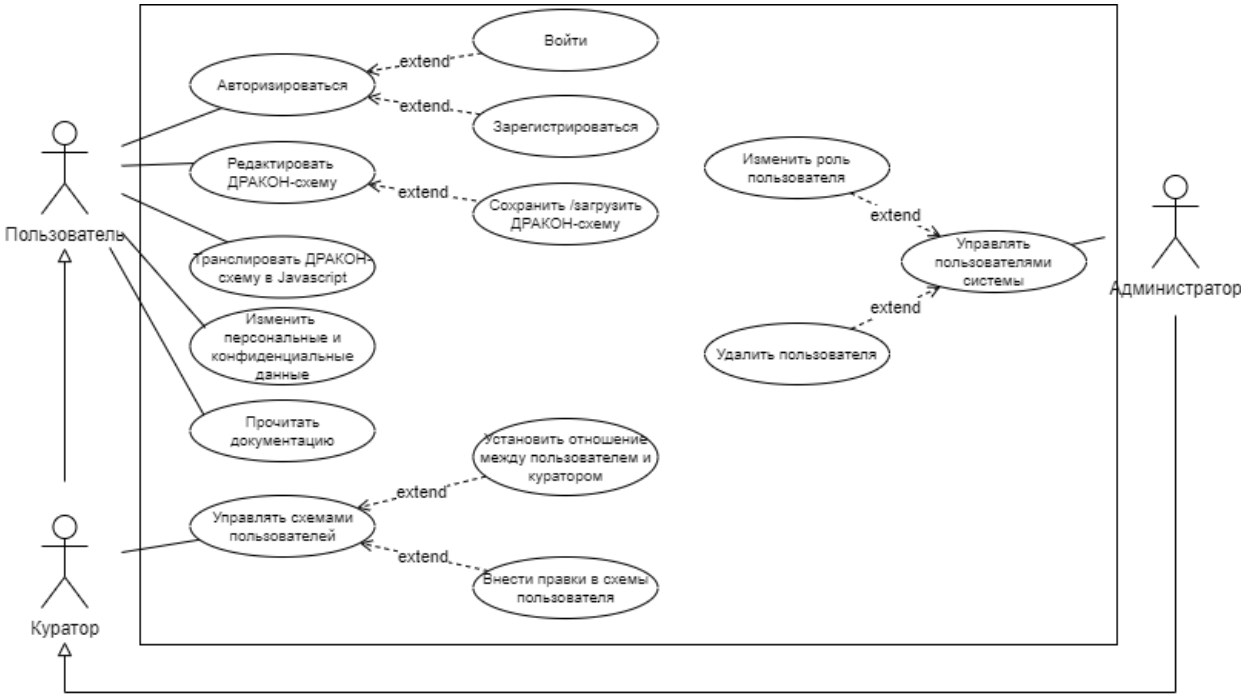

Рисунок 1.21 – Диаграмма вариантов использования

Основным актором в системе является пользователь. Пользователю разрешено управление собственными схемами, своими персональными данные, а также предоставляется возможность осуществлять конвертацию ДРАКОН-схем в JavaScript. Для идентификации используется механизм авторизации, для которой

необходимо предоставить уникальную комбинацию символов для имени пользователя и пароль, который предоставляет защиту данных пользователя от доступа третьих лиц.

Куратор обладает аналогичными с пользователем возможностями по взаимодействию с системой. Куратор может добавлять существующих пользователей к себе в группу, что даёт возможность осуществлять просмотр и редактирование схем других пользователей.

Администратор обладает полным спектром возможностей по взаимодействию с пользователями и схемами, однако основная его задача - контроль над работой системы: администратор выдаёт роли зарегистрированным пользователем, удаляет аккаунты пользователей, которые не удовлетворяют пользовательским требованиям. Например: некорректное имя пользователя, или учётная запись больше не является активной.

1.2.1 Спецификация вариантов использования

Спецификация вариантов использования для роли пользователь:

1. Зарегистрироваться.
   1.1. Краткое описание: пользователь в интерфейсе аутентификации вводит уникальное имя пользователя в системе, придумывает пароль, и нажимает кнопку «Регистрация».
   1.2. Ответ системы: система производит валидацию имени пользователя на уникальность и соответствие пароля минимальным требованиям, в случае успеха регистрирует пользователя в системе и позволяет войти под своими учётными данными.
2. Войти.
   2.1. Краткое описание: пользователь, ранее зарегистрированный в системе, вводит валидную комбинация имени пользователя и пароля и осуществляет вход через нажатие кнопки «Вход» интерфейса аутентификации системы.
   2.2. Ответ системы: система производит валидацию пользователя и выдаёт JWT-token для предоставления сессии пользователю и доступа к редактору ДРАКОН-схем.

3. Редактировать ДРАКОН-схему.
    3.1. Краткое описание: пользователь создаёт, удаляет или изменяет содержимое схемы или её название через интерфейс редактора ДРАКОН-схем.
    3.2. Ответ системы: система регистрирует изменения, произведенные пользователем, и обновляет данные в базе данных редактируемой схемы. В случае удаления схемы, операция осуществляется без возможности восстановления.
4. Транслировать ДРАКОН-схему в JavaScript.
    4.1. Краткое описание: пользователь, предварительно открыв схему, через интерфейс системы выбирает пункт «В JavaScript» или «Сохранить код».
    4.2. Ответ системы: в случае выбора пользователем пункта «Сохранить код» система осуществляет трансляцию текущей открытой схемы в язык JavaScript. После чего генерирует файл на стороне клиента, и, используя браузер, осуществляет сохранение на рабочей станции клиента файла с конвертированной программой в формате «.js». В случае выбора пункта «В JavaScript» система выводит конвертированную схему в специальном компоненте интерфейса редактора.
5. Изменить персональные данные.
    5.1. Краткое описание: оператор выбирает пункт «Изменить имя пользователя» или «Изменить пароль» и вводит обновленные данные для своей учётной записи. Или выбирает пункт «Удалить аккаунт».
    5.2. Ответ системы: система регистрирует изменения в базе данных после проверки на корректность входных данных. В противном случае уведомляет об ошибке.
6. Прочитать документацию.
    6.1. Краткое описание: оператор выбирает пункт в меню интерфейса «Документация».
    6.2. Ответ системы: система отображает основную информацию об иконах и макроиконах системы вместе с иллюстрациями и способами трансляции в язык JavaScript.

Спецификация вариантов использования для роли куратор:

1. Установить отношение между пользователем и куратором.
   1.1. Краткое описание: куратор в интерфейсе администрирования выбирает из списка или вводит имя пользователя. Для выбранного пользователя ему доступны следующие действия «Добавить/Удалить в список курируемых пользователей» и «Предоставить описание отношению между куратором и пользователем».
   1.2. Ответ системы: Система добавляет или исключает пользователя из списка курируемых для текущего куратора. Изменяет описание отношения в базе данных в ответ на ввод куратора.
2. Внести правки в схемы пользователя.
   2.1. Краткое описание: куратор в интерфейсе редактора в специальном выпадающем списке вводит имя курируемого пользователя, выбирает доступную схему из списка. После производит изменения в схеме, чтобы курируемый мог учесть правки.
   2.2. Ответ системы: система загружает схемы выбранного пользователя при наличии достаточных привилегий со стороны куратора. Регистрирует и сохраняет изменения схема в базе данных, включая информацию о времени и кураторе, который произвёл изменения.

Спецификация вариантов использования роли администратор:

1. Изменить роль пользователя.
   1.1. Краткое описание: администратор в интерфейсе администрирования выбирает из списка или вводит имя пользователя. Для выбранного пользователя «Администратор» выбирает желаемую роль и пункт «Установить роль»
   1.2. Ответ системы: Система проверяет привилегии пользователя, который производит изменение роли. В случае удовлетворительной проверки осуществляет обновление роли пользователя в базе данных.
2. Удалить пользователя
   2.1. Краткое описание: администратор в интерфейсе администрирования выбирает из списка или вводит имя пользователя. Для выбранного пользователя «Администратор» выбирает пункт «Удалить пользователя».

2.2. Ответ системы: система проверяет привилегии пользователя. В случае удовлетворительной проверки осуществляет удаление пользователя из базы данных, всех его схем и отношений.

1.2.2 Защита персональных данных

Одной из основных задач при проектирование многопользовательской информационной системы является обеспечение безопасности конфиденциальных данных.

Система осуществляет сбор и хранение логинов пользователей, их ДРАКОН-схем, а также почты. Все эти данные необходимо защитить от неавторизованного доступа и злоумышленников.

Каждому пользователю при регистрации необходимо указать пароль. На него накладывается ряд ограничений:

- Длина пароля не менее 6 символов;
- Пароль может состоять только из латинских символов, а также знака «_» и цифр;
- Пароль обязательно должен содержать одну заглавную букву или цифру;
- Не допускаются пароли, состоящие только из прописных или заглавных символов.

Система не осуществляет хранение паролей в «чистом» виде. Предварительно все пароли шифруются с помощью функции bcrypt.

Bcrypt является односторонней адаптивной функцией криптошифрования ключей безопасности переменной длинны. Эффективность алгоритма обуславливается тем, что время работы функции можно изменять с помощью коэффициента «salt», а сам ключ при каждой операции шифрования для одинаковой последовательности символов отличается. Для проверки пароля на корректность необходимо осуществить шифрование с той же «солью», комбинацией символов пароля и сравнить с изначально зашифрованной последовательностью.

Пароль пользователя при регистрации доставляется на сервер, шифруется через функцию bcrypt и сохраняется в базе данных в связке с уникальным именем пользователя. В последующем обратно дешифровать ключ невозможно.

Для аутентификации пользователя используется схема Http Basic (RFC 7617). По которой для входа в систему пользователю необходимо отправить связку из логина и пароля.

Если система находит пользователя в базе данных, и пароли совпадают, то пользователю возвращается специальный JWT Access токен для доступа к защищенным путям приложения.

Авторизация осуществляется по схеме Http Bearer (RFC 6750): в случае успешной авторизации пользователю выдаётся JWT токен, в котором зашифрованы его логин и роль в системе.

Все URL-пути приложения, которые осуществляют выдачу, обновление или удаление конфиденциальных данных защищены. Для выполнения любого запроса должен выполняться уровень привилегий пользователя.

Пользователь системы не может взаимодействовать со схемами других пользователей, изменять и получать конфиденциальные данные других пользователей.

Куратор обладает доступом к схемам только курируемых пользователей, не может изменять данные других пользователей.

Администратор получает полный доступ к схемам других пользователей, может удалять и изменять роль других учётных записей, но не имеет доступа к почте и паролю других пользователей системы.

1.2.3 Способы взаимодействия с ДРАКОН-схемой

Минимальной схемой в примитиве является блок «Заголовок-Конец». Пользователь не имеет возможности удаления этого блока. Графически связь представляется отрезком между двумя иконами, как представлено на рисунке 1.22.

Рисунок 1.22 – ДРАКОН-схема «Примитив»

Все последующие иконы вставляются между ними. Количество их неограниченно. Местом для вставки является центр линии между двумя иконами. Область, в которую можно вставить икону подсвечивается. Удаление происходит с помощью выбора соответствующего пункта и нажатием на саму икону, удаляемые иконы также подсвечиваются.

Графически отображаемая икона ограничена размером, и не все данные можно уместить в данном блоке. Одним из вариантов решения этой проблемы – увеличение размера самой иконы, но тут существует проблема снижения читаемости схемы. Одним из оптимальных способов решения данной проблемы – представление полного содержимого иконы в отдельном элементе интерфейса.

В теле самой иконы описываются действия, производимые в иконе, для описания комментария необходимо воспользоваться специальной иконой «Комментарий».

1.2.4 Методы представления ДРАКОН икон

Для проектируемой информационной системы необходимо реализовать представление ДРАКОН-схем в памяти программы. При этом контейнер должен обладать рядом качеств: предоставлять произвольный доступ к иконам, при этом обеспечивать последовательный доступ к иконам ДРАКОН-схемы для просмотра схемы от начала до конца. Учитывая наложенные ограничения, были разработаны классы, представленные на диаграмме классов в приложении 1.

Базовый абстрактный класс иконы устанавливает правила для всех наследуемых от него икон. Все иконы содержат информацию о своём типе, данные, которые они обязаны хранить в соответствии со своим типом и шаблон поведения на получение внешних данных. Помимо этого, каждая икона хранит информацию о своём родителе, макроиконе её содержащую. Все иконы внутри схемы имеют родителя.

Для проектируемого представления ДРАКОН-схем были разработаны следующие иконы:

- заголовок;
- конец;
- примитив;
- имя ветки;
- адрес;
- силуэт;
- действие;
- вопрос;
- ветка;
- цикл;

- вариант;
- выбор;
- разделитель;
- комментарий;
- пауза;
- ввод;
- вывод.

Каждая икона, наследуется от базового класса иконы, получая базовые поля для всех типов икон и методы взаимодействия друг с другом. Макроиконы содержат собственные дополнительные поля и правила поведения, которые используются с целью установки связи между другими иконами.

Каждая икона хранит массив вложенных в неё операторов, операторами могут служить как простые, так и макроиконы. Детали содержимого вложенных икон родительской неизвестны.

Вложенность икон удобно рассматривать как реализацию блочной видимости. Каждая родительская икона знает о своих вложенных операторах. В свою очередь о самой макроиконе знает информацию та икона, которая содержит её.

Для получения доступа к любой из икон, которые содержатся внутри другой, необходимо или пройти все уровни вложенности, или обратиться с помощью ассоциативного доступа по уникальному идентификатору иконы. Таким образом обеспечивается оптимальное хранение и представление схем. Любой простой оператор содержит пустой массив: он не может содержать вложенных инструкций языка ДРАКОН.

Некоторые иконы не имеют графического представления. К таким относится частный случай иконы «Ветка». Если она представляет собой ветку «Выбор» или ветку условной конструкции, то графически она никак не отображается, кроме как с помощью связи между условным оператором и иконой «Разделитель». Которая также не имеет собственного графического представления, но служит для сбора всех веток внутри макроиконы.

Икона «Вопрос» содержит массив указателей на основную и альтернативную ветку выполнения. После веток располагается специальная икона «Разделитель», которая обозначает границу условной конструкции. Каждая ветка представляет из

себя отдельную макроикону «Ветка», внутри которых содержится последовательность икон.

Каждая «Ветка» условно независима и является подобием «Примитива» по своему поведению, в неё так-же можно вставлять произвольное количество икон. Они формирует свою собственную последовательность операндов, правила отображения ДРАКОН-схем применяются точно такие же, разница заключается в том, что, с точки зрения языка ДРАКОН, эти две конструкции выполняют разные задачи. «Примитивы» в совокупности формируют «Силуэт», а «Ветки» являются содержимым различных макроикон условных конструкций. Реализация макроиконы «Вопрос» продемонстрирована на рисунке 1.23

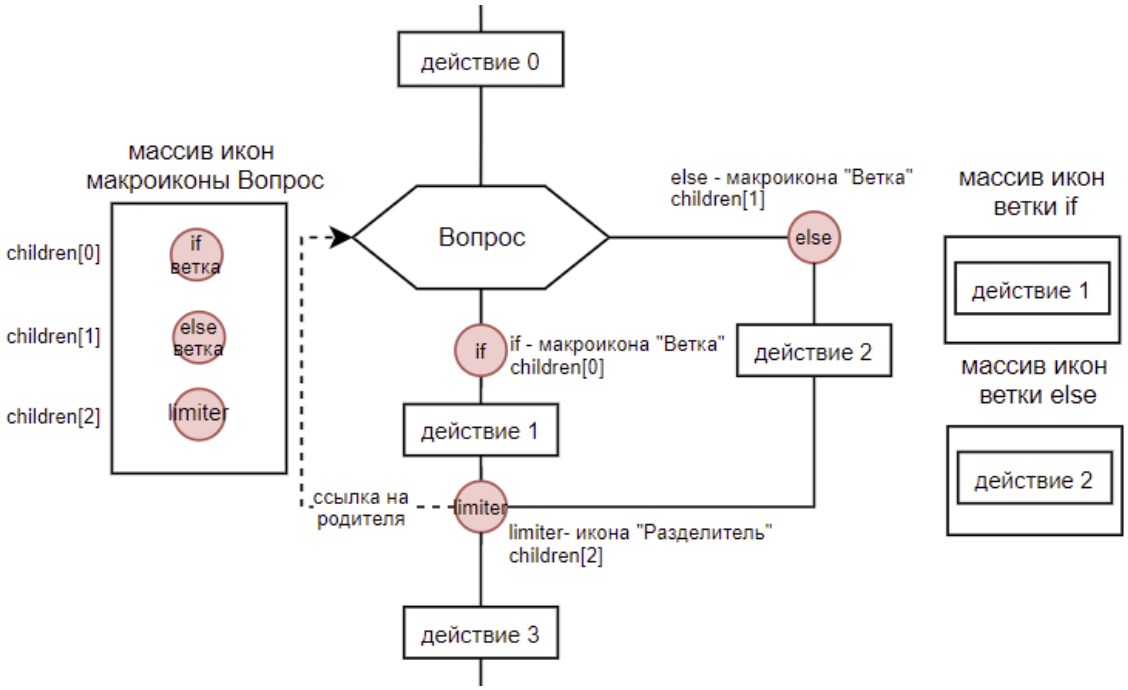

Рисунок 1.23 – Представление макроиконы «Вопрос»

Икона «Цикл» графически отображается в точности как «Вопрос» (см. рис 1.5), однако макроикона отличается направлением стрелки. Управление передаётся обратно, если условие не было выполнено.

«Цикл» содержит в себе обязательно одну «Ветку», в которой описывается набор инструкций, которые будут выполнены для попытки удовлетворения условию. Эта ветка является точкой входа в макроикону. Точкой выхода является сама икона «Цикл».

Вторая «Ветка» отсутствует, вместо неё икона цикла указывает на свою единственную ветку и на инструкцию, которая является внешней по отношению к ней (которая следует сразу за макроиконой «Цикл»).

Внешняя по отношению к циклу макроикона не знает подробностей реализации макроиконы «Цикл». Но в точке входа в неё автоматически считывается её внутреннее состояние, где доступна «Ветка» цикла и её содержимое.

На рисунке 1.24 показано представление макроиконы «Цикл».

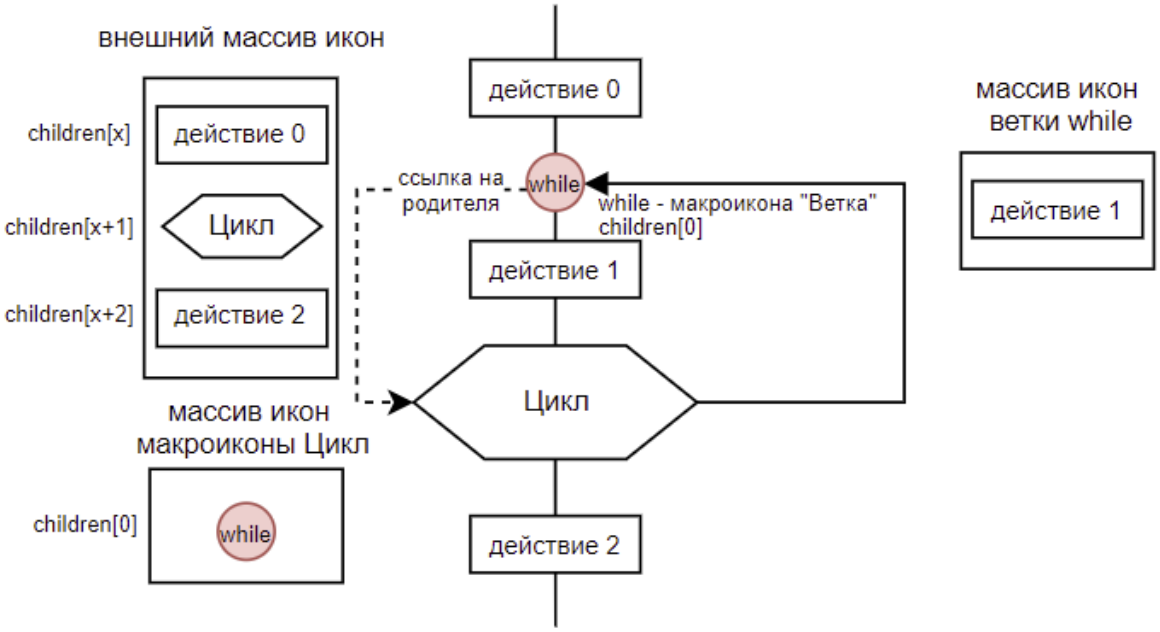

Рисунок 1.24 – Представление макроиконы «Цикл»

Макроикона «Вариант» подобно иконе «Вопрос» содержит несколько веток внутри себя. Но в отличии от неё количество этих веток условно неограниченно. В иконе «Вариант» указываются данные, состояние которых изменяется во времени, а сами состояния можно однозначно описать.

Каждая ветка именуются отлично друг от друга и располагается слева направо в зависимости от положения в массиве макроиконы «Вариант». Внутри каждой ветки свой набор икон. «Разделитель» хранит ссылку на икону «Варианта».

Сам же он располагается последним элементом в списке дочерних икон «Варианта». Таким образом макроикона «Вариант» и её границы можно однозначно вычислить.

Представление макроиконы «Вариант» показано на рисунке 1.25.

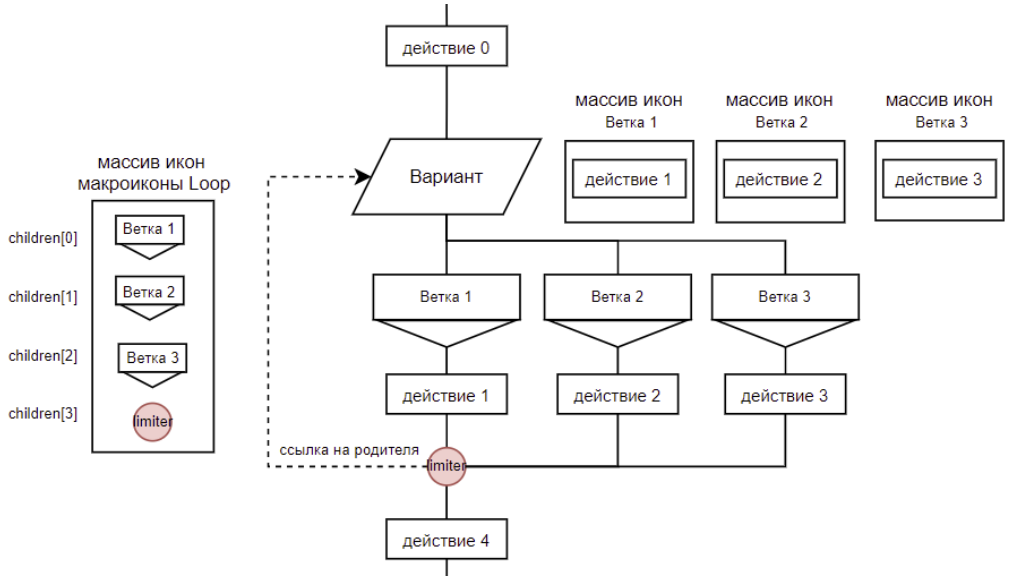

Рисунок 1.25 – Представление макроиконы «Вариант»

1.2.5 Методы представления ДРАКОН-схем

Для однозначной идентификации каждой иконе присваивается уникальный ID, по которому можно обратиться как к самой иконе, так и к её родителю, и всем вложенным в неё иконам. Благодаря уникальным номерам упрощается процесс вставки новых икон, их редактирование и удаление. Вторым преимуществом является упрощение сохранения дракон-схем в памяти компьютера. Каждой иконе соответствует уникальный идентификатор, что позволяет хранить иконы в любом желаемом виде, при условии читаемости данных на основе её ID.

К каждой иконе обеспечивается ассоциативный доступ. Для этого был разработан специальной контейнер - словарь. Для каждой иконы генерируется уникальный номер, который сохраняется во множестве использованных ключей, чтобы исключить вероятность дублирования идентификаторов. Ключ привязывается к иконе, устанавливаются ссылки на соседей, а сама икона сохраняется в словаре по указанному в ней ключу.

Благодаря такой реализации контейнер обладает качествами как контейнеров с ассоциативным доступом, так и с последовательным. Требуется это для того, чтобы при изменении (добавлении или удалении икон/макроикон) не было необходимости в обходе ДРАКОН-схемы.

При инициализации контейнера для ДРАКОН схемы генерируется базовая конструкция, которая включает в себя икону «Схема» и «Примитив», вложенную в

неё. Это является базовой заготовкой для будущего алгоритма. Таким образом пользователь освобождается от лишней работы по созданию каркаса ДРАКОН-схемы.

Главной макроиконой в контейнере является «Схема». Её уникальным отличием является отсутствие родительской иконы. Она содержит название схемы, которое впоследствии выводится в иконе «Заголовок», и массив примитивов.

«Примитив» в схеме может быть один. В таком случае графически примитив отображается с помощью заключения содержимого между иконами «Заголовок» и «Конец». Количество вложенных икон может быть любым, но все иконы примитива располагаются на одной вертикали, только макроиконы могут осуществлять ветвления вправо, однако ход выполнения всё равно возвращается на главную вертикаль.

Пример отображения «Примитива» представлен на рисунке 1.26.

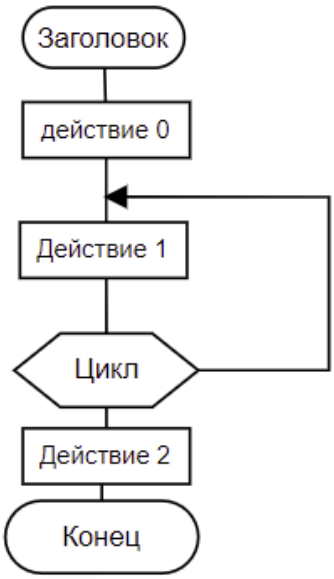

Рисунок 1.26 – Представление «Примитива»

Если отобразить все непечатаемые иконы, то можно графически показать на какие иконы ссылаются в качестве родителей инструкции в схеме. Например, как это продемонстрировано на рисунке 1.27, где для удобства содержимое икон «Схема» и «Примитив» выделены пунктирным квадратом.

А с помощью пунктирных стрелок на рисунке 1.27 показаны родительские отношения между иконами.

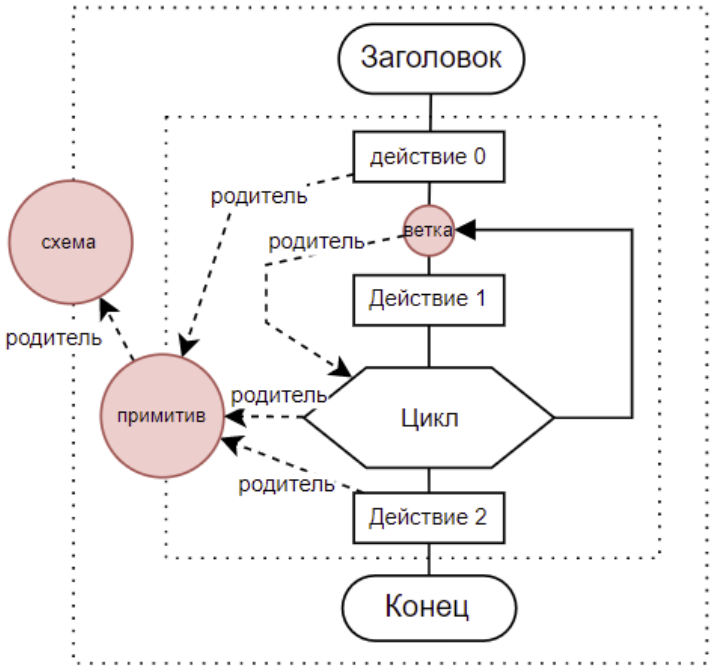

Рисунок 1.27 – Демонстрация родительской связи для икон «Ветка» и «Цикл»

Для иконы «Ветка» родителем является «Цикл». Для самой иконы «Цикл» родителем является «Примитив», который также содержит иконы «Действие 0» и «Действие 2». Сам «Примитив» является вложенной иконой «Схемы». Схема родительской иконы не имеет.

Стоит обратить внимание, что иконы «Заголовок» и «Конец» не представлены в схеме напрямую. «Заголовок» указывает на первую икону в «Схеме», а «Конец» на последнюю в иконе «Примитив», тем самым обеспечивается оптимизация при хранении ДРАКОН-схемы.

Для каждого типа икон есть шаблонные правила, по которым они могут быть вставлены в схему или быть удалены из неё. Таким иконам не нужна информация обо всей схеме, только данные родительской и соседних иконах, где производится сама операция.

1.2.6 Алгоритм вставки икон

Пользователь не может вставлять другие иконы во внутреннюю структуру макроикон, так как это может нарушить целостность всей схемы.

Для решения этой проблемы в каждой макроиконе есть специальные макроиконы «Ветка» и «Примитив», которые разрешают операцию вставки внутрь себя.

Пользователь может вставить до и после любой простой иконы в «Примитиве» и «Ветке». В случае, если они не содержат икон, то доступно только одно место для вставки: первым номером.

Доступные места для вставки отображаются графически с помощью круга зеленого цвета. Каждая такая икона содержит информацию о родителе, в котором она располагается, и адрес следующей иконы, если он существует.

В первом случае новая икона вставляется в массив перед иконой, обозначенной как следующая. Во втором случае вставка осуществляется в конец массива.

Для макроикон определены собственные правила для вставки. Так для иконы «Вопрос» создаются две «Ветки», которые не добавляются в родителя, но в словарь, содержащий ID всех икон в схеме. Аналогично для всех остальных комплексных икон.

Алгоритм вставки новых икон показаны на рисунке 1.28.

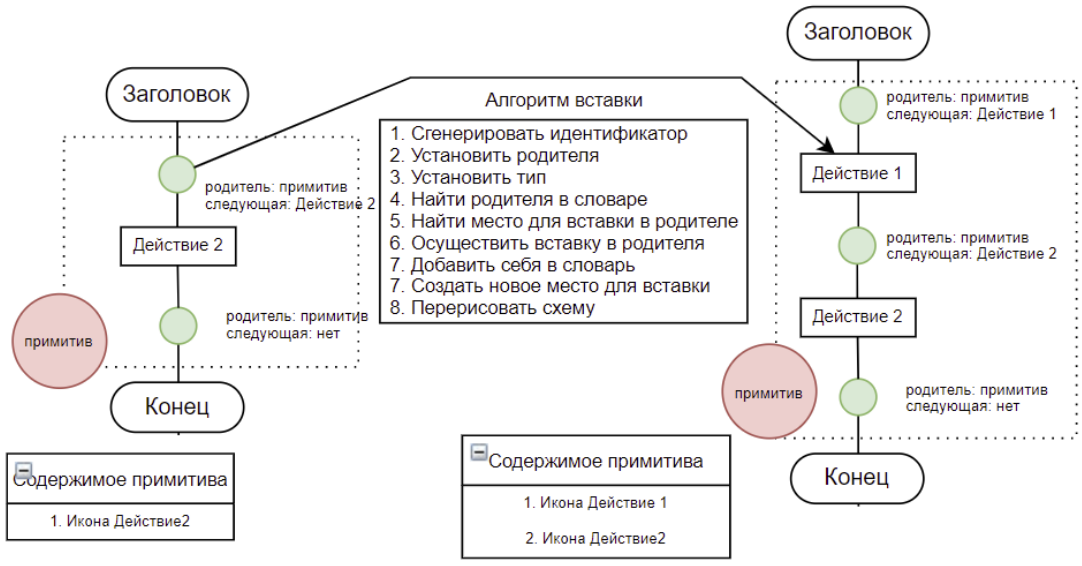

Рисунок 1.28 – Алгоритм вставки иконы

Каждая икона, добавляемая в схему, сохраняет себя в словаре, это позволяет получить доступ к её полям (см. приложение 1) по её идентификатору.

1.2.7 Алгоритм удаления икон

Удалять из схемы можно практически все отображаемые иконы, кроме «Заголовок» и «Конец», которые неявно представляют икону «Схема». Алгоритм схож с вставкой, но есть некоторые значимые отличия.

Простые иконы, которые не содержат в себе других инструкций, удаляются достаточно просто: по уникальному идентификатору в словаре находится ссылка на икону. После чего производится простая операция удаления иконы из массива со смещением. Также икона удаляется из словаря, после этого удаление можно считать завершенным.

Каждая удаляемая макроикона может содержать в себе некоторое множество других икон. В случае простого удаления все вложенные иконы останутся в висячем состоянии, когда формально иконы присутствуют схемы, но получить доступ к ним не представляется возможным. Поэтому все операции по удалению производятся рекурсивно.

Сначала производится поиск самой удаляемой иконы. Для всех икон, которые содержатся в удаляемой выполняется проверка: если содержимое присутствует, то осуществить переход в каждую икону и выполнить аналогичную проверку на наличие вложенных инструкций, в против случае произвести удаление.

1.2.8 Описание проектируемых интерфейсов

Помимо решения проблемы хранения и представления ДРАКОН схемы в памяти ЭВМ, необходимо графически представить как саму схему, там и способы взаимодействия с ней.

Каждой иконе присваиваются координаты и размер в осях X и Y. Позиция каждой последующей иконы вычисляется на основе предыдущей.

Для отрисовки соединительных линий вычисляется центр получившегося прямоугольника, сформированного координатами, начала иконы и её размера как для иконы родителя, так и для иконы потомка.

При добавлении или удалении икон осуществляется пересчёт всех координат, так-как могут потребоваться существенные изменения в текущем представлении ДРАКОН-схемы.

Принцип вычисления координат показан на рисунке 29.

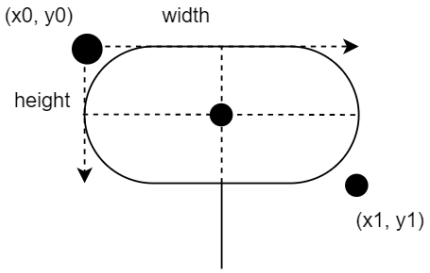

Рисунок 1.29 – Вычисление координат иконы

Места для вставки графически отображаются с помощью зеленого круга (см. рис. 1.28). Если пользователь хочет добавить икону в схему, для него необходимо графически показать все доступные места для вставки.

Все иконы, которые можно удалить также необходимо выделять. Если пользователь хочет исключить икону или макроикону из схемы, то необходимо подсветить все доступные для удаления иконы с помощью обводки красного цвета для всех икон, которые могут быть удалены из ДРАКОН-схемы.

Для ДРАКОН схемы необходима рабочая область, где будет отображаться сама схема. В ней также устанавливается разметочная сетка, которая служит для упрощения манипулирования схемой.

Манипуляции по добавлению новых икон должны проводиться через специальную панель, где пользователь выбирает нужную икону, а после место, куда требуется вставить икону.

Проектируемый интерфейс должен быть дружелюбен к пользователю. Весь основной функционал должен быть на главной странице приложения. Пользователь обязан иметь возможность как создавать и сохранять ДРАКОН схему, так и загружать уже существующую.

Программный продукт обязан предоставлять всю необходимую информацию о себе и его разработчике. Должен содержать в себе документацию, которая включает в себя учебник по работе с языком ДРАКОН и инструкцию по использованию самого программного продукта.

В рамках этой концепции был разработан прототип интерфейса редактора, показанный на рисунке 1.30.

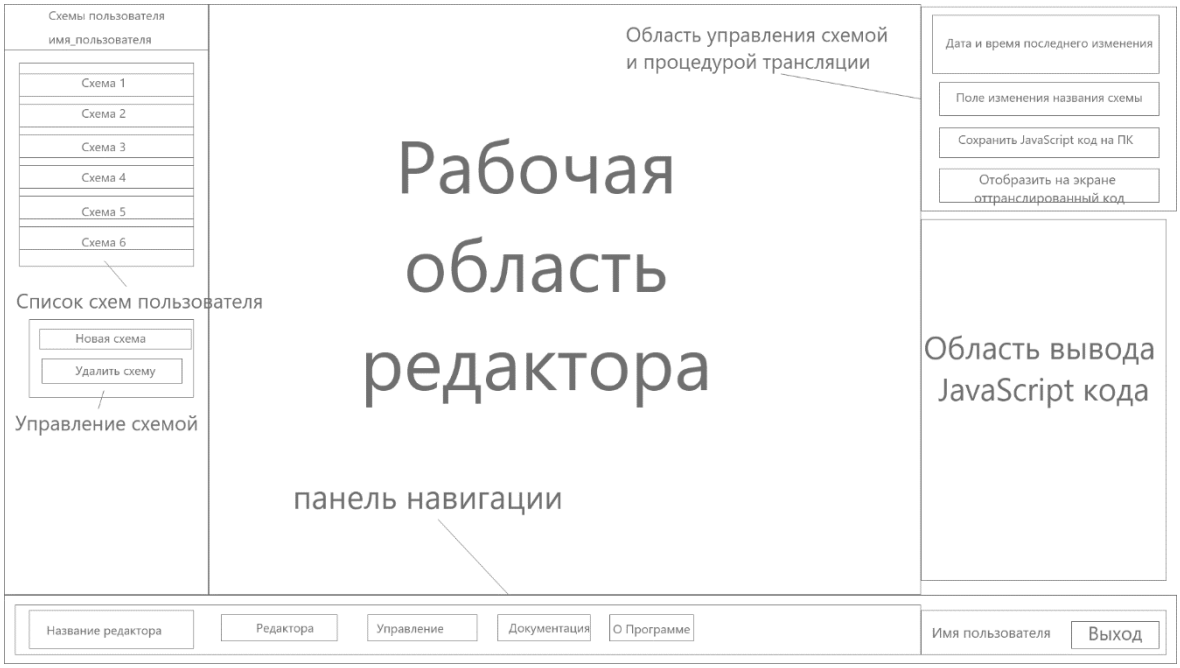

Рисунок 1.30 – Прототип графического интерфейса редактора

Редактор можно разделить на 4 основных блока:

- блок управления списком схем пользователя, куда включены методы добавления и удаления новых схем;
- рабочая область редактора, где отображаются ДРАКОН-схемы;
- блок управления схемой, в которой, в частности, выведена трансляция;
- панель навигации, которая содержит ссылки на основные страницы приложения и кнопку выхода.

На данном прототипе интерфейса основной графический блок с ДРАКОН-схемой располагается в центре. Именно на нём пользователь будет создавать дракон-схемы.

1.3 Инфологическая модель данных

Информационная система представляет из себя многопользовательское онлайн веб приложение, которое должно осуществлять сбор и обработку данных пользователей. База данных для разрабатываемой системы должна содержать следующее:

- информация о всех пользователях системы;
- роли пользователей в ИС;

- списки курируемых пользователей для кураторов;
- ДРАКОН-схемы каждого пользователя.

Таким образом, можно выделить следующие сущности базы данных, представленные в таблице 1.1.

Таблица 1.1 – Сущности базы данных

Сущность	Атрибуты
Пользователь	Идентификатор пользователя, имя пользователя, почта, пароль, роль
Курируемое отношение	Идентификатор отношения, идентификатор куратора, идентификатор пользователя, имя отношения
ДРАКОН-схема	Идентификатор схемы, название схемы, идентификатор пользователя, структура схемы, дата последнего изменения, идентификатор пользователя, сделавшего последнее изменение

Данные таблицы 1.1 также можно представить в виде ER-диаграмм (рис.1.31-1.33). На рисунке 1.31 изображен фрагмент ER-модели для сущности «Пользователь».

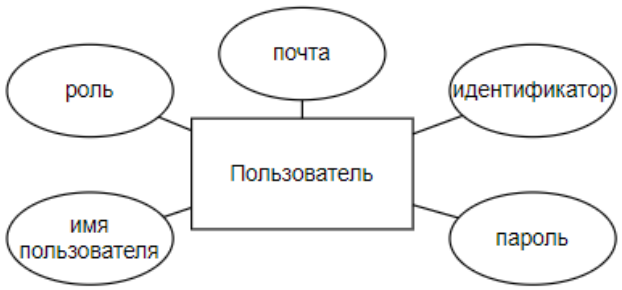

Рисунок 1.31 – Фрагмент ER-модели для типа сущности «Пользователь»

Фрагмент ER-модели сущности «Курируемый пользователь» показан на рисунке 1.32.

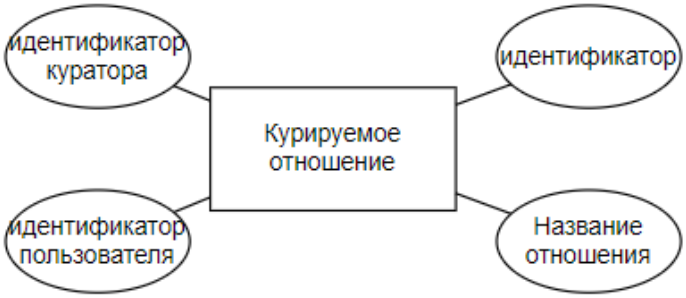

Рисунок 1.32 – Фрагмент ER-модели для типа сущности «Курируемое отношение»

Фрагмент ER-модели сущности «ДРАКОН-схема» показан на рисунке 1.33.

Рисунок 1.33 – Фрагмент ER-модели для типа сущности «ДРАКОН-схема»

Из сущностей, представленных на рисунках 1.31-1.33, сформирована диаграмма «Сущность-связь», представленная в приложении 2.

1.4 Входная и выходная информация

1.4.1 Входная информация

К входным данным относятся:

- Регистрационные данные пользователя (логин и пароль);
- Действия пользователя:
  - загрузка схем;
  - редактирование схем;
  - смена регистрационного имени;
  - смена пароля;
- Действия преподавателя в интерфейсе администрирования:
  - редактирование списка курируемых пользователей;
  - редактирование отношения между куратором и пользователем
- Действия администратора в интерфейсе администрирования:
  - выдача ролей;
  - удаление пользователей;

1.4.2 Выходная информация

К выходным данным относятся:

- Файл ДРАКОН-схемы в формате JSON;
- Файл с исходным кодом оттранслированной ДРАКОН-схемы на языке JavaScript;

- Список ДРАКОН-схем и их содержимое из базы данных;
- Список курируемых пользователей для каждого куратора.

1.5 Формат содержимого ДРАКОН-схемы

ДРАКОН-схема должна включать в себя способы собственного хранения на физических носителях, в частности в базе данных. В PostgreSQL наиболее удобным способом хранения набора данных с переменным числом атрибутов является формат JSON.

Для хранения дракон-схем был разработан специальный JSON-формат. Информацию о расположении хранить избыточно, так как представление осуществляется на стороне клиента. При загрузке дракон-схем производится перерасчёт расположения всех икон.

Каждая схема состоит из двух основных объектов: идентификатора макроиконы «Схема» и списка всех икон в схеме, которые содержат следующую информацию:

- «id» - атрибут, содержит уникальный uuid иконы;
- «type» - тип иконы;
- «parent» - содержит uuid родительской иконы;
- «children» - содержит список из uuid всех своих дочерних икон.

К каждой иконе можно получить доступ через её уникальный идентификатор.

Пример представления ДРАКОН-схемы в формате JSON представлен на рисунке 1.34.

Рисунок 1.34 – Представление ДРАКОН-схемы в формате JSON

1.6 Трансляция ДРАКОН-схемы в JavaScript

Любую ДРАКОН-схему можно однозначно представить в синтаксисе языка JavaScript. Каждой иконе сопоставляется конструкция языка:

- действие – операторы объявления переменных, констант, математические операции;
- комментарий – синтаксис многострочных комментариев языка Javascript;
- вывод – операторы вывода в консоль console.log();
- ввод – не обладает единой реализации в JavaScript в различных средах выполнения реализации отличаются друг от друга;
- пауза – операторы синхронного ожидания таймера await SetTimeout();
- вопрос – операторы if-else;
- цикл – оператор do-while;
- вариант – оператор switch;
- ветка – конструкция блока {};
- примитив – функция function MyFync(){}.

Трансляция начинается с чтения иконы «Схема». Изначально формируется пустая строка, содержащую только идентификатор иконы «Схема». Впоследствии рекурсивно происходит обход всех дочерних элементов для каждой иконы, где идентификаторы заменяется на конструкции языка JavaScript.

Пример сопоставления содержимого ДРАКОН-модели с программой на языке JavaScript показан на рисунке 1.35.

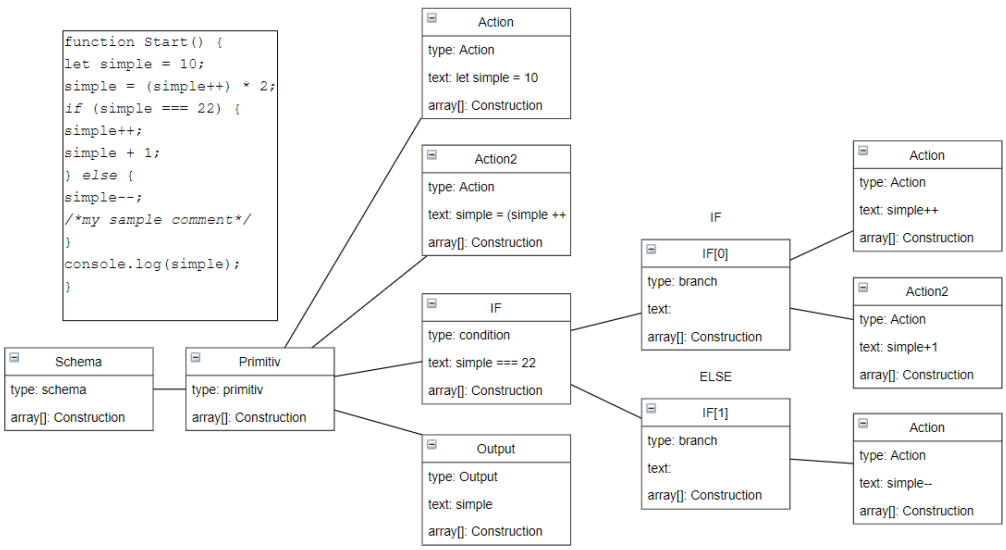

Рисунок 1.35 – сопоставление ДРАКОН-модели с кодом JavaScript

Слева направо показан уровень вложенности каждой иконы. Столбец представляет из себя массив икон, которые содержатся в родителе, с которым иконы в столбце связаны.

1.7 Требования к техническому и программному обеспечению

Рабочая станция:

- Intel-совместимый процессор с частотой не менее 2x1,6 ГГц;
- не менее 1024 МБ ОЗУ;
- не менее 200 МБ свободного места на диске;
- операционная система: ОС с поддержкой Chrome v.63+.

Сервер приложения и сервер базы данных:

- Intel-совместимый процессор с частотой не менее 8x3,4 ГГц;
- не менее 16 384 МБ ОЗУ;
- не менее 100 ГБ свободного места на диске;
- дисковод CD-ROM/DVD-ROM;
- операционная система: Windows Server 2008+ или схожая по функциональным возможностям UNIX-подобная ОС.

2 РАБОЧИЙ ПРОЕКТ

2.1 Общие сведения о работе системы

Программный продукт разработан в среде разработки Visual Studio Code v1.57 на языках TypeScript v4.0.5 и JavaScript ES6+.

Программный продукт использует клиент-серверную модель архитектуры.

Для построения сервера используется библиотека NestJS v.7.5.0. В качестве СУБД используется PostgreSQL v13.2. В базе данных хранятся данные о пользователях системы, их ДРАКОН-схемы и отношения пользователей в системе. Для связи СУБД и сервера используется библиотека TypeORM v0.2.3.

Все интерфейсы проектируемой системой разработаны с использованием библиотеки построения пользовательских интерфейсов – React v17.0.0.

Клиент – одностраничное (SPA) веб-приложение использующий динамическую подгрузку HTML, CSS и JavaScript для построения компонентов интерфейса.

2.2 Функциональное назначение программного продукта

Разработанный программный продукт предназначен для получения базовых знаний по основам алгоритмизация с помощью построения схем на языке ДРАКОН. Программа должна предоставлять пользователю возможность создавать, изменять и удалять схемы. Программа имеет следующие функциональные возможности:

- регистрация нового пользователя;
- аутентификация пользователя в системе с помощью логина и пароля;
- удаление учётной записи пользователя;
- обновление пароля пользователя;
- изменение логина пользователя;
- создание схемы;
- загрузка схемы из базы данных;
- редактирование схемы и её содержимого;
- удаление схемы;
- просмотр и изменение куратором схем других пользователей;
- добавление/удаление пользователя в список курируемых;
- изменение роли пользователя администратором;
- удаление учетной записи пользователя администратором.

Программа имеет следующие функциональные ограничения:

- не реализованы механизмы изменения масштаба и прокручивания области редактирования;
- отсутствует возможность создавать силуэты языка ДРАКОН;
- положение икон строго зафиксированы на рабочей области.

2.3 Выполнение программного продукта

Программный продукт располагается в папке dragon в каталоге с проектом. Для развертывания системы необходимо осуществить первоначальную настройку системы.

Информационной системе для развертывания необходима функционирующая база данных PostgreSQL v 13.0+. Сервер может располагаться как на удалённом сервере, так и на той же машине. Параметры подключения задаются в специальном конфигурационном файле «dragon/backend/ormconfig.json». Необходимо указать следующие параметры:

- host;
- port;
- username;
- password;

Предварительно необходимо инициализировать новую базу данных с именем «drakon_db» удобным для пользователя способом.

SQL скрипт для инициализации базы данных располагается по пути «dragon/backend/scripts/drakon_db.initscript.sql». После выполнения скрипта инициализации база данных готова к работе.

Для сборки программного продукта потребуется NodeJS версии 12.3 или выше. Для начала необходимо установить все зависимости для проекта «backend» с помощью команды npm install. После ввести последовательно команды npm run seed:config и npm run seed:run для создания роли администратора по умолчанию.

В файле .env в папке «dragon/backend» указать параметр PORT и HOSTNAME для прослушивания. Продублировать порт в «dragon/client-react/.env» в параметр REACT_APP_CLIENT_PORT и указать хост и порт в параметре REACT_APP_SERVER_DOMAIN.

Для запуска собственного сервера NodeJS достаточно выполнить команду npm run build в директориях «dragon/client-react» и «dragon/backend» для сборки

приложения с текущими параметры порта и хоста. После выполнить команду node dist/main.js в папке «dragon/backend», которая запустит сервер Node. Приложение готово к работе.

Для начала работы необходимо открыть браузер по адресу, который был задан для прослушивания (по умолчанию http://localhost:5000). При переходе на домашний URL будет открыта страницу входа в приложение. В системе предварительно есть уже зарегистрированная учётная запись администратора с комбинациями логина и пароля «Administrator». Перед осуществлением эксплуатации системы настоятельно рекомендуется войти в систему под этой учётной записью и изменить пароль в панели администратора.

Для регистрации необходимо создать нового пользователя, необходимо придумать имя, пароль и ввести почту для регистрации. В случае успешной регистрации будет получен JWT токен для доступа к защищённым URL приложения, а пользователь будет перенаправлен на домашнюю страницу приложения.

2.4 Физическая архитектура системы

Проектируемая система состоит из нескольких компонент. В качестве СУБД используется PostgreSQL версии 13.2. Сервер представляет из себя NestJS приложение, которое содержит библиотеку Express для обслуживания запросов от клиентов. Клиентами представляется одна или нескольких машин с доступом к сети, в которой работает сервер. Сервер предоставляет клиентам html страницы и данные из базы в ответ на запросы со стороны браузера. Диаграмма развертывания представлена в приложении 3 (см. рис. П.3).

2.5 Описание программы

Программный продукт следует разделить на 3 отдельных компонента:

- клиент;
- сервер;
- база данных.

2.5.1 Описание работы сервера

Сервер отвечает за обслуживание запросов со стороны клиента и взаимодействие с базой данной. NestJS предоставляет модульную архитектуру, в

которой каждый компонент подключается к основному модулю системы, расширяя функциональность сервера.

Пользовательские запросы обрабатываются через URL вида «домен/api/*» сквозь которые предоставляется API для получения информации из модулей сервера, которые обеспечивают связь с базой данных. Все модули спроектированы с учётом архитектурного стиля RESTful API.

В программе реализовано 7 модулей:

- UserModule;
- AuthModule;
- CuratorsModule;
- SchemaModule;
- TypeOrmModule;
- ServeStaticModule;
- ConfigModule.

UserModule отвечает за взаимодействие с сущностью «Пользователь» базы данных. Описание модуля приведено в таблице 2.1.

Таблица 2.1 – Описание модуля UserModule

Компонент	**Назначение**
UserService	Обработчик запросов к базе данных к сущности «Пользователь»
UserController	Обработчик вызовов к api/users

AuthModule обеспечивает механизмы регистрации новых пользователей и аутентификацию уже зарегистрированных. Его описание приведено в таблице 2.2.

Таблица 2.2 – Описание модуля AuthModule

Компонент	**Назначение**
AuthService	Обработчик запросов к базе данных к сущности «Пользователь» с целью аутентификации пользователя
AuthController	Обработчик вызовов к api/auth системы авторизации
BrcyptService	Сервис шифрования и валидации паролей

Продолжение таблицы 2.2

Компонент	Назначение
JwtStrategy	Сервис стратегии по управлению защищёнными путями URL
LocalStrategy	Сервис стратегии по валидации учетных данных пользователя

CuratorModule отвечает за взаимодействие с сущностью «Куриремое отношение» базы данных. Описание модуля приведено в таблице 2.3.

Таблица 2.3 – Описание модуля CuratorModule

Компонент	Назначение
CuratorService	Обработчик запросов к базе данных к сущности «Куратор»
CuratorController	Обработчик вызовов к api/curator

SchemaModule отвечает за взаимодействие с сущностью «ДРАКОН-схема» базы данных. Описание модуля приведено в таблице 2.4.

Таблица 2.4 – Описание модуля SchemaModule

Компонент	Назначение
CuratorService	Обработчик запросов к базе данных к сущности «Куратор»
CuratorController	Обработчик вызовов к api/curator

TypeOrmModule обеспечивает взаимодействие с базой данной. Его настройка осуществляется через файл «ormconfig.json». Помимо связи с базой данной, также предоставляет набор методов для осуществления запросов к словарям.

ServeStaticModule отвечает за обслуживание статических файлов. В программе таковыми файлами является собранный клиент приложения и его медиаресурсы (иконы, изображения, файлы markdown).

ConfigModule предоставляет поддержку конфигурационных файлов .env в среде NestJS.

Диаграмма компонентов сервера приведена в приложении 4.

2.5.2 Общее описание базы данных и даталогическая модель данных

Сервер осуществляет взаимодействие с данными через СУБД PostgreSQL v13.2. В ответ на SQL запросы по протоколу TCP/IP сервер получает и обновляет данные в базе. БД состоит из 3 справочников:

- пользователь;
- курируемое отношение;
- ДРАКОН-схема.

Справочник «Пользователь» предназначен для хранения информации о пользователях системы. Справочник содержит уникальный идентификатор пользователя, а также зашифрованный пароль с помощью функции bcrypt и информацию о роли пользователя в системе и почту. Идентификация пользователя осуществляется с помощью uuid, однако имя пользователя также должно быть уникальным для удобной идентификации через интерфейсы системы. Атрибуты справочника «Пользователи» представлены в таблице 2.5.

Таблица 2.5 – Справочник «Пользователи»

Параметр	**Тип**	**Размер**	**Примечание**
id	uuid, primary key	16	Код записи. Уникальное значение. Первичный ключ.
username	varchar	50	Уникальное имя пользователя, используется для идентификации пользователя клиентом
email	varchar	60	Почта учетной записи пользователя
password	text	переменный	Хешированный пароль пользователя
role	varchar	50	Пароль учетной записи пользователя

Справочник «Курируемое отношение предназначен» для представления связи между пользователем и куратором. На основе наличия в данном справочники принимается решения выдачи или не выдачи ДРАКОН-схем других пользователей. Атрибуты справочника «Курируемое отношение» показаны в таблице 2.6.

Таблица 2.6 – Справочник «Курируемое отношение»

Параметр	Тип	Размер	Примечание
id	serial, primary key	4	Номер записи. Уникальное значение. Первичный ключ.
id_curator	uuid, foreign key	16	Код записи куратора, Внешний ключ. Связан с первичным ключом таблицы «Пользователь». Вместе с id_user образуют уникальную пару куратор-пользователь.
id_user	uuid, foreign key	16	Код записи пользователя, Внешний ключ. Связан с первичным ключом таблицы «Пользователь». Вместе с id_curator образуют уникальную пару куратор-пользователь.
relation_name	varchar	100	Название для отношения кураор-пользователь

Справочник «ДРАКОН-схема» содержит содержимое схемы, идентификатор пользователя, создавшего её, а также название, информацию о времени и пользователе, внесшим последнее изменение. Информация о последнем изменении сохраняется для того, чтобы идентифицировать куратора или пользователя, который последним изменял схему. Атрибуты справочника «ДРАКОН-схема» представлены в таблице 2.7.

Таблица 2.7 – Справочник «ДРАКОН-схема»

Параметр	Тип	Размер	Примечание
id	uuid, primary key	4	Номер записи. Уникальное значение. Первичный ключ.
name	varchar	100	Название схемы
id_user	uuid	16	Код записи пользователя. Содержит id владельца схемы
data	varchar	100	Содержимое ДРАКОН-схемы

Продолжение таблицы 2.7

Параметр	Тип	Размер	Примечание
last_changed	datetime	8	Дата и время последнего изменения схемы
last_changed_by_id	uuid	16	id пользователя, изменения которого были последними.

2.5.3 Описание работы клиента

Клиент написан с использованием библиотеки React и реализован как одностраничное приложение. Это означает, что фактически в приложении существует одна единственная страница.

Сервер, при получении соответствующего URL запроса отправляет на клиент HTML страницу и скрипты JavaScript для взаимодействия с клиентской частью приложения. Для реализации возможности навигации в приложении клиента используется библиотеке «react-router», которая позволяет осуществлять рендер только тех интерфейсов системы, которые однозначно сопоставляются с URL путём.

Для рендера используется статическая модель, в которой все пути заранее известны. В системе выделяются следующие компоненты пользовательского интерфейса для рендера:

- страница регистрации;
- страница редактора ДРАКОН-схем;
- страница управления учетной записью и администрирования пользователей;
- страница документации по работе с ИС и языком ДРАКОН;
- страница информации о разработанном продукте;
- информационная страница о несуществующем пути приложения (форма ошибки 404).

Страница регистрации содержит все необходимые интерфейсы для авторизации пользователя в системе.

Страница редактора ДРАКОН-схем предоставляет функциональную возможность взаимодействия со схемами для пользователей системы.

На странице управление учетной записью пользователь может изменять конфиденциальные данные. Куратор редактировать список курируемых пользователей, а администратор отвечает за выдачу ролей в системе.

Интерфейс документации содержит всю основную информацию по способам взаимодействия с системой, а также содержит описание функциональных возможностей и ограничений программы, а также правила языка ДРАКОН.

Страница сведений о продукте содержит информацию о разработчике, разрабатываемой системе и все необходимые ссылки на рабочие репозитории проекта.

2.5.4 Описание процесса аутентификации

По умолчанию, по домашнему адресу приложения «/» осуществляется открытие интерфейса редактора. Но предварительно осуществляется проверка на наличие сохранённых данных пользователя.

Для хранения информации об учетной записи используется localStorage – интернет-хранилище в браузере клиента, которое позволяет осуществлять запись, чтение и удаление важных данных, которые необходимо сохранять локально на рабочей станции клиента.

Структура регистрационных данных пользователя, которая сохраняется в браузере клиента, имеет вид, представленный в таблице 2.8.

Таблица 2.8 – структура пользователя в localStorage

Поле	Тип	Примечание
username	Текстовый или NULL	Уникальное имя пользователя в системе
access_token	Текстовый или NULL	Сохранённый JWT токен доступа
uuid	Текстовый или NULL	Уникальный идентификатор пользователя
role	Текстовый или NULL	Роль пользователя в системе
email	Текстовый или NULL	Почта пользователя

Структура, описанная в таблице 8, полностью совпадает с сущностями из справочника «Пользователь» за исключением JWT токена. Если пользователь уже регистрировался в системе, то его данные буду сохранены в браузере, в противном случае необходимо пройти процедуру аутентификации.

Аутентификация пользователей осуществляется через специальную форму регистрации и входа.

Для успешного прохождения процесса регистрации необходимо ввести уникальное имя, по которому можно будет однозначно идентифицировать пользователя, пароль и почту. Если пользователь ранее зарегистрирован в системе, достаточно ввести валидную комбинацию логина и пароля.

На стороне клиента реализованы специальные сервисы формирования запросов и обработки ответов.

Сервис обработки запросов аутентификации отвечает за формирование пакета данных для отправки на сервер с целью регистрации пользователя, проверки валидности данных, выдачи и обновления JWT токенов. А также реализует обработчики для полученных с сервера ответов на запросы (Response).

Для валидации пользователя необходимо отправить на сервер форму с корректным именем пользователя и паролем. Сервер осуществит запрос на сравнение пользователя из базы данных с полученным от клиента именем. В случае, если пользователь найден, сервер получит объект сущности User, произведет хэширование пароля с той же солью, что и у пароля, хранимого в базе. Если пароли совпадают, сервер генерирует код возврата 200 и JWT токен регистрации, который будет валиден в течении 360 минут после успешного сеанса регистрации. После истечения срока валидности токена пользователю необходимо повторно ввести логин и пароль для обновления токена.

В случае возникновения ошибок при прохождении процедуры аутентификации выводится информационной сообщение со сведениями о произошедшей ошибке и о способах устранения.

Если аутентификация прошла успешна: пользователь в системе существует или же был создан, то ему выдаётся JWT токен доступа, который сохраняется в localStorage браузера вместе со всей информацией о текущем пользователе.

Такой подход позволяет сократить количество запросов к серверу, всю необходимую информацию для формирования запросов клиент хранит самостоятельно.

Эта же информация дублируется в специальный компонент библиотеки React – контекст. Контекст в React позволяет получить доступ к данным в любой точке приложения, принцип схож с глобальной переменной, однако для неё реализованы специальные методы по защите от случайных изменений.

2.5.5 Описание процесса взаимодействия с ДРАКОН-схемой

Если аутентификация пройдена успешна, а значит пользователю выдан JWT токен доступа, то осуществляется переход на главную страницу приложения, которой является интерфейс редактора ДРАКОН-схем.

На данной странице отображаются все необходимые элементы интерфейса для взаимодействия с ДРАКОН-схемами, сохранёнными в базе данных. По умолчанию каждый новый пользователь не имеет ДРАКОН-схем, закрепленных за ним. Однако интерфейс системы предусматривает создание новых ДРАКОН-схем, которые, в момент создания, добавляются в словарь «ДРАКОН-схема» и закрепляются за пользователем.

Редактор ДРАКОН-схем реализован с помощью графической библиотеки «Konva» по работе с элементом HTML Canvas. Редактор предоставляет функциональные возможности по созданию, редактированию и удалению икон и макроикон в выбранной пользователем схеме, позволяет перемещать схему в пределах рабочей области, а также заполнять иконы функциональным содержимым.

Изначально при входе в интерфейс редактора загружаются все идентификаторы ДРАКОН-схем из базы данных. Пользователю необходимо выбрать нужную ему схему для загрузки данных о её содержимом из базы данных.

ДРАКОН-схема, предварительно созданная пользователем, загружается в формате JSON. После осуществляется процесс её чтения с целью формирования ДРАКОН-модели, соответствующей экземплярам класса DragonModel (см. рис П.1).

На основе ДРАКОН-модели генерируется графическое отображение схемы библиотекой «Konva», где учитываются все требования, предъявляемые со стороны языка ДРАКОН.

Пользователь может добавлять или удалять иконы. Все возможные иконы для вставки предоставляются элементами интерфейса, как и сама возможность изменения содержимого ДРАКОН-схем.

После окончания редактирования схемы пользователю необходимо сохранить изменения, нажав соответствующую клавишу. Тогда схема снова преобразуется в формат JSON и отправится вместе с формой на сервер, который реализует механизмы обновление данных в БД.

2.5.6 Описание процесса управления пользовательскими данными

Каждому пользователю доступна возможность изменять свои конфиденциальные данные. Для этого в программе есть специальный интерфейс администрирования. Количество доступных вариантов использования зависит от роли пользователя в системе.

Обычному пользователю предоставляется возможность изменять своё уникальное имя в системе и пароль от учётной записи.

Куратору предоставляется возможность просмотра списка всех пользователей с функцией фильтра. Каждого простого пользователя Куратор может добавить в список курируемых, что позволит ему просматривать и изменять его схемы в интерфейсе редактора ДРАКОН-схем.

Администратор в системе необходим для установки ролей. Он также может просматривать любого пользователя в системе, но именно он назначает кто в системе является куратором, а кого назначить администратором. Дополнительно Администратор обладает функциональной возможностью удалять учётные записи пользователей из системы.

2.6 Обеспечение защиты данных

Для обеспечения безопасности хранимых в системе паролей используется шифрование. За него ответственен BcryptService модуля AuthModule. Его описание приведено в таблице 2.9.

Таблица 2.9 – Описание сервиса BcryptService

Метод	Назначение
hashPassword	Шифрование пароля с помощью функции bcrypt с заданной солью
checkPassword	Шифрование кандидата с помощью функции bcrypt с заданной солью и сравнение с валидным зашифрованным паролем.

Шифрование с помощью функции bcrypt является необратимым процессом. Зашифрованную последовательность символов невозможно вернуть в исходную. В базе данных все пароли хранятся в зашифрованном виде. Даже если злоумышленник получит доступ к базе данных, пароли останутся защищёнными.

2.7 Определение целостности сущностей

Каждая запись в справочниках обладает своим уникальным идентификатором и содержит набор данных.

В таблице «Пользователь» первичным ключом является поле ID, которое представляет из себя генерируемый уникальный uuid, с помощью которого можно уникально идентифицировать пользователя в базе. Чтобы у пользователей системы не было необходимость запоминать длинную комбинацию произвольных символов при аутентификации, каждому ID сопоставляется уникальное имя, которое пользователь выбирает самостоятельно.

В таблице «Куратор» первичным ключом служит поле ID. Данный ключ позволяет осуществлять быструю выборку из списка кураторских отношений, а также позволяет однозначно интерпретировать это отношение в базе. Но уникальность записи обеспечивает уникальная пара id_user и id_curator, что исключает возможность добавления нескольких вариантов отношений для одной пары куратор + пользователь.

В таблице «ДРАКОН-схема» первичным ключом служит поле ID. Уникальный идентификатор позволяет однозначно интерпретировать схему в системе, чтобы у пользователя всегда оставалась возможность манипулировать только своими, или только доступными ему, схемами.

2.8 Определение целостности атрибутов справочников

Атрибуты в справочниках не могут быть пустыми. Каждое поле важно для целостности записи.

Для таблицы «Пользователь» поля идентификатора, имени пользователя, почты и пароля нужны для идентификации пользователя в системе и обеспечения возможности входа. Роль необходима для предоставления различных привилегий. Для куратора такими привилегиями является возможность изменять схемы других пользователей, которые добавлены к нему в таблицу. Для администратора предоставляется возможность удаления пользователей и изменение ролей.

В таблице «Куратор» идентификаторы пользователя и куратора служат для однозначной интерпретации связи между двумя пользователями. А поле «название отношения» однозначно описывает эту связь.

В таблице «ДРАКОН-схема» поле названия необходимо для корректного отображения схемы на клиенте, поле «data» содержит саму схему, даже если схема

пустая, в ней содержится метаинформация для построения новой схемы. Дата последнего изменения и идентификатор пользователя, который произвёл последние манипуляции со схемой позволяют однозначно идентифицировать куратора, который изменил ДРАКОН-схему.

2.9 Определение ссылочной целостности атрибутов справочников

Ограничение внешнего ключа необходимо для соединения двух таблиц вместе и используется в таблицах «Куратор» и «ДРАКОН-схема».

Внешний ключ «id_куратор» в таблице «Куратор позволяет получить список всех курируемых пользователей из таблицы для текущего куратора. Внешний ключ «id_user» позволяет получить доступ к имени пользователя и его схемам в таблице «ДРАКОН-схема».

Внешний ключ «id_user» в таблице «ДРАКОН-схема» служит для получения только тех схем, которые принадлежат данному id. Внешний ключ «last_changed_by_id» позволяет получить имя куратора или пользователя, который последним производил изменения в таблице.

2.10 Основные пользовательские функции, триггерные функции, триггеры

Для управления пользователями, отношениями между кураторами и курируемыми, а также схемами, были реализованы триггеры и триггерные функции в базе данных, чтобы исключить возможность передачи неправильных входных данных со стороны сервера для уязвимых к ошибочным данным полей. Основные триггерные функции и триггеры представлены в таблице 2.10.

Таблица 2.10 – Основные триггерные функции и триггеры

Триггерная функция	Триггер	Назначение
user_registration	CREATE TRIGGER register_new_user_tr BEFORE INSERT ON users EXECUTE PROCEDURE *user_registration*();	Генерация uuid для нового пользователя
user_rename	CREATE TRIGGER user_rename_trg BEFORE UPDATE OR INSERT ON users EXECUTE PROCEDURE *user_rename*();	Проверка нового имени пользователя на наличие недопустимых символов

Продолжение таблицы 2.10

Триггерная функция	Триггер	Назначение
delete_user	CREATE TRIGGER delete_user_trg AFTER DELETE ON users EXECUTE FUNCTION *delete_user*();	Удаление данных о пользователе, всех его схем из таблицы «ДРАКОН-схема» и записей в таблице «Куратор»
curator_relation_role_check	CREATE TRIGGER validate_user_role BEFORE INSERT OR UPDATE ON curators EXECUTE PROCEDURE *curator_relation_role_check*();	Проверка отношения куратор-пользователь на корректность ролей.
scheme_insert	CREATE TRIGGER scheme_insert_trg BEFORE INSERT ON dragon_scheme EXECUTE PROCEDURE *scheme_insert*();	Генерация uuid ключа для таблица «ДРАКОН-схема»

2.11 Использование JSON-объектов

JSON необходим для удобного и компактного хранения сложного массива данных, количество полей в котором является динамическим. В таблице «ДРАКОН-схема» используется поле data типа jsonb, который хранит сложную структуру дракон схемы в виде единого объекта, что позволяет быть уверенным, что данные не мутируют без ведома системы и пользователя, а значит могут быть однозначно интерпретированы.

2.12 Обеспечение безопасности базы данных

Одним из важных аспектов в клиент-серверной архитектуре является организация и обеспечение безопасности как самой системы, так и данных, обрабатываемых и хранимых с помощью неё. В текущей версии информационной системы данные защищены следующими факторами:

- каждая триггерная функция, выполняемая до операции, обрабатывает исключения, что позволяет избежать сохранение некорректных данных;
- каждый DTO объект, который поступает с запросом на сервер, проходит валидацию на соответствие полей типу;

- каждый путь приложения, через который можно получить доступ к API базы данных защищён с помощью стратегии безопасности JWT, т.е. каждый пользователь должен произвести авторизацию и получить валидный токен;
- все роли имеют ограниченные возможности по обращению к базе данных только через соответствующие реализованные методы;
- пароли учетных записей невозможно дешифровать, сами пароли всегда хранятся только в зашифрованном виде с помощью функции bcrypt.

2.13 План обслуживания и резервного копирования базы данных

Резервное копирование базы данных осуществляется с помощью вспомогательной программы pg_dump.

2.14 Описание основных классов системы

2.14.1 Описание основных классов сервера

Для описания сущности «Пользователь» в системе был реализован класс пользователя, описание которого представлено в таблице 2.11.

Таблица 2.11 – Описание класса User

Поле	Тип	Назначение
uuid	string	Id пользователя
username	string	Имя пользователя
email	string	Почта
password	string	Пароль
role	string	Роль

Класс Curator представляет описание сущности «Курируемое отношение» внутри разрабатываемой системы. Описание класса размещено в таблице 2.12.

Таблица 2.12 – Описание класса Curator

Поле	Тип	Назначение
Id	number	Id отношения
uuid_curator	string	Id куратора

Продолжение таблицы 2.12

Поле	Тип	Назначение
uuid_user	string	Id пользователя
relation_name	string	Название отношения

Класс Schema представляет описание сущности «ДРАКОН-схема» для связи с базой данных. Описание класса Schema представлено в таблице 2.12.

Таблица 2.12 – Описание класса Schema

Поле	Тип	Назначение
uuid	string	Id схемы
name	string	Название схемы
id_user	string	Id пользователя
data	JSON	Содержимое схемы
last_changed	Date	Дата и время последнего изменения
last_changed_by_id	string	Id пользователя, сделавшего последнее изменение

Для авторизации используется связка из логина и пароля. Данные от клиента на сервер доставляются с помощью специальных DTO объектов, которые проходят проверку на корректность данных перед отправкой. Содержимое данного объекта приведено в таблице 2.13.

Таблица 2.13 – Описание LoginUserDTO

Поле	Тип	Назначение
username	string	Имя пользователя
password	string	Пароль

Для создания новой учетной записи используется класс CreateUserDTO. Важным отличием от LoginUserDTO является наличия поля для почта и специального декоратора для поля пароля, который осуществляет проверку содержимого на удовлетворение минимальным требования, которые описаны в подразделе 1.2.2.

Если пароль недостаточно надёжный, то возвращается статус 400 (Bad request) и сообщение «Слабый пароль». Описание класса CreateUserDTO представлено в таблице 2.14.

Таблица 2.14 – Описание CreateUserDTO

Поле	Тип	Назначение
username	string	Имя пользователя
email	string	Почта
password	string	Пароль

Каждый новый пользователь в системе получает роль, по умолчанию - «USER». Другие роли могут быть выданы администратором системы. Полный перечень ролей в системе представлен в таблице 2.15.

Таблица 2.15 – Описание ролей пользователя

Роль	Описание
USER	Базовый пользователь системы
CURATOR	Куратор, может добавлять к себе в список курируемых пользователей системы с целью просмотра и редактирования их схем
ADMIN	Администратор, распределяет роли, обладает возможностью удалять аккаунты и просматривать и редактировать чужие схемы

За обработку запросов к таблице «Пользователь» отвечает специальный UserController, который предоставляет набор URL путей, по которым пользователь может получить доступ к БД. Все пути в UserController защищены по схеме HTTP Bearer. Каждый запрос должен сопровождаться валидным JWT токеном в заголовке. Сопоставление методов контроллера с путями представлено в таблице 2.16.

Таблица 2.16 – Описание UserController

Метод	HTTP запрос	Описание
getAllNonPrivelegd()	GET host:port/api/users/nonprivileged	Получить список всех пользователей с ролью USER
findOneById(@Param('id', new ParseUUIDPipe({version: '4'})) id: string)	GET host:port/api/users/:id	Получить данные пользователя по ID

Продолжение таблицы 2.16

Метод	HTTP запрос	Описание
getAllNonPrivelegd()	GET host:port/api/users/nonprivileged	Получить список всех пользователей с ролью USER
update(@Body() payload:UserDto)	PUT host:port/api/users/update	Обновить данные пользователя
deleteUserByID(@Param('id', new ParseUUIDPipe({version: '4'})) id: string)	DELETE host:port/api/users/:id	Удалить данные пользователя

Сервисы NestJS осуществляет непосредственную обработку данных, которые предоставляются методами контроллера. Для каждого контроллера реализован минимум один сервис обслуживания запросов. Описание UserService, которые обслуживает запросы UserController, приведено в таблице 2.17.

Таблица 2.17 – Описание класса UserService

Поле	Тип	Назначение
usersRepository	Repository<User>	API к таблице «Пользователь»
Метод		**Назначение**
async findAll()		Получить все записи
async findByIds(users: string[])		Получить данные нескольких пользователей
async findAllNonPrivelegedUsers()		Получить список всех простых пользователей
async findOneByEmail(email: string)		Получить данные пользователя по его почте
async findOneById(id: string)		Поиск пользователя по id
async findOneByName(username : string)		Поиск пользователя по имени

Продолжение таблицы 2.17

Поле	Тип	Назначение
usersRepository	Repository<User>	API к таблице «Пользователь»
async create(userData : CreateUserDto)		Создание нового пользователя
async update(id: string, payload: Create UserDto)		Обновление конфиденциальных данных пользователя
async updatePassword(userid: string, p ayload: Partial<CreateUserDto>)		Обновить пароль пользователя
async delete(uuid: string)		Удаление пользователя по id

Для реализации сервиса авторизации необходим UserService, который предоставляет доступ к таблице «Пользователь». AuthController предоставляет набор URL путей для обеспечения механизма авторизации. Сопоставление методов контроллера с путями представлено в таблице 2.18.

Таблица 2.18 – Описание AuthController

Метод	HTTP запрос	Описание
login(@Request() req)	POST host:port/api/auth/login	Вход в систему зарегистрированного пользователя
signup(@Body() user: CreateUserDto)	POST host:port/api/ auth/signup	Регистрация нового пользователя
signup(@Body() user: CreateUserDto)	POST host:port/api/ auth/signup	Регистрация нового пользователя
verify(@Param('token') token: string)	GET host:port/api/ auth/verify/:token	Проверка валидности JWT токена
deleteUser(@Headers() req)	DELETE host:port/api/ auth/login	Удалить учётную запись пользователя
changePassword(@Bod y() changePasswordDto : ChangePasswordDto,)	PATCH host:port/api/ auth/changePassword	Обновить пароль

AuthController для изменения пароля использует специальный интерфейс ChangePasswordDTO, описание которого приведено в таблице 2.19. Для нового пароля используются декоратор проверки сложности пароля, аналогичный тому, который применяется для CreateUserDTO.

Таблица 2.19 – Описание ChangePasswordDTO

Поле	Тип	Назначение
username	string	Имя пользователя
oldPassword	string	Старый пароль
newPassword	string	Новый пароль

Реализация защиты путей приложения от несанкционированного доступа осуществляется с помощью библиотеки «Passport» и реализуемых ею стратегий LocalStrategy и JwtStrategy, которые предоставляют методы авторизации по связке логин плюс пароль или через токен соответственно.

Защищены все URL пути приложения, кроме запросов регистрации нового пользователя и проверки токенов на валидность.

Методы авторизации реализуются через AuthService, описание которого представлено в таблице 2.20.

Таблица 2.20 – Описание класса AuthService

Поле	Тип	Назначение
userService	UserService	Сервис пользователей для получения доступа к таблице «Пользователь»
jwtService	JwtService	Сервис безопасности и валидации JWT для идентификации и выдачи токенов
cryptoService	CryptoService	Сервис хеширования и валидации пароля
configService	ConfigService	Сервис управления файлами конфигурации
Метод		**Назначение**
async signUp(user : CreateUserDto)		Регистрация нового пользователя
async logIn(username: string, password: string)		Вход зарегистрированного пользователя и выдача токена
async createToken(user)		Создание и подпись токена

Продолжение таблицы 2.20

Метод	Назначение
async verify(payload)	Проверка, что пользователь зарегистрирован в системе
async verifyToken(token)	Проверка токена на валидность
async changePassword(changePasswordDto: ChangePasswordDto)	Изменение пароля учетной записи пользователя

За обработку запросов к таблице «Курируемое отношение» отвечает специальный CuratorController. Его описание представлено в таблице 2.21.

Таблица 2.21 – Описание CuratorController

Метод	HTTP запрос	Описание
async getAll()	GET host:port/api/curators	Получить все курируемые отношения
getOneById(@Param('id') id : number)	GET host:port/api/curators/:id	Получить курируемое отношение по id
getByCuratorId(@Param('id', new ParseUUIDPipe({version: '4'})) id: string)	GET host:port/api/curators/create	Получить все курируемые отношения для конкретного куратора
async createRelations(@Body() curatorObject : CreateCuratorDto)	POST host:port/api/curators	Создать новое отношение
async update(@Body() payload: CuratorDto)	PUT host:port/api/curators/:id	Обновить существующее отношение
async removeRelation(@Param() relationId: number)	DELETE host:port/api/auth/changePassword	Удалить отношение из базы данных

Для создания нового курируемого отношения используется интерфейс CreateCuratorDTO. Его описание представлено в таблице 2.22.

Таблица 2.22 – Описание CreateCuratorDTO

Поле	Тип	Назначение
uuid_curator	string	id куратора
uuid_user	string	id пользователя
relation_name	string	название курируемого отношения

Описание сервиса по взаимодействие с таблицей «Курируемое отношение» представлено в таблице 2.23.

Таблица 2.23 – Описание CuratorService

Поле	Тип	Назначение
curatorRepository	Repository <Curator>	API к таблице «Курируемое отношение»
Метод		**Назначение**
private async findAll()		Получить все записи
async findStudents(id_curator: number)		Получить всех пользователей, привязанных к куратору
async findOneById(id: string)		Поиск отношения куратор-пользователь по id
async findCurators(id_user : string)		Получить список всех кураторов, к которым привязан пользователь
async create(curator: CuratorDto)		Создание нового отношения
async update(id: number, payload: CuratorDto)		Обновление отношения куратора и пользователя
async delete(id: number)		Удаление отношения куратора и пользователя

За выдачу статических данных, такие как страница и медиаресурсы отвечает модуль ServeStaticModule, который входит в состав NestJS.

Для обеспечения взаимодействия с базой данной используется модуль TypeOrmModule, который входит в состав NestJS.

2.14.2 Описание основных классов клиента

Взаимодействие с сервисами сервера осуществляется через методы и функции, реализованные на стороне клиента. Для управления состояниям приложения было разработано четыре сервиса:

- Сервис управления авторизацией;
- Сервис пользователя;
- Сервис управления схемой;
- Сервис управления курируемыми пользователями.

За управление авторизацией в приложении отвечает класс AuthStateService. Он хранит состояние текущего пользователя на протяжении всей его работы. Описание класса приведено в таблице 2.24. При проектировании сервисов на стороне клиента применяется паттерн «Одиночка».

Таблица 2.24 – Описание AuthStateService

Поле	**Тип**	**Назначение**
static _instance	AuthStateServie	Экземпляр класса
static _accessToken	string	JWT токен доступа
static _user	IUser	Данные авторизованного пользователя
Метод		**Назначение**
getInstance(): AuthStateService		Получить экземпляр класса
getToken() : string		Получить JWT токен
getRole() : string		Получить роль
getUUID(): string		Получить id пользователя
getUsername(): string		Получить имя пользователя
getEmail(): string		Получить почту пользователя
async TokenVerification(token: string): Promise<boolean>		Проверка токена на валидность
async Authentificate(username: string, password: string): Promise<ResponsePayload>		Аутентификация пользователя по логину и паролю
async RegisterUser(username: string, password: string, email: string)		Регистрация нового пользователя

Продолжение таблицы 2.24

Метод	Назначение
Logout(): void	Осуществить выход из системы
async DeleteRegistrationData(uuid: string)	Удалить учетную запись пользователя
async create(curator: CuratorDto)	Создание нового отношения
async changePassword(oldPassword: string, newPassword: string)	Изменить пароль пользователя

Для чтения ответов сервера на вызовы методов авторизации реализован интерфейс ResponsePayload. Описание которого приведено в таблице 2.25.

Таблица 2.25 – интерфейс ResponsePayload

Поле	Тип	Назначение
status	number	Код ответа сервера
status_text	string	Сообщения, сопровождающее ответ
body	loginResponseDTO	Содержимое ответа сервера

В ответ на корректные данные аутентификации пользователя сервер возвращает JWT токен для доступа к защищённым путям приложения и поля класса User, которые помещаются в интерфейс loginResponseDTO.

Получение данных о других пользователях осуществляется через UserStateService клиента. Также данный сервер отвечает за изменение персональных данных текущего пользователя, поэтому для его работы необходим экземпляр класса AuthStateService. Описание класса UserStateService приведено в таблице 2.26.

Таблица 2.26 – Описание UserStateService

Поле	Тип	Назначение
static _instance	UserStateService	Экземпляр класса
static _authService	AuthStateService	Экземпляр класса регистрации
Метод		**Назначение**
getInstance():UserStateService		Получить экземпляр класса
setTokenBearer () : JSON		Установка заголовка с токеном авторизации для выполнения запросов
getUserinfoById(uuid: string): Promise<UserDTO>		Получить данные о пользователе по его ID

Продолжение таблицы 2.26

Метод	Назначение
GetUnprevilegedUsers(): Promise<UserDTO[]>	Получить список всех пользователей с ролью USER
GetAllUsers(): Promise<UserDTO[]>	Получить список всех пользователей
async UpdateData(user: UserDTO)	Обновить персональные данные пользователя

Взаимодействие с курируемыми пользователями осуществляется через CuratorStateService. Описание класса приведено в таблице 2.27.

Таблица 2.27 – Описание класса CuratorStateService

Поле	Тип	Назначение
static _instance	CuratorStateService	Экземпляр класса
static _authService	AuthStateService	Экземпляр класса регистрации
Метод		**Назначение**
getInstance():UserStateService		Получить экземпляр класса
createRelation(relation : CreateRelationDTO)		Создание нового отношения между куратором и пользователем
DeleteRelation(id: number)		Удаление существующего отношения
getRelationsByCurator(curator: string)		Получить список всех курируемых пользователей по id куратора
updateRelation(relation: RelationDTO)		Обновить выбранное отношение

Взаимодействие со схемами пользователя на удалённом хранилище в базе данных осуществляется с помощью SchemaService. Описание данного класса приведено в таблице 2.28. Выбранная для редактирования схема помещается в статическую переменную класса.

Таблица 2.28 – Описание класса SchemaService

Поле	Тип	Назначение
static _instance	SchemaService	Экземпляр класса
static _authService	AuthStateService	Экземпляр класса регистрации
static _schema	DragonModel	Модель ДРАКОН-схемы

Продолжение таблицы 2.28

Метод	Назначение
getInstance():UserStateService	Получить экземпляр класса
public getModel(model: DragonModel)	Получить модель ДРАКОН-схемы
public setModel(model: DragonModel)	Установить данные для модели ДРАКОН-схемы
async getUserSchemas(user_id : string) : Promise<SchemaDTO[]>	Получить все схемы выбранного пользователя
async createNewSchema(user_id: string) : Promise<SchemaDTO | undefined>	Создание новой схемы для текущего пользователя
updateSchema(schema: SchemaDTO): Promise<SchemaDTO | undefined>	Обновление схемы
deleteSchema(id: string): Promise<any>	Удаление схемы

Для представления модели данных были разработаны классы, которые представлены в приложении 1. Описание класса ModelDragon представлено в таблице 2.29.

Таблица 2.29 – Описание класса ModelDragon

Поле	Тип	Назначение
_head	string	Указатель на макроикону «Схема»
_container	map<string, DragonInstruction>	Словарь для хранения ссылок на все иконы ДРАКОН-схемы
Метод		**Назначение**
Insert(instruction: DragonInstruction, next?: string): void		Вставка иконы в ДРАКОН-схему
toJSON(): JSON		Преобразование схемы в формат JSON
public setModel(model: DragonModel)		Присваивание
Delete(parent: string, uuid: string)		Удаление иконы
private RecursiveDelete(parent: DragonInstruction)		Рекурсивное удаление всех вложенных икон
validate()		Проверка на висячие иконы
Find(uuid: string)		Поиск инструкции в ДРАКОН-схеме

Продолжение таблицы 2.29

Метод	Назначение
getDeepestLeftChild(uuid: string)	Поиск последней иконы в первой ветке
Update(uuid: string, text: string)	Обновление данных иконы
private castInstruction<T extends DragonInstruction>(instruction: T, instruction_meta: metadataInterface)	Преобразование абстрактной иконы к конкретной реализации
private parseInstructionFromJSON(instruction_id: string, schema: any)	Чтение иконы в формате JSON
static restoreFromJSON(schema: any)	Статическая версия алгоритма чтения схемы в формате JSON
public restoreFromJSON(schema: any)	Чтение схемы в формате JSON
public parseInstruction(instruction: DragonInstruction, code: string, deep = 0): string	Трансляция иконы в язык JavaScript
public toJavaScript(): string	Трансляция схемы в язык JavaScript

Каждая икона представляет собой экземпляр от наследуемого класса DragonInstruction. Который описывает базовое поведение для всех икон. Описание его содержимого представлено в таблице 2.30.

Таблица 2.30 – Описание класса DragonInstruction

Поле	Тип	Назначение
_head	string	Указатель на макроикону «Схема»
_container	map<string, DragonInstruction>	Словарь для хранения ссылок на все иконы ДРАКОН-схемы
Метод		**Назначение**
Insert(instruction: DragonInstruction, next?: string): void		Вставка иконы в ДРАКОН-схему
toJSON(): JSON		Преобразование схемы в формат JSON
public setModel(model: DragonModel)		Присваивание

Продолжение таблицы 2.30

Метод	Назначение
Delete(parent: string, uuid: string)	Удаление иконы
private RecursiveDelete(parent: DragonInstruction)	Рекурсивное удаление всех вложенных икон

У каждой иконы есть свой тип, а также наследуемый от DragonInstruction класс. Полный список реализованных икон представлен в таблице 2.31.

Таблица 2.31 – Типы икон

Тип	Описание	Реализуемый класс
SCHEMA	ДРАКОН-схема	DragonSchemaInstruction
PRIMITIVE	Макроикона «Примитив»	DragonPrimitiveInstruction
ACTION	Икона «Действие»	DragonActionInstruction
CONDITION	Макроикона «Вопрос»	DragonConditionInstruction
LIMITER	Ограничитель условных конструкций	DragonLimiterInstruction
BRANCH	Макроикона «Ветка»	DragonBranchInstruction
INPUT	Икона «Ввод»	DragonInputInstruction
OUTPUT	Икона «Вывод»	DragonOutputInstruction
LOOP	Макроикона «Цикл»	DragonLoopInstruction
SWITCH	Макроикона «Вариант»	DragonSwitchInstruction
COMMENT	Икона «Комментарий»	DragonCommentInstruction
SLEEP	Икона «Таймер»	DragonSleepInstruction

Каждая реализуемый класс иконы отличается от базового DragonInstruction реализацией конструктора. Для каждого типа макроикон производится дополнительная настройка в виде создания зависимых от неё вложенных икон, установление ссылок и добавление набора икон в словарь, который содержит ссылки на все иконы в схеме.

2.15 Реализованные меню и интерфейсы

При первом посещении веб-приложения пользователю отображается окно входа и регистрации. По умолчанию оно имеет вид, представленный на рисунке 2.1.

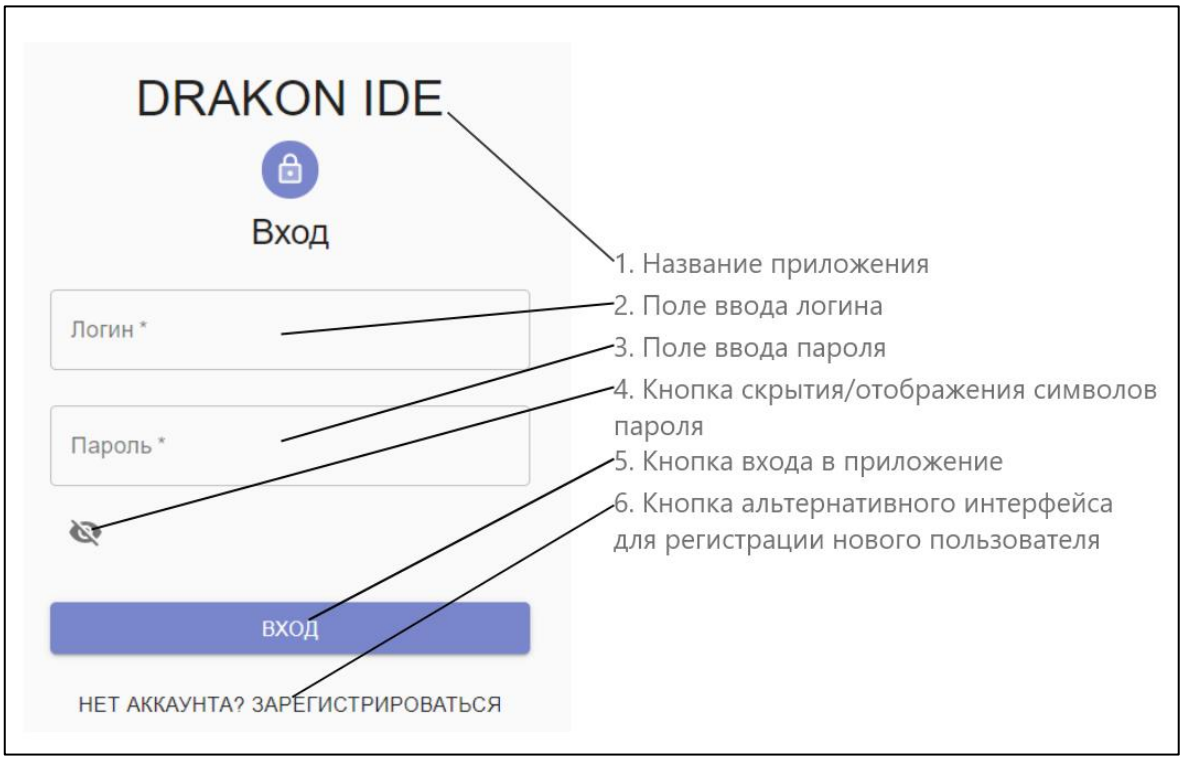

Рисунок 2.1 – Окно входа в систему

Для входа в систему пользователю необходимо ввести в поля 2 и 3 логин и пароль от своей учётной записи. После чего нажать кнопку «Вход», обозначенной на рисунке 2.1 цифрой 5.

В случае возникновения ошибок они будут выведены в специальном окне, его вид показан на рисунке 2.2.

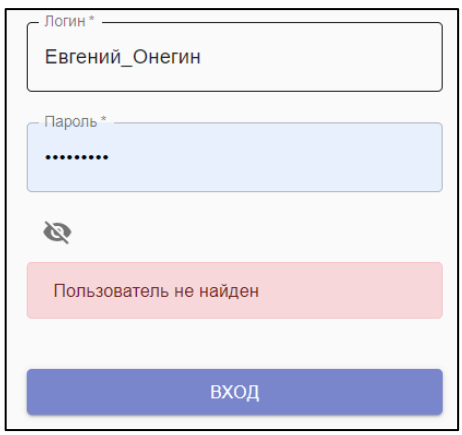

Рисунок 2.2 – Демонстрация окна ошибки

Если пользователь не зарегистрирован, то ему необходимо открыть форму регистрации. Для этого необходимо нажать кнопку 6 (см. рис 2.1). Откроется интерфейс регистрации нового пользователя. Его внешний вид представлен на рисунке 2.3

Рисунок 2.3 – Окно регистрации в системе

Каждый новый пользователь должен придумать уникальный идентификатор пользователя. Ввести ранее незарегистрированную в систему почту и придумать пароль. Все данные для регистрация вводятся в поля 2-5. После успешного заполнения полей необходимо нажать кнопку 7. Если регистрация пользователя произведена успешно, то он будет перенаправлен на главную страницу приложения – интерфейс редактор, в противном случае будет выведено сообщение с описанием ошибки (см. рис 2.2.).

Возврат на предыдущую форму (см. рис 2.1.) осуществляется через нажатие кнопки 8.

Для переключения отображения/скрытия текста в поле ввода пароля пользователя реализован специальный переключатель в виде кнопки с иконой с изображением глаза.

Интерфейс редактора продемонстрирован на рисунке 2.4.

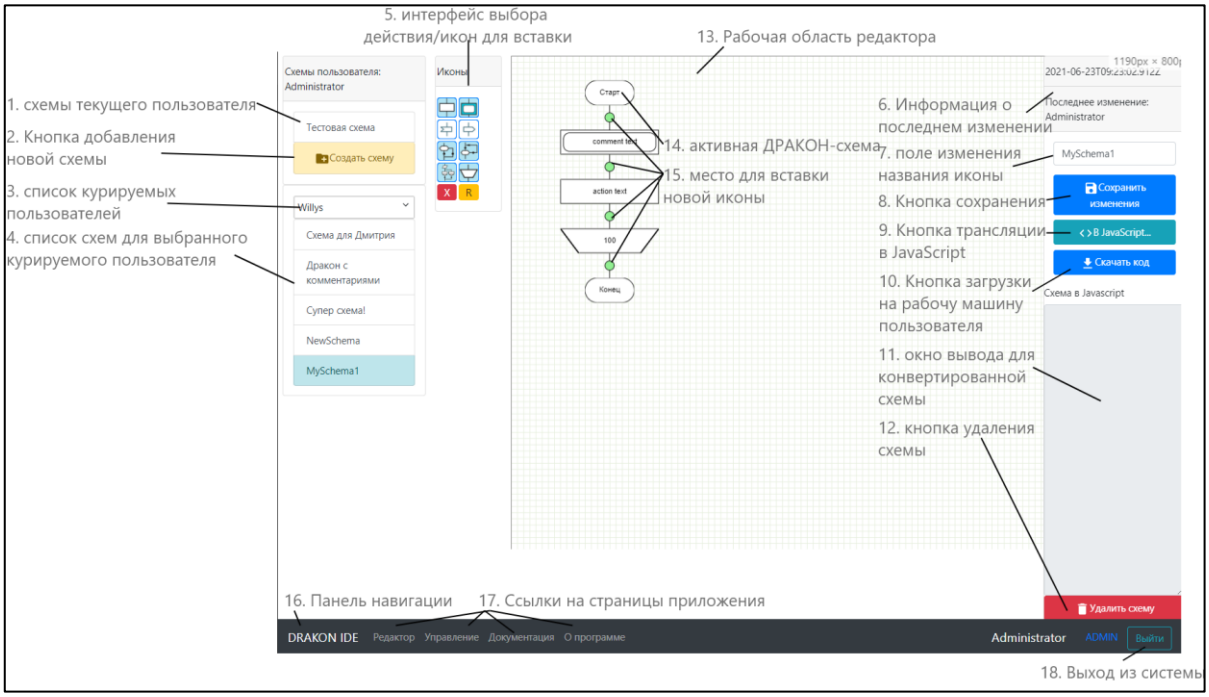

Рисунок 2.4 – Интерфейс ДРАКОН-редактора

Только что зарегистрированные пользователи не обладают ни одной ДРАКОН-схемой. Список всех ДРАКОН-схем пользователя выводится в окне интерфейса системы под номером 1 на рисунке 2.4.

Новые схемы создаются с помощью кнопки под номером 2. Новая схема после добавления выбирается как активная, что обозначается бирюзовым цветом в списке 1. Удаление ДРАКОН-схем осуществляется с помощью кнопки 12.

Ниже списка ДРАКОН-схем пользователя для кураторов и администраторов выводится список курируемых пользователей (элемент интерфейса 3 на рисунке 2.4). Ещё ниже выводится список всех схем выбранного пользователя.

Выбранная ДРАКОН-схема отображается в рабочей области редактора. Для изменения её содержимого необходимо выбрать в интерфейсе под номером 5 действие, которое пользователь хочет произвести. Пользователь может как удалять иконы, так и добавлять новые, все возможные иконы для вставки представлены в компоненте интерфейса 5.

Место для вставки новых икон помечается зеленым кругом, как показано на рисунке 2.4 в элементе интерфейса 15.

Для изменения содержимого иконы, необходимо двойным щелчком мыши нажать на любую икону, кроме иконы «Конец». После откроется специальная

форма, куда необходимо ввести изменения в текст иконы. Фокус устанавливается автоматически, ввод осуществляется с помощью клавиатуры, закрытие формы производится нажатием клавиши «Enter». Вид данной формы представлен на рисунке 2.5.

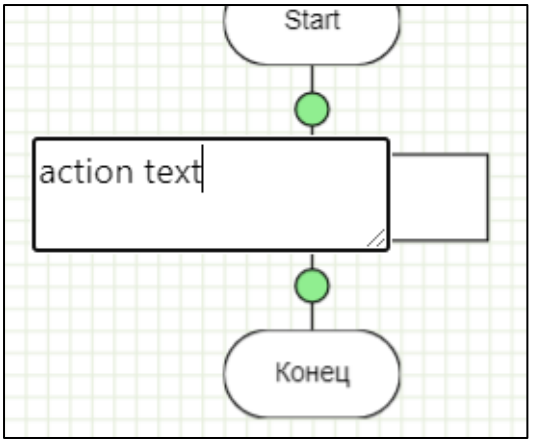

Рисунок 2.5 – Форма изменения содержимого иконы

Название схемы изменяется в поле под номером 7. Все изменения в ДРАКОН-схеме производятся локально. Для сохранения и выгрузки обновленной схемы в базу данных необходимо нажать клавишу под номером 8.

Клавиши 9 и 10 отвечают за трансляцию ДРАКОН-схемы в язык JavaScript. Кнопка под номером 9 осуществляет вывод программы после трансляции в специальную форму, которая на рисунке 2.4 указана под номером 11.

Кнопка под номером 10 сохраняет программу на компьютере пользователя. Имя файла имеет следующий формат: «[Название_схемы].js».

Под интерфейсом редактора располагается общая для всех страниц панель навигации, которая позволяет переключаться между доступными страницами (см. компонент интерфейса 17 на рис. 2.4). А также осуществить выход из системы при нажатии клавиши 18.

Помимо этого, в ней выводится имя текущего пользователя в правом нижнем углу. Рядом располагается отображение его роли в системе.

С помощью интерфейса навигации (см. рис 2.4) пользователь может открыть меню управления и администрирования. Количество отображаемых элементов интерфейса зависит от роли пользователя.

Страница управления для роли «Администратор» приведена на рисунке 2.6.

Рисунок 2.6 – Страница управления и администрирования

Каждый пользователь системы может изменять свои персональные данные. Для изменения имени пользователя необходимо нажать клавишу 1. Откроется компонент изменения логина. Аналогично осуществляется взаимодействие с кнопкой 2, которая открывает интерфейс изменения пароля. Вид интерфейсов изменения персональных данных представлен на рисунке 2.7.

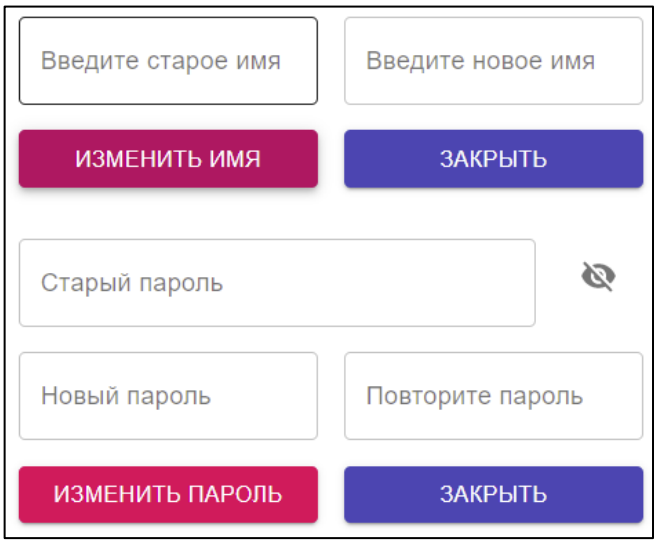

Рисунок 2.7 – Интерфейс изменения персональных данных.

В случае ввода некорректных данных пользователю будет показано информационное сообщение с описанием ошибки и способами устранения.

Содержимое окна изменяется в зависимости от типа получаемой ошибки. Пример внешнего вида такого окна представлен на рисунке 2.8.

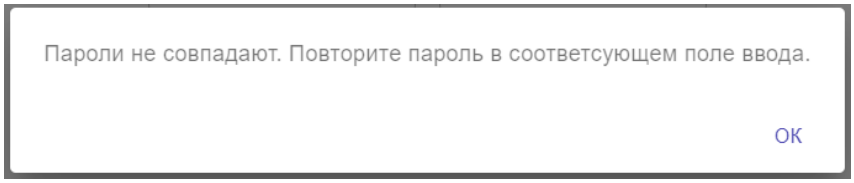

Рисунок 2.8 – Интерфейс окна уведомления.

Пользователь в любой момент может удалить свою учётную запись пользователя. Для этого необходимо нажать клавишу 3. Будет выведено окно подтверждения, где пользователю необходимо решить: готов ли он к удалению своей учётной записи. Интерфейс окна подтверждения представлен на рисунке 2.9.

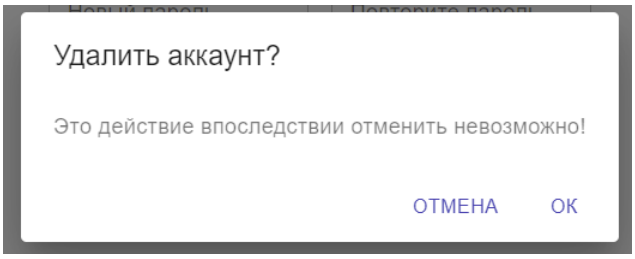

Рисунок 2.9 – Интерфейс окна подтверждения.

Элемент интерфейса под номером 4 на рисунке 2.6 виден только кураторам и администраторам. Он содержит Список пользователей системы, статус: является ли пользователь курируемым для текущей учётной записи. Статус отображается с помощью иконы «Сердце». Если в таблице «Курируемое отношение» существует соответствующая запись, то в компоненте интерфейса под номером 7 будет выведено название для этого отношения, которое пользователь может изменять с помощью клавиатуры.

Пункты интерфейса 8 и 9 для изменения доступны только Администраторам. Куратор видит роль пользователи, но изменить её не может.

Кнопка удаления пользователя доступна только администратору, перед удалением будет выведено окно подтверждения (см. рис. 2.9).

Интерфейс документации содержит всю необходимую информацию о языке ДРАКОН, способах представления икон и способах трансляции в язык JavaScript. Документация представлена в виде файла «Markdown», являющийся удобным форматом для представления документации.

Интерфейс документации представлен на рисунке 2.10.

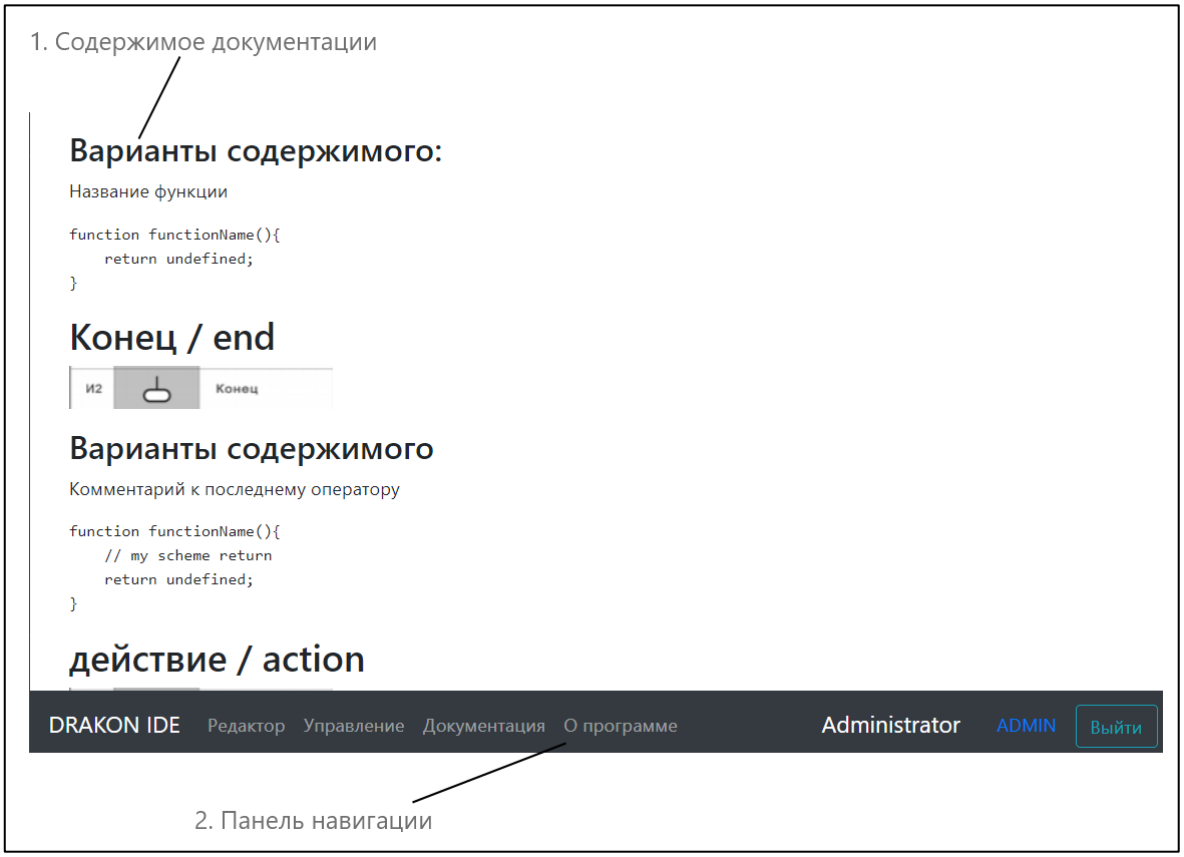

Рисунок 2.10 – Интерфейс страницы документации.

Последней страницей в панели навигации является «О программе», в которой кратко указана информацию о разрабатываемой системе, её разработчике. А также размещена кнопка перехода на страницу с репозиторием проекта на сайте «Github». Внешний вид данной страницы представлен на рисунке 2.11

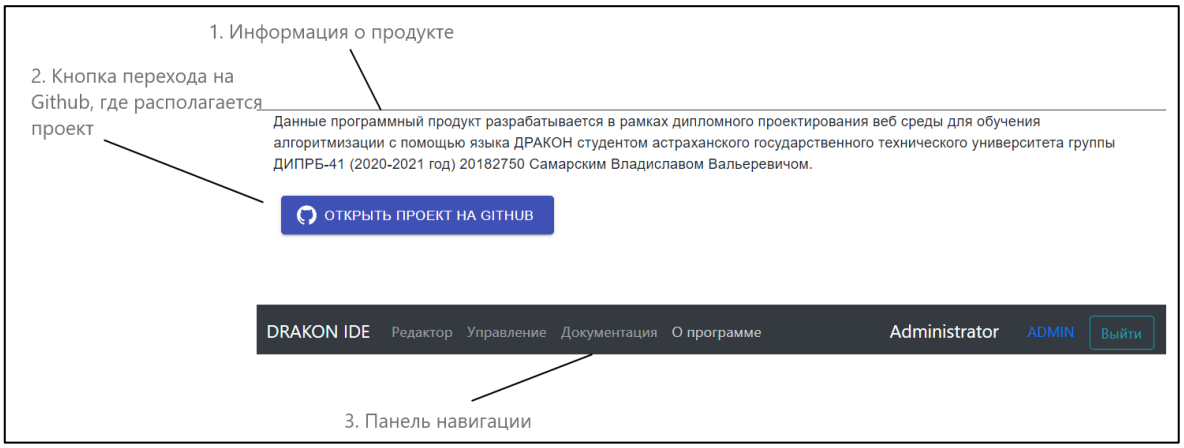

Рисунок 2.11 – Интерфейс страницы «О программе».

При попытке перейти на несуществующую страницу будет отображено уведомление, что страницы не существует. Интерфейс уведомления показан на рисунке 2.14.

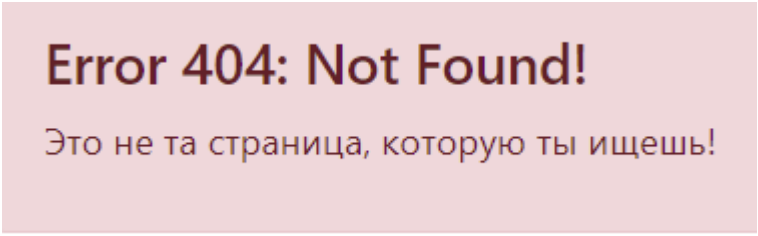

Рисунок 2.12 – Интерфейс страницы «Не найдено»

2.16 Сообщения системы

ИС осуществляет общение с пользователем с помощью информационных сообщений. Все ошибки отображаются в виде всплывающих сообщений, в которых сообщается вся требуемая информация для пользователя, исключая технические подробности.

В таблице 2.32 приведены все основные сообщения системы.

Таблица 2.32 – сообщения системы

№	Сообщение системы	Описание
1	Нет ответа от сервера	Сервер недоступен
2	Пароли не совпадают. Повторите пароль в соответствующем поле ввода.	Пользователь не смог продублировать придуманный пароль в интерфейсе регистрации
3	Удалить аккаунт? Это действие впоследствии отменить невозможно!	Пользователь удаляет учетную запись. Требуется подтверждение
4	Имя пользователя успешно изменено!	Пользователь изменил логин учётной записи
5	Проверьте ввод! Старое имя введено некорректно!	Введенное имя не совпадает с именем пользователя
6	Имя пользователя должно состоять из букв латинского и кириллического алфавита, цифр и знака "_"	Имя пользователя содержит запрещенные символы

Продолжение таблицы 2.32

№	Сообщение системы	Описание
7	Поля пароля не должны быть пустыми	Пользователь не ввел пароль в интерфейсе регистрации или в интерфейсе смены пароля
8	Текущий пароль введен неверно	Пароль от учётной записи введен неверно в интерфейсе смены пароля
9	Пароль успешно изменен	Пароль пользователя успешно изменен на новый
10	Слабый пароль. Пароль должен состоять из букв латинского алфавита, содержать хотя бы 1 заглавную и 1 прописную букву или цифру. Длина не менее 6 символов	Новый пароль не удовлетворяет требованиям безопасности
11	Пользователя не существует	Пользователь не может быть найден в базе данных
12	Вы не администратор	Пользователь пытается воспользоваться привилегиями администраторы через API вызовы.
13	Отношения не существует	Не найдено отношение между куратором и пользователем в базе данных
14	Схема не найдена	ДРАКОН-схема не найдена в базе данных
15	Пользователь не найден	Пользователь не найден в базе данных
16	Некорректный токен	JWT токен некорректен или неактуален
17	Не предоставлена почта	Пользователь не ввел почту в интерфейсе регистрации, или формат записи не интерпретируется как почта
18	Неправильный пароль	Пользователь неверно ввел пароль в интерфейсе входа

В случае возникновения других сообщений необходимо обратиться к разработчику программного продукта.

3 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

Тестирование интерфейсов ДРАКОН-редактора:

1. Пользователь открывает браузер и открывает главную страницу приложения (по умолчанию URL приложения localhost:5000). Система должна отобразить интерфейс входа в систему (см.рис. 2.1).
2. Пользователь нажимает кнопку «Нет аккаунта? Зарегистрироваться», в ответ система открывает интерфейс регистрации нового пользователя (см. рис 2.3).
3. В поле логина пользователь должен ввести имя «Евгений_Онегин». Пароль – «Pushkin». Почту не указывать. В поле дублирования пароля скопировать пароль и вставить. После нажать клавишу «Регистрация». Убедиться, что было выведено сообщение 17.
4. Ввести данные почты «sample@sample.com». Нажать повторно клавишу «Регистрация». Система должна осуществить перенаправление на страницу редактора (см. рис 2.4).
5. Пользователь нажимает кнопку «Создать схему». В ответ система генерирует базовый примитив ДРАКОН-схемы, сохраняет его в базе данных и возвращает результат пользователю, отображая новую добавленную схему с именем «NewSchema» в списке схем пользователя (см. рис 2.4).
6. Пользователь должен убедиться, что созданная схема отображена и выделена как активная, а её содержимое отображается на основной рабочей области редактора.
7. Пользователь выбирает икону «Действие» в интерфейсе выбора икон для вставки. В ответ система подсвечивает для активной ДРАКОН-схемы места для вставки с помощью зеленого круга (см. рис 2.5). Пользователь должен убедиться, что место для вставки есть только одно.
8. Пользователь клавишей мыши нажимает на зеленый круг. В ответ система осуществляет вставку иконы в ДРАКОН-схему. Пользователь должен убедиться, что система отобразила вставленную икону.
9. Пользователь повторяет пункт 8 для одного из зеленых кругов. Убеждается, что иконы выглядят одинаково и их содержимое ничем друг от друга не отличается.

10. Пользователь выбирает икону «Таймер» в интерфейсе выбора икон для вставки, повторяет пункт 8 для самого первого доступного места вставки в схеме. Система генерирует другое представление для иконы и отображает её в рабочей области.
11. Пользователь двойным щелчком мыши нажимает на икону «Таймер». В ответ система открывает окно редактирования содержимого иконы (см. рис 2.5).
12. Пользователь стирает содержимое и вводит текст «2000» и нажимает клавишу «Enter». Система закрывает окно редактирования и отображает изменения.
13. Пользователь выбирает кнопку «X» в интерфейсе выбора действия. Система в ответ окрашивает в красный все три иконы, которые были добавлены в схему.
14. Пользователь нажимает кнопкой мыши на вторую красную иконы «Действие». В ответ система осуществляет удаление иконы из схемы. Пользователь должен убедиться, что система отобразила изменения.
15. Пользователь повторяет пункт 12 для оставшейся иконы «Действие» и вводит текст «let b =10». Система в ответ должна обновить содержимое иконы.
16. Пользователь повторяет пункт 8 для иконы «Вывод» для самой последней иконы в схеме. Убеждается, что в конец система добавила соответствующую икону.
17. Пользователь повторяет пункт 12 для иконы «Вывод». Вводит текст «b» . Система отображает изменение в иконе.
18. Пользователь нажимает клавишу «В JavaScript…» в интерфейсе редактора. В ответ система осуществляет трансляцию текущей иконы и выводит содержимое в соответствующем компоненте интерфейса (см. рис 2.4).
19. Пользователь нажимает клавишу «Скачать код». В ответ система осуществляет генерацию файла «NewSchema.js» и загружает её на машину клиента через функционал браузера.
20. Пользователь открывает консоль браузер и копирует в неё код из файла «NewSchema.js». Добавляет в конец текст «Start()» и нажимает клавишу

Enter. Браузер выполняет код, который передает пользователь и выводит сообщение «10» в консоль.

21. Пользователь закрывает консоль браузера. Вводит имя «Моя схема» в соответствующем поле для ввода имени схема и нажимает клавишу «Сохранить изменения». Система обновляет данные схемы и возвращает пользователю обновленную, с новым названием и сохранёнными добавленными иконами.
22. Чтобы убедиться, что изменения сохранены, пользователю необходимо повторить пункт 5. Пользователь должен перезагрузить приложение или осуществить повторный вход с целью проверки, что система генерирует разные отображения для каждой из ДРАКОН-схем.
23. Завершить тестирование интерфейса редактора

Тестирование интерфейса управления и администрирования:

1. Пользователь входит в систему, введя следующие регистрационные данные: логин – Administrator, пароль – Administrator. Система в ответ на нажатие клавиши «Вход» должна отобразить интерфейс редактора.
2. В меню навигации пользовать должен нажать на ссылку «Управление». В ответ система откроет соответствующую страницу (см. рис. 2.6).
3. Пользователь нажимает клавишу «Изменить имя пользователя». Вводит в соответствующее поле свой логин – Administrator. В поле «Введите новое имя» - «Главный_Администратор», и нажимает клавишу «Изменить имя». В ответ система выведет сообщение 4, а в панели навигации обновится имя текущего пользователя.
4. Пользователь нажимает кнопку «Выйти». Повторяет пункт 1. Убеждается, что было выведено сообщение 15.
5. Пользователь изменяет логин на «Главный_Администратор». Убеждается, что система осуществила вход.
6. Пользователь нажимает клавишу «Изменить пароль», вводит пароль из пункта 1 в качестве старого. В качестве нового вводит пароль «WeakPassword» и нажимает клавишу «Изменить пароль». В ответ система обновляет данные о пароле пользователя.
7. Для проверки изменений пользователь повторяет пункт 4, но в этот раз должно быть выведено сообщение 18.

8. Пользователь повторяет вход с правильным паролем из пункта 6, убеждается, что система осуществила авторизацию пользователя и перенаправила его на страницу управления.
9. В списке, где указано «Начните вводить имя пользовался», пользователь должен нажать на стрелку. Система загрузит список пользователей и выведет их на экран. Выбрать в списке пользователя «Евгений_Онегин».
10. Убедиться, что икона «Курируемый» не светится розовым. Нажать на неё, в ответ система поменяет её цвет, а пользователя добавит в список курируемых.
11. Для проверки, что пользователь добавлен в таблицу «Курируемое отношение» необходимо открыть интерфейс редактора. Открыть выпадающий список в левой части интерфейса (см. рис. 2.4) и убедиться, что имя пользователя «Евгений_Онегин» там присутствует.
12. Нажать на пользователя «Евгений_Онегин». В ответ система загрузит из базы данных схемы выбранного пользователя и отобразит их в виде списка.
13. Открыть схему «Моя схема», убедиться, что её содержимое отображается в редакторе. Добавить любую икону в схему и нажать сохранить.
14. Открыть схему повторно, убедиться, что изменения сохранены и логин пользователя, который произвел последнее изменение совпадает с логином текущего пользователя.
15. Вернуться на страницу управления. Установить роль «Куратор» для пользователя «Евгений Онегин». Для этого пользователь должен выбрать в списке соответствующий вариант и нажать кнопку «Установить роль». В ответ система осуществит обновление данных пользователя, в частности обновит его роль в системе.
16. Убедиться, что роль «Евгений_Онегин» стала «Куратор». Нажать клавишу «Удалить пользователя». Убедиться, что было выведено уведомление (см. рис 2.9) с сообщением 3.
17. Пользователь нажимает клавишу подтверждения. В ответ система удаляет пользователя из базы данных.
18. Пользователь повторяет пункт 11, убеждается, что пользователь «Евгений_Онегин» не фигурирует в списке, аналогично для списка из пункта 9.

19. Завершить тестирование интерфейса администрирования

Тестирование путей приложения:

1. Для тестирования путей приложения пользователь последовательно нажимает на ссылке в панели навигации. Система в ответ предоставляет соответствующий интерфейс страницы.
2. Пользователь открывает интерфейс администрирования и добавляет любую букву в адресную строку в конец URL. После нажимает клавишу ENTER. В ответ система отобразит сообщение о невозможности найти страницу (см. рис. 2.12).

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была разработана многопользовательская единая веб-среда обучения основам алгоритмизации с помощью языка ДРАКОН.

Программный продукт отвечает поставленным требованиям и может быть использован для автоматизации процесса обучения проектированию алгоритмов.

Текущая версия программного продукта может быть усовершенствована с помощью:

- реализации всех основных макроикон языка ДРАКОН;
- расширения стандарта языка ДРАКОН для сопоставления икон с конструкциями языка JavaScript;
- создания специальных тестовых заданий для выполнения учениками.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Браун Э. Изучаем JavaScript. Руководство по созданию современных веб-сайтов 3-е изд. –М.: O'REILLY, 2017. – 363 с.
2. Плаксин М.А. Тестирование и отладка программ для профессионалов будущих и настоящих [Электронный ресурс] 2-е изд. (эл.) – М.: БИНОМ. Лаборатория знаний, 2013. – 167 с.: ил.
3. Паронджанов В.Д. Язык ДРАКОН, краткое описание. 4-е изд. – М.: Дело, 2002 – 124 с.
4. Леоненков, А.В. Самоучитель UML. – 2-е изд., перераб. и доп. – СПб., БХВ-Петербург, 2004. – 432 с.: ил.
5. Якобсон А., Буч Г., Рамбо Дж. Унифицированный процесс разработки программного обеспечения. – СПб.: Питер, 2002. – 496 с: ил.
6. Файн Я., Моисеев А. TypeScript быстро . – М.:ПИТЕР-СПБ 2020 г. – 528с.
7. Мардан А. React быстро. Веб-приложения на React, JSX, Redux и GraphQL 1-е изд. – М.:ПИТЕР-СПБ 2019 г. – 560с.
8. Ханс-Юрген Шониг, Mastering PostgreSQL 12: Advanced Techniques to Build. 3-е изд. – М.: Packt Publishing 2019г. – 470с.
9. ГОСТ 19.201-78 ЕСПД. Техническое задание. Требование к содержанию и оформлению – М.: Стандартинформ, 1980. – 4 с.
10. ГОСТ 19.301-79 ЕСПД. Программа и методика испытаний. Требования к содержанию и оформлению – М.: Стандартинформ, 1980. – 2 с.
11. ГОСТ 19.106-78 ЕСПД.. Единая система программной документации. Требования к программным документам, выполненным печатным способом – М.: Стандартинформ, 2010. – 12 с.
12. ГОСТ 19.701-90 (ИСО 5807-85). ЕСПД. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М.: Изд-во стандартов, 1991. –26 с.
13. ГОСТ Р 52633.0-2006 Защита информации. Основные термины и определения. – М.: Изд-во стандартов, 2008. – 8 с.
14. ГОСТ Р 58833-2020 Защита информации. Идентификация и аутентификация. – М.: Стандартинформ, 2020. – 27 с.

ПРИЛОЖЕНИЕ 1

Диаграмма классов ДРАКОН-схемы

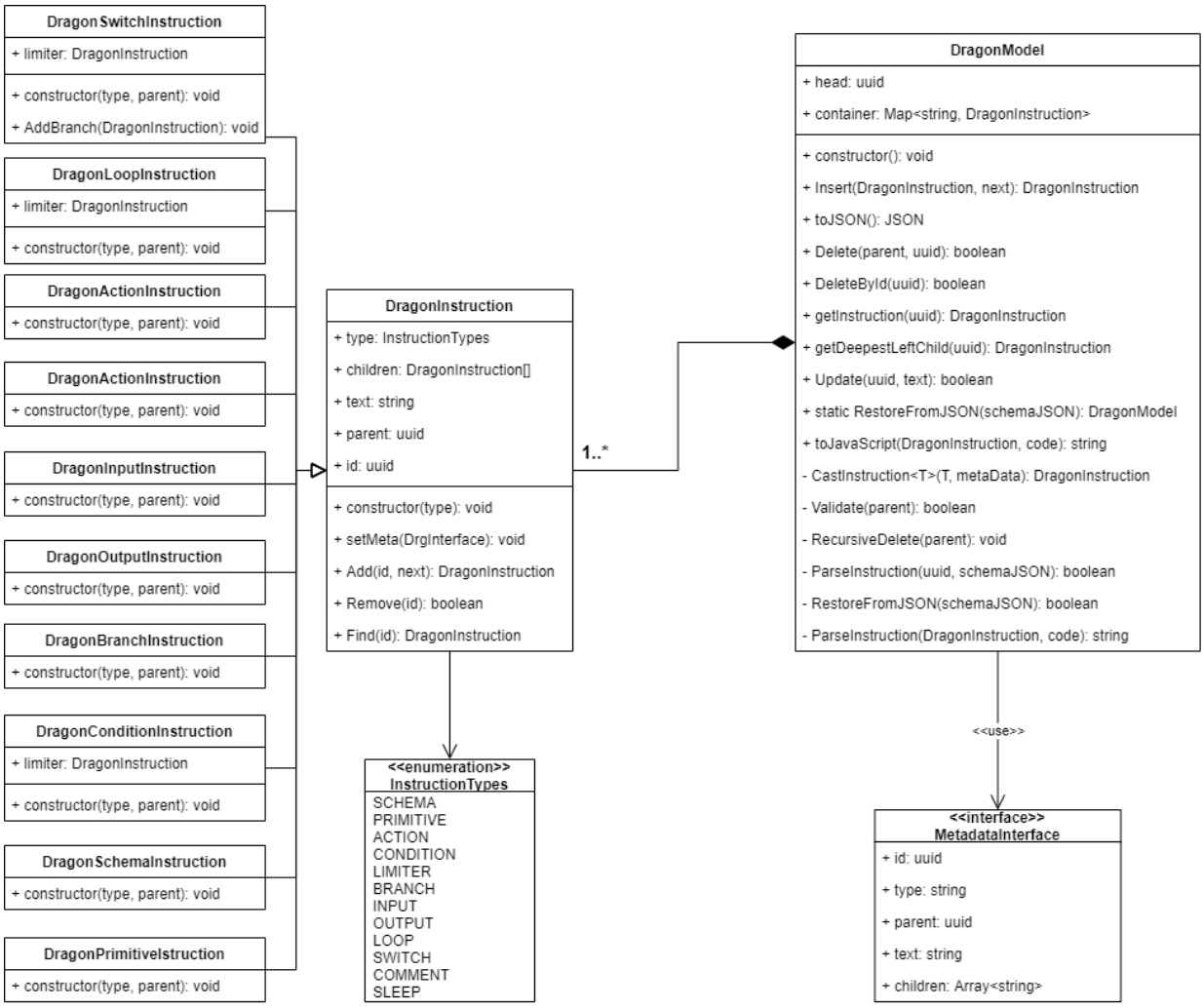

Рисунок П.1 – Диаграмма классов для представления ДРАКОН-схем

ПРИЛОЖЕНИЕ 2

Диаграмма сущность-связь

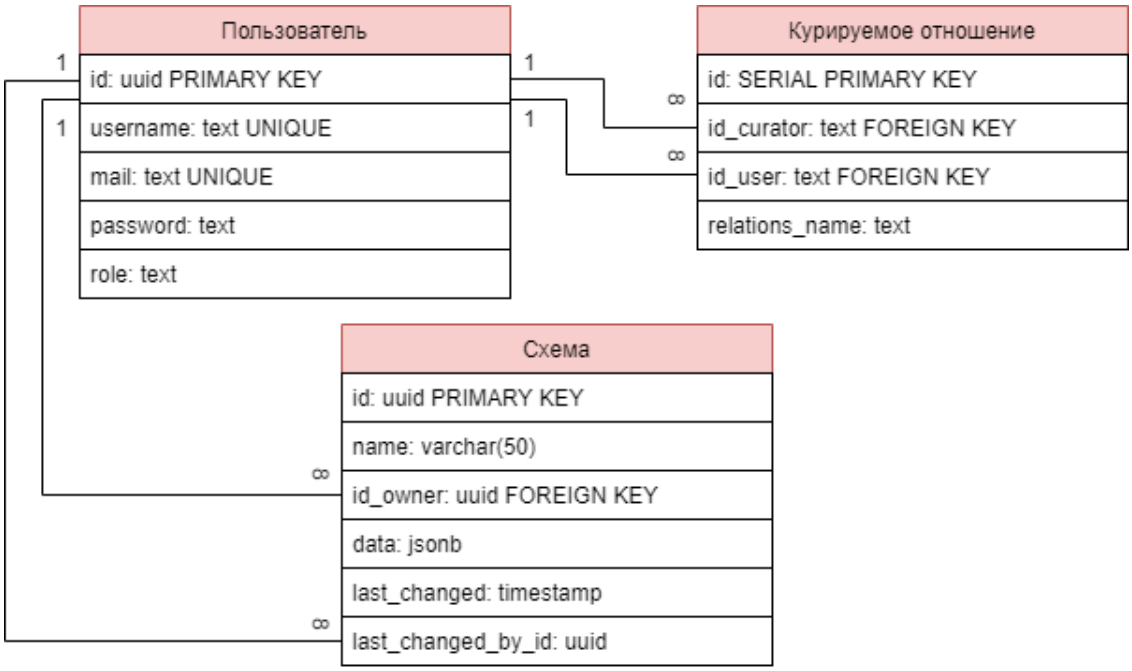

Рисунок П.2 – Диаграмма сущность-связь базы данных «БД_Дракон»

ПРИЛОЖЕНИЕ 3

Диаграмма развертывания

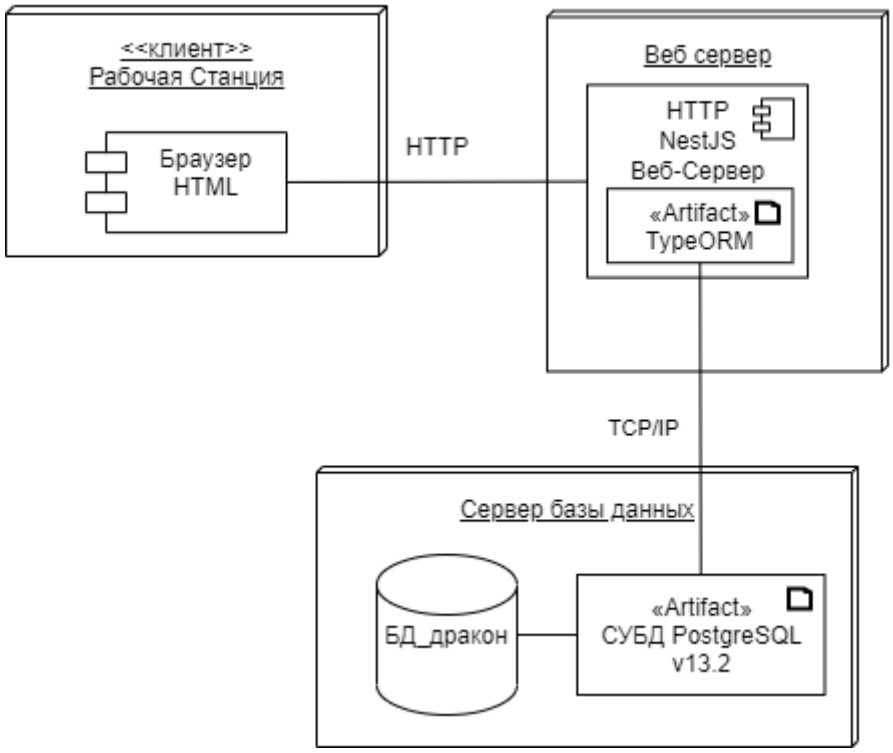

Рисунок П.3 – Диаграмма развертывания среды по обучению алгоритмизации DRAKON IDE

ПРИЛОЖЕНИЕ 4

Диаграмма компонентов сервера

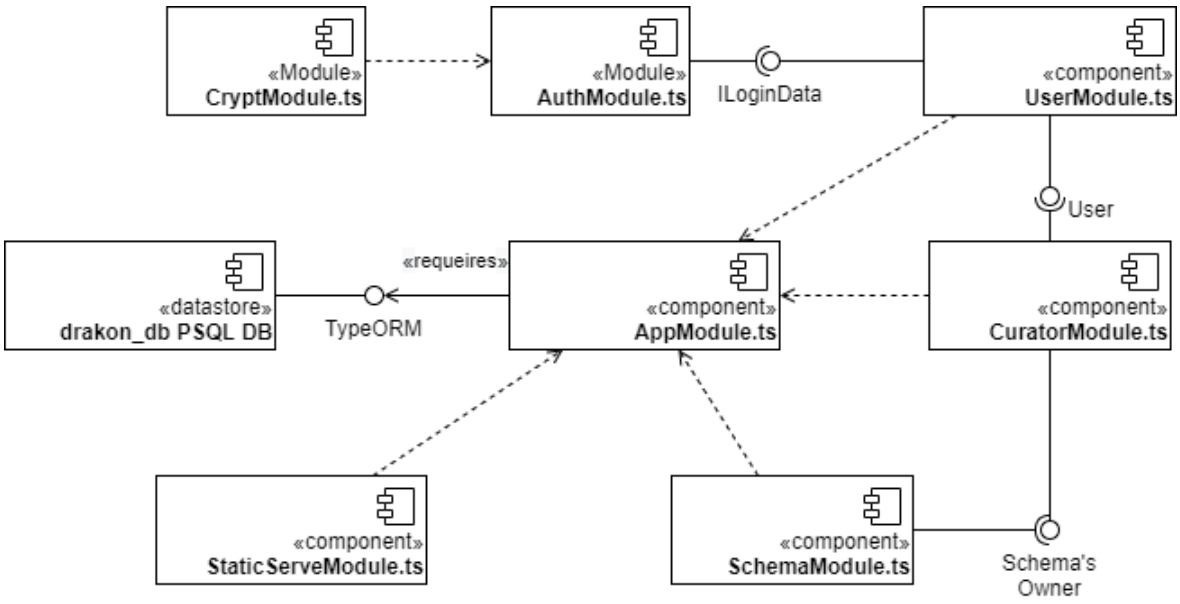

Рисунок П.4 – Диаграмма компонентов сервера NestJS