|  |  |  |
| --- | --- | --- |
|  | ***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 09.03.04 Программная инженерия  ПрофильРазработка программно-информационных систем  КафедраАвтоматизированные системы обработки информации и управления  от \_\_\_\_.\_\_\_\_.202\_\_ № \_\_\_\_\_\_  **ОТЧЕТ ПО ПРАКТИКЕ**  Вид практики: производственная  Место прохождения практики: ***ФГБОУ «АГТУ», кафедра АСОИУ*** | | |
|  | | Отчет выполнил:  обучающийся группы ДИПРб-31  \_\_\_\_\_\_\_\_\_\_\_Самарский Владислав Валерьевич |
| Руководитель практики  от профильной организации  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н. доц.  Лаптев Валерий Викторович  «\_\_\_\_» \_\_\_\_\_\_\_\_\_ 202\_\_ г. | | Руководитель практики от Университета  к.т.н., доцент.  \_\_\_\_\_\_\_\_\_\_\_\_ Седова Яна Анатольевна  Результаты защиты отчета  Оценка, полученная на защите  «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»  Члены комиссии:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_ 202\_\_ г. |
| Астрахань 2020 | | |

**ИНДИВИДУАЛЬНЫЙ ПЛАН/ЗАДАНИЕ**

**Вид практики**: учебная / производственная / преддипломная

**Тип практики:***Производственно-технологическая практика*

**Способ проведения практики:** выездная / стационарная

**Обучающийся**       *Самарский Владислав Валерьевич,*

**Курс   3**    **группа**  *ДИПРб-31*

**Направление**  *09.03.04   Программная  инженерия* .  
**Профиль**         *Разработка  программно-информационных  систем* .  
**Место проведения практики** Кафедра АСОИУ .

**Задание:**

Проектирование интегрированной среды для обучения алгоритмизации на языке ДРАКОН, разработка серверной части и базы данных для проектируемой системы. Система должна удовлетворять следующим требованиям:

проектируемая система должна быть реализована в виде веб-приложения, предоставляющее возможность создавать и редактировать дракон-схемы с возможностью загрузки и сохранения схем, конвертации в синтаксис языка JavaScript.

**Объем и краткое содержание (виды работ) практики**

| **№**  **п/п** | **Раздел практики** | **Формы текущего контроля успеваемости** |
| --- | --- | --- |
| **1** | **1 этап (подготовительный)**  1. Получение индивидуального задания на практику.  2. Инструктаж по технике безопасности при работе на персональном компьютере и правилами внутреннего распорядка на предприятии  3. Ознакомление с основными видами деятельности организации и его организационной структурой | Регистрация в журнале по технике безопасности |
| **2** | **2 этап (теоретический )**  **Выполнение индивидуального задания: анализ предметной области**  1. Проведение анализа предметной области в соответствии с утвержденным планом;  2. Поиск и анализ информации по индивидуальному заданию; сбор данных, необходимых для решения поставленных задач.  3 Разработка технического задания, технического проекта, а также программы и методики испытаний программного продукта | Техническое задание.  Разделы отчёта по практике «Технический проект» и «Программа и методика испытаний» для разрабатываемого программного продукта |
| **3** | **3 этап (экспериментальный)**  **Выполнение индивидуального задания: обработка и анализ полученной информации**  1. Разработка и тестирование программного продукта.  2. Обработка и анализ полученной информации, сравнение полученных результатов с известными данными, при необходимости – внесение изменений в программный код.  3. Обоснование полученных выводов. | Раздел отчёта по практике – «Рабочий проект» для разработан­ного программного продукта |
| **4** | **4 этап: подготовка отчета по практике**  Подготовка отчета, в котором должны быть отражены результаты работы.  Составление пояснительной записки в виде файла и твердой копии.  Подготовка презентации для выступления на защите в комиссии кафедры | Материал по результатам работы: программа, отвеча­ющая поставленным требованиям, письмен­ный отчет, электрон­ная презентация |
| **5** | **Заключительный этап:**  Защита отчета по практике на кафедре | Отчет по результатам практики |
|  | **Форма отчетности по практике** | Зачет с оценкой |

Руководитель практики от Университета: *к.т.н., доц. Седова Яна Анатольевна*

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_ г. \_\_\_\_\_\_\_\_\_\_

*подпись*

Задание получил: *Самарский Владислав Валерьевич*

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_ г. \_\_\_\_\_\_\_\_\_\_

*подпись*

**Согласовано:**

Руководитель практики от профильной организации

*к.т.н. доц. Лаптев Валерий Викторович*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАБОЧИЙ ГРАФИК (ПЛАН) ПРОВЕДЕНИЯ ПРАКТИКИ**

2019/2020 учебный год

**Направление**  *09.03.04  Программная  инженерия* .  
**Профиль**         *Разработка  программно-информационных  систем* .  
**Курс**                   3,    **группа**  *ДИПРб-31* .

**Место прохождения практики** Кафедра «АСОИУ» .

Руководитель практики от Университета    к.т.н., доц.  Седова Я.А .

**Вид практики:** учебная / производственная / преддипломная

**Тип практики:***Производственно-технологическая практика[[1]](#footnote-1)*

**Способ проведения практики**: выездная/стационарная

**Срок прохождения практики:** с 13.07.2020 по 19.07.2020.

|  |  |
| --- | --- |
| **Дата (сроки)** | **Планируемые формы работы (раздел практики)** |
| 13.07.2020 | Знакомство с правилами внутреннего распорядка, изучение структуры организации, инструктаж по технике безопасности.  Получение индивидуального задания на практику |
| 14.07.2020 | Анализ предметной области, сбор информации, необходимой для разработки технического задания, технического проекта и программы испытаний программного продукта |
| 15.07.2020 | Проектирование интерфейсов разрабатываемой системы. Проектирование серверной части программного продукта |
| 16.07.2020 | Тестирование программного продукта.  Обработка и анализ полученной информации, сравнение полученных результатов исследований с известными данными, при необходимости – внесение изменений в программный код. |
| 17.07.2020 | Подготовка отчета. Составление пояснительной записки в виде файла и твердой копии. Подготовка презентации для выступления на защите в комиссии кафедры |
| 18.07.2020 | Защита практики |

Руководитель практики от университета *к.т.н., доц. Седова Яна Анатольевна*

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_ г. \_\_\_\_\_\_\_\_\_\_

*подпись*

**Согласовано:**

Руководитель от профильной организации

к.т.н. доц. Лаптев Валерий Викторович

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_ г. \_\_\_\_\_\_\_\_\_\_

Обучающийся *Самарский Владислав Валерьевич*

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_ г. \_\_\_\_\_\_\_\_\_\_

*подпись*

**ДНЕВНИК ПРАКТИКИ**

**Вид практики:** учебная / производственная/ преддипломная

**Способ проведения практики:** выездная/стационарная

**Обучающийся** *Самарский Владислав Валерьевич*, группа *ДИПРб-31*

**Направление**  *09.03.04  Программная  инженерия*  
**Профиль**         Разработка  программно-информационных  систем.

**Место проведения практики** кафедра «АСОИУ»

Дата начала практики «13» июля 2020 г.

Дата окончания практики «19» июля 2020 г.

| **Дата** | **Наименование и ход работ,** | **Краткое описание работы** |
| --- | --- | --- |
| 13.07.2020 | Знакомство с организацией, в которой проводится практика. Цели и задачи практики. Техника безопасности, правила внутреннего распорядка. | Изучена инструкция по технике безопасности и правила внутреннего распорядка. Получено задание на практику. |
| 14.07.2020 | Обследование предметной области. Разработка технического задания. | Собран и обследован материал предметной области |
| 15.07.2020-16.07.2020 | Проектирование БД для хранения ДРАКОН-схем | Составлен прототип базы данных в докер контейнере |
| 17.07.2020 | Проектирование сервера для обслуживания БД | Разработан сервер NestJS для обслуживания базы PostgreSQL |
| 18.07.2020 | Защита практики |  |
| Руководитель практики  от профильной организации  *к.т.н. доц. Лаптев Валерий Викторович*  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_ г. \_\_\_\_\_\_\_\_\_\_ | | Руководитель практики от Университета к*.т.н., доцент. Седова Яна Анатольевна* «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_ г. \_\_\_\_\_\_\_\_\_\_  *подпись* |
| Обучающийся *Самарский Владислав Валерьевич*  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_ г. \_\_\_\_\_\_\_\_\_\_  *подпись* | | |

**СОДЕРЖАНИЕ**

[Введение 8](#_Toc69296695)

[1 Технический проект 10](#_Toc69296696)

[1.1 Анализ предметной области 10](#_Toc69296697)

[1.1.1 Язык ДРАКОН 10](#_Toc69296698)

[1.1.2 Правила написания дракон-схем 12](#_Toc69296699)

[1.2 Технология обработки информации 14](#_Toc69296700)

[1.2.1 Спецификация вариантов использования 15](#_Toc69296701)

[1.2.2 Шифрование паролей 16](#_Toc69296702)

[1.2.3 Описание способов взаимодействия пользователя с ДРАКОН-схемой 16](#_Toc69296703)

[1.2.4 Описание методов представления ДРАКОН икон 17](#_Toc69296704)

[1.2.5 Описание методов представления ДРАКОН-схем 18](#_Toc69296705)

[1.2.6 Алгоритм вставки простой иконы 19](#_Toc69296706)

[1.2.7 Алгоритм вставки комплексных икон 20](#_Toc69296707)

[1.2.8 Алгоритм удаления икон 21](#_Toc69296708)

[1.3 Инфологическая модель данных 21](#_Toc69296709)

[1.3.1 Сущность базы данных 21](#_Toc69296710)

[1.3.2 Диаграмма сущность-связь 22](#_Toc69296711)

[1.4 Входная и выходная информация 23](#_Toc69296712)

[1.4.1 Входная информация 23](#_Toc69296713)

[1.4.2 Выходная информация 24](#_Toc69296714)

[1.4 Формат содержимого ДРАКОН-схемы 25](#_Toc69296715)

[1.5 Требования к техническому и программному обеспечению 26](#_Toc69296716)

[2 Рабочий проект 27](#_Toc69296717)

[2.1 Общие сведения о работе системы 27](#_Toc69296718)

[2.2 Функциональное назначение программного продукта 27](#_Toc69296719)

[2.3 Выполнение программного продукта 28](#_Toc69296720)

[2.4 Описание физической архитектуры системы 28](#_Toc69296721)

[2.5 Описание программы 28](#_Toc69296722)

[2.6 Даталогическая модель данных 30](#_Toc69296723)

[2.7 Определение целостности сущностей 31](#_Toc69296724)

[2.8 Определение целостности атрибутов 32](#_Toc69296725)

[2.9 Определение ссылочной целостности 32](#_Toc69296726)

[2.10 Основные пользовательские функции, триггерные функции, триггеры 33](#_Toc69296727)

[2.11 Использование JSON-объектов 33](#_Toc69296728)

[2.12 Обеспечение безопасности базы данных 34](#_Toc69296729)

[2.13 План обслуживания и резервного копирования базы данных 34](#_Toc69296730)

[2.14 Описание основных классов системы 34](#_Toc69296731)

[2.15 Реализованные меню и интерфейсы 39](#_Toc69296732)

[2.16 Сообщения системы 40](#_Toc69296733)

[3 Программа и методика испытаний 41](#_Toc69296734)

[Заключение 42](#_Toc69296735)

[Список использованных источников 43](#_Toc69296736)

[Приложение 1 Диаграмма классов ДРАКОН-схемы 44](#_Toc69296737)

[Приложение 2 Диаграмма сущность-связь для базы данных 45](#_Toc69296738)

[Приложение 3 Диаграмма развертывания 46](#_Toc69296739)

# 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 подготовить к более комплексным 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. В итоге получается продукт без синтаксический 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 вопрос, а что будет, если заменить текстовое программирование на визуальное? Научное сообщество ответило на этот вопрос в 1996 году, представив язык ДРАКОН – Дружелюбный Русский Алгоритмический Язык, 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 описанию 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.

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:

* платформонезавимость. 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 практической работы является проектирование клиент-серверной информационной системы, которая осуществляет хранение в базе данных пользовательских ДРАКОН-схем, распределяет роли между пользователями системы с целью автоматизации процесса обучения и отображает графическое представление ДРАКОН-схемы.

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

# ТЕХНИЧЕСКИЙ ПРОЕКТ

## **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. Иконы – минимальная единица в алгоритме, представляет какое-либо действие, событие. Объединяясь в группы иконы формируют макроиконы.

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



Рисунок 1.1 – Икона «Заголовок»

Икона «Конец» 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 (рис 1.2).



Рисунок 1.2 – Икона «Конец»

Икона «Действие» содержит в себе некую операцию, проводимую над доступными данными, которая проводится безусловно (рис 1.3).



Рисунок 1.3 – Икона «Действие»

Икона «Вопрос» представляет из себя развилку из двух вариантов, в зависимости от результата проверки условия, хранящегося в главной иконе (рис 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 называются макроиконами, так-как образованы объединением икон в группы для решения одной конкретной задачи.



Рисунок 1.5 – Макроиконы «Вопрос» и «Цикл»

Икона «Ввод» отвечает за входные данные. Икона «Вывод», соответственно, за вывод информации. Обе иконы представлены на рисунке 1.6.



Рисунок 1.6 – Иконы «Вывод» и «Ввод»

Икона «Комментарий» не выполняет функциональной роли, только благоприятствует пониманию алгоритма (рис.1.7).



Рисунок 1.7 – Икона «Комментарий»

В уникальном представлении каждой иконы и заключается принцип наглядности: каждый тип иконы отличается от другой. 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 написания дракон-схем**

В языке ДРАКОН схему можно представить в виде «Шампура (рис 1.8).

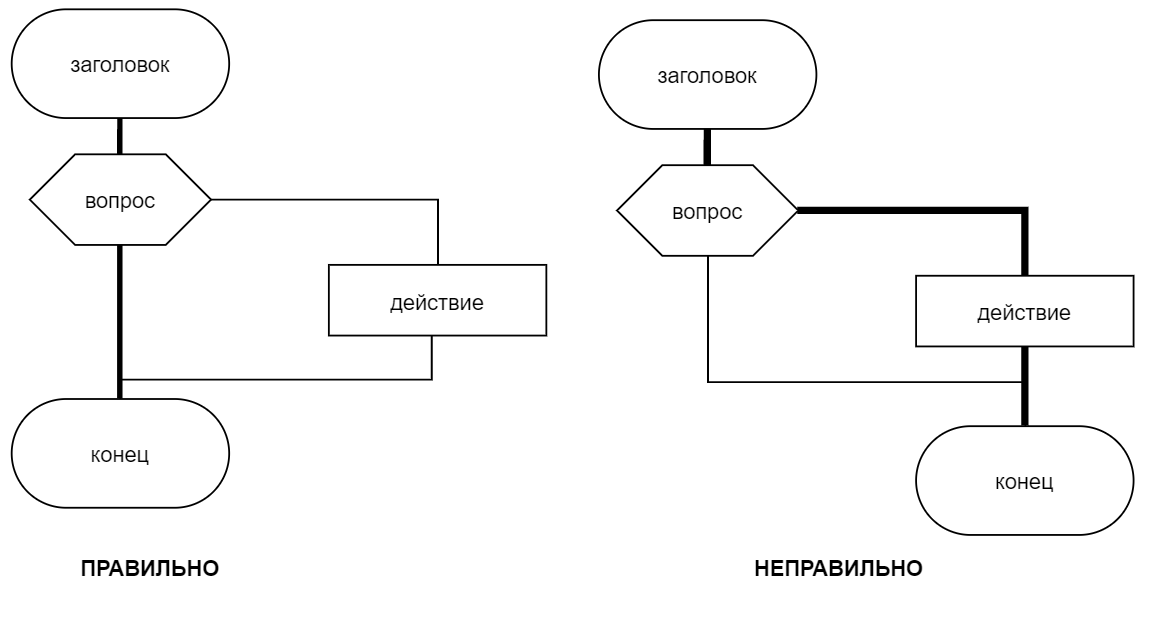


Рисунок 1.8 – Графическое представление правила «Шампура»

«Шампур» - вертикальная линия, которая соединяет иконы «Заголовок» и «Конец». Между этими двумя иконами может располагаться сколь угодно других икон, но все они должны лежать на одной вертикали. Желательно запретить нарушение этого правила с помощью программной реализации, а именно связать координаты x и у всех икон на главной вертикали.

Другое важное правило, которое вытекает из первого: главный маршрут алгоритма всегда должен идти по «Шампуру» (рис 1.9).

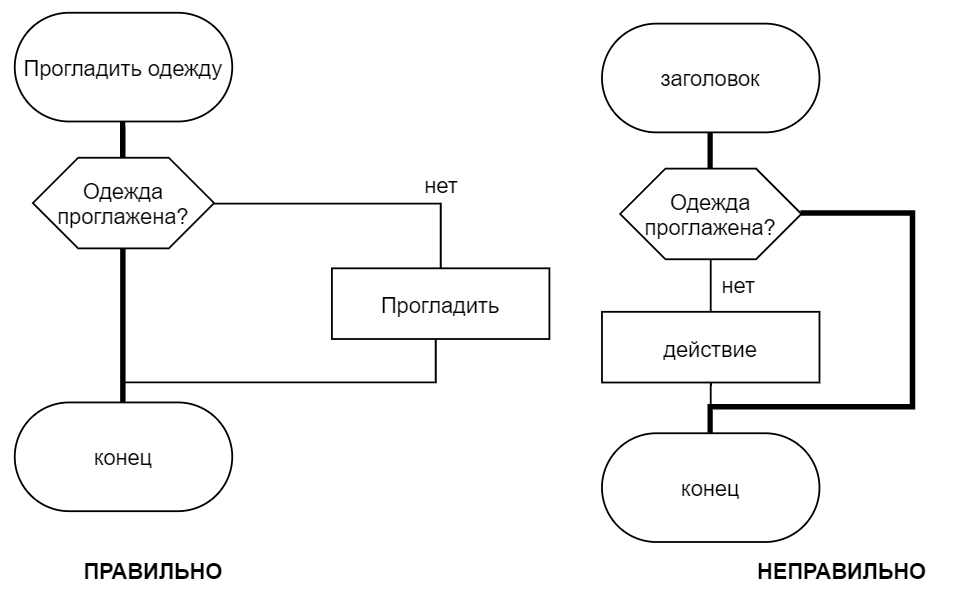


Рисунок 1.9 – Графическое представление правила: главный маршрут по вертикали

Под 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 неблагоприятно развитие событий, тем правее оно описано. Это необходимо для улучшения читаемости алгоритма. Также его можно описать другими словами: «Чем правее, тем хуже».

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 нереализуемо 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 иконки 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 пересечения линий всегда можно преобразовать к виду, в котором они пересекаться не будут.

## **1.2 Технология обработки информации**

Проектируемая системы должна осуществлять сбор, хранение и предоставление ДРАКОН-схем пользователю. Система должна предоставлять возможность идентификации пользователя. Для организации процесса обучения в системе должна присутствовать роль «Куратор», который может добавлять пользователей к себе в группу, что позволит осуществлять просмотр и редактирование ДРАКОН-схем других пользователей. «Администратор» в системе осуществляет контроль учетных записей в системе. Диаграмма вариантов использования представлена на рисунке 1.10.

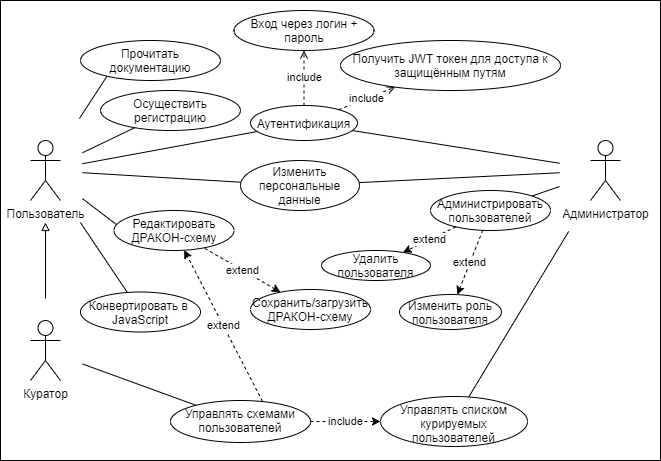


Рисунок 1.10 – Диаграмма вариантов использования

В системе присутствуют следующие роли:

* пользователь;
* куратор;
* администратор.

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 регистрации.

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 выполняет задачи контроля над работой системы: выдаёт роли зарегистрированным пользователем, удаляет аккаунты пользователей, которые не удовлетворяют пользовательским требованиям (некорректное имя пользователя).

### **1.2.1 Спецификация вариантов использования**

1. Осуществить регистрацию.
   1. Краткое описание: пользователь в интерфейсе аутентификации вводит уникальное имя пользователя в системе, придумывает пароль и нажимает кнопку «Регистрация».
   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. Аутентификация.
   1. Краткое описание: пользователь, ранее зарегистрированный в системе, вводит валидную комбинация имени пользователя и пароля и осуществляет вход через нажатие кнопки «Войти» интерфейса системы.
   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.
   1. Краткое описание: пользователь создаёт, удаляет или изменяет содержимое схемы или её название.
   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. Конвертировать в JavaScript.
   1. Краткое описание: пользователь через интерфейс системы выбирает пункт «Конвертация в JS».
   2. Ответ системы: система осуществляет конвертацию активной ДРАКОН-схемы в синтаксис языка JavaScript и размещает его в текущей папке проекта.
5. Прочитать документацию по работе с ИС.
   1. Краткое описание: оператор выбирает пункт в меню интерфейса «Help-Editor», в котором в виде списка описываются главы, содержание которых отражает всю функциональность проектируемой ИС, описанной в текстовой форме.
   2. Ответ системы: система отображает в левой части интерфейса список с главами, в правой содержание выбранной пользователем главы с возможностью навигации по документу.
6. Прочитать документацию по работе с языков ДРАКОН.
   1. Краткое описание: оператор выбирает пункт в меню интерфейса «Help-DRAGON», где в виде списка описываются главы, в которых содержится описание методов представления алгоритмов с помощью языка ДРАКОН.
   2. Ответ системы: система отображает в левой части интерфейса список с главами, в правой содержание выбранной пользователем главы с возможностью навигации по документу.

### **1.2.2 Шифрование паролей**

Для обеспечения безопасности хранимых в базе данных паролей пользователей необходимо использовать криптографическую функцию 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 отправляется на сервер, шифруется через функцию bcrypt и сохраняется в базе данных в связке с уникальным именем пользователя. В последующем обратно дешифровать ключ невозможно.

### **1.2.3 Описание способов взаимодействия пользователя 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 (рис 1.11).



Рисунок 1.11 – Дракон-схема «Примитив»

Все последующие блоки вставляются между ними. Количество их неограниченно. Местом для вставки является центр линии между двумя иконами. Область, в которую можно вставить икону подсвечивается. Удаление происходит выбором соответствующей иконы и нажатием клавиши Delete.

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. Компромиссным решением служит увеличение размеры иконы по вертикали.

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

### **1.2.4 Описание методов представления ДРАКОН икон**

Для проектируемой информационной системы необходимо 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 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.

Для проектируемой концепции представления ДРАКОН-схем были разработаны следующие иконы:

* заголовок;
* конец;
* действие;
* вопрос;
* ограничитель условной конструкции;
* пустая икона;
* комментарий;
* 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.

«Пустые» иконы являются служебными. Они не отображаются графически, необходимы для обеспечения механизма вставки, удаления и представления макроикон.

Иконы «Заголовок» и «Конец» содержат указатели друг на друга. Реализовано это с целью последующей вставки дракон-схемы в другую дракон схему в виде макроиконы, чтобы каждый из модулей имел информацию и своём начале и конце.

Икона «Вопрос» содержит в себе указатель на икону альтернативной ветки. В базовом случае – это простая икона, которая формирует последовательность действий в случае отрицательного результата проверки в иконе «Вопрос». Для установки границ макроиконы «Вопрос» в ней также содержится указатель на служебную икону «Ограничитель условной конструкции», которая завершает ветку «Если-то». Сам же ограничитель содержит указатель на родительскую икону «Вопрос», это необходимо для корректного синтаксического анализа дракон-схемы. Демонстрация этого механизма представлена на рисунке 1.12.

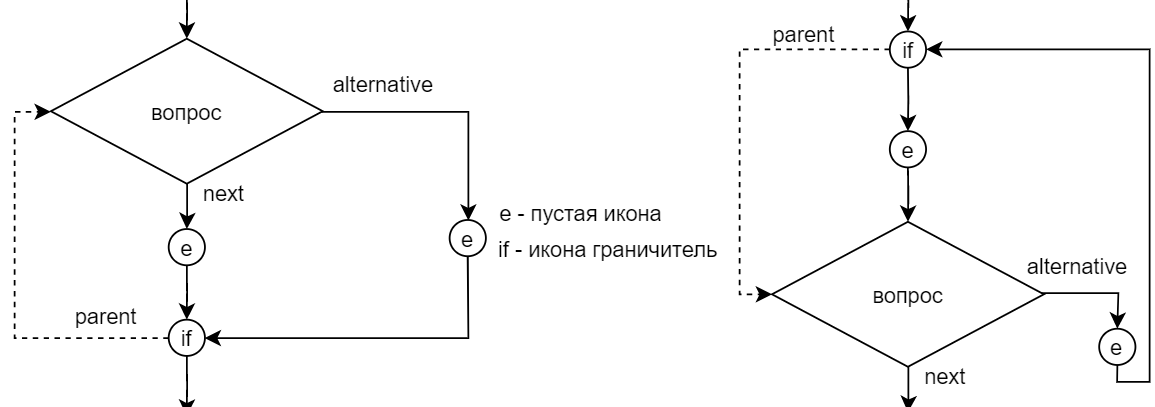


Рисунок 1.12 – Демонстрация реализации макроиконы вопрос

### **1.2.5 Описание методов представления ДРАКОН-схем**

Для однозначной идентификации каждой иконе присваивается уникальный ID, по которому можно обратиться как к самой иконе, так и ко всем её соседям. Благодаря уникальным номерам, упрощается процесс вставки новых икон и их удаление. Вторым преимуществом является упрощение сохранения дракон-схем в памяти компьютера. Каждая икона содержит в себе указатель на соседние иконы (выше и ниже), если одна из них отсутствует, как например указатель previous для иконы «Заголовок», то значение устанавливается в null (рис 1.13).

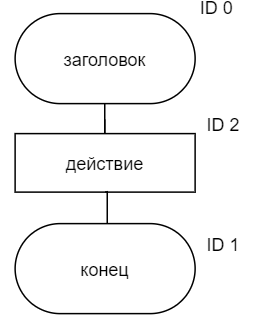


Рисунок 1.13 – Уникальные идентификаторы икон

Для представления ДРАКОН-схем 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 просматривать дракон схему целиком.

При инициализации контейнера для ДРАКОН схемы генерируется базовая конструкция, которая состоит из иконы «Начало», «Пустая икона» и «Конец». Что освобождает пользователя от ненужной работы по созданию обязательного базового каркаса.

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 соседних, где производится сама операция.

Схема может содержать указатели на другие схемы, тем самым позволяя осуществлять декомпозицию проекта на функции, для этого предусмотрена специальная икона.

Механизм вставки указателей на модуль позволяет в рамках одного проекта реализовать несколько ДРАКОН-схем с их последующим объединением в цепочку исполнения (рис 1.14).

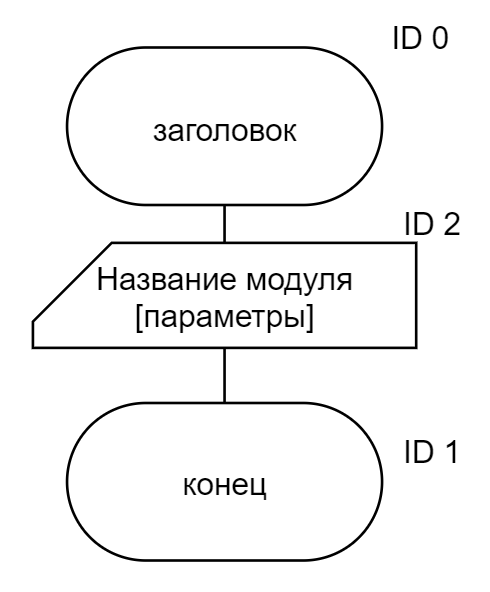


Рисунок 1.14 – Графическое представление вложенной схемы

### **1.2.6 Алгоритм вставки простой иконы**

В параграфе 1.2.2 было представлено описание «пустой» иконы. Добавление новых икон/макроикон осуществляется только в «пустые».

На рисунке 1.15 демонстрируется метод вставки простой иконы.

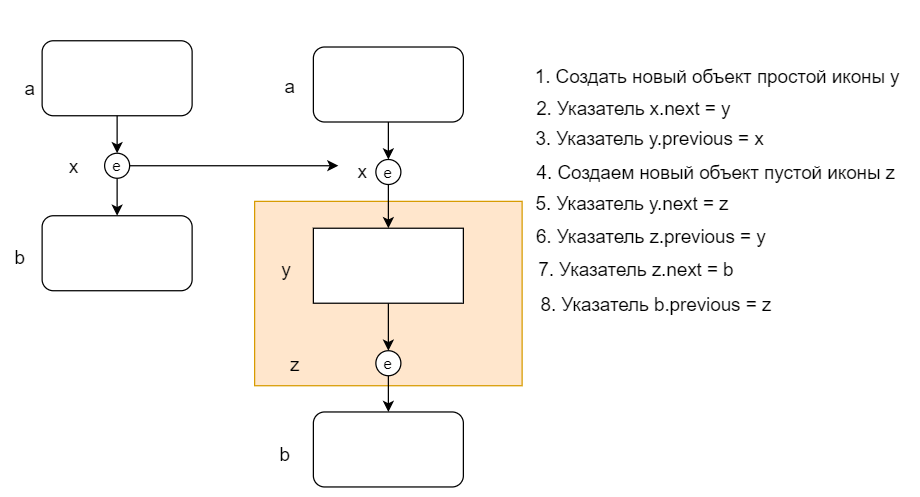


Рисунок 1.15 – Алгоритм вставки простой иконы

Присутствие «Пустой иконы» обуславливается механизмом изменения ДРАКОН схемы. Новые иконы могут быть вставлены только в строго определённые места, для обозначения таких мест используется неявная икона-пустышка. Поэтому любая операция вставки реализует механизм вставки именно макроиконы, пусть и неявно.

### **1.2.7 Алгоритм вставки комплексных икон**

Каждая макроикона реализует собственный метод вставки, так-как их поведение отличается от стандартного. На текущем этапе разработки такой иконой является «Вопрос». Реализация алгоритма добавления данной макроиконы представлено на рисунке 1.16.

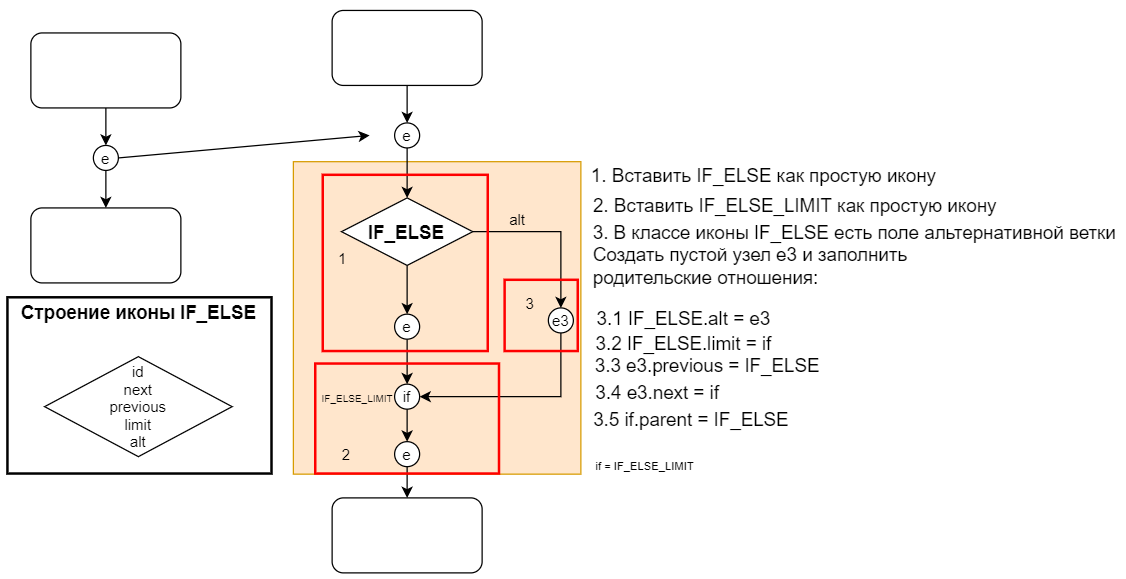


Рисунок 1.16– Алгоритм вставки макроиконы «Вопрос»

### **1.2.8 Алгоритм удаления икон**

В противоположность от правила вставки, удаление возможно всех икон, кроме «Заголовок», «Конец», и служебных («Пустая» и «Ограничитель» икон). Правила удаления икон представлены на рисунке 1.17.

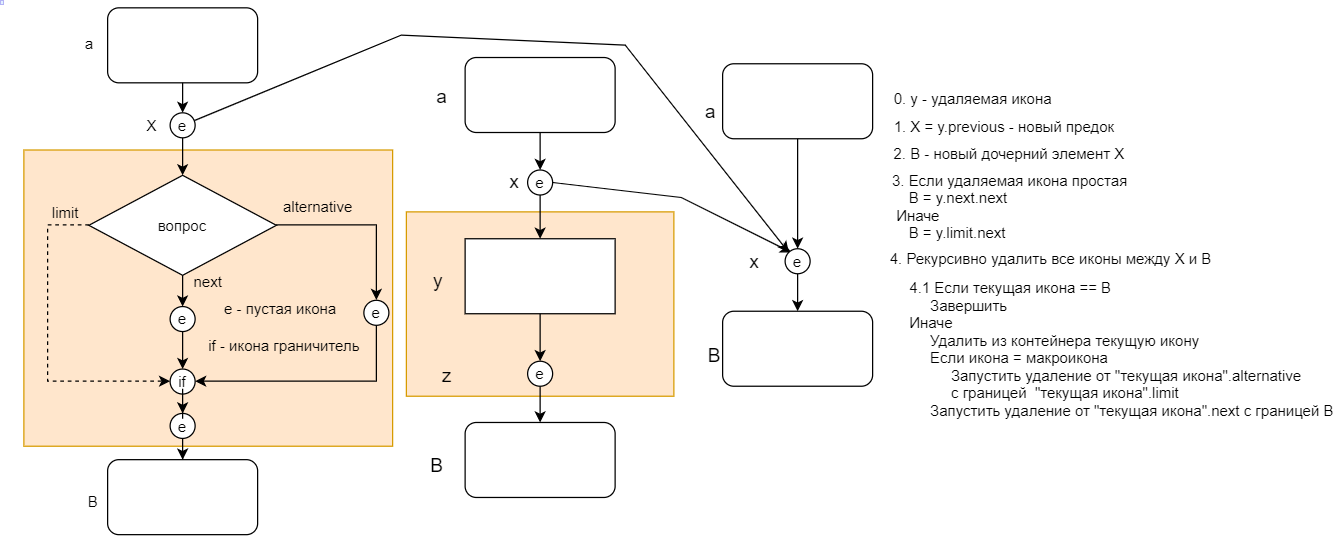


Рисунок 1.17– Алгоритм удаления макро и простых икон

Иконы удаляются рекурсивно, каждая макроикона содержит указатель на свой ограничитель: когда итерация удаления доходит до такой иконы, то инициализируется удаление альтернативных веток до границы конструкции, после чего управление передаётся основному потоку рекурсии и происходит удаление основной ветки.

Такой подход позволяет добиться безопасности проведения операции удаления, «ограничитель» не будет удалён, пока все альтернативные ветки не будут исключены из контейнера.

## **1.3 Инфологическая модель данных**

### **1.3.1 Сущность базы данных**

База данных должна содержать следующие данные: информация о всех пользователях системы, их роли в ИС, списки курируемых пользователей для кураторов и 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, пароль, роль |
| Куратор | Идентификатор отношения, идентификатор куратора, идентификатор пользователя |

Продолжение таблицы 1.1

|  |  |
| --- | --- |
| **Сущность** | **Атрибуты** |
| 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.1.18-1.20). На рисунке 1.18 изображен фрагмент ER-модели для сущности «Пользователь».

|  |
| --- |
|  |
| Рисунок 1.18 – Фрагмент ER-модели для типа сущности «Пользователь» |

Фрагмент ER-модели сущности «Куратор» показан на рисунке 1.19.

|  |
| --- |
|  |
| Рисунок 1.19 – Фрагмент ER-модели для типа сущности «Куратор» |

Фрагмент ER-модели сущности «ДРАКОН-схема» показан на рисунке 1.20.

|  |
| --- |
|  |

Рисунок 1.20 – Фрагмент ER-модели для типа сущности «ДРАКОН-схема»

### **1.3.2 Диаграмма сущность-связь**

Из сущностей, представленных на рисунках 1.18-1.20, сформирована диаграмма «Сущность-связь», представленная в приложении 2.

## **1.4 Входная и выходная информация**

### **1.4.1 Входная информация**

Справочник «Пользователь» предназначен для хранения информации о пользователях системы. Атрибуты справочника «Пользователь» представлены в таблице 1.2.

Таблица 1.2 – Справочник «Пользователь»

| **Параметр** | **Тип** | **Размер** | **Примечание** |
| --- | --- | --- | --- |
| ID\_пользователя | uuid | длинное строковое | Первичный ключ таблицы |
| Имя\_пользователя | текстовый | длинное строковое | Название роли |
| Пароль | текстовый | длинное строковое | Пароль учётной записи пользователя |
| Роль | текстовый | 20 | Название роли пользователя в системе |

Справочник «Куратор» предназначен для хранения информации о связи куратор - пользователь. Куратор может просматривать и изменять схемы только тех пользователей, которые привязаны к куратору. Атрибуты справочника «Куратор» представлены в таблице 1.3.

Таблица 1.3 – Справочник «Куратор»

| **Параметр** | **Тип** | **Размер** | **Примечание** |
| --- | --- | --- | --- |
| ID\_отношения | целый, серийный | длинное целое | Первичный ключ таблицы |
| ID\_куратора | uuid | длинное строковое | Идентификатор куратора. Внешний ключ. Связан с ключом справочника «Пользователь» |
| ID\_пользователя | uuid | длинное строковое | Идентификатор пользователя. Внешний ключ. Связан с ключом справочника «Пользователь» |
| Имя\_отношения | текстовый | длинное строковое | Имя отношения между куратором и пользователем |

Справочник «ДРАКОН-схема» предназначен для хранения информации о схемах пользователей. Атрибуты справочника «ДРАКОН-схема» представлены в таблице 1.4.

Таблица 1.4 – Справочник «Дракон-схема»

| **Параметр** | **Тип** | **Размер** | **Примечание** |
| --- | --- | --- | --- |
| ID\_схемы | uuid | длинное строковое | Первичный ключ таблицы |
| Имя\_схемы | текстовый | длинное строковое | Название схемы |
| ID\_пользователя | uuid | длинное строковое | Идентификатор пользователя. Внешний ключ. Связан с ключом справочника «Пользователь» |
| Содержимое\_схемы | jsonb | длинное строковое | Содержимое ДРАКОН-схемы |
| Время \_последнего \_изменения | дата/время | Длинное строковое | Дата и время последнего изменения |
| ID\_пользователя для\_последнего  изменения | uuid | длинное строковое | Идентификатор пользователя, который производил последнее изменение. Внешний ключ |

К входным данным также относятся:

* имя пользователя и пароль при процессе регистрации;
* ввод содержимого в иконы ДРАКОН-схем;
* изменение размера и положения икон ДРАКОН-схем;

### **1.4.2 Выходная информация**

Представление выходных данных при процессе авторизации представлено в таблице 1.5.

Таблица 1.5 - Представление JWT токена доступа к защищённым API приложения:

|  |  |
| --- | --- |
| **Описание** | **Тип** |
| JWT токен доступа | string |
| uuid пользователя | string |
| Роль пользователя | string |

Представление выходных данных о пользователе системы представлено в таблице 1.6.

Таблица 1.6 - Представление информации о пользователе:

|  |  |
| --- | --- |
| **Описание** | **Тип** |
| uuid | string |
| Роль | string |
| Имя пользователя | string |

Представление выходных данных списка курируемых пользователей представлено в таблице 1.7.

Таблица 1.7 - Представление курируемых пользователей:

|  |  |
| --- | --- |
| **Описание** | **Тип** |
| uuid куратора | string |
| uuid пользователей | string |

Представление выходных данных при получении ДРАКОН-схем с базы данных представлено в таблице 1.8.

Таблица 1.8 - Представление ДРАКОН-схем:

|  |  |
| --- | --- |
| **Описание** | **Тип** |
| uuid схемы | string |
| Содержимое схемы | string |
| Дата последнего изменения | Date |
| Имя пользователя, сделавшего последнее изменение | string |

**1.4 Формат содержимого ДРАКОН-схемы**

Для хранения дракон-схем был разработан xml-формат для хранения ДРАКОН-схем. Элемент «icons» хранит в себе все существующие иконы и связи между ними. Информацию о расположении хранить избыточно, так как представление осуществляется на стороне клиента.

При загрузке дракон-схем производится перерасчёт расположения всех икон. Каждый элемент записи представляет из себя икону, которая содержит следующую информацию:

* «ID» - атрибут, содержит уникальный id;
* «type» - обозначает тип иконы;
* «previous» - содержит id предыдущей иконы;
* «next» - содержит id следующей иконы;
* «alternative» - содержит id альтернативной ветки для икон выбора;
* «text» - содержит основную операцию, проводимую данной иконой;
* «value» - дополнительный атрибут с внутренним содержимым значения иконы.

Пример хранения дракон-схемы представлен на рисунке 1.21. Содержимое схемы помещается в формат JSON в поле «данные».

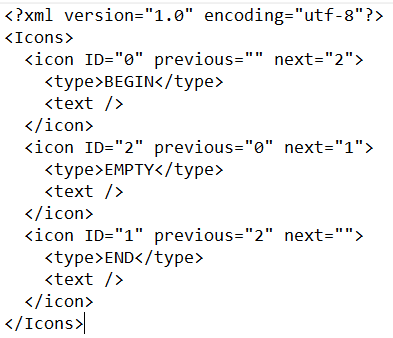


Рисунок 1.21 – пример ДРАКОН-схемы в xml формате

**1.5 Требования к техническому и программному обеспечению**

Рабочая станция:

* Intel-совместимый процессор с частотой не менее 1,6 ГГц;
* не менее 512 МБ ОЗУ;
* не менее 50 MБ 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 1,6 ГГц;
* не менее 8192 МБ ОЗУ;
* не менее 512 MБ свободного места на диске;
* дисковод CD-ROM/DVD-ROM;
* операционная система: ОС с поддержкой среды выполнения NodeJS 14.16+.

**2 РАБОЧИЙ ПРОЕКТ**

**2.1 Общие сведения о работе системы**

Программный продукт разработан в среде разработки Visual Studio Code на языке TypeScript, JavaScript с использованием библиотеки NestJs v.7.5.0 + NodeJs как платформы для сервера, СУБД PostgreSQL 12.3 для организации хранилища данных. Библиотеки TypeORM v.0.2.3 для связи базы данных с сервером. React для написания интерфейсов системы (рис 2.1).

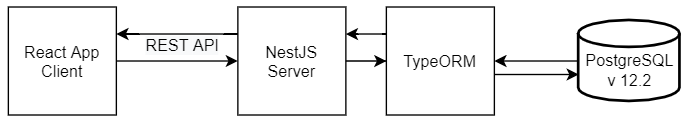


Рисунок 2.1 – Демонстрация архитектуры системы.

**2.2 Функциональное назначение программного продукта**

Разработанный программный продукт предназначен для загрузки и изменений дракон-схем, а также получения данных из базы данных. Программа имеет следующие функциональные возможности:

* добавление нового пользователя;
* вход в систему уже зарегистрированных пользователей;
* удаление учётной записи пользователя;
* обновление пароля пользователя;
* добавление/удаление пользователя в список курируемых;
* загрузка схемы из базы данных;
* просмотр и изменение куратором схем других пользователей;
* создание схемы;
* редактирование схемы и её содержимого;
* удаление схемы;
* редактирование положения и содержимого икон.

Программа имеет следующие функциональные ограничения:

* отсутствуют компоненты интерфейсов для вызовов API к базе данных;
* ограничения уровня языка ДРАКОН не реализованы;
* конвертер в ЯП не реализован.

**2.3 Выполнение программного продукта**

Для выполнения программы необходимо открыть папку с проектом. Для запуска сервера необходимо установить на серверную машину NodeJS версии 12.3 или выше и PostgreSQL версии 12.3 или выше. Запустить NodeJS в папке проекта, после чего ввести npm run start:dev:db. Данный скрипт создаст и инициализирует базу данных. После чего ввести npm run:start:dev. Команда запустит дев сервер на порту localhost:5000. Сервер готов к прослушиванию. При переходе на домашний URL будет открыта страницу входа в приложение. Для регистрации необходимо создать нового пользователя, введя имя пользователя и пароль. В случае успешной регистрации будет получен JWT токен для доступа к защищённым URL приложения. Данный токен позволит открыть страницу редактора (localhost:5000/sample) и осуществлять запросы к api (localhost:5000/api/\*).

**2.4 Описание физической архитектуры системы**

Проектируемая система состоит из нескольких компонент. В качестве СУБД используется PostgreSQL версии 12.3. Сервер представляет из себя NodeJS приложение, развернутое на порту для прослушивания запросов от клиентов. Клиентами представляется одна или нескольких машин с доступом к сети, в которой работает сервер. Диаграмма развертывания представлена в приложении 3 (рис. П.3).

**2.5 Описание программы**

Программный продукт состоит из клиентской и серверной части. Клиентская часть проектируемой системы на текущий момент состоит только из страницы регистрации.

Аутентификация пользователей осуществляется через специальную форму. Необходимо ввести уникальное имя пользователя и пароль, после чего нажать одну из клавиш: «Регистрация» или «Вход» в зависимости от наличия или отсутствия учетной записи пользователя. Модуль аутентификации содержит в себе два метода для обработки пользовательских запросов (табл.2.1).

Таблица 2.1 – Описание модуля регистрации

|  |  |
| --- | --- |
| **Метод** | **Назначение** |
| handleLogInButtonClick | Вызов к API для входа в систему через связку логин+пароль |
| handleSignUpButtonClick | Вызов к API для регистрации в системе через связку логин+пароль |

Шифрование пароля осуществляется без возможности дешифровки. Для валидации пользователя необходимо отправить на сервер форму с валидным именем пользователя и паролем. Сервер осуществит запрос 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 60 минут после успешного сеанса регистрации. После истечения срока валидности токена пользователю необходимо повторно ввести логин и пароль.

Обслуживание клиента осуществляется с помощью сервера. URL вида «домен/api/\*» используются для API вызовов к базе данных и получения информации из неё. Для обеспечения RESTful API для NestJS было реализовано четыре модуля (табл. 2.2-2.5).

В таблице 2.2 представлено описание модуля UserModule.

Таблица 2.2 – Описание модуля UserModule

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

В таблице 2.3 представлено описание модуля AuthModule.

Таблица 2.3 – 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 |
| JwtStrategy | Сервис стратегии по управлению защищёнными путями URL |
| LocalStrategy | Сервис стратегии по валидации учетных данных пользователя |

В таблице 2.4 представлено описание модуля CuratorModule.

Таблица 2.4 – Описание модуля CuratorModule

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

В таблице 2.5 представлено описание модуля SchemeModule.

Таблица 2.5 – Описание модуля SchemeModule

|  |  |
| --- | --- |
| **Компонент** | **Назначение** |
| SchemeService | Обработчик запросов к базе данных к сущности «ДРАКОН-схема» |
| SchemeController | Обработчик вызовов к api/schema |

**2.6 Даталогическая модель данных**

Справочник «Пользователи» предназначен для хранения информации о пользователях системы. Справочник содержит уникальный идентификатор имя пользователя. Атрибуты справочника «Пользователи» представлены в таблице 2.6.

Таблица 2.6 – 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 | |
| password | text | переменный | Хешированный пароль пользователя |
| role | varchar | 50 | Пароль учетной записи пользователя |

Атрибуты справочника «Куратор» представлены в таблице 2.7.

Таблица 2.7 – Справочник «Куратор»

| **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. |

Продолжение таблицы 2.7

|  |  |  |  |
| --- | --- | --- | --- |
| **Параметр** | **Тип** | **Размер** | **Примечание** |
| 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 |

Атрибуты справочника «ДРАКОН-схема» представлены в таблице 2.8.

Таблица 2.8 – 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 |
| last\_changed | datetime | 8 | Дата и время последнего изменения схемы |
| last\_changed\_by\_id | uuid | 16 | id пользователя, изменения которого были последними. |

**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 выборки 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**

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

Для таблицы «Пользователь» поля идентификатора, имени пользователя 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 с схемой позволяют однозначно идентифицировать куратора или пользователя, который произвёл изменения и время совершения операции.

**2.9 Определение ссылочной целостности**

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**

Основные триггерные функции и триггеры представлены в таблице 2.9.

Таблица 2.9 – 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 |
| 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, обрабатываемой и хранимой ей. В текущей итерации программный данные в программном продукте защищены следующими факторами:

* каждая триггерная функция, выполняемая до операции, обрабатывает исключения, что позволяет избежать занесения некорректных данных в базу;
* 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**

Класс User представляет описание сущности «Пользователь» для связи с базой данных (табл. 2.10).

Таблица 2.10 – Описание класса User

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Назначение** |
| uuid | string | Id пользователя |
| username | string | Имя пользователя |
| password | string | Пароль |
| role | string | Роль |

Класс Curator представляет описание сущности «Куратор» для связи с базой данных (табл 2.11).

Таблица 2.11 – Описание класса Curator

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Назначение** |
| id | number | Id отношения |
| uuid\_curator | string | Id куратора |
| uuid\_user | string | Id пользователя |
| relation\_name | string | Название отношения |

Класс Schema представляет описание сущности «ДРАКОН-схема» для связи с базой данных (табл 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 |

При авторизации используется связка логин + пароль. Данные от клиента на сервер доставляются с помощью специальных DTO объектов, которые проходят проверку на корректность данных (табл 2.13).

На сервер пароль доставляется в виде простого текста. Предполагается, что протокол передачи данных от клиента к серверу будет HTTPS.

Таблица 2.13 – Описание LoginUserDTO

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Назначение** |
| username | string | Имя пользователя |
| password | string | Пароль |

Роль каждому новому пользователю выдаётся по умолчанию «USER». Другие роли может выдавать только администратор (табл 2.14).

Таблица 2.14 – Описание ролей пользователя

|  |  |
| --- | --- |
| **Роль** | **Описание** |
| USER | Простой 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 чужие схемы |

Сервисы обслуживают запросы пользователя, на текущем этапе проектирования реализовано 5 сервисов (табл. 2.15-2.19). В таблице 2.15 представлено описание класса UserService.

Таблица 2.15 – Описание класса UserService

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Назначение** |
| usersRepository | Repository<User> | API к таблице «Пользователь» |
| **Метод** | | **Назначение** |
| private async findAll() | | Получить все записи |
| async findOneById(id: string) | | Поиск пользователя по id |
| async findOneByName(username : string ) | | Поиск пользователя по имени |
| async create(userData : CreateUserDto) | | Создание нового пользователя |

Продолжение таблицы 2.15

|  |  |
| --- | --- |
| **Метод** | **Назначение** |
| async update(id: string, payload: CreateUserDto) | Обновление конфиденциальных данных пользователя |
| async delete(uuid: string) | Удаление пользователя по id |

В таблице 2.16 представлено описание класса CryptoService.

Таблица 2.16 – Описание класса CryptoService

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Назначение** |
| saltRounds | number | Соль для уровни сложности функции bcrypt |
| **Метод** | | **Назначение** |
| public async hashPassword(password: string) | | Хеширование пароля с помощью bcrypt |
| public async checkPassword(candidate: string, saltedPassword: string) | | Проверка пароля на соответствие хешированному |

В таблице 2.17 представлено описание класса AuthService.

Таблица 2.17 – 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 |
| **Метод** | | **Назначение** |
| async signUp(user : CreateUserDto) | | Регистрация нового пользователя |
| async logIn(username: string, password: string) | | Вход зарегистрированного пользователя и выдача токена |

Продолжение таблицы 2.17

|  |  |
| --- | --- |
| **Метод** | **Назначение** |
| async createToken(user) | Создание и подпись токена |
| async verify(payload) | Проверка, что пользователь зарегистрирован в системе |

В таблице 2.18 представлено описание класса CuratorService.

Таблица 2.18 – Описание класса CuratorService

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

В таблице 2.19 представлено описание класса SchemaService.

Таблица 2.19 – Описание класса SchemaService

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Назначение** |
| schemaRepository | Repository<Schema> | API к таблице «ДРАКОН-схема» |
| **Метод** | | **Назначение** |
| private async findAll() | | Получить все записи |

Продолжение таблицы 2.19

|  |  |
| --- | --- |
| **Метод** | **Назначение** |
| async findOneById(id: string) | Найти ДРАКОН-схему по id |
| create(schema: CreateSchemaDto) | Создать ДРАКОН-схему |
| async findAllByUserId(user\_id: string) | Получить все схемы пользователя |
| async update(id: number, payload: CuratorDto) | Обновление данных ДРАКОН-схемы |
| async delete(id: number) | Удаление ДРАКОН-схемы |

Каждый путь в приложении обслуживает определенный контроллер. Сопоставление путей URL и ответственного роутера представлено в таблице 2.20.

Таблица 2.20 – Описание класса SchemaService

|  |  |
| --- | --- |
| **Контроллер** | **Путь** |
| UsersController | [порт]:/[домен]/api/users\* |
| AuthController | [порт]:/[домен]/api/auth\* |
| CuratorController | [порт]:/[домен]/api/curator\* |
| SchemaController | [порт]:/[домен]/api/schema\* |
| PageController | [порт]:/[домен]/[страница] |

**2.15 Реализованные меню и интерфейсы**

На рисунке 2.2 изображен прототип интерфейса редактора.

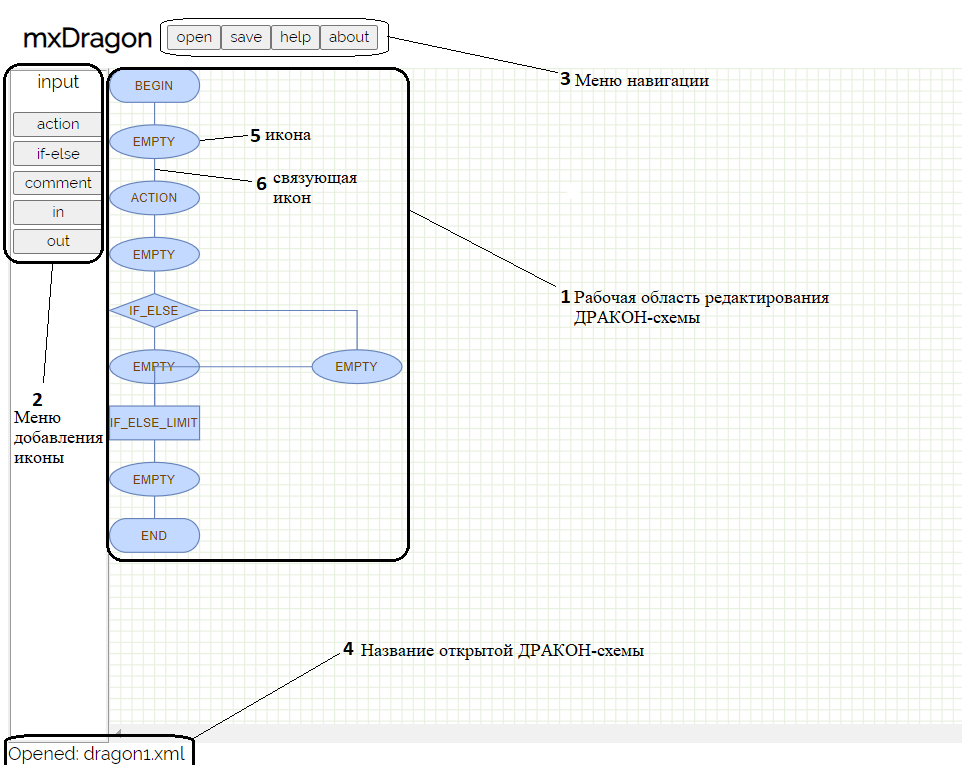


Рисунок 2.2 – Окно редактора

На данный момент текущий прототип служит для демонстрации интерфейсов системы редактирования схем и не является функциональным.

Форма аутентификации представлена на рисунке 2.3. Необходимо ввести имя пользователя и пароль для входа или регистрации, если учетной записи не существует.  
В случае успешной идентификации пользователя будет осуществлено перенаправление на главную страницу редактора.

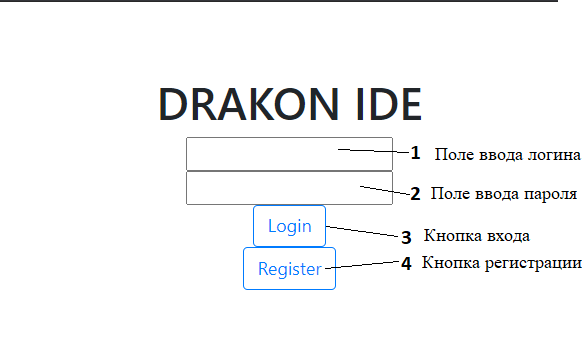


Рисунок 2.3 – Окно входа в систему

**2.16 Сообщения системы**

В таблице 2.21 указаны сообщения системы.

Таблица 2.21 – сообщения системы

|  |  |  |
| --- | --- | --- |
| **№** | **Сообщение системы** | **Описание** |
| 1 | «User not found» | Пользователь не найден |
| 2 | «Invalid password» | Некорректный пароль |
| 3 | «User with that name already  registered» | Пользователь с таким именем уже зарегистрирован |
| 4 | «Relation not found» | Связь Куратор-пользователь не найдена |
| 5 | «Scheme not found» | Схема не найдена в БД |
| 6 | «username has forbidden symbols» | Введено некорректное имя пользователя |
| 7 | «curators can add only USER roles» | Куратор не может добавлять администраторов и других кураторов к себе в группу |
| 8 | «insufficient permission» | Роль пользователя не позволяет выполнить запрос |
| 9 | «JWT Token is incorrect or expired» | JWT токен некорректен или неактуален |

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

**3 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ**

1. Для развертывания разрабатываемого проекта пользователь запускает скрипт развёртывания базы данных из корневой папки с проектом через yarn start:dev:db.
2. Система создаст базу данных «drakon\_db» в СУБД PostgreSQL.
3. Для запуска дев-сервера пользователь запускает из корневой папки проекта команду yarn start:dev.
4. Платформа NodeJS запускает дев-сервер на порту 5000.
5. Пользователь открывает браузер и переходит по URL: localhost:5000.
6. Браузер открывает окно регистрации
7. В верхнем поле для ввода данных пользователь вводит логин «Джим», в нижнем поле «helloworld». Нижнее текстовое поле является паролем, содержимое поля скрыто.
8. Пользователь нажимает кнопку «Register».
9. Система выдаёт пользователю JWT токен для доступа к защищённым путям и сообщение об успешной регистрации. Если такой пользователь уже существует, система выдаст сообщение №3.
10. Пользователь открывает страницу localhost:5000/sample.
11. Браузер отображает главное окно редактора программы (рис 2.2).
12. Пользователь выделяет схемы долгим зажатием ЛКМ, перемещает схему и отдельные компоненты.
13. Система ведёт икону вслед за курсором, отображая изменения.
14. С помощью пользователь Postman осуществляет запрос к localhost:5000/api/users, используя JWT токен для доступа как параметр Authenticate в заголовке запроса.
15. Система выдаёт список пользователей.
16. Пользователь выбирает случайный uuid и обращается к серверу с запросом к localhost:5000/api/users/«uuid».
17. Система в ответ выдаёт информацию о нём в формате, описанном в таблице 1.6.
18. С помощью Postman или аналогичного ПО пользователь отправляет метод delete по тому же url.
19. Система выводит сообщение 2 об успешном удалении пользователя.
20. Пользователь повторяет пункт 9.
21. Система выдаёт сообщение 1, сообщая, что такого пользователя не существует.
22. Завершить работу.

**ЗАКЛЮЧЕНИЕ**

В результате прохождения производственной практики на кафедре «АСОИУ» АГТУ была спроектирована и разработана база данных с использованием PostgreSQL версии 12.3 для проектируемой системы. Реализован сервер для обслуживания базы данных и распределения доступа пользователей. Спроектированы прототипы интерфейсов системы.

Программа отвечает поставленным требованиям и может использоваться как сервер для клиентской части интегрированной среды разработки DRAKON IDE.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Белов С.В., Лаптев В.В., Морозов А.В., Толасова В.В., Мамлеева А.Р. Требования к оформлению студенческих работ. – Астрахань, АГТУ, 2017. 104 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.

**ПРИЛОЖЕНИЕ 1 Диаграмма классов ДРАКОН-схемы**

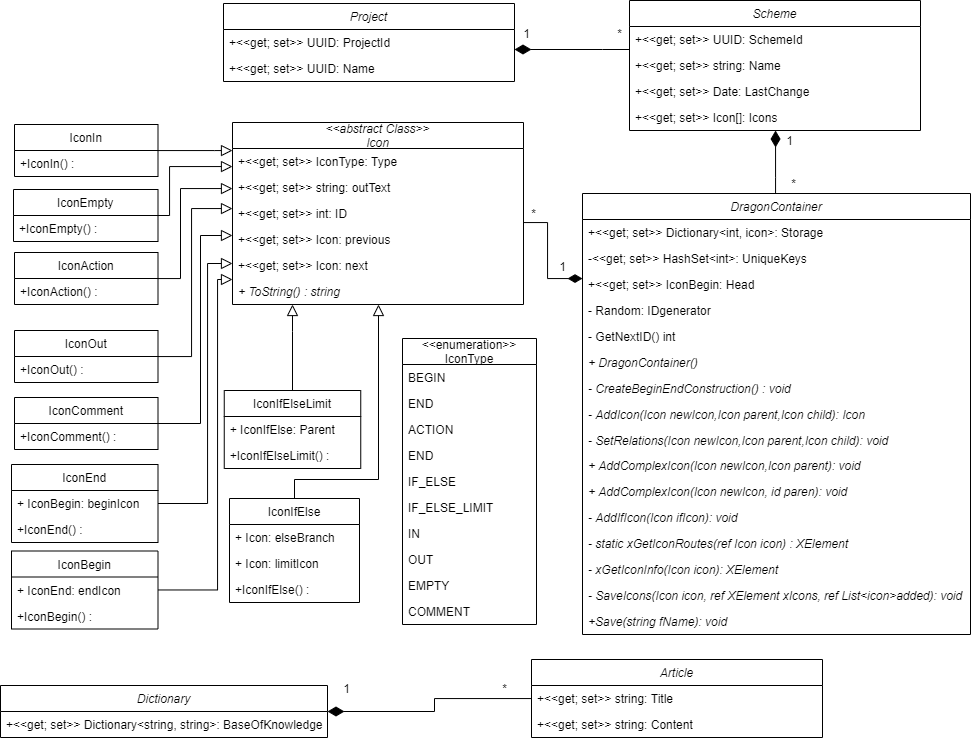


Рисунок П.1 – Диаграмма классов для представления ДРАКОН-схем

**ПРИЛОЖЕНИЕ 2 Диаграмма сущность-связь для базы данных**

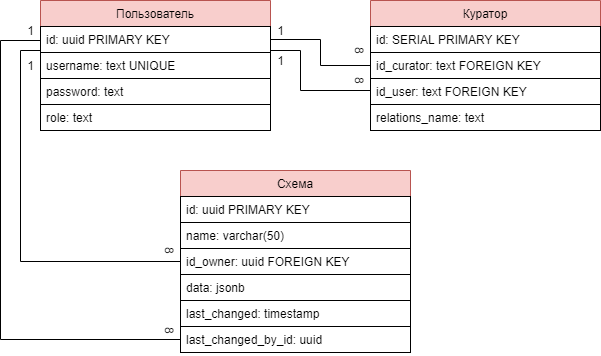


Рисунок П.2 – Диаграмма сущность-связь базы данных «БД\_Дракон»

**ПРИЛОЖЕНИЕ 3 Диаграмма развертывания**

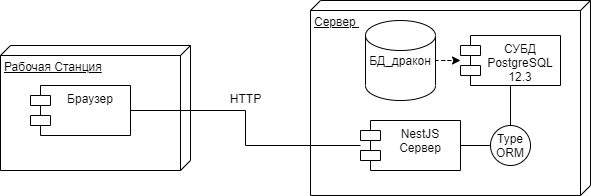


Рисунок П.3 – Диаграмма развертывания среды по обучению алгоритмизации DRAKON IDE

1. [↑](#footnote-ref-1)