Санкт-Петербургский государственный университет

Ким Юния Александровна

Выпускная квалификационная работа

Визуальный редактор планов исполнения запросов PosDB на базе REAL.NET

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5006.2018 «Математическое обеспечение и администрирование информационных систем»*

Профиль *Информационные системы и базы данных*

Научный руководитель:
доцент кафедры системного программирования, к.т.н.
Ю. В. Литвинов

Консультант:
ассистент кафедры информационно-аналитических систем
Г. А. Чернышев

Рецензент:
аналитик данных ООО «Нордиджи» Н. В. Бобров

Санкт-Петербург
2022

Saint Petersburg State University

Yuniya Kim

Bachelor's Thesis

Visual editor of PosDB query execution plans based on REAL.NET

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5006.2018 "Software and Administration of Information Systems"*

Profile: *Information Systems and Databases*

Scientific supervisor:
C.Sc., System Programming chair docent
Y. V. Litvinov

Consultant:
Information and Analytical Systems chair assistant
G. A. Chernishev

Reviewer:
data engineer at Nordigy Ltd.
N. V. Bobrov

Saint Petersburg
2022

Оглавление

Введение

В современном мире существует ряд различных задач, для решения которых эффективнее всего использовать визуальные предметно-ориентированные языки программирования. В качестве примера подобной задачи и соответствующих языков можно привести среды для обучения программированию TRIK Studio [26], LEGO RoboLab [3] и Scratch [19] и их языки. Визуальные предметно-ориентированные языки хороши тем, что они направлены на решение задач в конкретной предметной области и, будучи визуальными языками, выступают в качестве аналогов текстовым языкам и позволяют пользователям писать программы с помощью манипуляций графическими моделями и объектами.

В настоящее время на кафедре системного программирования СПбГУ ведётся разработка платформы для предметно-ориентированного моделирования REAL.NET [27], которая призвана облегчить реализацию и работу с различными визуальными языками. Изначально платформа разрабатывалась как десктопная система, однако в условиях современных реалий среда постепенно переносится в веб. Сейчас у REAL.NET имеется веб-редактор, который позволяет работать с визуальными языками.

К команде REAL.NET обратился один из разработчиков движка СУБД PosDB [13] и предложил задачу по реализации визуального редактора для создания планов выполнения запросов в PosDB. План выполнения запросов – это последовательность операций, которые необходимо выполнить для исполнения SQL-запроса в СУБД. Проанализировав такой план, можно понять, как уменьшить время выполнения запроса и увеличить производительность, чем обычно занимаются администраторы баз данных. Для удобства восприятия информации планы выполнения запросов можно визуализировать с помощью специальных программ. Разработчики движков, в свою очередь, сначала выписывают несколько типовых запросов, а затем строят и визуализируют их возможные планы и анализируют, какое преобразование в план лучше

для оптимизации работы создаваемого движка СУБД. Реализованный визуальный редактор планируется использовать для построения планов выполнения запросов, а также в образовательных целях для обучения студентов созданию планов, так как их построение имеет набор правил и ограничений, которые нельзя нарушать. В отличие от обычных редакторов для построения диаграмм, реализованный редактор будет иметь систему проверки упомянутых ограничений.

1. Постановка задачи

Целью данной работы является разработка визуального редактора для создания планов выполнения запросов движка PosDB на базе платформы REAL.NET. Для достижения данной цели были поставлены следующие задачи.

1. Провести обзор движка и существующих визуализаторов планов.
2. Разработать визуальный предметно-ориентированный язык для создания планов запросов в PosDB.
3. Реализовать новую необходимую функциональность в пользовательском интерфейсе.
4. Добавить систему проверки ограничений при построении планов.
5. Провести апробацию визуального редактора.

2. Обзор

2.1. Обзор движка

2.1.1. PosDB

PosDB – движок распределённой дисковой колоночной СУБД. Дисковые системы хранят большинство данных на диске и используют оперативную память для кэширования или временного хранения информации. Данная система была создана для изучения распределённого выполнения запросов в колоночных системах. Колоночные СУБД хранят каждый атрибут в отдельном файле в отличие от классических, хранящих данные построчно. Несмотря на большое количество исследований в области колоночных СУБД, тема оптимизации запросов всё ещё плохо изучена, что связано с отсутствием систем, которые позволяли бы проводить подобные исследования.

Архитектуру PosDB можно представить в виде набора следующих модулей: каталог, синтаксический анализатор, упроститель и оптимизатор запросов, исполнитель запросов, система хранения и управления памятью и утилиты. Говоря о PosDB, нужно понимать, что данные могут представляться в кортежном и позиционном виде. В первом случае данные представляются в виде кортежей непосредственно значений, во втором –- в виде номеров позиций в колонке. Важно отметить, что нельзя просто так перейти от одного представления к другому, нужно произвести специальную операцию, называемую материализацией. Материализация -– это составление кортежей, которое происходит из-за того, что данные на диске хранятся и обрабатываются в колоночном виде, а результаты запросов нужно представить в виде кортежей. Именно материализация порождает ограничения и правила, которые нужно учитывать при построении планов.

2.2. Обзор существующих визуализаторов планов

Так как в ходе обзора визуальные редакторы планов не были обнаружены, было решено проанализировать существующие визуализаторы, чтобы установить, каким образом специалисты в сфере администрирования баз данных представляют себе визуализацию планов. Для обзора были взяты самые распространённые визуализаторы планов, которые есть в свободном доступе, как веб-приложения, так и десктопные. Были проанализированы такие составляющие, как визуальный язык, внешнее устройство, различные полезные для пользователя функции, а также дизайн и удобство использования.

2.2.1. explain.dalibo.com

explain.dalibo.com [21] – веб-сервис, позволяющий анализировать и визуализировать планы выполнения запросов для СУБД PostgreSQL [14]. Инструмент использует JavaScript-библиотеку PEV2 [6] для анализа и отображения планов. Можно отметить, что сервис предлагает пользователю различные виды и функции по отображению планов, например, компактное и очень компактное отображение схемы, подсветка и скрытие поддерева узла. Все операции в плане представлены в виде однотипных узлов с названиями, соответствующими выполняемому действию. Что касается связей, то все они имеют одинаковый вид. Сам план представлен в виде дерева с корнем сверху. Инструмент может показывать дополнительную необходимую информацию, выделять затратные операции соответствующими значками и цветами и сохранять визуализированный план в архив, однако не позволяет передвигать узлы и связи. Из недостатков можно отметить трудность восприятия больших планов из-за одинаковости узлов, а также невозможности перемещения и изменения размеров узлов. На рисунке 1 представлен пример визуализации плана с помощью explain.dalibo.com.

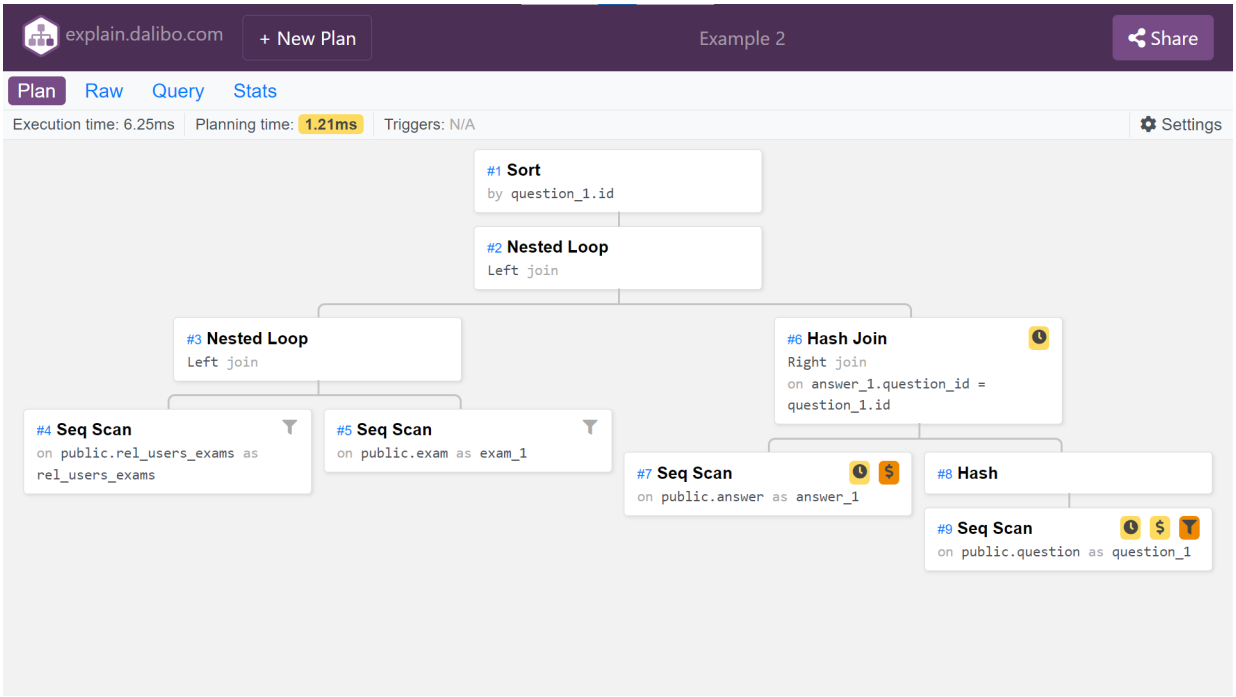

Рис. 1: Пример визуализированного плана в explain.dalibo.com

2.2.2. Explain PostgreSQL

Explain PostgreSQL [4] – веб-инструмент для анализа и визуализации планов любой сложности, предназначенный для запросов в СУБД PostgreSQL. Сервис обладает большим количеством встроенных функций для анализа, показывает подробную информацию и советы по оптимизации при наведении на нужный элемент, отображает узлы в виде картинок с соответствующей операцией и изменяет цвет его ободка и толщину связи для затратных действий, а также имеет историю визуализаций. Кроме того, можно отметить, что инструмент не предусматривает перемещение элементов на сцене и отображает план с корнем справа. Пример визуализированного плана в Explain PostgreSQL можно увидеть на рисунке 2.

2.2.3. DBeaver Enterprise Edition

DBeaver Enterprise Edition – одна из версий десктопного кросс-платформенного приложения DBeaver [2] для администрирования баз данных. Данная версия является платной, но предоставляет двухнедельный пробный период. Инструмент умеет работать с большим количе-

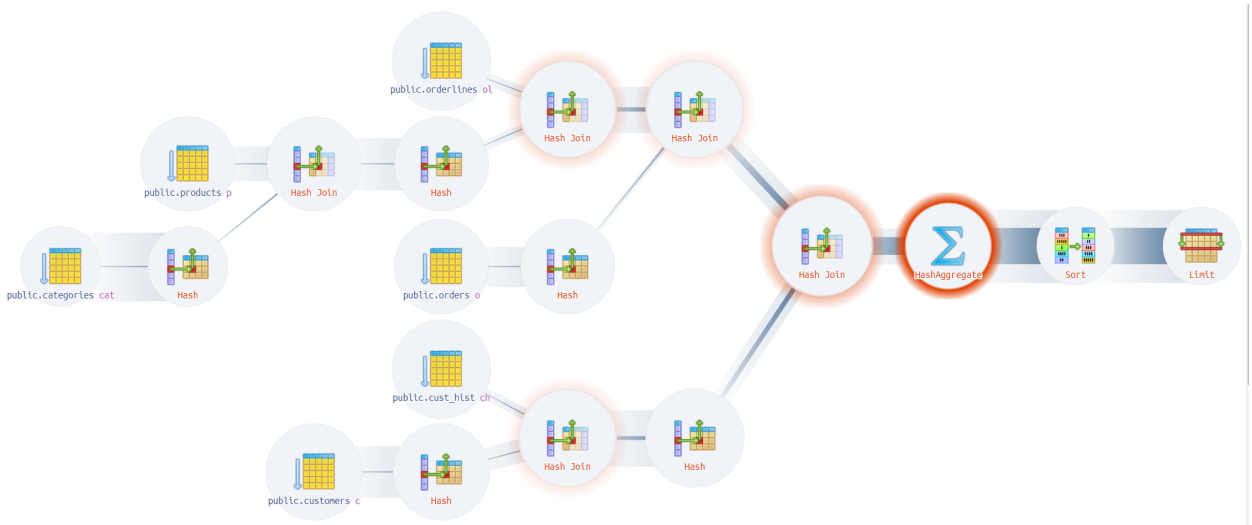

Рис. 2: Пример визуализированного плана в Explain PostgreSQL

ством различных баз данных, в том числе такими, как PostgreSQL, Microsoft SQL Server [10], Oracle [12], MySQL [11], а также NoSQL [20]. Широкая функциональность приложения включает в себя инструмент для визуализации и взаимодействия с планами выполнения запросов. DBeaver позволяет производить манипуляции с элементами плана: перемещать узлы, изменять их размер, откреплять связи и добавлять на них точки перегиба, сворачивать поддеревья, а также отображать и скрывать детали. Помимо этого, в приложении реализованы операции отмены и возврата, множественное выделение элементов и раскраска узла в различные цвета в зависимости от затратности соответствующего действия. Пример визуализации плана в DBeaver Enterprise Edition представлен на рисунке 3.

2.3. Используемые технологии

2.3.1. REAL.NET

REAL.NET – среда для создания визуальных предметно-ориентированных языков, которая разрабатывается на кафедре системного программирования СПбГУ. Первая реализованная версия была десктопной и создавалась с помощью платформы .NET.

На данный момент REAL.NET представляет собой веб-приложение

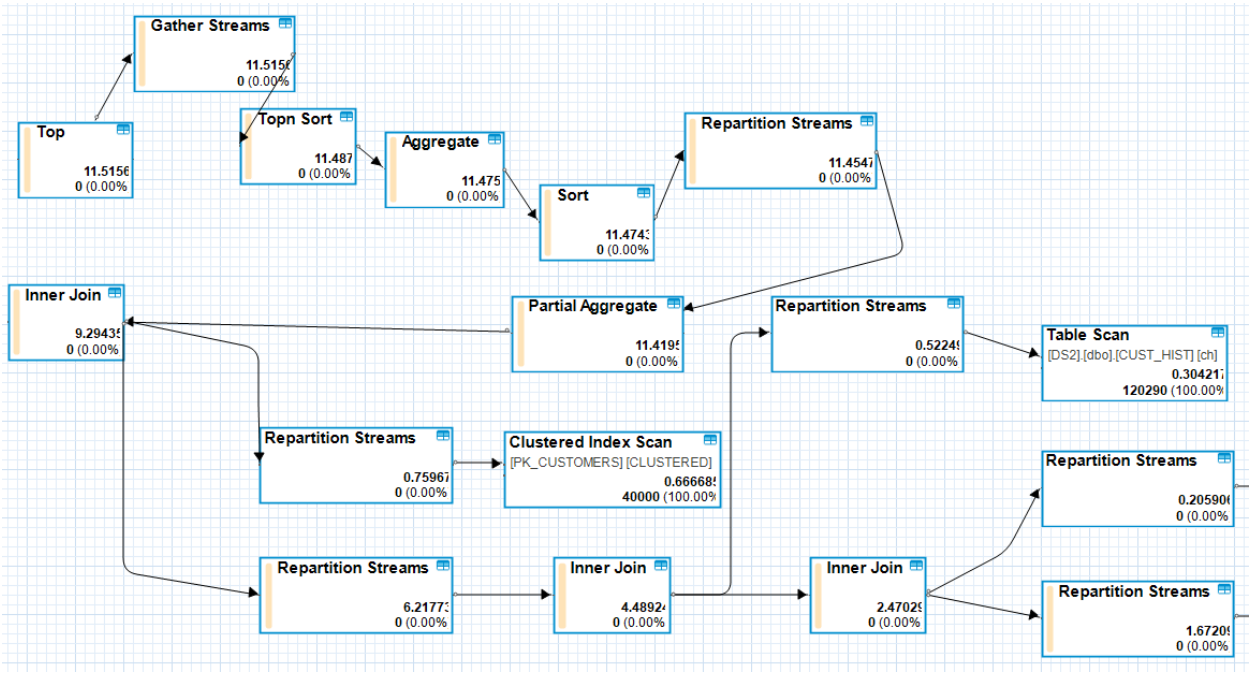

Рис. 3: Пример визуализированного плана в DBeaver Enterprise Edition

и имеет серверную и клиентскую части. Серверная часть реализует микросервисную архитектуру и состоит из нескольких компонентов, основным из которых является репозиторий. Он хранит модели и метамодели, а также позволяет управлять ими: создавать, изменять, удалять. Диаграмма архитектуры серверной части представлена на рисунке 4. Её более подробное описание см. в работе [24].

Клиентская часть представляет собой веб-редактор, разработанный с использованием библиотеки React [17] и библиотеки React Flow [15] для отображения и построения диаграмм. Веб-редактор состоит из трёх частей: палитры, которая содержит элементы метамодели; сцены, на которой строятся модели, и панели работы со свойствами, в которой можно изменять их значения. Компоненты взаимодействуют с репозиторием бэкенда через серверную часть фронтенда, спроектированную по паттерну Gateway для обеспечения более удобного интерфейс доступа к API. Внешний вид веб-редактора REAL.NET представлен на рисунке 5. Для более подробного описания клиентской части см. работу [25].

В десктопную версию был также добавлен модуль проверки ограничений [22]. Для создания ограничений на какой-либо язык создается

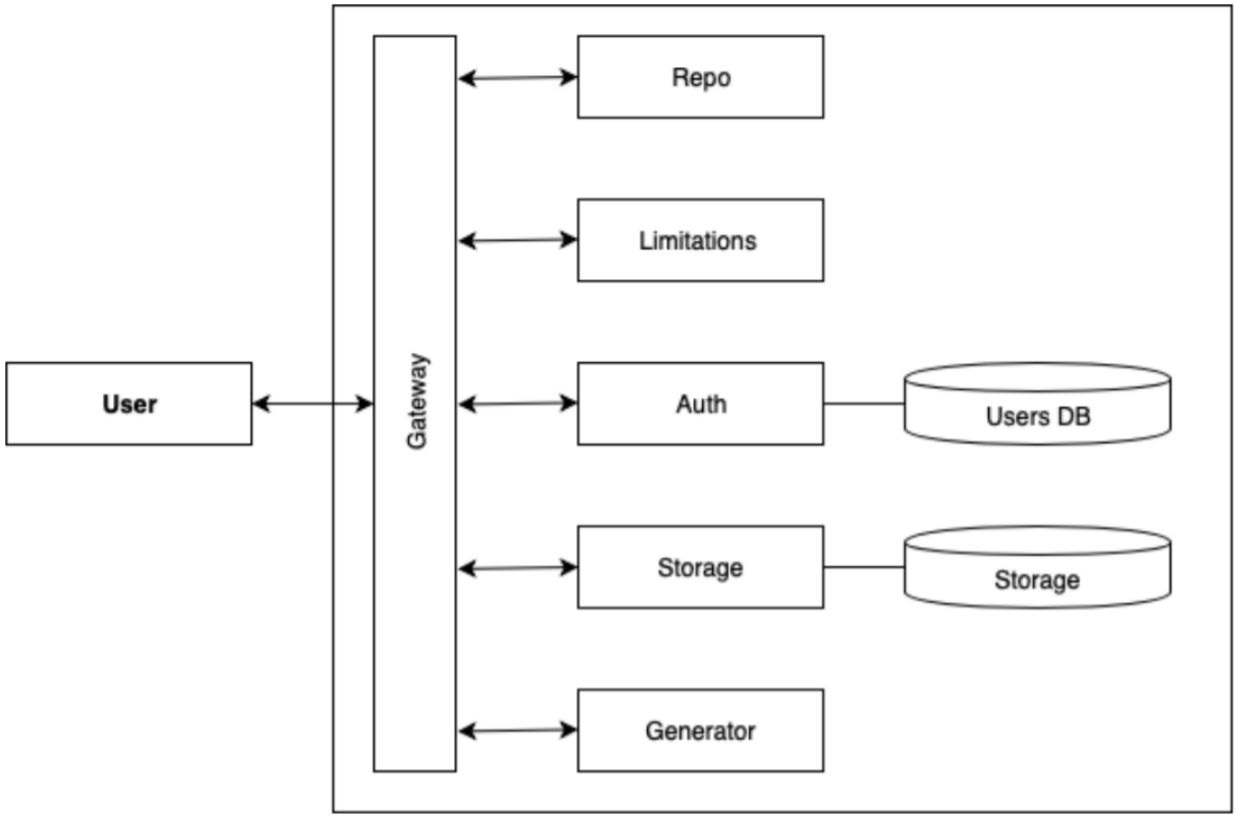

Рис. 4: Диаграмма архитектуры серверной части REAL.NET

новая метамодель. В неё включаются две группы элементов: элементы исходного визуального языка и элементы проверки ограничений, а именно логические операции и связи, которые специфичны для данного языка ограничений.

2.3.2. React

React – JavaScript-библиотека для создания пользовательских интерфейсов, которая призвана предоставить приложению высокую скорость, простоту и масштабируемость. Основными особенностями библиотеки React являются использование JSX-разметки, работа с виртуальным DOM (Document Object Model), компонентный подход, а также методы жизненного цикла и хуки.

JSX-разметка – расширение синтаксиса JavaScript, которое позволяет применять HTML-подобный синтаксис для описания структуры интерфейса. JSX может создавать React-элементы и использовать любые корректные JavaScript-выражения.

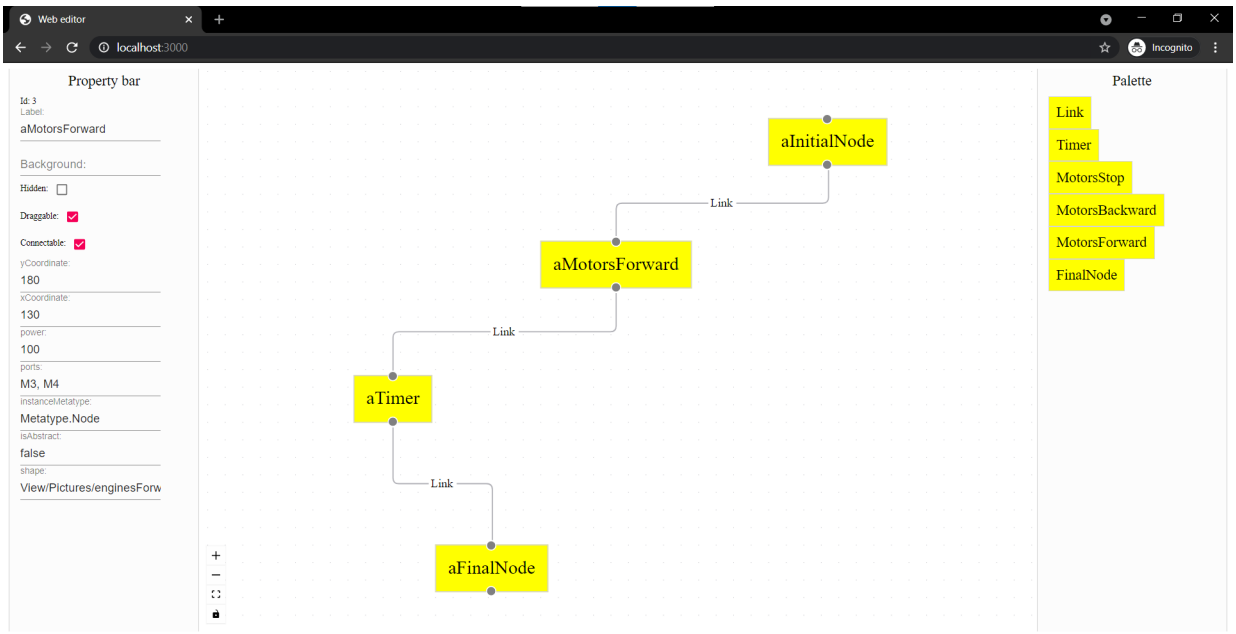

Рис. 5: Внешний вид веб-редактора REAL.NET

Следующей особенностью React является виртуальный DOM, который представляет собой легковесную копию обычного DOM, и React работает именно с виртуальным DOM, а не обычным. Если приложению нужно узнать информацию о состоянии элементов, то происходит обращение к виртуальному DOM, если же необходимо изменить элементы веб-страницы, то соответствующие изменения сначала вносятся в виртуальный DOM. После этого новое состояние виртуального DOM сравнивается с текущим и если они различаются, то React находит минимальное количество манипуляций, которые необходимо осуществить для обновления обычного DOM до нового состояния, и производит их. Такая схема взаимодействия с элементами веб-страницы работает гораздо быстрее и эффективнее, чем классическая, когда происходит обращение к DOM из JavaScript напрямую.

Ещё одна отличительная черта данной библиотеки – это использование компонентного подхода. Компоненты – это относительно маленькие, повторно используемые части кода, которые подобно обычным JavaScript-функциям принимают произвольные входные данные и возвращают React-элементы для отображения на странице. React позволяет определять компоненты как классы или функции. Важной со-

ставляющей компонентов являются props и state. Props – это входные данные React-компонентов, передаваемые от родительского компонента к дочернему и предназначенные только для чтения. State – это объект, описывающий внутреннее состояние компонента, определяющийся внутри него и доступный только там же.

Что касается методов жизненного цикла, то в React существуют специальные настраиваемые функции, с помощью которых можно определить, какие действия будут происходить на различных этапах жизни компонента. Наиболее распространёнными являются метод constructor, представляющий собой конструктор, в котором происходит инициализация компонента; метод render, отвечающий за отрисовку компонента; componentDidMount, метод, который вызывается после рендеринга компонента, и componentWillUnmount, который вызывается перед удалением компонента из DOM.

Последней особенностью библиотеки являются хуки. Хуки – это функции, с помощью которых можно «подцепиться» к состоянию и методам жизненного цикла React из функциональных компонентов. Примерами хуков являются useState, предназначенный для управления состоянием компонентов; useEffect, используемый для перехвата различного рода изменений в компонентах, которые нельзя обработать внутри компонентов; useCallback, который позволяет управлять функциями обратного вызова. Также нужно отметить, что в React возможно создавать свои хуки, если в этом есть необходимость.

Таким образом, учитывая все вышеперечисленные особенности, React является удобным и мощным инструментом для разработки визуального веб-редактора.

2.3.3. React Flow

React Flow – open-source библиотека, написанная на языке TypeScript с использованием React. Данная библиотека позволяет строить различные диаграммы, производить манипуляции с узлами и связями, создавать вложенные узлы, настраивать порты и другие параметры элементов. Кроме того, инструмент отличается высокой гибкостью,

а также обладает большим количеством различных полезных функций и обработчиков событий и современным пользовательским интерфейсом. Благодаря этим преимуществам библиотеку удобно использовать в интерактивных системах, в которых существуют различные классы элементов и нужно реагировать на большое количество разных событий.

3. Описание решения

3.1. Метамодель

На рисунке 6 представлен пример плана, который можно будет построить с помощью разрабатываемого редактора.

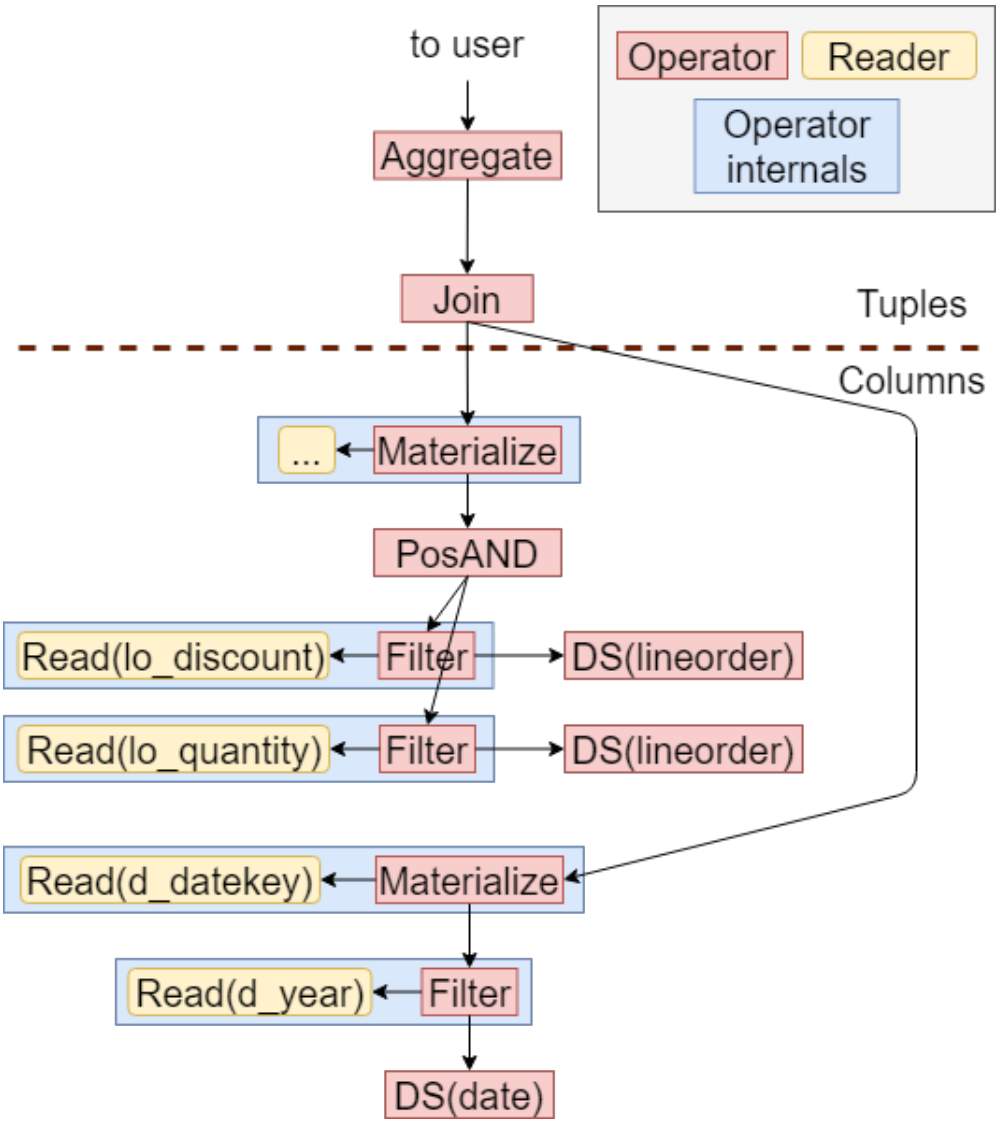

Рис. 6: Пример плана ожидаемого результата

Визуальный предметно-ориентированный язык для PosDB создавался самостоятельно с использованием наработок из похожих работ прошлых лет [23].

Для того, чтобы разработать язык, нужно в первую очередь определить, из каких сущностей должны состоять диаграммы. Согласно [5],

планы выполнения запросов в PosDB, для моделирования которых и разрабатывается язык, являются деревьями, то есть каждый узел может иметь родителя и потомков. Кроме того, имеются только три типа блоков: Operator с различными операциями, Reader для считывания данных и Operator internals, чтобы разграничить содержимое операторов; также есть планка материализации Materializaton line и два вида связей между узлами: local access для доступа к локальным данным и remote access, чтобы взаимодействовать с удалёнными данными.

На рисунке 7 представлена метамодель языка планов выполнения запросов в PosDB. Корнем метамодели является блок AbstractQueryBlock, от которого наследуются все остальные элементы метамодели, в том числе Operator, Reader, Operator internals и Materialization line.

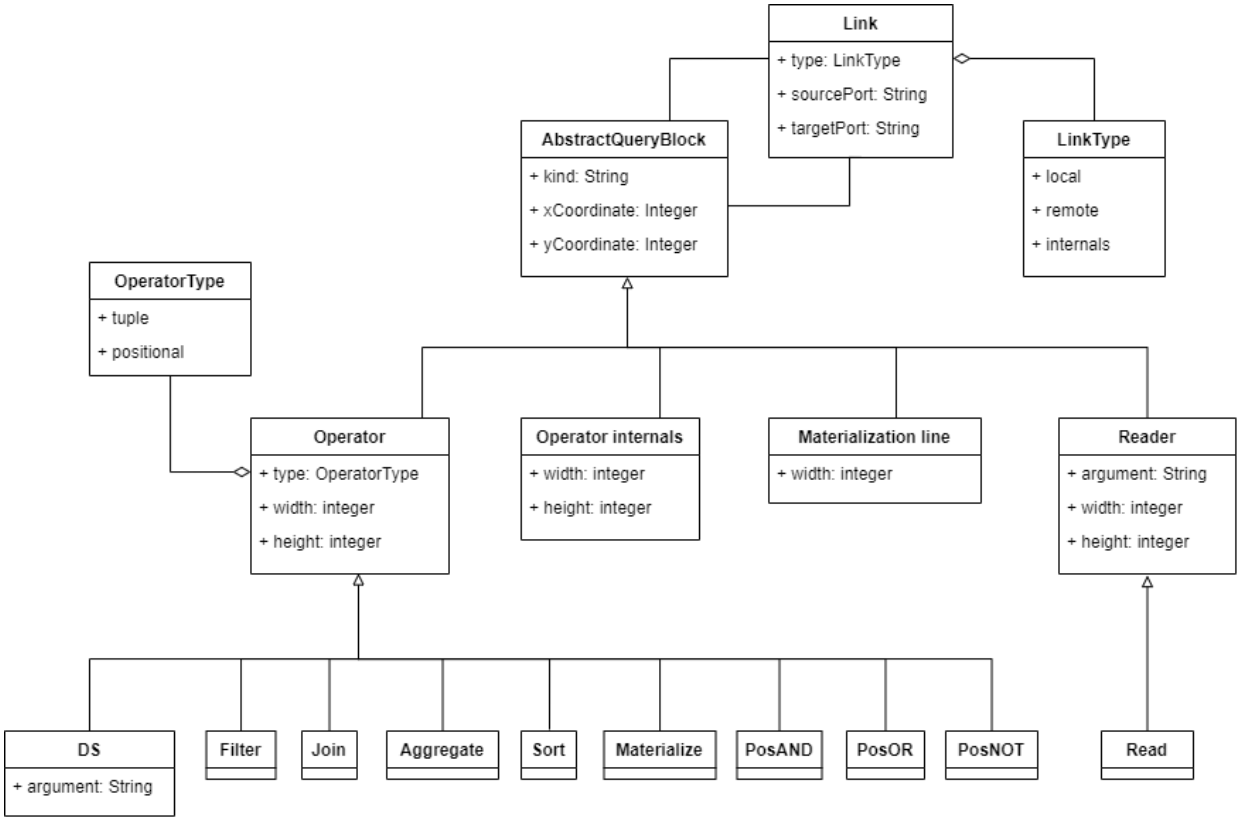

Рис. 7: Метамодель языка планов выполнения запросов в PosDB

У всех узлов имеются следующие атрибуты: kind, представляющий вид узла, – Operator, Reader, Operator internals или Materialization line; xCoordinate и yCoordinate – координаты по оси X и Y соответственно.

У элемента Operator есть атрибут type, содержащий тип оператора: позиционный (positional) или кортежный (tuple). У элемента Reader имеется атрибут argument, содержащий информацию об аргументе считывателя. Кроме того, у всех узлов есть атрибуты с размерами: шириной (width) и высотой (height).

На данный момент в метамодель было добавлено несколько операторов, которые соответствуют различным операциям: оператор доступа к базе данных DS (DataSource), оператор сортировки Sort, оператор соединения данных Join, оператор агрегации Aggregate, оператор фильтрации Filter, оператор материализации Materialize, а также три оператора PosAND, PosOR, PosNOT, выполняющих логические операции AND, OR, NOT соответственно над позициями, и считыватель данных Read.

Текущие результаты представлены в репозитории микросервиса Repo на GitHub [8]

3.2. Система проверки ограничений

Система проверки ограничений уже была встроена в веб-редактор в виде составляющей части микросервиса по работе с репозиторием на серверной части веб-редактора. Задачей было адаптировать и настроить систему для работы с ограничениями, специфичными для построения планов в PosDB.

Архитектуру системы проверки ограничений можно увидеть на рисунке 8. Для её реализации использовался паттерн Стратегия.

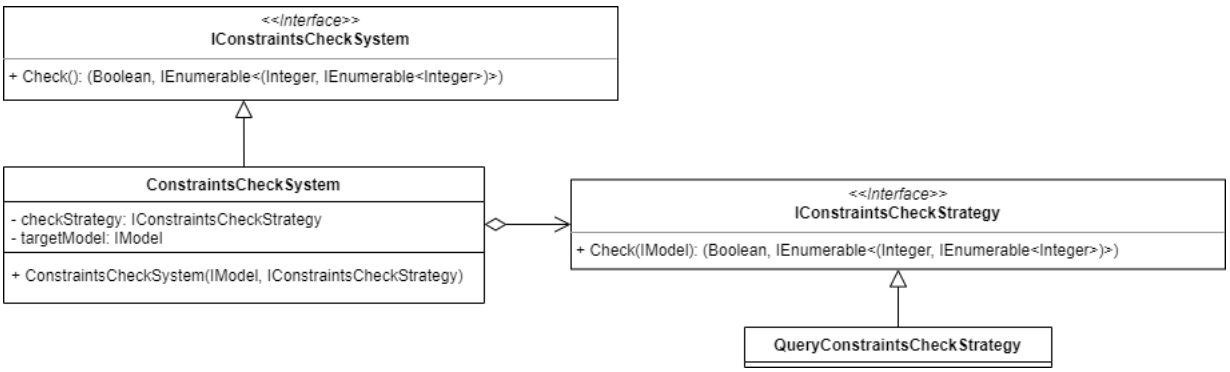

Рис. 8: Архитектура системы проверки ограничений

Класс ConstraintsCheckSystem, реализующий интерфейс IConctraintsCheckSystem, представляет собой систему проверки ограничений и с помощью метода Check() позволяет производить проверку выполнения ограничений. Метод Check() возвращает кортеж из двух значений: булевого результата проверки, равного true, если проверка успешно пройдена, и false иначе, и перечисляемой коллекции, реализующей интерфейс IEnumerable. Коллекция содержит информацию об имеющихся ошибках: код непройденного ограничения и список идентификаторов элементов, которые нарушают данное правило.

Проверка, в свою очередь, определяется стратегией, реализующей интерфейс IConstraintsCheckStrategy с вышеупомянутым методом Check(). Для ограничений при построении планов в PosDB была создана стратегия QueryConstraintsCheckStrategy.

На данный момент поддерживается проверка следующих ограничений.

- После материализующего оператора должны идти только кортежные операторы.
- На выход плана запроса должен выдаваться результат в кортежном представлении.
- В листах дерева всегда должны быть операторы DataSource.
- У позиционных операторов должны быть заданы считыватели.
  - У некоторых позиционных бинарных операторов (например, соединение) должны быть заданы как минимум два считывателя.
- У кортежных операторов не должно быть считывателей.
- У n-арного кортежного оператора все поставщики данных должны быть кортежными.
- У n-арного позиционного оператора все поставщики данных должны быть позиционными.

Текущие результаты представлены в репозитории бэкенд-части проекта на GitHub [7].

3.3. Визуальный редактор

На данный момент визуальный редактор имеет следующий вид.

- Справа представлена палитра с узлами, разделёнными на группы по их виду: позиционные и кортежные, а также узел с содержимым оператора и планка материализации.
- Слева располагается панель работы со свойствами, позволяющая изменять значения параметров элементов, а также манипулировать некоторыми визуальными свойствами.
- Среднюю и основную часть занимает сцена, на которой создаются и изменяются диаграммы.
- В правом верхнем углу сцены могут отображаться предупреждения о нарушении ограничений.

Внешний вид визуального редактора представлен на рисунке 9.

Редактор позволяет создавать новые узлы через перетаскивание соответствующего элемента из палитры на сцену. Некоторые позиционные операторы (например, Filter, Sort, Join) создаются сразу с блоком типа Operator internals и минимальным числом считывателей. Для того, чтобы добавить новый Reader в блок Operator internals, необходимо перетащить считыватель внутрь узла либо со сцены, либо из палитры. Что касается связей, то они создаются через соединение портов узлов с помощью зажатия левой кнопки мыши. Редактор поддерживает отображение двух видов связей: сплошная стрелка обозначает доступ к локальным данным, а пунктирная – доступ к удалённым данным.

Для удаления элементов можно использовать клавишу Delete или же клик правой кнопкой мыши по нужному элементу и выбор соответствующего пункта в контекстном меню. Стоит отметить, что при удалении узла типа Operator internals удаляется и всё его содержимое.

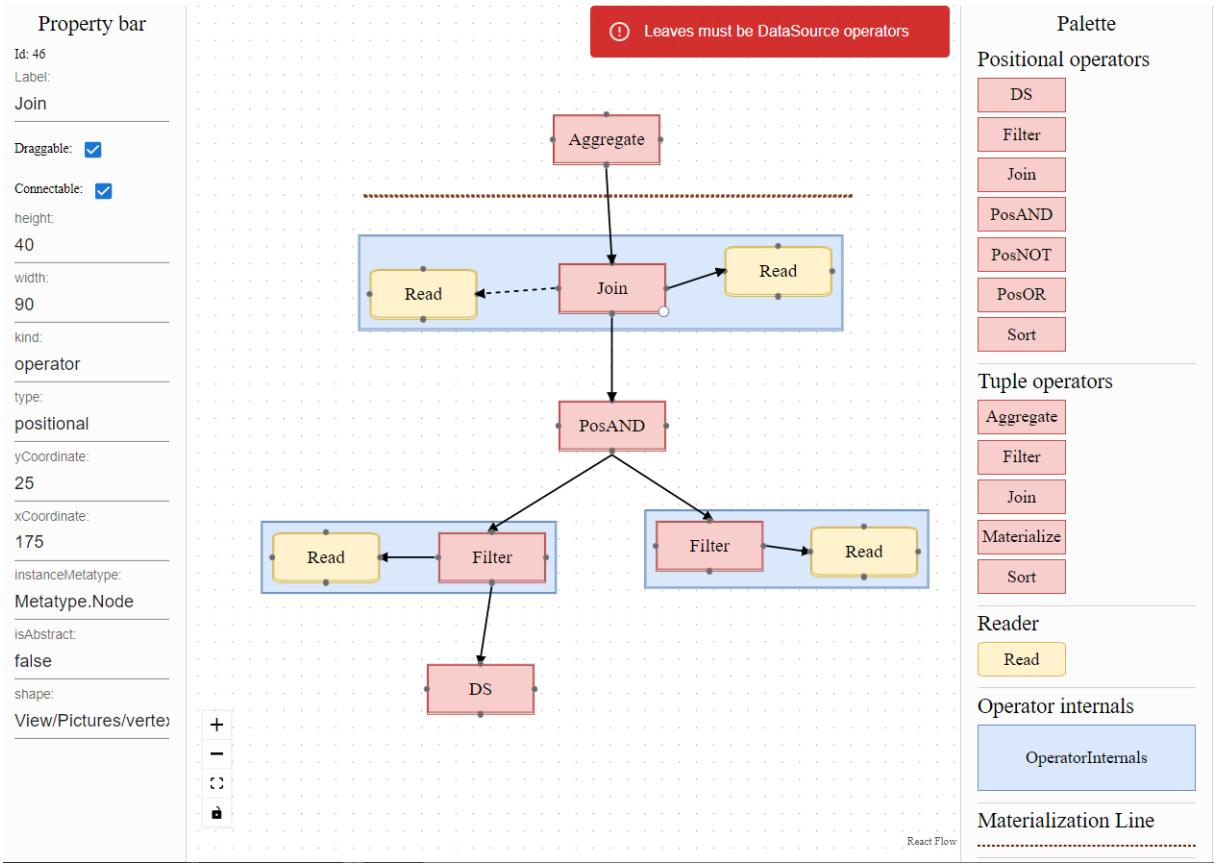

Рис. 9: Внешний вид визуального редактора

Также в редакторе есть возможность множественного выделения элементов на сцене для перемещения с помощью мышки и клавиши Shift.

Размеры всех узлов на сцене, включая планку материализации, можно изменять, потянув за ползунок в правом нижнем углу выделенного элемента. Кроме того, узел Operator internals умеет по нажатию специальной кнопки самостоятельно подстраивать свои размеры под размеры блоков, которые лежат внутри него. Для этого вычисляются координаты внутренних узлов и определяются их самые крайние верхние, нижние, правые и левые границы, после чего и высчитываются новые координаты и размеры блока Operator internals. Чтобы реализовать функциональность по изменению размеров узлов на сцене, использовалась библиотека React-Resizable [16].

После выполнения любой операции с элементами вызывается проверка системы ограничений. В случае если она возвращает false, то есть модель нарушает какие-либо правила, редактор отображает соответ-

ствующие предупреждения в виде списка с описаниями ошибок. После их исправления предупреждения исчезают.

Текущие результаты представлены в репозитории фронтенд-части проекта на GitHub [9].

4. Апробация

Апробация проводилась с использованием метода Single Ease Question (SEQ) [18] и шкалы System Usability Scale (SUS) [1]. В ней будут участвовали 8 человек: разработчики движка PosDB, а также их студенты, которые обучаются рисованию планов запросов.

Метод SEQ позволяет оценить усилия, которые пользователь прикладывает для выполнения той или иной задачи, и понять, насколько легко или сложно осуществить задание. Для этого используется семибалльная шкала, где 1 – это «очень сложно» и 7 – «очень легко». Чтобы вычислить значение SEQ считается среднее арифметическое всех оценок.

Шкала SUS – это опросник, состоящий из десяти вопросов, который направлен на оценку удобства использования продукта для пользователей. Для вычисления итогового балла SUS использует специальную формулу, которая выдаёт значение от нуля до ста. Ответ интерпретируется в соответствии с существующей шкалой.

Участникам апробации предлагалось выполнить несколько заданий, таких как «Добавьте из палитры два оператора, а также планку материализации и расположите их так, чтобы сверху был первый оператор, снизу – другой, а между ними была планка материализации», «Создайте связь от одного оператора к другому, затем измените тип любой связи с local на remote» и другие. После выполнения каждого задания нужно было оценить его сложность согласно методу SEQ. В конце анкетирования респонденты отвечали на вопросы шкалы SUS.

Результаты опроса представлены в таблице 1.

Средний балл по оценке SEQ составил 6, что выше среднестатистического значения в 5.5. Среднее количество баллов по шкале SUS равно 90, что соответствует буквенной оценке «A+» («Потрясающе»).

После использования респонденты отмечали удобство системы в целом и тот факт, что редактор не требует много времени на обучение работе. Помимо этого, в ходе анкетирования были выявлены некоторые недостатки редактора: маленький размер портов для проведения

Респондент	Оценка SEQ				Оценка SUS
	Задание 1	Задание 2	Задание 3	Задание 4	
1	7	7	7	6	92.5
2	7	7	7	7	95
3	5	7	6	5	90
4	3	5	4	4	82.5
5	7	7	7	7	100
6	7	6	7	6	97.5
7	7	6	6	6	87.5
8	6	4	6	4	80

Таблица 1: Результаты опроса

связей между узлами, маленькая область действия у некоторых кнопок и элементов на сцене и медленный отклик системы. Также респонденты оставили свои предложения по улучшению системы: реализовать возможность динамического добавления других операторов и добавить кнопку экспорта нарисованных планов в виде pdf или png.

Заключение

В рамках выпускной квалификационной работы работы были достигнуты следующие результаты.

- Проведён обзор движка и существующих визуализаторов планов.
- Разработан визуальный предметно-ориентированный язык для создания планов запросов в PosDB.
- Реализован визуальный редактор с необходимой функциональностью в пользовательском интерфейсе.
- Добавлена система проверки ограничений при построении планов.
- Проведена апробация.

Список литературы

[1] Bangor Aaron, Kortum Philip T., Miller James T. An Empirical Evaluation of the System Usability Scale // International Journal of Human–Computer Interaction. — 2008. — Vol. 24, no. 6. — P. 574–594.

[2] DBeaver. — URL: https://dbeaver.com (дата обращения: 29.04.2022).

[3] Erwin Ben, Cyr Martha, Rogers Chris. LEGO Engineer and RoboLab: Teaching Engineering with LabVIEW from Kindergarten to Graduate School // International Journal of Engineering Education. — 2000. — Vol. 16, no. 3. — P. 181–192.

[4] Explain PostgreSQL. — URL: https://explain.tensor.ru (дата обращения: 29.04.2022).

[5] Galaktionov Viacheslav, Klyuchikov Evgeniy, Chernishev George. Position Caching in a Column-Store with Late Materialization: An Initial Study // Proceedings of the 22nd International Workshop on Design, Optimization, Languages and Analytical Processing of Big Data co-located with EDBT/ICDT 2020 Joint Conference, DOLAP@EDBT/ICDT 2020, Copenhagen, Denmark, March 30, 2020. — Copenhagen, Denmark, 2020. — P. 89–93.

[6] GitHub репозиторий PEV2. — URL: https://github.com/dalibo/pev2 (дата обращения: 29.04.2022).

[7] GitHub репозиторий бэкенд-части проекта. — URL: https://github.com/REAL-NET/web-editor-backend/tree/query (дата обращения: 29.04.2022).

[8] GitHub репозиторий микросервиса Repo. — URL: https://github.com/REAL-NET/Repo/tree/query (дата обращения: 29.04.2022).

[9] GitHub репозиторий фронтенд-части проекта. — URL: https:

//github.com/REAL-NET/web-editor-frontend/tree/query (дата обращения: 29.04.2022).

[10] Microsoft SQL Server documentation. — URL: https://docs.microsoft.com/en-us/sql (дата обращения: 29.04.2022).

[11] MySQL documentation. — URL: https://dev.mysql.com/doc (дата обращения: 29.04.2022).

[12] Oracle documentation. — URL: https://docs.oracle.com (дата обращения: 29.04.2022).

[13] PosDB: обзор архитектуры / Г. А. Чернышев, В. А. Галактионов, В. Д. Григорьев и др. // Программирование. — 2018. — № 1. — С. 60–76.

[14] PostgreSQL documentation. — URL: https://www.postgresql.org/docs (дата обращения: 29.04.2022).

[15] React Flow documentation. — URL: https://reactflow.dev/docs (дата обращения: 29.04.2022).

[16] React-Resizable GitHub repository. — URL: https://github.com/react-grid-layout/react-resizable (дата обращения: 29.04.2022).

[17] React documentation. — URL: https://reactjs.org/docs/getting-started.html (дата обращения: 29.04.2022).

[18] Sauro Jeff, Dumas Joseph S. Comparison of three one-question, post-task usability questionnaires // CHI '09: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. — Boston, MA, USA, 2009. — P. 1599–1608.

[19] The Scratch Programming Language and Environment / John Maloney, Mitchel Resnick, Natalie Rusk et al. // ACM Transactions on Computing Education. — 2010. — Vol. 10, no. 4. — P. 1–15.

[20] Wikipedia. NoSQL // Wikipedia, The Free Encyclopedia. — URL: https://ru.wikipedia.org/wiki/NoSQL (дата обращения: 29.04.2022).

[21] explain.dalibo.com. — URL: https://explain.dalibo.com (дата обращения: 29.04.2022).

[22] Алымова Д. А. Визуальный язык задания ограничений на модели в REAL.NET. — Сайт кафедры системного программирования СПбГУ. — 2019. — URL: https://se.math.spbu.ru/thesis/texts/Alymova_Dar%27ja_Andreevna_Bachelor_Thesis_2019_text.pdf (дата обращения: 29.04.2022).

[23] Ивашева В. М. Визуальная технология обработки медицинских изображений при помощи библиотеки MIRF на основе REAL.NET. — Сайт кафедры системного программирования СПбГУ. — 2021. — URL: https://se.math.spbu.ru/thesis/texts/Ivasheva_Valerija_Mihajlovna_Bachelor_Thesis_2021_text.pdf (дата обращения: 29.04.2022).

[24] Кидянкин М. В. Микросервисная архитектура DSM-платформы REAL.NET Web. — Сайт кафедры системного программирования СПбГУ. — 2020. — URL: https://se.math.spbu.ru/thesis/texts/Kidjankin_Mihail_Vladimirovich_Bachelor_Report_2020_text.pdf (дата обращения: 29.04.2022).

[25] Ким Ю. А. Веб-редактор для платформы REAL.NET. — Сайт кафедры системного программирования СПбГУ. — 2021. — URL: https://se.math.spbu.ru/thesis/texts/Kim_Junija_Aleksandrovna_Bachelor_Report_2021_text.pdf (дата обращения: 29.04.2022).

[26] Литвинов Ю. В., Кириленко Я. А. TRIK Studio: среда обучения программированию с применением роботов // V Всероссийская конференция «Современное технологическое обучение: от компьютера к роботу» (сборник тезисов). — СПб, 2015. — С. 5–7.

[27] Среда предметно-ориентированного визуального моделирования REAL.NET / Ю. В. Литвинов, Е. В. Кузьмина, И. Ю. Небогатиков, Д. А. Алымова // СПИСОК-2017. Материалы 7-й всероссийской научной конференции по проблемам информатики. — СПб, 2017. — С. 80–89.