**ОГЛАВЛЕНИЕ**

ВЕДОМОСТЬ ОБЪЕМА ДИПЛОМНОГО ПРОЕКТА

[ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ 7](#_Toc136519502)

[ВВЕДЕНИЕ 8](#_Toc136519503)

[1 АНАЛИЗ ВОЗМОЖНОСТИ РЕАЛИЗАЦИИ ИНТЕРАКТИВНОГО ТРЕНАЖЕРА ОБУЧЕНИЯ ПОСТРОЕНИЮ ПРОЕКЦИЙ ТОЧЕК И ОТРЕЗКОВ НА ЭПЮРЕ МОНЖА 9](#_Toc136519504)

[1.1 Анализ способов реализации интерактивного тренажера обучения построению проекций точек и отрезков на эпюре Монжа 9](#_Toc136519505)

[1.2 Обзор программ-аналогов для моделирования бизнес-процессов 11](#_Toc136519506)

[1.3 Цели и постановка задачи 14](#_Toc136519507)

[2 ВЫБОР ПРОГРАММНО-ТЕХНИЧЕСКИХ СРЕДСТВ РЕАЛИЗАЦИИ WEB-ПРИЛОЖЕНИЯ «ИНТЕРАКТИВНЫЙ ТРЕНАЖЕР ОБУЧЕНИЯ ПОСТРОЕНИЮ ПРОЕКЦИЙ ТОЧЕК И ОТРЕЗКОВ НА ЭПЮРЕ МОНЖА» 16](#_Toc136519508)

[2.1 Выбор стека технологий для серверной части 16](#_Toc136519509)

[2.2 Выбор системы управления базой данных 17](#_Toc136519510)

[2.3 Выбор стека технологий для клиентской части 18](#_Toc136519511)

[3 ПРОЕКТИРОВАНИЕ WEB-ПРИЛОЖЕНИЯ «ИТЕРАКТИВНЫЙ ТРЕННАЖЕР ОБУЧЕНИЯ ПОСТРОЕНИЮ ПРОЕКЦИЙ ТОЧЕК И ОТРЕЗКОВ НА ЭПЮРЕ МОНЖА» 20](#_Toc136519512)

[3.1 Общая структура взаимодействия компонентов в web-приложении 20](#_Toc136519513)

[3.2 Диаграмма вариантов использований 21](#_Toc136519514)

[3.3 Концептуальная модель базы данных 24](#_Toc136519515)

[3.4 Логическая модель базы данных 26](#_Toc136519516)

[3.5 Физическая модель базы данных 29](#_Toc136519517)

[4 РЕАЛИЗАЦИЯ WEB-ПРИЛОЖЕНИЯ «ИТЕРАКТИВНЫЙ ТРЕННАЖЕР ОБУЧЕНИЯ ПОСТРОЕНИЮ ПРОЕКЦИЙ ТОЧЕК И ОТРЕЗКОВ НА ЭПЮРЕ МОНЖА» 31](#_Toc136519518)

[4.1 Реализация серверной части 31](#_Toc136519519)

[4.2 Реализация клиентской части 39](#_Toc136519520)

[4.2.1 Раздел управления платформами 42](#_Toc136519521)

[4.2.2 Конструктор задач 43](#_Toc136519522)

[4.2.3 Раздел выполнения узлов и задач 48](#_Toc136519523)

[5 ТЕСТИРОВАНИЕ WEB-ПРИЛОЖЕНИЯ «ИТЕРАКТИВНЫЙ ТРЕННАЖЕР ОБУЧЕНИЯ ПОСТРОЕНИЮ ПРОЕКЦИЙ ТОЧЕК И ОТРЕЗКОВ НА ЭПЮРЕ МОНЖА» 52](#_Toc136519524)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 57](#_Toc136519525)

[6.1 Раздел общей информации 57](#_Toc136519526)

[6.2 Раздел управления платформами 58](#_Toc136519527)

[6.3 Раздел конструирования задач 61](#_Toc136519528)

[6.4 Раздел выполнения узлов и задач 65](#_Toc136519529)

[6.5 Пример использования web-приложения 67](#_Toc136519530)

[7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПП 75](#_Toc136519531)

[7.1 Определение единовременных затрат на создание программного продукта (разработку модели) 75](#_Toc136519532)

[7.1.1 Определение трудоемкости разработки ПП 75](#_Toc136519533)

[7.1.2 Определение себестоимости создания ПП 78](#_Toc136519534)

[7.1.3 Определение минимальной цены ПП 82](#_Toc136519535)

[7.2 Определение ожидаемого прироста прибыли в результате внедрения 83](#_Toc136519536)

[7.2.1 Определение годовых эксплуатационных расходов при ручном решении задач 83](#_Toc136519537)

[7.2.2 Определение годовых текущих затрат, связанных с эксплуатацией задачи 84](#_Toc136519538)

[7.2.3 Определение дополнительной прибыли пользователя за период использования ПП 85](#_Toc136519539)

[7.2.4 Определение максимальной, планируемой и отпускной цены 86](#_Toc136519540)

[7.3 Расчет показателей эффективности использования программного продукта 86](#_Toc136519541)

[7.4 Оценка конкурентоспособности объекта проектирования 88](#_Toc136519542)

[8 ОХРАНА ТРУДА 93](#_Toc136519543)

[8.1 Производственая санитария, техника безопасности и пожарная профилактика 93](#_Toc136519544)

[8.1.1 Метеоусловия 93](#_Toc136519545)

[8.1.2 Вентиляция и отопление 94](#_Toc136519546)

[8.1.3 Освещение 95](#_Toc136519547)

[8.1.4 Шум 96](#_Toc136519548)

[8.1.5 Электробезопасность 96](#_Toc136519549)

[8.1.6 Излучение 97](#_Toc136519550)

[8.1.7 Пожарная безопасность 98](#_Toc136519551)

[8.2 Требования к помещениям для работы с ВДТ, ЭВМ и ПЭВМ 99](#_Toc136519552)

[ЗАКЛЮЧЕНИЕ 101](#_Toc136519553)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_Toc136519554)

ГРАФИЧЕСКАЯ ЧАСТЬ

[ПРИЛОЖЕНИЕ А](#_Toc136519555)

# ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

В настоящей пояснительной записке применяются следующие определения и сокращения:

* БД – база данных;
* СУБД – система управления базами данных; ПО – программное обеспечение;
* ПП – программный продукт;
* API – программный интерфейс приложения;
* ACID – требования к СУБД, обеспечивающие наиболее надёжную и предсказуемую её работу;
* REST – архитектурный стиль взаимодействия компонентов распределённого приложения в сети;
* HTTP – протокол прикладного уровня передачи данных;
* Скриптинг – написание сценариев (скриптов) на языках программирования.

# ВВЕДЕНИЕ

В настоящее время на рынке компьютерных технологий существует множество программных инструментов, позволяющих обследовать предметную область и построить модель. Целью данного проекта является создание web-приложения для обучения построению проекций точек на эпюре Монжа посредством организации потока задач.

Актуальность темы визуального представления и организации потока задач заключается в том, что помимо простого моделирования бизнес-процесса в виде блок-схемы, каждому блоку можно назначить свою задачу, которую он будет выполнять. Таким образом сформированная модель будет нести в себе не только информативный характер, но и обладать поведением, автоматизируя решение проблемы. Правильная организация бизнес-процесса позволяет значительно повысить качество и скорость создания продукта.

В сфере обучения бизнес-процесс может рассматриваться как набор учебных задач, которые сперва создаются, затем решаются и проверяются на правильность решения. Таким образом происходит оценка усвоения определенных знаний, а также обучение на допущенных ошибках. Одними из наиболее важных и сложных являются графические задачи. В том числе по начертательной геометрии, инженерной графике и 3D моделированию.

В данный момент современные системы позволяют использовать только фиксированный набор типов задач, которые представляют набор текстовых полей, списков, выбор вариантов ответов. На этом все достижения заканчиваются. Такие задачи предназначены только для проверки знаний. Они не дают комментариев, не предлагают для решения более простые задачи. У таких систем нет плана обучения, который смог бы подстроиться под знания испытуемого. А ограниченные типы задач часто затрудняют создания качественных тестов.

Решением данной проблемы является разработка программной платформы для создания задач, которую можно расширять внешними программными проектами, а также получать полный отчет о выполнении этих задач, иметь возможность обучаться, а не просто получать результат, а также комбинировать задачи в схемы, путем соединения входных и выходных параметров. В принципе такой подход позволит создавать задачи, которые смогут охватить практически любую предметную область, а повторное использование позволит создавать более сложные задачи, тем самым позволяя формировать целые сценарии обучения, которые могут автоматически адоптироваться к обучению конкретно взятого обучаемого.

# АНАЛИЗ ВОЗМОЖНОСТИ РЕАЛИЗАЦИИ ИНТЕРАКТИВНОГО ТРЕНАЖЕРА ОБУЧЕНИЯ ПОСТРОЕНИЮ ПРОЕКЦИЙ ТОЧЕК И ОТРЕЗКОВ НА ЭПЮРЕ МОНЖА

## Анализ способов реализации интерактивного тренажера обучения построению проекций точек и отрезков на эпюре Монжа

В сфере обучения существует множество решений, направленных на повышение эффективности усвоения полученных знаний. К одному из таких решений можно отнести электронное тестирование. Тестирование происходит путем создания учебных задач и дальнейшего их выполнения. Для удобства использования задачи группируются в тесты, которые в свою очередь можно изобразить в виде линейного бизнес-процесса. В рамках разработки web-приложения «Интерактивный тренажер обучения построению проекций точек и отрезков на эпюре Монжа» упор был сделан на предоставление инструментария для разработки многофункциональных заданий для проверки знаний, который позволит не ограничиваться на встроенных возможностях, а предоставит функционал для разработки собственных типов заданий. Так же был сделан упор на доступность и простоту в использовании данного инструментария.

Бизнес-процессом называется организованный комплекс взаимосвязанных действий, которые в совокупности дают ценный для клиента результат. На выходе бизнес-процесса клиент обязательно получает некий результат (может быть, не окончательный) [1]. Одним из способов представления бизнес-процессов является графический метод, который представляет из себя блок-схему. В нем все операции и данные изображаются специальными символами. Метод применяется для отображения логической последовательности действий процесса. Гибкость – главное достоинство данного метода.

Процесс может быть представлен множеством способов. Но для выполнения поставленных задач дипломного проектирования наиболее подходящим будет IDEF0. Методология IDEF0 дает возможность создать модель функции процесса. На рисунке 1.1 изображаются главные функции процесса, входы, выходы, управляющие воздействия и устройства, взаимосвязанные с основными функциями. Существует возможность расписать процесс до более низких уровней. Каждая IDEF0-диаграмма содержит блоки и дуги. Блоки изображают функции моделируемой системы. Дуги связывают блоки вместе и отображают взаимодействия и взаимосвязи между ними. Функциональные блоки (работы) на диаграммах изображаются прямоугольниками, означающими поименованные процессы, функции или задачи, которые происходят в течение определенного времени и имеют распознаваемые результаты. Блоки в IDEF0 размещаются по степени важности, как ее понимает автор диаграммы. Этот относительный порядок называется доминированием. Доминирование понимается как влияние, которое один блок оказывает на другие блоки диаграммы. Например, самым доминирующим блоком диаграммы может быть либо первый из требуемой последовательности функций, либо планирующая или контролирующая функция, влияющая на все другие. Наиболее доминирующий блок обычно размещается в верхнем левом углу диаграммы, а наименее доминирующий – в правом углу.

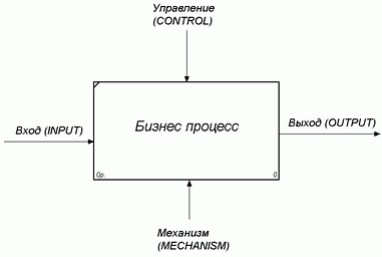


Рисунок 1.1 – Общая схема IDEF0

Таким образом у процесса есть входные и выходные параметры, а также задача, которую он выполняет. Поэтому, процессы можно объединять в цепочки путем передачи выходных параметров во входные следующих процессов. По такому же принципу работают визуальные конструкторы и языки программирования. Наиболее ярким примером визуального программирования можно отметить технологию Blueprint, которая используется в игровом движке Unreal Engine [17, 18].

Blueprint – это система визуального скриптинга, которая является быстрым способом создания прототипов игр. Вместо построчного написания кода всё можно делать визуально: перетаскивать узлы (ноды), задавать их свойства в интерфейсе и соединять их. Пример использования технологии Blueprint приводится на рисунке 1.2.

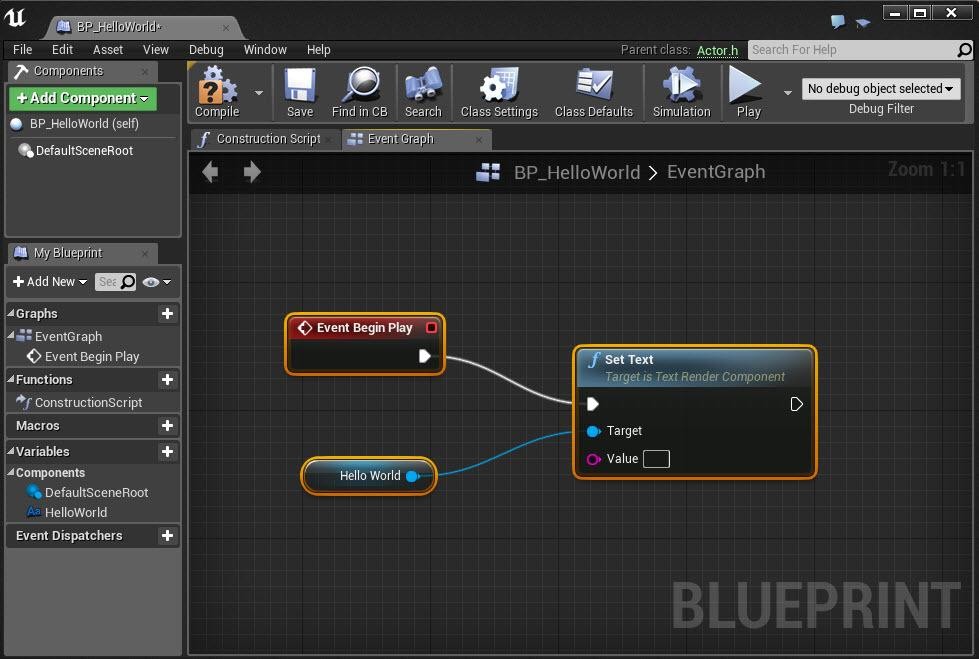


Рисунок 1.2 – Визуальное программирование на Blueprint

В последнее время визуальное программирование стало набирать большую популярность и отлично зарекомендовало себя в области разработки игр и обработке компьютерной графики. Написание программного кода путем соединения узлов является более простым и наглядным для понимания человека. А схема соединенных между собой блоков уже представляет собой бизнес-процесс.

Одним из примеров бизнес-процессов в образовательной деятельности можно выделить тестирование. Процесс тестирования представляет собой набор задач, заранее подготовленных на определенную тему, после выполнения которых можно получить результат усвоения тех или иных знаний. Если построить блок-схему данного процесса, то она будет выглядеть так, как показано на рисунке 1.3. Как можно заметить недостатком теста является то, что все задачи выполняются, последовательно. Хотя и в некоторых программных продуктах (далее «ПП») предусмотрена возможность выполнения задач случайным образом, можно сказать, что они все равно выполняются последовательно, так отсутствует логический смысл их расстановки.

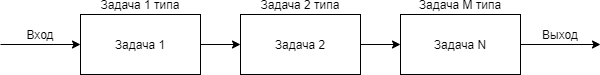


Рисунок 1.3 – Блок-схема теста

Еще одним серьезным недостатком является то, что набор разновидностей задач сильно ограничен. Базовый набор задач содержит следующие варианты:

* задачи закрытого типа, которые представляют собой текстовое поле и ожидают ввода точного ответа на вопрос;
* выбор одного решения. Предполагается выбор одного единственно-верного ответа из предложенного списка вариантов;
* выбор нескольких решений. Предполагается выбор нескольких правильных ответов из предложенного списка вариантов;
* установка правильной последовательности. Предполагает расстановку вариантов ответов в списке в нужном порядке.

Таким образом, из-за ограниченного списка типов стандартных задач, грамотно и точно описать более сложные производные задачи не всегда является возможным.

## Обзор программ-аналогов для моделирования бизнес-процессов

В настоящее время разработано множество инструментов для моделирования бизнес-процессов. Их можно разделить на следующие категории:

* конструкторы блок-схем;
* имитаторы бизнес-процессов;
* системы, которые используются в учебном процессе.

К категории конструкторов блок-схем можно отнести хорошо зарекомендовавший себя за долгое время программный продукт Microsoft Visio [19].

Visio – это программное обеспечение для рисования различных диаграмм: блок-схем, организационных диаграмм, планов зданий, поэтажных планов, диаграмм потоков данных, технологических схем и моделирования бизнес-процессов. На рисунке 1.4 можно увидеть внешний вид интерфейса Visio.

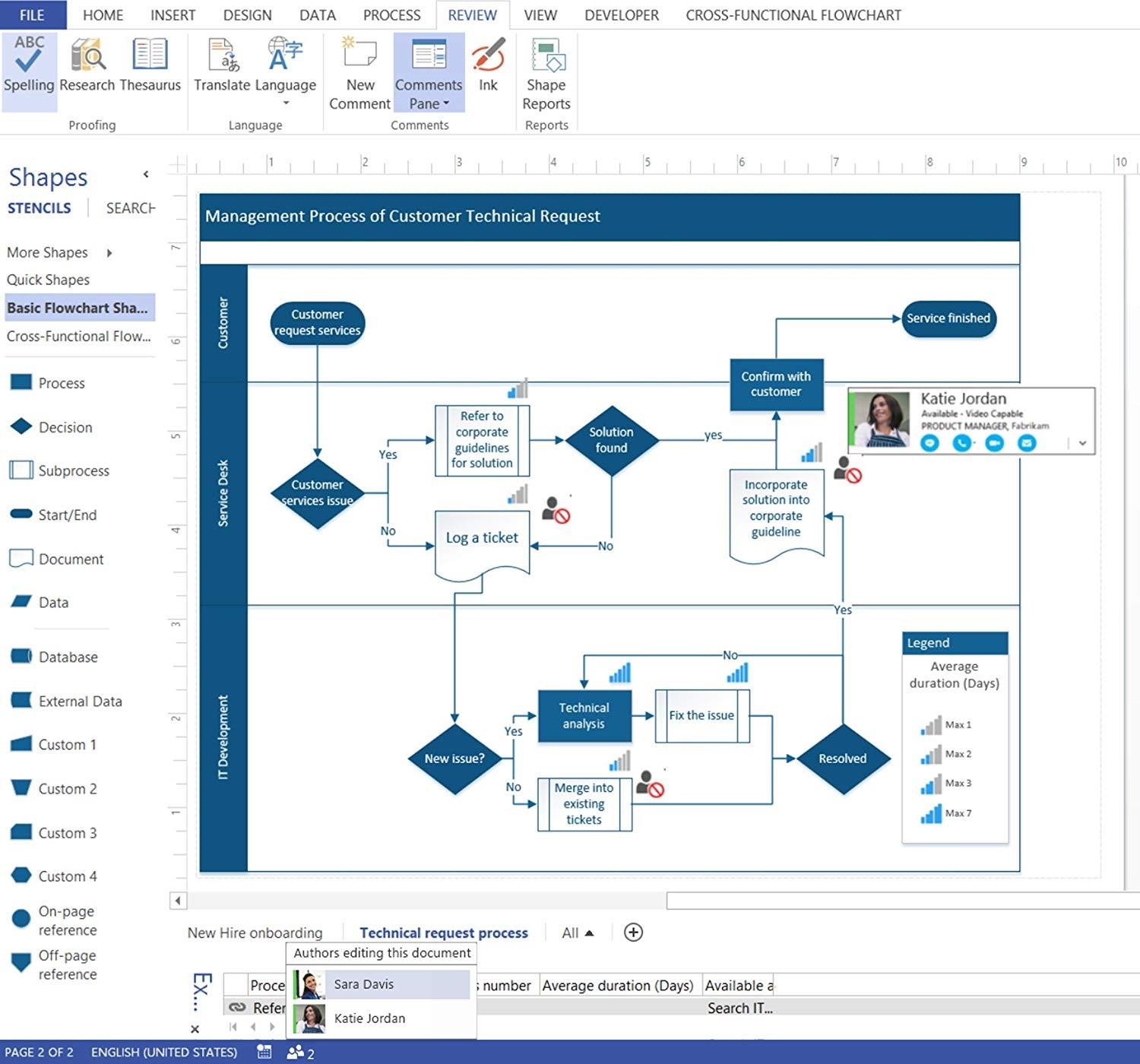


Рисунок 1.4 – Внешний вид интерфейса Visio

Visio предоставляет инструменты для удобного конструирования блок-схем, позволяет создавать вложенные друг в друга схемы. Но недостатком такого подхода является то, что схемы остаются схемами и носят в себе только информативный характер. Разрабатываемый программный проект решает данную проблему, путем предоставления функционала для назначения поведения блоков (задач) схемы.

К категории имитаторов бизнес-процессов можно отнести такие инструменты, как Blueprint, интерфейс которого показан на рисунке 1.2, а также инструмент CX- Supervisor, который показан на рисунке 1.5 и служит для имитации поведения физического оборудования.

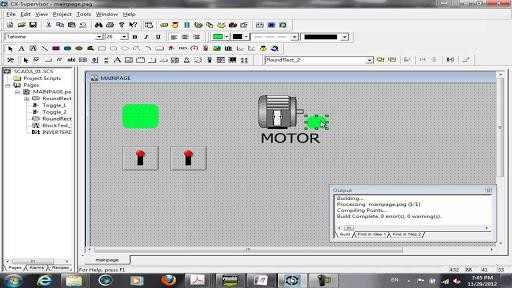


Рисунок 1.5 – Внешний вид интерфейса CX-Supervisor

Данный инструмент служит виртуальной оболочкой, которая имитирует поведение реального оборудования, такого как: мотор, включатель, индикатор и тд. Касаемо рассматриваемой темы дипломного проекта, к недостаткам данного инструмента можно отнести отсутствие возможности добавления собственных блоков для конструирования. Данная проблема в разрабатываемом дипломном проектировании решается путем разработки web-приложения, что автоматически делает его кроссплатформенным, а также позволяет внедрять в себя другие web- приложения.

В качестве примера систем, которые используются в учебном процессе, можно привести систему тестирования Indigo, внешний вид интерфейса которой показан на рисунке 1.6. Indigo представляет собой мультифункциональный комплекс программного обеспечения, позволяющий автоматизировать процесс проведения тестирования и обработки результатов [2]. Использование данного инструмента происходит следующим образом: создаются тесты с набором заданий, затем эти тесты выполняются, а результаты тестирования отображаются в сводных таблицах. По такому принципу работает абсолютное большинство систем тестирования. Зачастую такие системы ограничены как функциональными, так и программными возможностями. Ограниченный набор типов задач затрудняет разработку заданий в сферах, где необходима графическая визуализация, например начертательная геометрия и 3D графика, а в виду того, что зачастую такие инструменты разрабатываются под настольные системы, у них отсутствует кроссплатформенность. Это негативно сказывается на удобстве использования и зачастую из-за сложности в применении такие программные инструменты не набирают должной популярности среди пользователей.

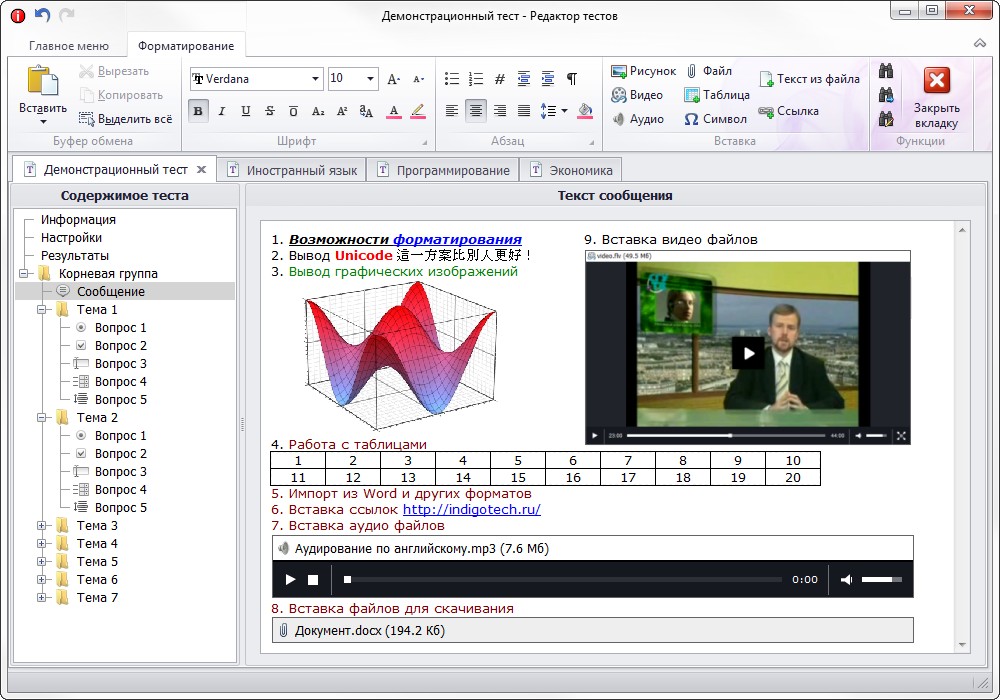


Рисунок 1.6 – Внешний вид интерфейса Indigo

Таким образом разрабатываемый программный инструмент совмещает в себе все перечисленные категории систем моделирования. Инструмент одновременно является и симулятором, и системой построения блок-схем, и системой тестирования, при рассмотрении ее в частном случае.

## Цели и постановка задачи

Целью дипломного проекта является разработка web-приложения «Интерактивный тренажер обучения построению проекций точек и отрезков на эпюре Монжа». Данное web-приложение должно быть реализовано на серверной и клиентской части. Клиентская часть должна включать в себя функционал для добавления новых типов задач (узлов), функционал для комбинирования этих узлов в процессы (задачи), а также их запуск.

Для реализации поставленных целей необходимо решить следующие задачи:

1. определиться с используемыми технологиями;
2. сформулировать сущности организации задач для обучения;
3. построить физическую и логическую модели данных;
4. разработать серверную часть;
5. разработать клиентскую часть;
6. провести тестирование приложения;
7. описать полученные результаты, сформулировать выводы и заключения.

Функционал web-приложения «Интерактивный тренажер обучения построению проекций точек и отрезков на эпюре Монжа» должен включать в себя следующее:

1. создание, редактирование, удаление платформ. Платформа служит для группирования узлов по схожим предметным областям для более удобного поиска и хранения узлов;
2. создание, редактирование, удаление узлов. Узел ­ сущность, которая служит связующим звеном между разрабатываемой системой и внешними системами;
3. создание, редактирование, удаление параметров в узлах. Параметры предназначены для определения входных и выходных параметров узла, содержит тип и направление передаваемого в параметр значения;
4. создание, редактирование, удаление задач. Задача­ ­­­­­служит для организации хранения сконструированных из узлов, соединенных между собой;
5. выполнение узлов и задач;
6. просмотр списков платформ, узлов, параметров и задач.

# ВЫБОР ПРОГРАММНО-ТЕХНИЧЕСКИХ СРЕДСТВ РЕАЛИЗАЦИИ WEB-ПРИЛОЖЕНИЯ «ИНТЕРАКТИВНЫЙ ТРЕНАЖЕР ОБУЧЕНИЯ ПОСТРОЕНИЮ ПРОЕКЦИЙ ТОЧЕК И ОТРЕЗКОВ НА ЭПЮРЕ МОНЖА»

## Выбор стека технологий для серверной части

Серверная часть web-приложения состоит из нескольких компонентов, обычно называемых уровнями или слоями приложения. Наиболее распространенные слои веб- приложения:

1. уровень представления – относится к пользовательскому интерфейсу приложения, включает в себя код HTML, CSS и JavaScript, который запускается в браузере для отображения веб-страниц, составляющих интерфейс приложения, и управления их динамическим поведением при взаимодействии пользователя со страницей;
2. прикладной уровень – обрабатывает логику, которая обеспечивает функционирование веб-приложения, выполняя любые операции и алгоритмы, обеспечивающие работу бизнес-функций приложения;
3. уровень данных – хранит, организует и управляет доступом к данным приложения с использованием базы данных и реализуется в коде на стороне сервера.

В качестве языка программирования серверной части был выбран C#. C# – это объектно-ориентированный язык программирования, разработанный компанией Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML [4].

Ключевым фактором выбора данного языка является то, что он отлично интегрируется с платформой .NET и всей экосистемой продуктов Microsoft. В качестве фреймворка для разработки web-приложения был взят ASP .NET Core. Платформа ASP.NET Core представляет технологию от компании Microsoft, предназначенную для создания различного рода web-приложений: от небольших web-сайтов до крупных web-порталов и web-сервисов. ASP.NET Core является продолжением развития платформы ASP.NET и полностью является opensource-фреймворком. Все исходные файлы фреймворка доступны на GitHub. ASP.NET Core может работать поверх кроссплатформенной среды .NET Core, которая может быть развернута на основных популярных операционных системах: Windows, Mac OS, Linux. И таким образом, при помощи ASP.NET Core можно создавать кроссплатформенные приложения [5].

В качестве инструментария разработки используются последние выпуски Visual Studio, начиная с версии Visual Studio 2019. Также для обеспечения работы серверной части, было принято решения об использовании NuGet библиотеки Entity Framework Core, которая служит посредником между API (от англ. Application Programming interface) и базой данных.

В результате было решено использовать для серверной части язык программирования C# на базе фреймворка .NET Core Web API с использованием библиотеки Entity Framework Core. Выбор этой технологий позволяет перейти к выбору СУБД.

## Выбор системы управления базой данных

Главным фактором при принятии решения является то, как выглядят наши данные. Если данные в основном структурированы, то следует выбирать реляционную базу данных. Каждая строка в реляционной базе является отдельной сущностью, а каждый столбец является атрибутом, который описывает эту сущность. Из-за этих четких, структурированных отношений между строками и столбцами в таблице базы данных SQL лучше всего подходят, когда требуется соответствие ACID. ACID означает:

* атомарность – это условие, при котором либо транзакция успешно выполняется целиком, если какая-либо из ее частей не выполняется, вся транзакция отменяется;
* согласованность – это условие, при котором данные, записанные в базу данных, должны быть действительными в соответствии со всеми определенными правилами;
* изолированность – когда транзакции выполняются одновременно, они не конкурируют друг с другом и действуют так, как если бы они выполнялись последовательно;
* стойкость – после того, как транзакция была зафиксирована в базе данных, она считается постоянной, даже в случае сбоя системы.

Нереляционная СУБД позволяет следующее:

* создавать документы без тщательного определения их структуры заранее;
* добавить поля в базу данных без изменения полей существующих документов;
* хранить документы, которые имеют свою уникальную структуру;
* иметь несколько баз данных с различной структурой и синтаксисом [6].

Таким образом из приведенных выше факторов был сделан выбор в пользу реляционных СУБД, так как сущности можно четко представить в виде таблиц, и необходимо соответствие ACID для обеспечения надежности системы. Исходя из того, что для серверной части была выбрана технология Microsoft – .NET Core, целесообразно было выбрать реляционную СУБД SQL Server.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире. Данная СУБД подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

Центральным аспектом в MS SQL Server, как и в любой СУБД, является база данных. База данных представляет хранилище данных, организованных определенным способом. Нередко физически база данных представляет файл на жестком диске, хотя такое соответствие необязательно. Для хранения и администрирования баз данных применяются системы управления базами данных (database management system) или СУБД (DBMS). И как раз MS SQL Server является одной из такой СУБД [7].

## Выбор стека технологий для клиентской части

Так как клиентская часть ПП является ключевой частью проекта и подразумевает огромное количество динамической отрисовки компонентов, выбор был сделан в сторону фреймворка React. React – это платформа для создания одностраничных клиентских приложений с использованием HTML и TypeScript. React написан на TypeScript. Он реализует основные и дополнительные функции в виде набора библиотек TypeScript, которые могут быть импортированы в приложение. Архитектура приложения React опирается на определенные фундаментальные концепции. Основными строительными блоками являются TSX файлы, которые предоставляют контекст компиляции для компонентов. TSX собирают связанный код в функциональные наборы. В приложении всегда есть по крайней мере корневой модуль, который включает загрузку, и обычно он имеет гораздо больше функциональных модулей. Компоненты определяют представления, представляющие собой наборы элементов экрана, которые React может выбирать и изменять в соответствии с логикой вашей программы [8].

Для обеспечения надежности, а также простоты разработки, в качестве языка программирования был выбран TypeScript в замену JavaScript. Решающим фактором стало наличие строгой типизации. TypeScript – язык программирования, представленный Microsoft в 2012 году и позиционируемый как средство разработки web-приложений, расширяющее возможности JavaScript. TypeScript является обратно совместимым с JavaScript и компилируется в последний [9].

В качестве языка разметки интерфейса, был выбран HTML. Это наиболее известный и используемый вариант во всем мире. HTML – стандартизированный язык разметки документов во всемирной паутине. Большинство web-страниц содержат описание разметки на языке HTML. Во всемирной паутине HTML-страницы, как правило, передаются браузерам от сервера по протоколам HTTP или HTTPS, в виде простого текста или с использованием шифрования. Текстовые документы, содержащие разметку на языке HTML (такие документы традиционно имеют расширение .html или .htm), обрабатываются специальными приложениями, которые отображают документ в его форматированном виде. Такие приложения, называемые «браузерами» или «интернет-обозревателями», обычно предоставляют пользователю удобный интерфейс для запроса web-страниц, их просмотра (и вывода на иные внешние устройства) и, при необходимости, отправки введённых пользователем данных на сервер. Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения) [10].

Для написание более дружелюбного интерфейса использовался язык стилей CSS. Он используется создателями web-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих web-страниц. Основной целью разработки CSS являлось отделение описания логической структуры web-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой web-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом [11].

Для реализации дружелюбного и удобного интерфейса было решено использовать шаблон графического интерфейса Argon Design System. Argon Design System состоит из более чем 100 отдельных компонентов, что дает свободу выбора и комбинирования. Все компоненты могут иметь разные цвета, которые можно легко изменить с помощью файлов SASS. Это экономит много времени при переходе от прототипа к полнофункциональному коду, потому что все элементы реализованы. Эта система дизайна поставляется с готовыми примерами, поэтому процесс разработки проходит без проблем. Каждый элемент имеет несколько состояний для цветов, стилей, наведения, фокуса, к которым можно легко получить доступ и использовать [12].

# ПРОЕКТИРОВАНИЕ WEB-ПРИЛОЖЕНИЯ «ИТЕРАКТИВНЫЙ ТРЕННАЖЕР ОБУЧЕНИЯ ПОСТРОЕНИЮ ПРОЕКЦИЙ ТОЧЕК И ОТРЕЗКОВ НА ЭПЮРЕ МОНЖА»

## Общая структура взаимодействия компонентов в web-приложении

Web-приложение – это клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – web-сервер. Основная часть приложения, как правило, находится на стороне web-сервера, который обрабатывает полученные запросы в соответствии с бизнес-логикой продукта и формирует ответ, отправляемый пользователю. На этом этапе в работу включается браузер, именно он преобразовывает полученный ответ от сервера в графический интерфейс, понятный рядовому пользователю [13]. Схема взаимодействия компонентов показана на рисунке 3.1.



Рисунок 3.1 – Схема взаимодействия компонентов приложения

Первой и одной из ключевых особенностей web-приложений является их архитектура. Web-приложение представлено следующими составляющими:

1. клиент;
2. web-сервер;

3) база данных.

Клиент – это браузер для и того, чтобы пользователь увидел графический интерфейс приложения, браузер должен обработать полученный ответ web-сервера, в котором будет содержаться информация, реализованная с применением HTML, CSS, TypeScript. Именно эти технологии дают понять браузеру, как именно необходимо отрисовать все, что он получил в ответе. Именно на клиенте происходит отображение, добавление, изменение и удаление данных. Клиент предоставляет возможности по управлению платформами и узлами, конструктор для конструирования задач, а также интерфейс для их выполнения. Для изменения данных клиент посылает HTTP запросы на API web-сервера.

Web-сервер – это сервер, принимающий HTTP-запросы от клиентов и выдающий им HTTP-ответы. Web-сервером называют как программное обеспечение, выполняющее функции web-сервера, так и непосредственно компьютер, на котором это программное обеспечение работает. API web-сервера имеет собственный интерфейс, через который сервер общается с клиентом, а также интерфейс для взаимодействия с базой данных. Так же web-сервер содержит в себе определение бизнес-логики и поведения сущностей, которые используются в проекте. Для обеспечения целостности данных на сервере используются методы первоначальной валидации данных, которые присылает клиент. Так же обеспечивается логирование ошибок для их более простого поиска и решения.

Серверная часть посредством обращения к API-методам будет предоставлять возможность для создания, чтения, обновления и удаления хранимых в базе данных сущностей, таких как: платформы, узлы, задачи, и т.д., а также управлять выполнением и проверкой задач.

База данных представляет собой информационную модель, которая позволяет упорядоченно хранить данные об объекте или группе объектов, обладающих набором свойств, которые можно категорировать. Базы данных функционируют под управлением систем управления базами данных. В разрабатываемом программном приложении используется СУБД Microsoft SQL Server. БД содержит данные о сущностях платформ, узлов, задач и т.д.

В БД будут храниться все данные о созданных платформах, узлах, задачах, а также данные параметров узлов и задач, такие как тип данных, их значения. База данных также будет содержать информацию о взаимосвязях вышеперечисленных сущностей между собой.

## Диаграмма вариантов использований

Диаграммы вариантов использования являются необходимым средством при анализе требований, планировании и управлении итеративной разработкой. Работа с вариантами использования является одной из самых важных на стадии уточнения. Каждый вариант использования – это потенциальное требование к системе, и пока оно не выявлено, невозможно запланировать его реализацию.

Суть диаграммы вариантов использования состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой [14].

При анализе задач и требований, поставленных при разработке программного приложения, была спроектирована диаграмма вариантов использования, которая приводится ниже на рисунке 3.2.

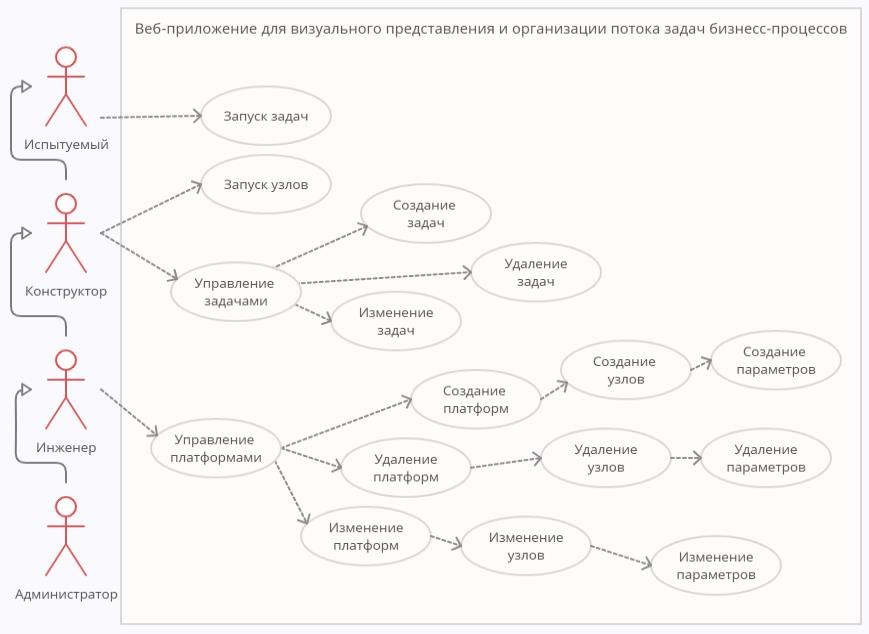


Рисунок 3.2 – Диаграмма вариантов использования

На рисунке 3.2 отображены следующие действующие лица:

* испытуемый (предоставляются права на решение задач);
* конструктор (предоставляются права на создание, изменение, удаление задач);
* инженер (предоставляются права на создание, изменение и удаление платформ, а также узлов и параметров);
* администратор (доступны все возможности системы).

Описание вариантов использования:

1. запуск задач:

* система предоставляет возможность запускать (выполнять) задачи;
* существует возможность просматривать список доступных задач;
* существует возможность вести поиск по записям.

1. запуск узлов:

* система предоставляет возможность запускать (выполнять) узлы;
* существует возможность просматривать список доступных узлов;
* существует возможность вести поиск по записям.

1. управление задачами:

* система предоставляет возможность создания, редактирования и удаления задач;
* при создании или редактировании открывается отдельная страница с формой для ввода данных;
* после удаления открывается страница со списком задач;
* существует возможность просматривать список доступных задач;
* существует возможность вести поиск по записям.

1. управление платформами:

* система предоставляет возможность создания, редактирования и удаления платформ;
* при создании или редактировании открывается отдельная страница с формой для ввода данных;
* после удаления открывается страница со списком платформ;
* существует возможность просматривать список доступных платформ; существует возможность вести поиск по записям.

1. создание, изменение, удаление узлов:

* система предоставляет возможность создания, редактирования и удаления узлов;
* при создании или редактировании открывается отдельная страница с формой для ввода данных;
* после удаления открывается страница редактирования платформы;
* существует возможность просматривать список доступных узлов;
* существует возможность вести поиск по записям.

1. создание, изменение, удаление параметров:

* система предоставляет возможность создания, редактирования и удаления параметров;
* при создании или редактировании открывается отдельная страница с формой для ввода данных;
* после удаления открывается страница редактирования узла;
* существует возможность просматривать список доступных параметров;
* существует возможность вести поиск по записям.

## Концептуальная модель базы данных

Концептуальное проектирование – это процедура конструирования информационной модели, не зависящей от каких-либо физических условий реализации. Основной задачей концептуального проектирования является определение предметной области (ПО) системы с учетом требований пользователей.

Концептуальная модель ПО представляет собой описание структуры и динамики, характера информационных потребностей пользователей в терминах, понятных пользователю и не зависимых от реализации базы данных (БД). Это описание выражается в терминах не отдельных объектов ПО и связей между ними, а их типов, связанных с ними ограничений целостности и тех процессов, которые приводят к переходу предметной области из одного состояния в другое.

Для описания структуры данных предметной области были выделены следующие сущности:

* платформа (platform) – служит для группирования узлов по схожим предменым областям для более удобного поиска и хранения узлов;
* узел (node) – сущность, которая служит связующим звеном между разрабатываемой системой и внешними системами;
* параметр узла (pin) – предназначен для определения входных и выходных параметров узла, содержит тип и направление передаваемого в параметр значения;
* задача (flow) – служит для организации хранения сконструированных задач из узлов, соединенных между собой;
* параметр задачи (alias) – используется для определения входных и выходных параметров задачи, содержит тип и направление передаваемого в параметр значения;
* связь (connector) – определяет связи между узлами для организации непрерывной цепочки выполнения узлов.

При проектировании базы данных также необходимо учитывать производительность системы. Это может включать создание индексов для ускорения выполнения запросов, оптимизацию структуры таблиц, кэширование данных и другие меры для обеспечения эффективной работы с данными.

Тестирование является неотъемлемой частью проектирования базы данных. Оно позволяет проверить правильность и работоспособность базы данных, а также ее соответствие требованиям. В процессе тестирования можно выявить и исправить ошибки, а также провести оптимизацию для улучшения производительности.

Остальные сущности, показанные в таблице 3.1, используются для организации связей между описанными сущностями.

Таблица 3.1 – Сущности предметной области

|  |  |
| --- | --- |
| Название сущности | Описание |
| Platform | Содержит данные о платформах |
| Node | Содержит данные об узлах |
| Pin | Содержит данные о параметрах |
| Flow | Содержит данные о задачах |
| Flow\_Node | Содержит данные узлов для конкретных заданий |
| PinValue | Содержит данные о значениях параметров |
| Alias | Содержит данные о параметрах задачи |
| PinValue\_Alias | Содержит данные о параметрах для конкретных заданий |
| Connector | Содержит связи PinValue с PinValue |

В таблице 3.2 приведены атрибуты сущностей предметной области.

Таблица 3.2 – Атрибуты сущностей

|  |  |  |
| --- | --- | --- |
| Сущность | Атрибут | Назначение |
| 1 | 2 | 3 |
| Platform | Id | Id платформы |
| Name | Название платформы |
| Description | Описание платформы |
| Author | Автор платформы |
| IsActive | Доступность платформы |
| Node | Id | Id узла |
| PlatformId | Id платформы |
| Name | Название узла |
| Command | Команда узла |
| CommandType | Тип команды |
| IsActive | Доступность узла |
| Pin | Id | Id параметра |
| NodeId | Id узла |
| Name | Название параметра |
| Direction | Направление параметра |
| ValueType | Тип значения параметра |
| IsPublic | Доступность параметра |
| Flow | Id | Id задачи |
| Name | Название задачи |
| IsActive | Доступность задачи |
| X | Координата X в конструкторе при редактировании задачи |
| Y | Координата Y в конструкторе при редактировании задачи |

Продолжение таблицы 3.2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Flow\_Node | Id | Id принадлежности узла к задаче |
| FlowId | Id задачи |
| NodeId | Id узла |
| Name | Синоним названия узла |
| X | Положение узла по координате X |
| Y | Положение узла по координате Y |
| PinValue | Id | Id значения параметра |
| PinId | Id параметра |
| FlowNodeId | Id принадлежности узла к задаче |
| Value | Значение параметра |
| Alias | Id | Id параметра задачи |
| FlowId | Id задачи |
| Name | Название параметра задачи |
| Direction | Направление параметра задачи |
| ValueType | Тип значения параметра |
| PinValue\_Alias | Id | Id принадлежности значения параметра к параметру задачи |
| AliasId | Id параметра задачи |
| PinValueId | Id значения параметра |
| Connector | Id | Id связи |
| FlowId | Id задачи |
| StartPinValueId | Id начального параметра |
| EndPinValueId | Id конечного параметра |

Сущность «Задача» описывается следующими полями:

* идентификатор (id) – является уникальным значением, которое идентифицирует конкретную задачу в системе, поле является обязательным для заполнения и генерируется автоматически при добавлении записи;
* название (name) – определяет название задачи и является обязательным для заполнения;
* статус (isActive) – используется для отражения статуса активности задачи, и запрещает выполнения задачи если состояние установлено как неактивное;
* координаты положения (x, y) – служат для сохранения положения экрана при конструировании задачи.

Выработка концептуальной модели позволяет перейти к логическому моделированию базы данных.

## Логическая модель базы данных

Логическим моделирование БД формально можно считать процесс создания модели мира самой базы данных, без той системы, которая будет позволять ей работать, и прочих физических деталей. Основными компонентами такой модели являются сущности, их атрибуты и связи между ними. Точность и полнота играют в этом процессе ключевую роль. Одним из главных преимуществ этого этапа является то, что всегда можно взять черновой проект, отложить его в сторону и начать все заново или просто внести желаемые поправки. Гораздо легче менять те или иные детали на этапе проектирования, чем иметь дело с проблемами уже реализованной производственной базы данных, которая плохо спроектирована.

Логическая модель расширяет концептуальную путем определения для сущностей их атрибутов, описаний и ограничений, уточняет состав сущностей и взаимосвязи между ними. Oграничения сущностей приведены в таблице 3.3.

Таблица 3.3 – Атрибуты и ограничения сущностей предметной области

|  |  |  |  |
| --- | --- | --- | --- |
| Сущность | Имя атрибута | Тип данных | Ограничения |
| 1 | 2 | 3 | 4 |
| Platform | Id | INT | PRIMARY KEY |
| Name | NVARCHAR (50) | NOT NULL |
| Description | NVARCHAR (250) | NULL |
| Author | NVARCHAR (50) | NOT NULL |
| IsActive | BIT | NOT NULL |
| Node | Id | INT | PRIMARY KEY |
| PlatformId | INT | FOREIGN KEY |
| Name | NVARCHAR (50) | NOT NULL |
| Command | NVARCHAR (250) | NULL |
| CommandType | INT | NOT NULL |
| IsActive | BIT | NOT NULL |
| Pin | Id | INT | PRIMARY KEY |
| NodeId | INT | FOREIGN KEY |
| Name | NVARCHAR (50) | NOT NULL |
| Direction | INT | NOT NULL |
| ValueType | INT | NOT NULL |
| IsPublic | BIT | NOT NULL |
| Flow | Id | INT | PRIMARY KEY |
| Name | NVARCHAR (50) | NOT NULL |
| IsActive | BIT | NOT NULL |
| X | INT | NOT NULL |
| Y | INT | NOT NULL |
| Flow\_Node | Id | INT | PRIMARY KEY |
| FlowId | INT | FOREIGN KEY |
| NodeId | INT | FOREIGN KEY |
| Name | NVARCHAR (50) | NOT NULL |
| X | INT | NOT NULL |
| Y | INT | NOT NULL |
| PinValue | Id | INT | PRIMARY KEY |

Продолжение таблицы 3.3

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
|  | PinId | INT | FOREIGN KEY |
| FlowNodeId | INT | FOREIGN KEY |
| Value | NVARCHAR (250) | NOT NULL |
| Alias | Id | INT | PRIMARY KEY |
| FlowId | INT | FOREIGN KEY |
| Name | NVARCHAR (50) | NOT NULL |
| Direction | INT | NOT NULL |
| ValueType | INT | NOT NULL |
| PinValue\_Alias | Id | INT | PRIMARY KEY |
| AliasId | INT | FOREIGN KEY |
| PinValueId | INT | FOREIGN KEY |
| Connector | Id | INT | PRIMARY KEY |
| FlowId | INT | FOREIGN KEY |
| StartPinValueId | INT | FOREIGN KEY |
| EndPinValueId | INT | FOREIGN KEY |

Для создания таблиц использовались SQL скрипты, один из которых показан на рисунке 3.3. Скрипт создания таблицы состоит из определения полей таблицы, их типов данных и значений по умолчанию. Так же скрипт содержит ограничения на поле Id, которое определяет его первичным ключом. Помимо основных полей, описанных в логической модели БД также используются технические поля, такие как: AddDate, AddSource, ChangeDate, ChangeSource. Данные поля используются для фиксирования времени и источника добавления и обновления данных. Это позволяет легко отслеживать, когда и кем были внесены изменения. На рисунке 3.3 под скриптом создания таблицы располагаются скрипты для определения связей с другими таблицами.

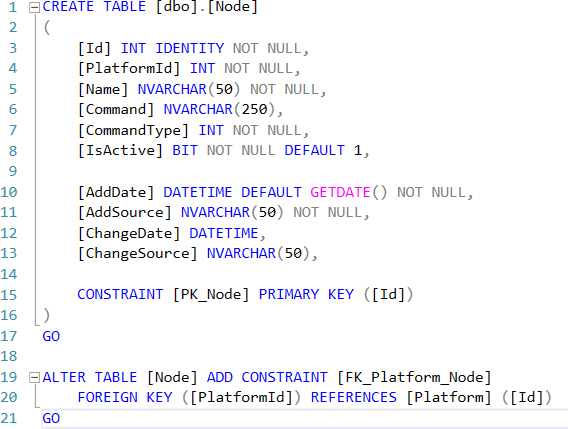


Рисунок 3.3 – SQL скрипт создания таблицы Node

Таким образом структура таблиц БД была вынесена в отдельный проект. Это позволяет обеспечить организацию версионности базы данных.

## Физическая модель базы данных

Физическое моделирование осуществляется с использованием таблиц, описанных в логическом моделировании и СУБД Microsoft SQL Server. Физическая организация данных оказывает влияние на эксплуатационные характеристики проектируемой базы, так как на этом уровне идет привязка к физической памяти. На этапе физического моделирования для сохранения целостности БД создаются первичные и вторичные ключи, применяются ограничения Not Null, а также сами таблицы реализуются с применением правил нормальных форм. Также устанавливается отношения между сущностями.

Разработанная схема БД представлена на рисунке 3.4. Стоит отметить, что в спроектированной схеме БД присутствует связь «многие ко многим», которая обеспечивает организацию соединения узлов задач в цепочки, Данная связь имеет частный вид (т.к. в стандартном варианте исполнения для организации используются три таблицы, а в текущем две) и организуется между таблицами PinValue и Connector.

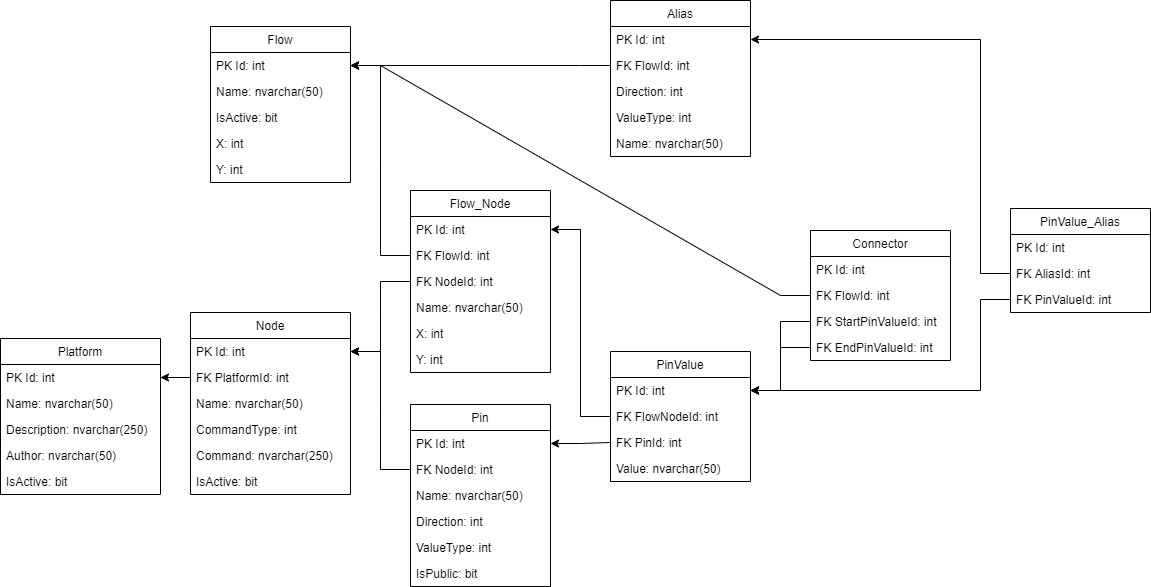


Рисунок 3.4 – Схема базы данных

В результате было разработано 9 таблиц, для поддержания функционала работы системы. Полученная схема базы данных имеет нестандартную структуру из-за наличия циклических связей. Это обусловлено тем, что при создании задач узлы параметров могут соединяться, а также из-за того, что, одна задача одновременно может включать в себя несколько одинаковых узлов. На рисунке 3.5 представлен скрипт, с помощью которого создавалась таблица «Platform».

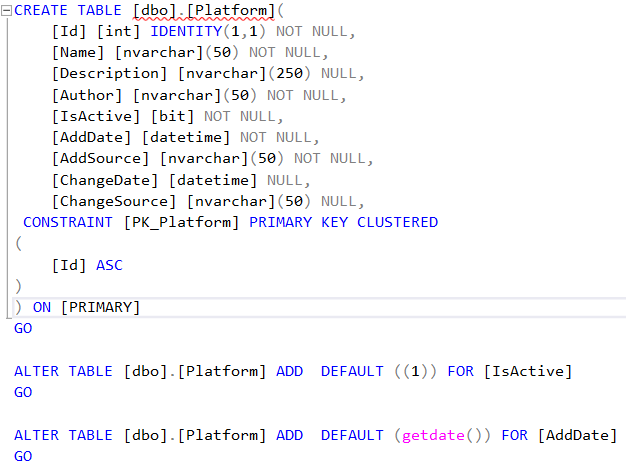


Рисунок 3.5 – Скрипт создания таблицы Platform

На рисунке 3.6 представлен скрипт, с помощью которого создавалась таблица «Node».



Рисунок 3.6 – Скрипт создания таблицы Node

Команда «Create Table» используется для создания таблицы. Далее идет описание атрибутов сущности, которые будут храниться в таблице. Ключевое слово «Constraint» используется для установки ограничений. В данных таблица установлены ограничения для атрибута «Id», который также является первичным ключом. Установлены стандартные значения: 1 для атрибута «IsActive» и текущая дата для атрибута «AddDate» Также для зfписей в таблице Node установлено ограничение для проверки наличия в таблице «Platform» записи с первичным ключом, соответствующим значению атрибута «PlatformId».

Остальные таблицы в базе данных были созданы аналогичным путем.

# РЕАЛИЗАЦИЯ WEB-ПРИЛОЖЕНИЯ «ИТЕРАКТИВНЫЙ ТРЕННАЖЕР ОБУЧЕНИЯ ПОСТРОЕНИЮ ПРОЕКЦИЙ ТОЧЕК И ОТРЕЗКОВ НА ЭПЮРЕ МОНЖА»

## Реализация серверной части

Разработка серверной части происходила с использованием интегрированной среды разработки Microsoft Visual Studio и фреймворков ASP .NET Core и Entity Framework Core. В проекте использовался подход REST. REST – это архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. На рисунке 4.1 показана схема взаимодействия серверной и клиентской части согласно под ходу REST.

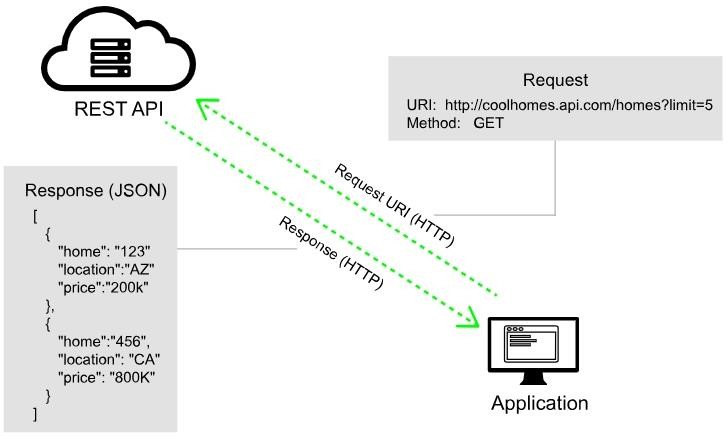


Рисунок 4.1 – Модель разработки REST API

Для удобной разработки, а также более гибкого расширения, серверная часть была разделена на несколько отдельных проектов. Каждый проект представляет из себя обособленный уровень, который выполняет ключевую роль в приложении.

В качестве первого проекта рассматривается проект «Common», который содержит базовые реализации, используемые в других проектах. Содержимое проекта не несет в себе никакой бизнес-логики и содержит только общие классы и методы. Структура проекта показана на рисунке 4.2. Благодаря тому, что что проект не связан с web-приложением, его можно многократно использовать в других разработках. Зачастую такой проект является шаблонным, в него вынесено все, что может пригодиться при разработке большинства приложений. Поэтому проект может содержать в себе много разновидностей классов, таких как базовые модели, сервисы, перечисления или какие-то полезные классы-помощники.

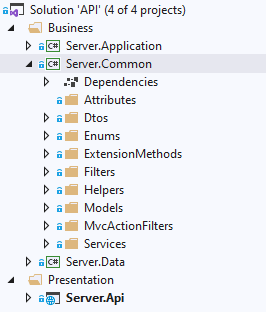


Рисунок 4.2 – Структура проекта Common

В качестве примера класса-помощника можно привести класс, переопределяющий ответ сервера, который показан на рисунке 4.5. Данный класс позволяет обработать большую часть ошибок, которые могут произойти и обернуть их в нужный формат ответа, который понятен клиентской части web-приложения. Это позволяет получать более информативные сообщения об ошибках.

Например, без использования данного класса, ответ сервера при возникновении ошибки отобразит полный путь возникновения ошибки, и будет выглядеть как показано на рисунке 4.3.

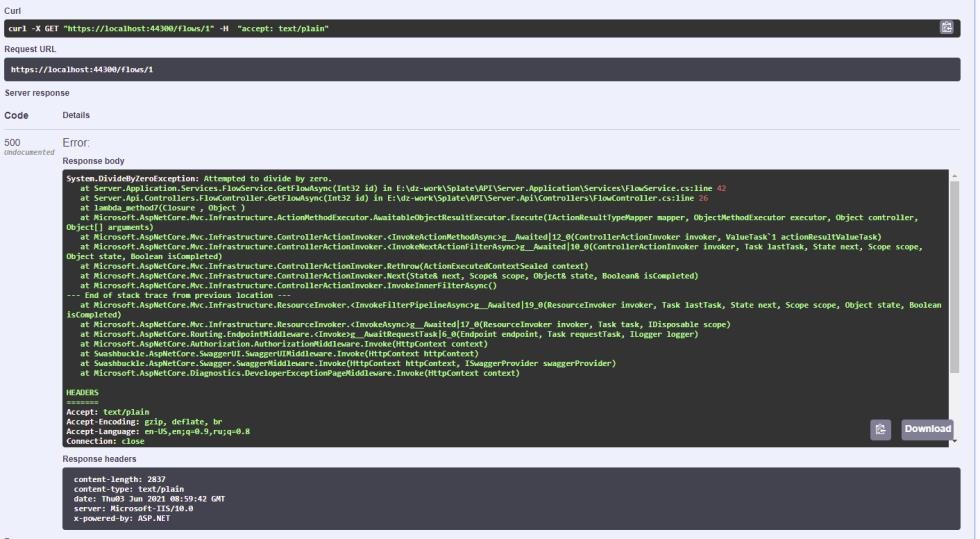


Рисунок 4.3 – Ответ сервера без обработки ошибок

Таким образом при обработке ошибок ответ будет иметь структурированную модель, которая будет содержать в себе список ошибок, как показано на рисунке 4.4.



Рисунок 4.4 – Ответ сервера с обработкой ошибок

В дальнейшем переопределенный ответ без проблем распознается клиентской частью приложения и отображается пользователю. Такой механизм позволяет быстро и качественно организовать обработку исключительных ситуаций на стороне сервера.

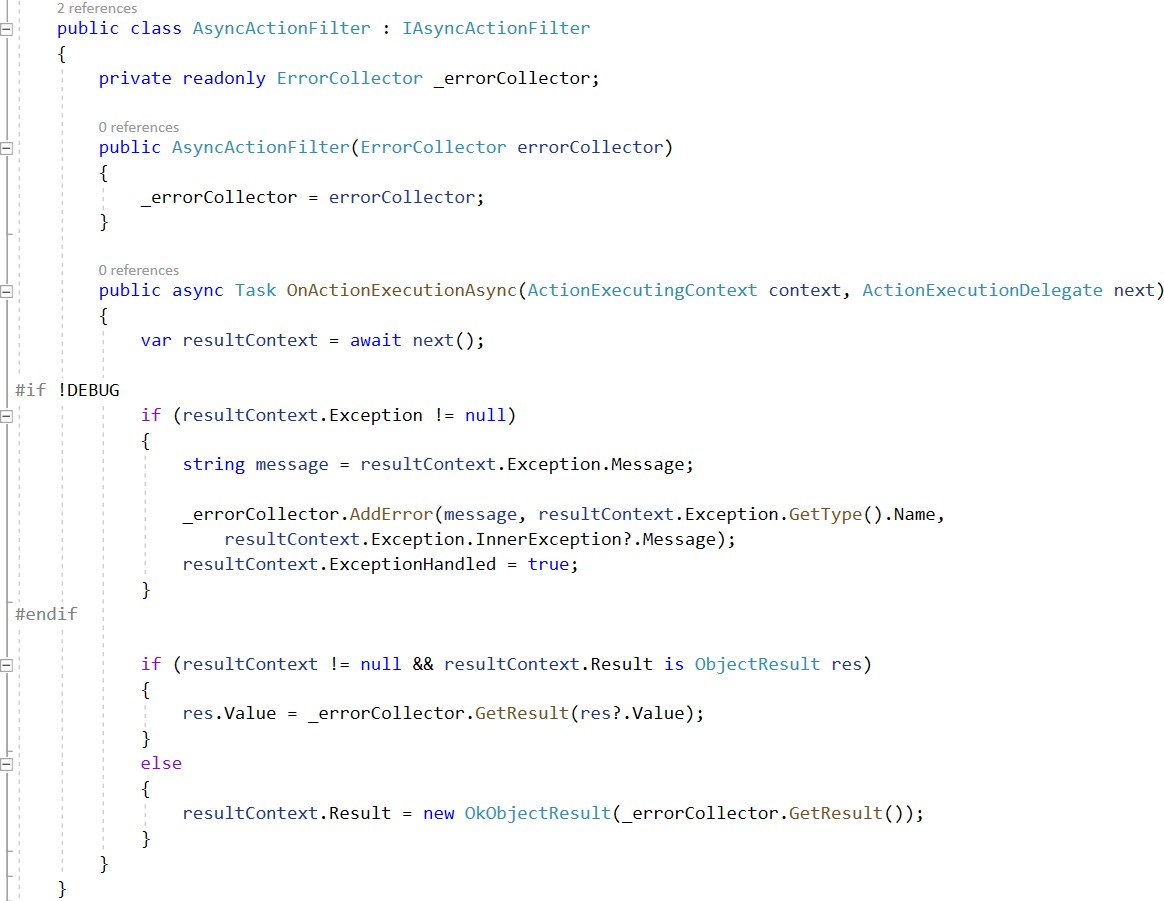


Рисунок 4.5 – Листинг кода переопределение ответа сервера

Следующим проектом является проект «Data». Данный проект служит для предоставления доступа к базе данных. В проекте реализуется паттерн Repository, идея которого заключается в том, что на каждую сущность БД создается специальный класс- репозиторий. В свою очередь репозиторий предоставляет методы для получения, создания, изменения и удаления данных. Структура проекта показана на рисунке 4.6.

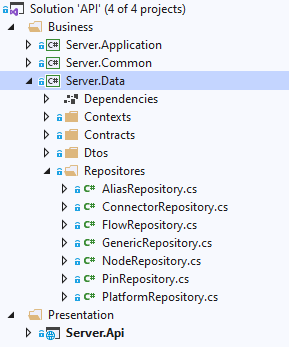


Рисунок 4.6 – Структура проекта Data

Для получения доступа к базе данных используется библиотека Entity Framework Core [20]. При использовании Entity Framework в приложении существует три подхода для организации взаимодействия Entity Framework с базой данных:

* Code-First – сперва создается класс модели данных, которые будут храниться в БД, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы;
* Model-First – в первую очередь создается модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере;
* Database-First – Entity Framework создает набор классов, которые отражают модель конкретной базы данных.

Для разработки web-приложения был выбран подход Database-First. Решающим фактором стала более гибкая возможность при проектировании базы данных, а также возможность вынести схему БД в отдельный проект. Таким образом это позволяет включить схему базы данных в систему контроля версий.

Для работы Entity Framework необходим класс-контекст, который будет содержать в себе все возможные определения сущностей базы данных. Листинг кода такого класса приводится на рисунке 4.7.

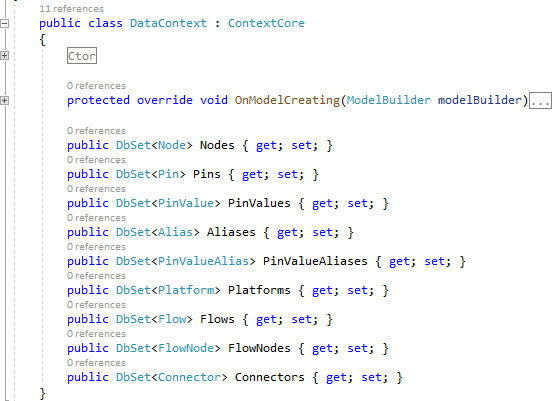


Рисунок 4.7 – Листинг класса-контекста

В контексте в виде свойств перечисляются названия используемых таблиц из БД. Названия таблиц сопоставляются путем сравнения их с названием классов-сущностей. Если требуется нестандартное сопоставление, или определение дополнительных ограничений на сущностях, это можно сделать путем переопределения метода OnModelCreating показанное на рисунке 4.8. Далее контекст используется в репозиториях. При создании объекта контекста открывается соединение с базой данных и закрывается после вызова деструктора.

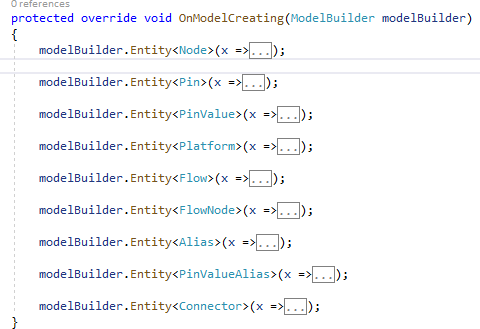


Рисунок 4.8 – Переопределение метода OnModelCreating

Третьей частью сервера приложения является проект «Application», который включает в себя всю бизнес-логику, относящуюся к приложению, а также входные и выходные модели. В проекте реализуется паттерн Service Layer, который предназначен для разделения логики на уровни. Многоуровневая архитектура позволят легко расширять проект. Структура данного проекта показана на рисунке 4.9.

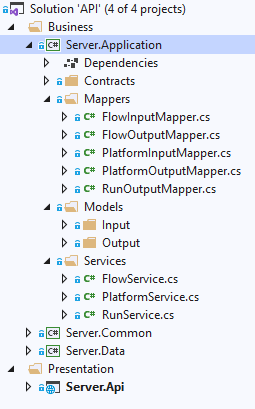


Рисунок 4.9 – Структура проекта Application

Так же проект включает в себя входные и выходные модели, которые используются для общения сервера с клиентской частью веб приложения. Для преобразования сущностей в модели используются классы-мапперы, которые добавляют методы расширения на моделях и сущностях, пример класса-маппера приводится на рисунке 4.10.

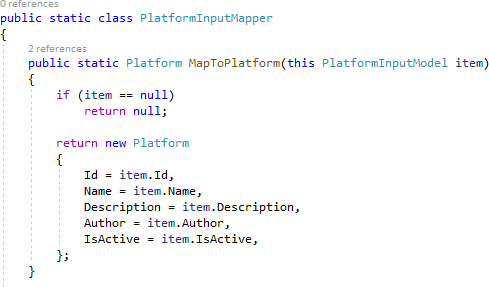


Рисунок 4.10 – Листинг класса-маппера

Заключительной частью проекта является Api (далее «АПИ»). Данный проект служит для предоставления публичного интерфейса, через который можно взаимодействовать с данными системы. Структура приводится на рисунке 4.11. В основе проекта лежит шаблон проектирования Model-View-Controller. Было принято разделить проект на три контроллера, кратно разделам web-приложения. Такое разделение позволяет уменьшить риск затронуть работоспособность одного раздела, разрабатывая другой. Также это позволяет уменьшить входные и выходные модели, что положительно сказывается на скорости работы web-приложения.

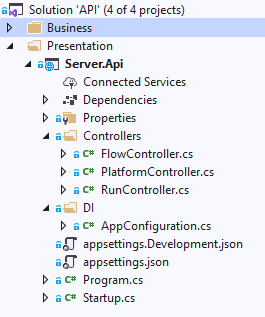


Рисунок 4.11 – Структура проекта API

API является исполняемым проектом и представляет собой web-сервер приложения. Проект API не использует репозитории напрямую. Это происходит через сервисы проекта Application. Для того чтобы эти сервисы работали, сперва их нужно зарегистрировать в проекте. Делается это путем добавления нужного сервиса в конфигурационный класс, показанный на рисунке 4.12. Помимо репозиториев и сервисов в этом классе регистрируются контроллеры, контексты и классы-помощники. У репозиториев и сервисов при регистрации указывается время жизни, которое обеспечивает своевременное предоставление обновленных данных. Таким образом время жизни указывается путем использования метода AddScoped, в котором задается интерфейс и реализация конкретного сервиса или репозитория. Затем все зарегистрированные сервисы можно использовать путем объявления параметра нужного типа в конструкторе контроллера, или сервиса. Подстановка сервисов происходит автоматически, путем использования встроенных возможностей в Entity Framework Core.

Таким образом, такая организация регистрации сервисов позволяет сократить время разработки, а также увеличить читаемость и безопасность программного кода.

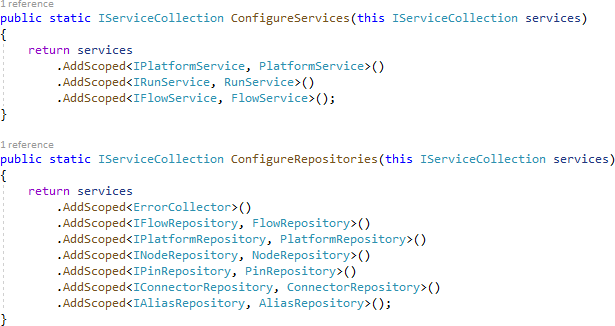


Рисунок 4.12 – Листинг конфигурационного класса API

Также в конфигурационном классе настраивается Swagger. Swagger – это фреймворк для спецификации RESTful API. Oн дает возможность не только интерактивно просматривать спецификацию API, но и отправлять запросы – так называемый Swagger UI (от англ. User Interface), изображение которого показано на рисунке 4.13.

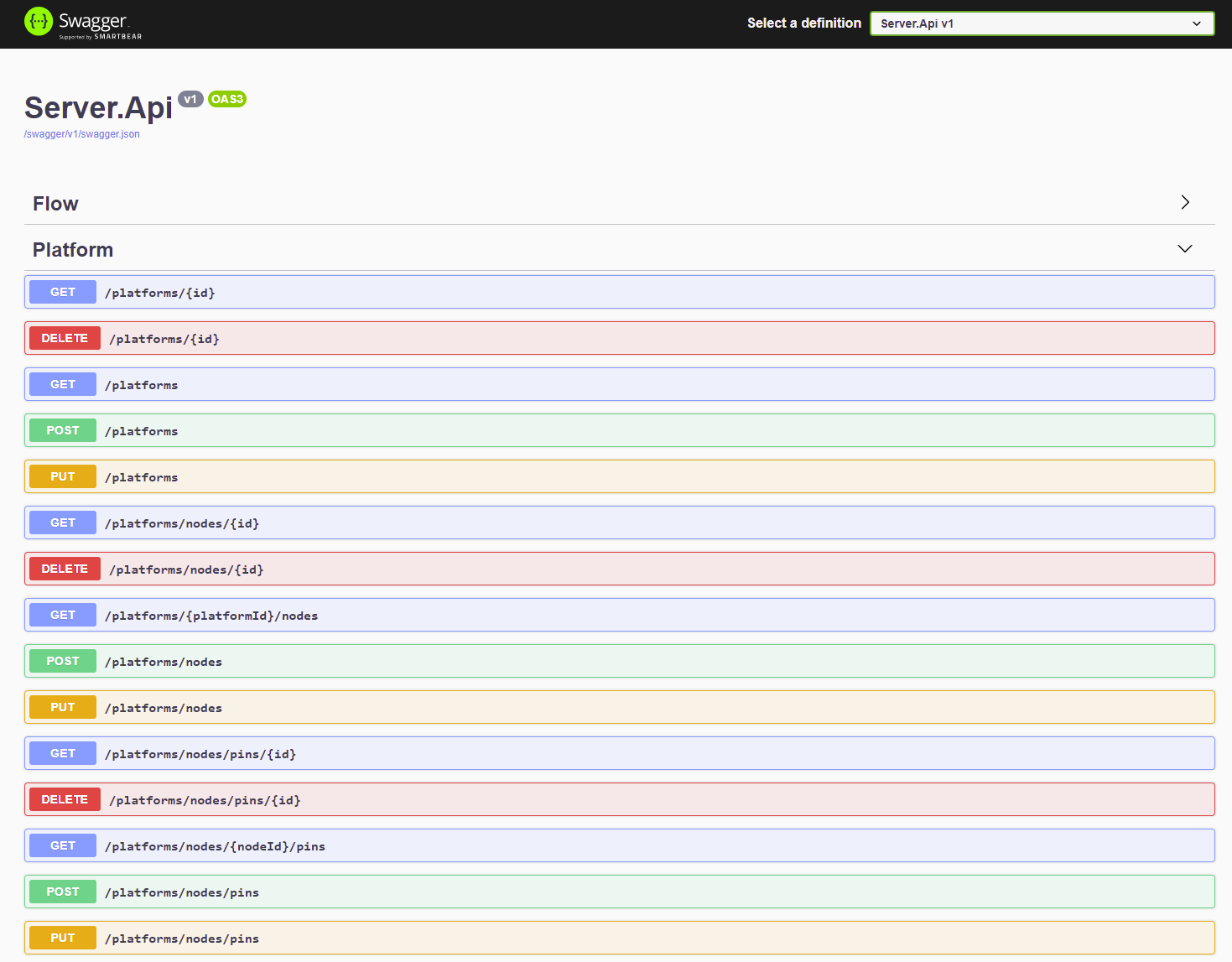


Рисунок 4.13 – Интерфейс Swagger UI

Данный интерфейс генерируется автоматически и включает в себя все возможные точки доступа API, которые декларируются в контроллерах. Это позволяет легко и быстро отлаживать API, путем запуска нужных запросов. Также это предоставляет возможность для использования API сторонними разработчиками или в случае, если серверную часть разрабатывает одна команда, а клиентскую – другая.

## Реализация клиентской части

Клиентская часть была реализована в текстовом редакторе Visual Studio Code с использованием фреймворка React. Входной точкой в приложение является компонент «Index.tsx», листинг кода которого показан на рисунке 4.14. React-компоненты – это небольшие, повторно используемые части кода, которые возвращают React-элементы для отображения на странице. Компоненты можно разбить на отдельные части в зависимости от выполняемой функции и использовать внутри других компонентов. Компоненты могут возвращать другие компоненты, массивы, строки и числа. Имена компонентов всегда должны начинаться с заглавной буквы.



Рисунок 4.14 – Листинг исполняемого компонента «Index.tsx»

Для упрощения разработки клиентской части, а также стандартизации интерфейса были разработаны компоненты, которые можно многократно использовать в разных разделах web-приложения.

Table – таблица, которая предназначена для отображения списка сущностей. Данный компонент показан на рисунке 4.15

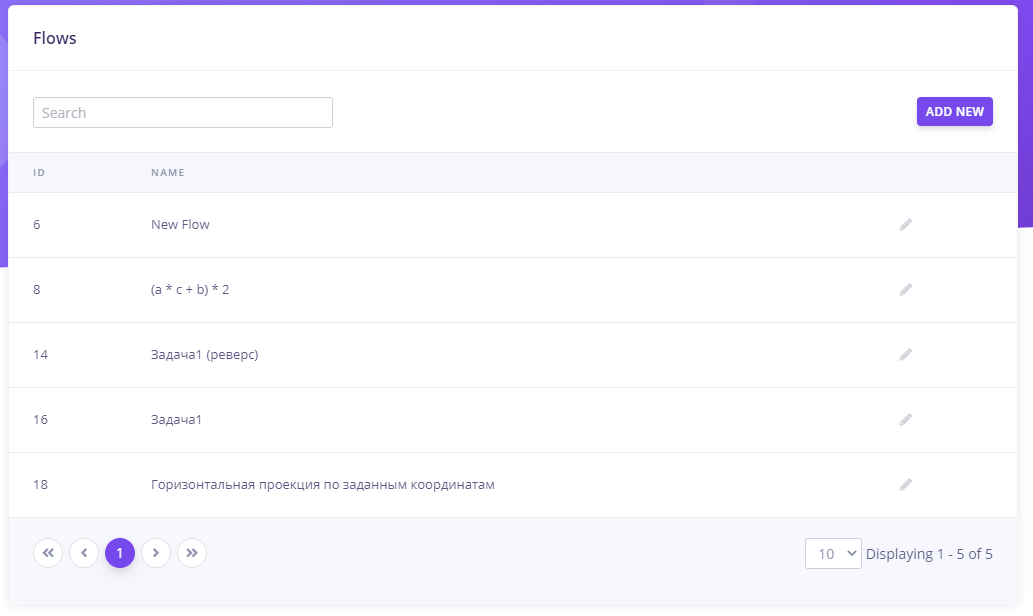


Рисунок 4.15 – Компонент Table

Компонент состоит из следующих частей:

* заголовок таблицы;
* строка поиска;
* кнопка перехода на страницу добавления новой записи;
* элементы таблицы;
* кнопка редактирования элемента;
* пагинация таблицы;
* количество отображаемых элементов на странице.

У компонента присутствует своя строка загрузки, поэтому загрузка данных происходит асинхронно.

Navbar – компонент, который показывает текущее положение открытой страницы, относительно домашней. Представление компонента можно увидеть на рисунке 4.16.

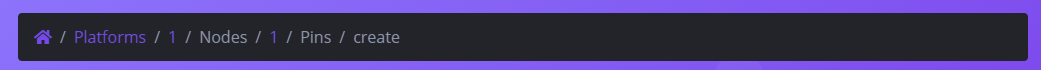


Рисунок 4.16 – Компонент Navbar

Текст в данном компоненте является кликабельным и при нажатии на него перенаправит на нужную страницу. При нажатии на кнопку «Дом» откроется главная страница.

FormModel – компонент который позволяет легко создать форму добавления и редактирования сущности. Отображение компонента можно увидеть на рисунке 4.17.

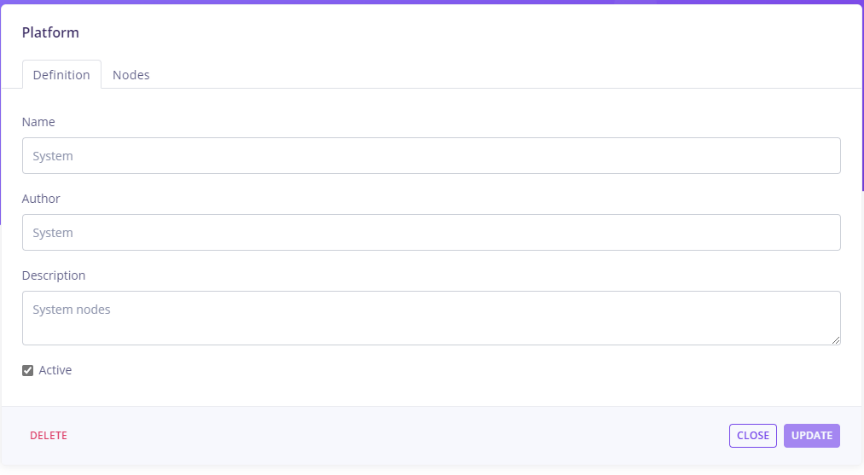


Рисунок 4.17 – Компонент «FormModel»

Компонент состоит из следующих частей:

* вкладки – на первой вкладке всегда находится форма, на остальных вкладках можно размещать собственные компоненты, такие как таблица;
* поля формы;
* кнопка удаления – доступна только при изменении сущности;
* кнопка сохранения;
* кнопка возврата на страницу со списком.

На главной странице клиентской части web-приложения отображаются карточки, сделанные в виде кнопок, изображение которых показано рисунке 4.18. При клике на карточку открывается страница выбранного раздела. Карточка состоит из названия раздела, и иконки, которая ассоциируется с этим разделом.

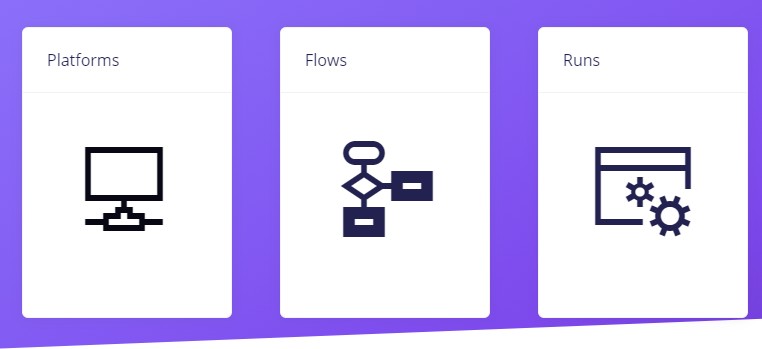


Рисунок 4.18 – Главная страница web-приложения

Таким образом клиентская часть web-приложения состоит из трех разделов: раздел управления платформами, конструктор задач, раздел выполнения узлов и задач.

### Раздел управления платформами

На главной странице раздела отображается компонент «Table» со списком платформ. При открытии страницы редактирования отображается компонент FormModel, в котором настроены поля сущности платформы. При переходе на вкладку

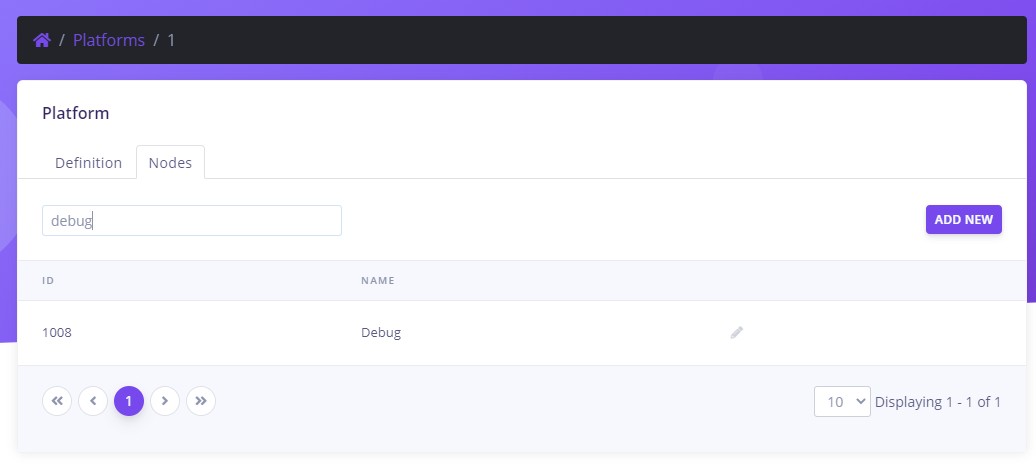
«Nodes» отображается встроенная в форму таблица, которая показана на рисунке 4.19.

Рисунок 4.19 – Вкладка «Nodes»

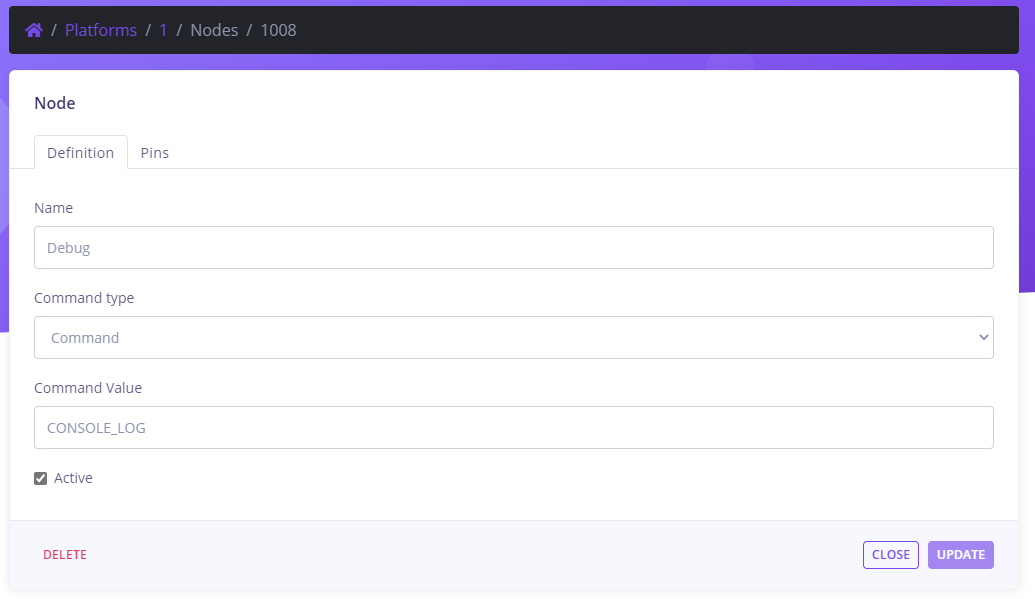
В таблице отображается список узлов. При нажатии на кнопку редактирования или создания нового узла открывается станица форма узла, которая показана на рисунке 4.20. Стоит отметить, что узел содержит внутри себя команду. Если выбран тип команды «Страница», то значением команды должен быть адрес этой страницы.

Рисунок 4.20 – Страница редактирования узла

Все страницы раздела «Platforms» сделаны в одном стиле, при переходе навкладку «Pins» отобразится список параметров узла в виде таблицы. И при нажатии на кнопку добавления откроется форма создания параметра.

Таким образом React позволяет создавать собственные компоненты, и в дальнейшем использовать их в разных местах.

### Конструктор задач

Конструктор задач, изображение которого показано на рисунке 4.21 является ключевой частью web-приложения. Он позволяет конструировать задачи путем соединения параметров узлов. Интерфейс конструктора можно разделить на несколько составных частей:

* рабочая сетка;
* панель поиска узлов;
* графическое представление узла;
* связи между узлами;
* меню настроек задачи узла и его связей;
* кнопки сохранения изменения и удаления задачи.

Рабочая сетка конструктора предназначена для размещения на ней узлов будущей задачи. При удерживании левой кнопки мыши и перетаскивании, сетка изменяет свое положение относительно осей X и Y.

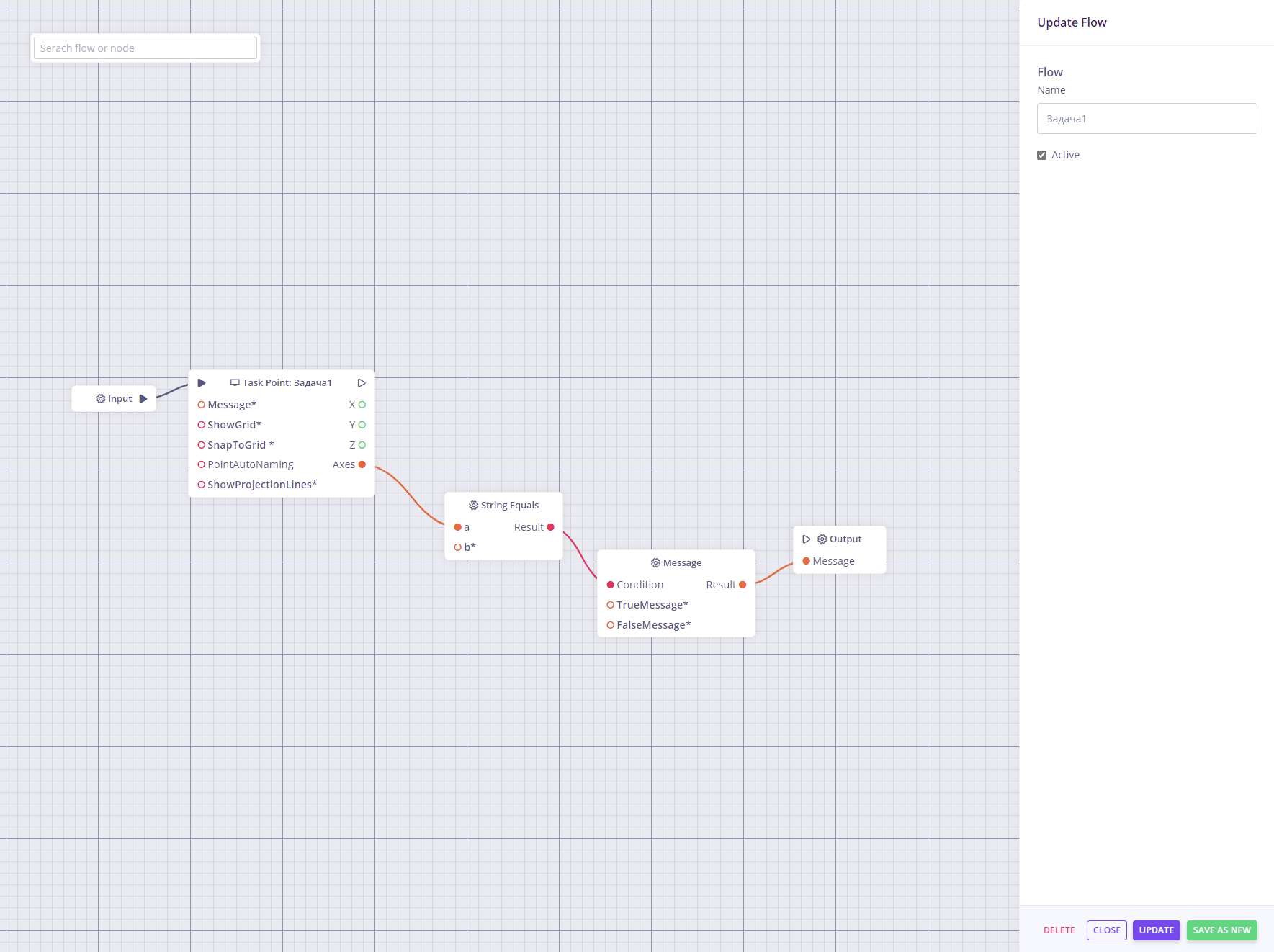


Рисунок 4.21 – Страница конструктора задач

Данная возможность позволяет разрабатывать сколь угодно большие задачи, которые могут не вмещаться на экране пользовательского интерфейса. При перемещении сетки создается эффект того, что ее размер кажется бесконечным. На самом деле изображение сетки на заднем фоне перемещается только на размер большого квадрата, затем положение изображения сбрасывается, и оно перемещается заново. В свою очередь, отображение узлов происходит в другом контейнере, который имеет прозрачный фон, а изменение координат положения происходит в соответствии с перемещением курсора. Таким образом достигается эффект бесконечной сетки.

Панель поиска узлов представляет собой строку поиска, которая разворачивается в список, если перевести на нее фокус. В развернутом состоянии она отображает список доступных для добавления узлов. При нажатии на узел он добавляется на сетку. На рисунке 4.22 показана панель поиска узлов.

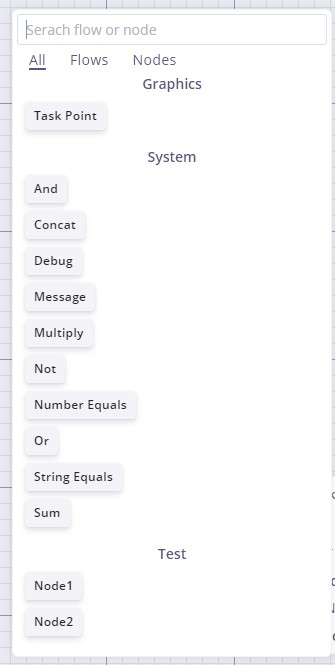


Рисунок 4.22 – Список доступных узлов

При выделении узла, в меню отображаются его входные и выходные параметры. Стоит отметить, что выходные параметры, а также входные, которые соединены линиями являются неактивными. У каждого параметра есть свой тип данных, согласно которому подбирается свой тип поля для ввода. На рисунке 4.23 можно увидеть меню параметров выбранного узла.

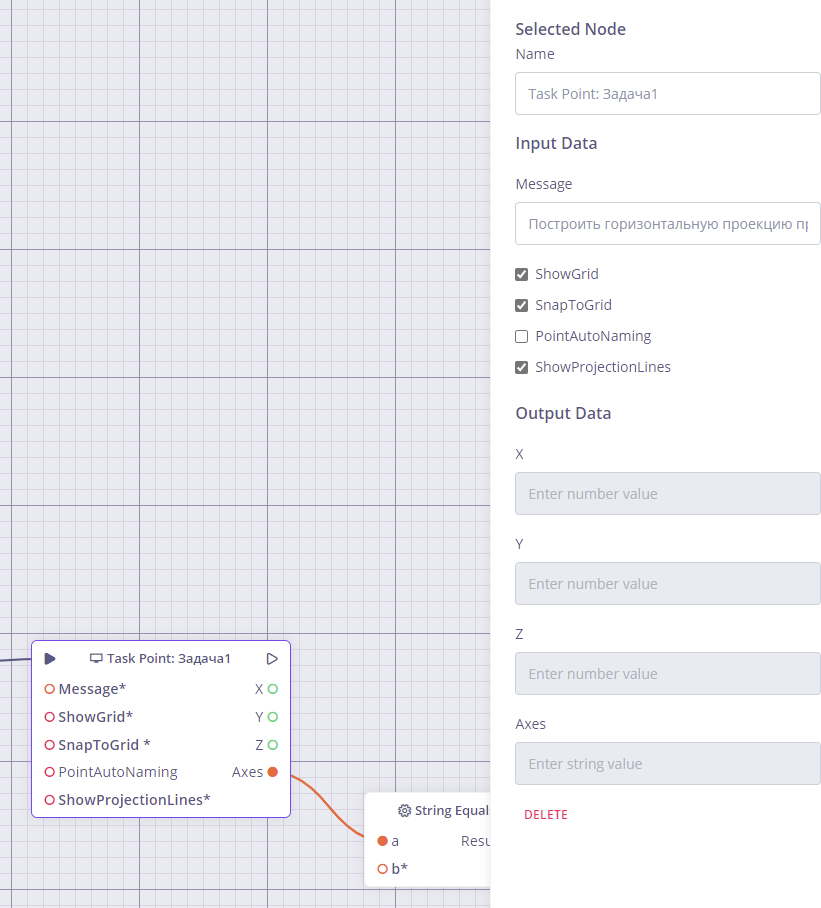


Рисунок 4.23 – Меню параметров выделенного узла

В конструкторе задач используется слишком много динамики, т.е. много изменений, которые связаны между собой. Например, при перетаскивании узла надо заново отрисовать все его соединения, при этом чтобы не жертвовать производительностью, все остальные компоненты лучше не отрисовывать заново. К сожалению, React не поддерживает такой возможности и ее пришлось реализовать вручную.

Для этого был разработан класс ObservableValue, со свойством value, а также двумя методами: subscribe и notify. При вызове метода subscribe происходит подписка на изменения значения. При вызове notify у всех подписок срабатывает повторная отрисовка. Метод notify встроен внутрь сеттера свойства value. Листинг кода класса показан на рисунке 4.24.

Для того чтобы реагировать на сообщение повторной отрисовки был разработан компонент Observer, листинг кода которого показан на рисунке 4.25. Суть компонента заключается в том, что у него есть обязательный входной параметр ObservableValue. При инициализации компонента происходит подписка на изменения ObservableValue.

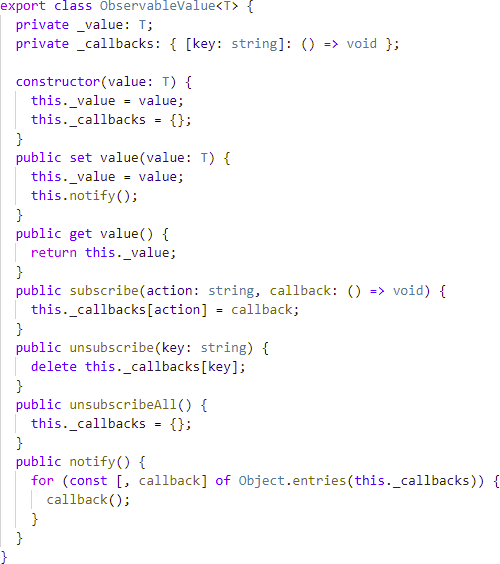


Рисунок 4.24 – Листинг кода класса ObservableValue

В методе subscribe задается функция, которая изменяет состояние текущего компонента стандартным образом. Такое действие вызывает встроенную в React повторную отрисовку. После того как компонент удаляется необходимо отписаться от ObservableValue. Для этого вызывается метод unsubscribe. Таким образом если нужно повторно отрисовать изменения в нескольких изолированных местах, то можно использовать там компонент Observer. Это значительно экономит ресурсы устройства, на котором выполняется работа, а также увеличивает скорость отрисовки компонентов. При необходимости метод notify можно вызвать в любом удобном месте, что является огромным плюсом при разработке данного конструктора.



Рисунок 4.25 – Листинг кода компонента Observer

Для отображения связей между узлами используются сдвоенные кривые Безье, изображенные на рисунке 4.26, которые обеспечивают плавный переход от выходных параметров узла ко входным параметрам следующего узла.

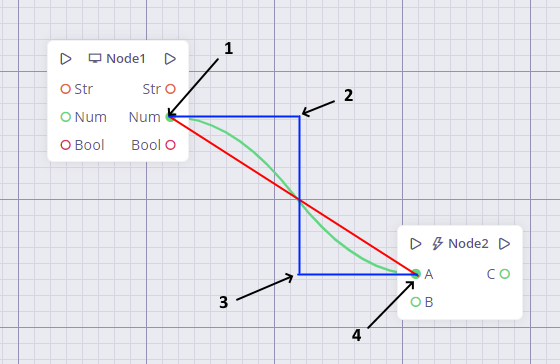


Рисунок 4.26 – Отображение связей между узлами

Сдвоенная кривая состоит из упрощенной модели кривой Безье и содержит четыре набора координат, согласно рисунку 4.26, точки 1 и 4 задают начало и конец кривой, а точки 2 и 3 располагаются по средине, относительно начальной и конечной координаты X. Координата Y у точки 2 совпадает с точкой 1, так же как и координата Y точки 3 совпадает с координатой Y точки 4. При перемещении узлов координаты точек кривой отрисовываются заново.

### Раздел выполнения узлов и задач

Данный раздел предназначен для отладки узлов, а также запуска созданных в конструкторе задач. На главной странице раздела отображается список задач в виде таблицы на вкладке Flows и список улов на вкладке Nodes, как показано на рисунке 4.27.

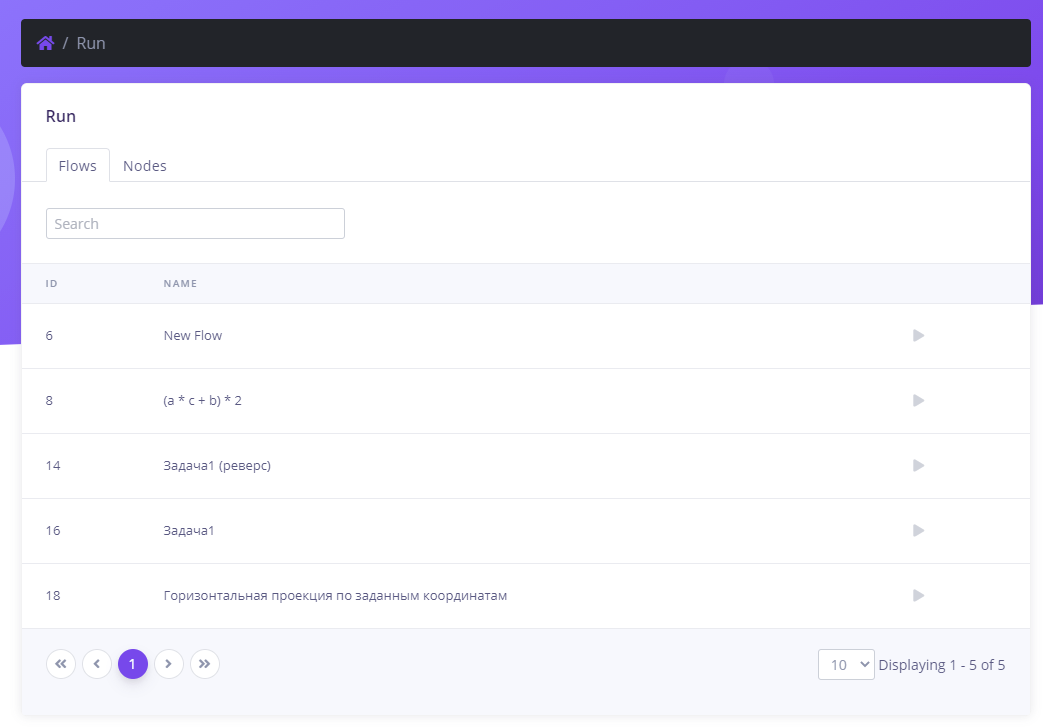


Рисунок 4.27 – Раздел выполнения узлов и задач

При нажатии на кнопку выполнения открывается страница выполнения задачи, как показано на рисунке 4.28. В верху страницы отображается название задачи, а также текущий узел. По центру отображаются входные параметры узла. В нижней части отображаются кнопки закрытия задачи, а также перехода к следующей задаче.

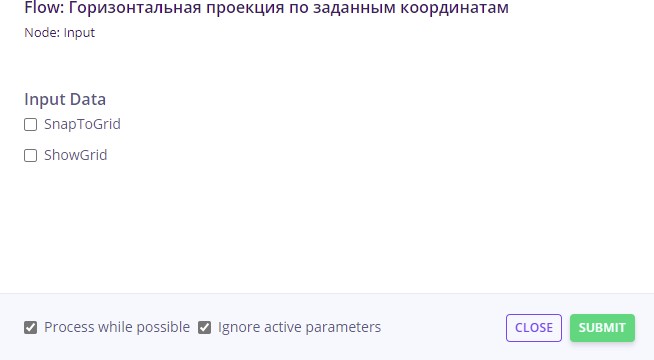


Рисунок 4.28 – Страница выполнения задачи

Если текущая задача имеет тип команды «Команда», то при переходе к следующей задаче, выходные параметры текущей задачи заполняются путем обработки команды по ее коду. Обработчики команд прописываются в коде и выглядят как показано на рисунке 4.29.

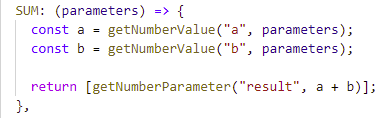


Рисунок 4.29 – Обработчик команды «Sum»

Данная команда производит суммирование параметров a и b и возвращает результат в параметре result.

Если текущая задача имеет тип команды «Страница», отображается страница, которая была указана в качестве команды. Выглядит это как показано на рисунке 4.30. Страница внешнего графического конструктора отображается внутри iframe web- приложения. Общение между iframe и web-приложением происходит за счет обработчика события «message» и метода send. Это позволяет отправить параметры во внешний интерфейс, и затем получить результат, после выполнения каких-то действий. После получения результата происходит переход к следующему узлу. Таким образом каждый узел выполняет свою команду и получает какой-то результат. Важным моментом является то, что выполняются только те узлы, у которых известны все параметры, если какой-либо параметр неизвестен, узел ожидает пока он заполнится.

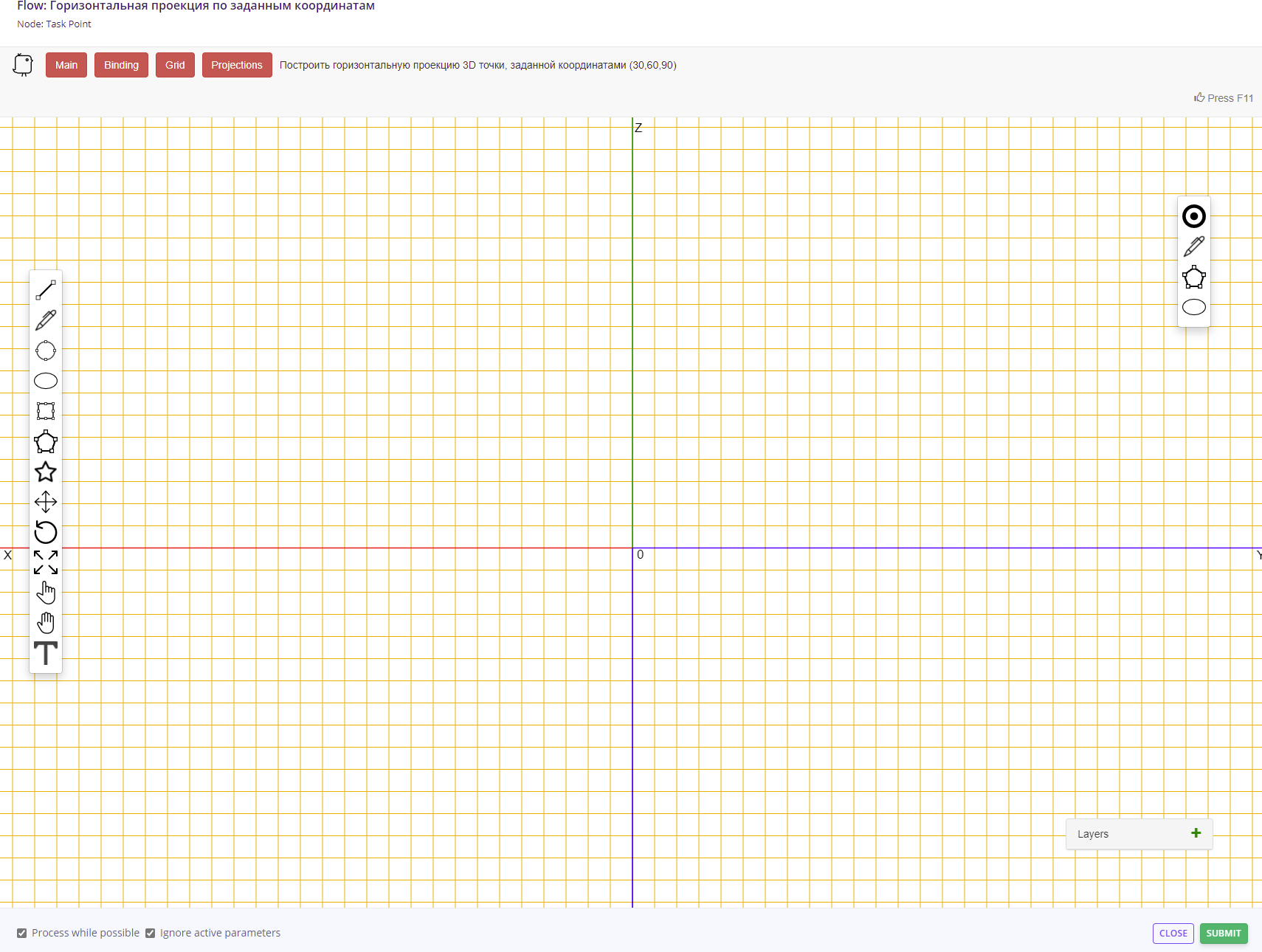


Рисунок 4.30 – Отображение узла с командой «Страница»

Для того чтобы web-приложение смогло обработать команду «Страница» необходимо реализовать небольшой программный код на стороне внешнего конструктора. Код обработчика команды «Страница» показан на рисунке 4.31.

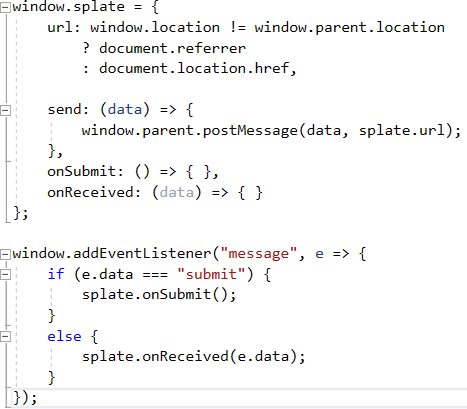


Рисунок 4.31 – Листинг кода обработчика команды «Страница»

Для того чтобы обработчик заработал необходимо переопределить функции onSubmit и onReceived.

Функция onSubmit срабатывает, когда нажимается кнопка Submit. В этой функции обязательно должна вызваться функция send и передаться туда объект с ответом, который затем преобразуется в выходные параметры узла.

Функция onReceived выполняется в момент, когда запускается данный узел, в качестве параметра функции передаются входные данные узла. Если в течение 10 секунд после вызова onSubmit не отправить ответ функцией send, узел отклонит свою обработку.

# ТЕСТИРОВАНИЕ WEB-ПРИЛОЖЕНИЯ «ИТЕРАКТИВНЫЙ ТРЕННАЖЕР ОБУЧЕНИЯ ПОСТРОЕНИЮ ПРОЕКЦИЙ ТОЧЕК И ОТРЕЗКОВ НА ЭПЮРЕ МОНЖА»

Функциональное тестирование является одним из ключевых видов тестирования, задача которого – установить соответствие разработанного программного обеспечения (ПО) исходным функциональным требованиям заказчика. То есть проведение функционального тестирования позволяет проверить способность информационной системы в определенных условиях решать задачи, нужные пользователям.

В зависимости от степени доступа к коду системы выделяют 2 типа функционального тестирования:

1. тестирование black box (черный ящик) – проведение функционального тестирования без доступа к коду системы;
2. тестирование white box (белый ящик) – функциональное тестирование с доступом к коду системы.

Тестирование black box проводится без знания внутренних механизмов работы системы и опирается на внешние проявления ее работы. При этом тестировании проверяется поведение ПО при различных входных данных и внутреннем состоянии систем. В случае тестирования white box создаются тест-кейсы, основанные преимущественно на коде системы ПО.

Ключевые преимущества функционального тестирования:

1. полностью имитирует фактическое использование системы;
2. позволяет своевременно выявить системные ошибки ПО и, тем самым, избежать множества проблем при использовании;
3. экономия за счет исправления ошибок на более раннем этапе жизненного цикла ПО [15].

Функциональное тестирование ПО в рамках дипломного проекта представлено критическим видом тестирования.

Критическое тестирование – основной тип тестовых испытаний, во время которого значимые элементы и функции приложения проверяются на предмет правильности работы при стандартном их использовании. Критическое тестирование является одним из самых распространенных видов функционального тестирования, в частности, для web-проектов. Плюс ко всему позволяет выявить самые быстро находимые дефекты и исправить приложение в более сжатые сроки. Чаще всего на практике на данном уровне тестирования проверяется основная масса требований к продукту.

Критическое тестирование может быть как позитивным, так и негативным:

1. позитивный тест критического пути – это проверка работоспособности функций программного продукта, с которыми пользователь сталкивается ежедневно;
2. негативный тест критического пути – это проверка всевозможных вариантов нестандартного использования функциональности, используемой пользователем каждый день.

Для данного вида тестирования пишутся наиболее подробные и глубокие тест- кейсы, чтобы покрыть всю возможную функциональность приложения.

Критическое тестирование является одним из самых распространенных видов функционального тестирования. Частота проведения данного тестирования обусловлена в первую очередь необходимостью периодической проверки всего приложения в сжатые сроки. А также позволяет выявить самые быстро находимые дефекты и исправить приложение в более сжатые сроки [16]. Список граничных и эквивалентных значений для критического тестирования приведен в таблице 5.1.

Таблица 5.1 – Список граничных и эквивалентных значений для критического тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| Названия поля | Формат данных | Граничные значения | Эквивалентные значения |
| 1 | 2 | 3 | 4 |
| Название платформы | 1. Содержит любые символы. 2. Длина от 1 до 50 символов. 3. Обязательное поле для заполнения. | Строки длиной 1  символ и 50 символов | Строки длиной 2 символа, 49 символа и 20 символов. |
| Описание платформы | 1. Содержит любые символы. 2. Длина от 1 до 250 символов. 3. Обязательное поле для заполнения. | Строки длиной 1  символ и 250 символов | Строки длиной 2 символа, 249 символа и 125 символов. |
| Автор платформы | 1. Содержит любые символы. 2. Длина от 1 до 50 символов. 3. Обязательное поле для заполнения. | Строки длиной 1  символ и 50 символов | Строки длиной 2 символа, 49 символа и 20 символов. |
| Название узла | 1. Содержит любые символы. 2. Длина от 1 до 50 символов. 3. Обязательное поле для заполнения. | Строки длиной 1  символ и 50 символов | Строки длиной 2 символа, 49 символа и 20 символов. |
| Команда узла | 1. Содержит любые символы. 2. Длина от 1 до 250 символов. 3. Обязательное поле для заполнения. | Строки длиной 1  символ и 250 символов | Строки длиной 2 символа, 249 символа и 125 символов. |

Продолжение таблицы 5.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Название параметра | 1. Содержит любые символы. 2. Длина от 1 до 50 символов. 3. Обязательное поле для заполнения. | Строки длиной 1  символ и 50 символов | Строки длиной 2 символа, 49 символа и 20 символов. |
| Название задачи | 1. Содержит любые символы. 2. Длина от 1 до 50 символов 3. Обязательное поле для заполнения | Строки длиной 1  символ и 50 символов | Строки длиной 2 символа, 49 символа и 20 символов. |
| Название узла задачи | 1. Содержит любые символы. 2. Длина от 1 до 50 символов. 3. Обязательное поле для заполнения. | Строки длиной 1  символ и 50 символов | Строки длиной 2 символа, 49 символа и 20 символов. |
| Значение параметра узла задачи | 1. Содержит любые символы. 2. Длина от 1 до 250 символов. 3. Обязательное поле для заполнения. | Строки длиной 1  символ и 250 символов | Строки длиной 2 символа, 249 символа и 125 символов. |
| Название параметра задачи | 1. Содержит любые символы. 2. Длина от 1 до 50 символов. 3. Обязательное поле для заполнения. | Строки длиной 1  символ и 50 символов | Строки длиной 2 символа, 49 символа и 20 символов. |

В таблице 5.2 представлены некоторые тестовые случаи для проверки работы с данными: добавление, удаление, редактирование, просмотр, поиск.

Таблица 5.2 – Примеры тестовых случаев критического тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название модуля | Описание тестового случая | Ожидаемые результаты | Тест пройден? |
| 1 | 2 | 3 | 4 | 5 |
| 1 | Создание платформы | Создание новой платформы:   1. Перейти в раздел платформ. 2. Нажать «Add New». 3. Ввести корректные данные. 4. Нажать на кнопку «Save». | 1. Открывается страница добавления платформы. 2. Информация появляется в полях. 3. Платформа сохраняется. 4. Платформа отображается в таблице на главной странице раздела. | Да |
| 2 | Редактирование платформы | Редактирование существующей платформы:   1. Перейти в раздел платформ. 2. Нажать «Edit». 3. Изменить существующие данные. 4. Нажать на кнопку «Update». | 1. Открывается страница редактирования платформы. 2. Информация появляется в полях. 3. Данные изменяются на актуальные. | Да |

Продолжение таблицы 5.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 3 | Удаление платформы | Удаление существующей платформы:  Перейти в раздел платформ.  Нажать «Edit».  Нажать «Delete». | Открывается страница редактирования платформы.  Платформа удаляется.  Открывается главная страница раздела. | Да |
| 4 | Поиск платформы | Поиск платформы:  Перейти в раздел платформ.  Ввести название в поле поиска. | 1) Отображается отфильтрованный список платформ. | Да |
| 5 | Создание узла | Создание нового узла:   1. Открыть редактированиеплатформы. 2. Выбрать вкладку «Nodes». 3. Нажать «Add New». 4. Ввести корректные данные. 5. Нажать на кнопку «Save. | 1. Открывается страница добавления узла 2. Информация появляется в полях. 3. Узел сохраняется. 4. Узел отображается в таблице на странице редактирования платформы. | Да |
| 6 | Редактирование узла | Редактирование существующего узла:   1. Открыть редактирование платформы. 2. Выбрать вкладку «Nodes». 3. Нажать «Edit». 4. Изменить существующие данные. 5. Нажать на кнопку «Update». | 1. Открывается страница редактирования узла. 2. Информация появляется в полях. 3. Данные изменяются на актуальные. | Да |
| 7 | Удаление узла | Удаление существующего узла:   1. Открыть редактирование платформы. 2. Выбрать вкладку «Nodes». 3. Нажать «Edit». 4. Нажать «Delete». | 1. Открывается страница редактирования узла. 2. Узел удаляется. 3. Открывается страница редактирования платформы. | Да |
| 8 | Поиск узла | Поиск платформы:   1. Открыть редактирование платформы. 2. Выбрать вкладку «Nodes». 3. Ввести название в поле поиска. | 1) Отображается отфильтрованный список узлов. | Да |
| 9 | Создание параметра | Создание нового параметра:   1. Открыть редактирование узла. 2. Выбрать вкладку «Pins». 3. Нажать «Add New». 4. Ввести корректные данные. 5. Нажать на кнопку «Save». | 1. Открывается страница добавления параметра. 2. Информация появляется в полях. 3. Параметр сохраняется. 4. Параметр отображается в   таблице на странице редактирования узла. | Да |
| 10 | Редактирование параметра | Редактирование существующего параметра:   1. Открыть редактирование узла. 2. Выбрать вкладку «Pins». 3. Нажать «Edit». 4. Изменить существующие данные. 5. Нажать на кнопку «Update». | 1. Открывается страница редактирования параметра. 2. Информация появляется в полях. 3. Данные изменяются на актуальные. | Да |

Продолжение таблицы 5.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 11 | Удаление параметра | Удаление существующего параметра:   1. Открыть редактирование узла. 2. Выбрать вкладку «Pins». 3. Нажать «Edit». 4. Нажать «Delete». | 1. Открывается страница редактирования параметра. 2. Параметр удаляется. 3. Открывается страница редактирования узла. | Да |
| 12 | Поиск параметра | Поиск параметра:   1. Открыть редактирование узла. 2. Выбрать вкладку «Pins». 3. Ввести название в поле поиска. | 1) Отображается отфильтрованный список параметров. | Да |
| 13 | Создание задачи | Создание новой задачи:   1. Перейти в раздел задач. 2. Нажать «Add New». 3. Ввести корректные данные. 4. Нажать на кнопку «Save». | 1. Открывается страница добавления задачи. 2. Информация появляется в полях. 3. Задача сохраняется. 4. Задача отображается в таблице на странице раздела. | Да |
| 14 | Редактирование задачи | Редактирование существующей задачи:   1. Перейти в раздел задач. 2. Нажать «Edit». 3. Изменить существующие данные. 4. Нажать на кнопку «Update». | 1. Открывается страница редактирования задачи. 2. Информация появляется в полях. 3. Данные изменяются на актуальные. | Да |
| 15 | Удаление задачи | Удаление существующей задачи:   1. Перейти в раздел задач. 2. Нажать «Edit». 3. Нажать «Delete». | 1. Открывается страница редактирования задачи. 2. Задача удаляется. 3. Открывается главная страница раздела. | Да |
| 16 | Поиск задачи | Поиск задачи:   1. Перейти в раздел задач. 2. Ввести название в поле поиска. | 1) Отображается  отфильтрованный список задач. | Да |
| 17 | Запуск узла | Запуск узла:   1. Перейти в раздел запуска узлов. 2. Запустить узел. 3. Ввести входные параметры 4. Нажать «Submit». | 1. Открывается страница выполнения узла. 2. Заполняются входные параметры. 3. Происходит выполнение команды. 4. Появляются выходные параметры. | Да |
| 18 | Запуск задачи | Запуск задачи:   1. Перейти в раздел запуска задач. 2. Запустить задачу. 3. Ввести входные параметры Нажать «Submit». | 1. Открывается страница выполнения задачи. 2. Заполняются входные параметры. 3. Происходит выполнение команды. 4. Появляются выходные параметры. | Да |

В результате проведения тестирования была проверена функциональность разработанного web-приложения. Ошибок обнаружено не было.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## Раздел общей информации

В ходе дипломного проектирования было разработано web-приложение «Интерактивный тренажер обучения построению проекций точек и отрезков на эпюре Монжа». На рисунке 6.1 можно увидеть общий алгоритм работы приложения.

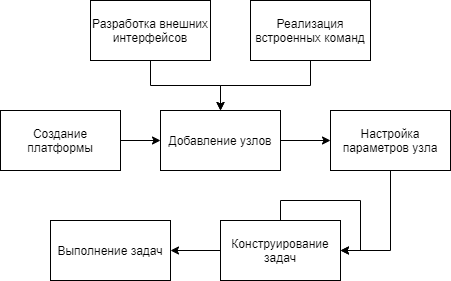


Рисунок 6.1 – Общий алгоритм работы ПП

На главной странице web-приложения расположено меню, изображенное на рисунке 6.2. Данное меню служит для навигации между страницами. Такой вариант расположения был выбран для того, чтобы сэкономить свободное пространство на экране.

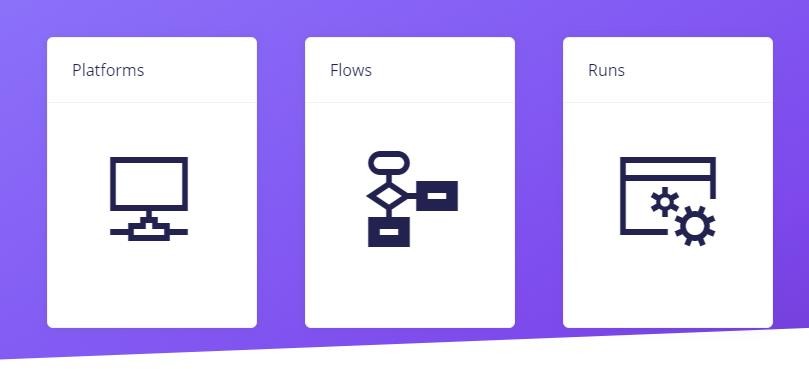


Рисунок 6.2 – Главное меню web-приложения

Из главного меню можно попасть во все разделы:

* Platforms – раздел управления платформами.
* Flows – раздел конструирования задач.
* Runs – раздел выполнения узлов и задач.

## Раздел управления платформами

Данный раздел предназначен для конфигурирования узлов, которые в дальнейшем можно использовать для конструирования задач. На главной странице раздела расположена панель навигации и таблица с платформами, изображенные на рисунке 6.3.

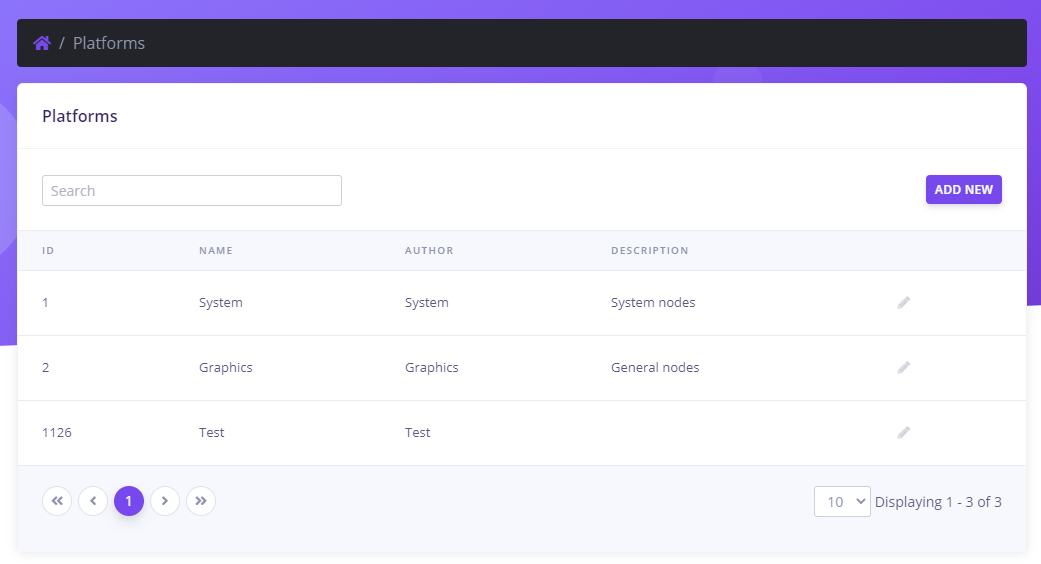


Рисунок 6.3 – Главная страница раздела Platforms

При нажатии на кнопку «Add New» или кнопку «Edit» можно попасть на страницу добавления или редактирования платформы, которая показана на рисунке 6.4.

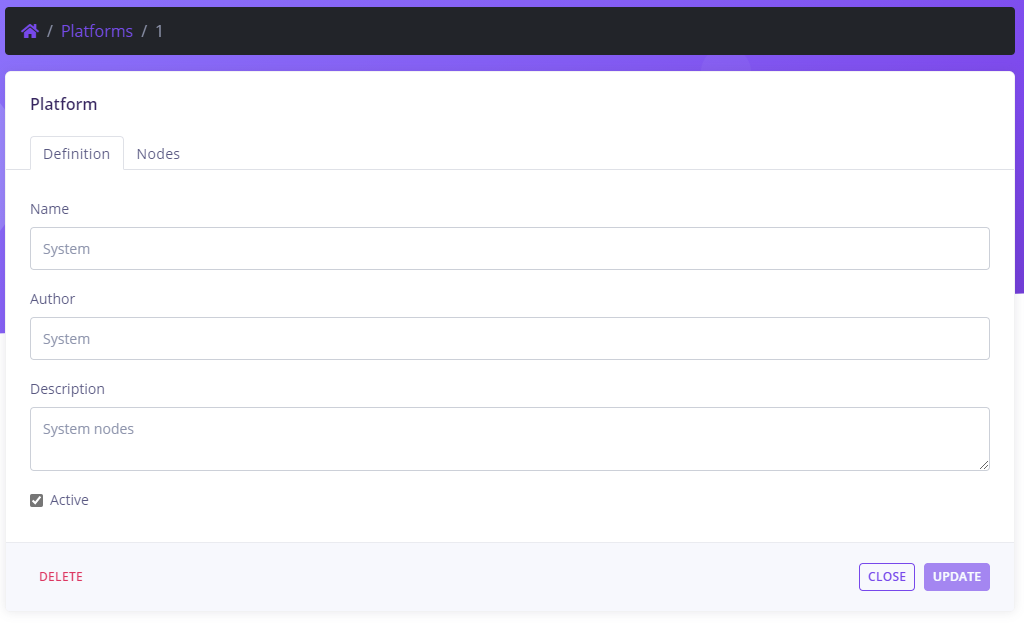


Рисунок 6.4 – Страница редактирования платформы

При нажатии на кнопку «Delete» можно удалить платформу. Если перейти на вкладку «Nodes», то можно увидеть список узлов, относящихся к платформе, как показано на рисунке 6.5. Стоит отметить, что данная вкладка активна только при редактировании платформы.

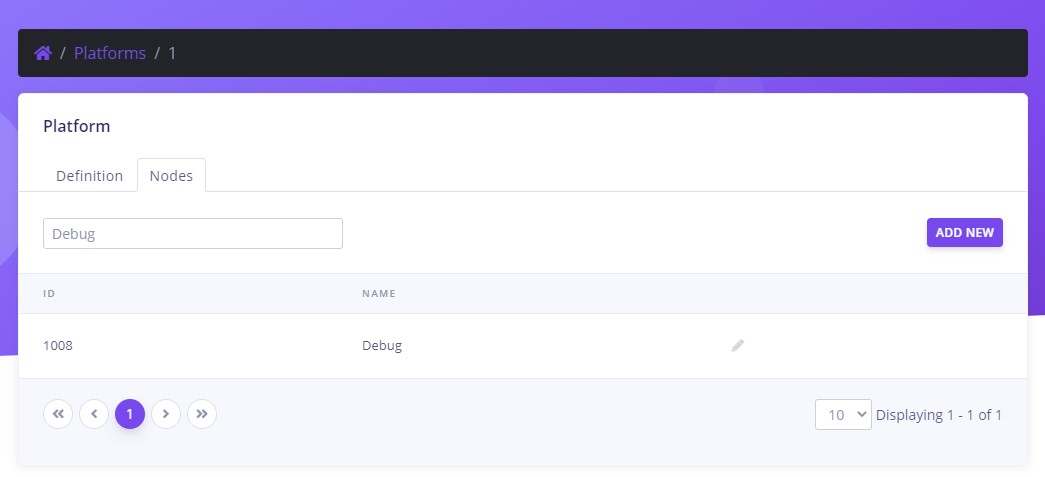


Рисунок 6.5 – Вкладка «Nodes» на странице редактирования платформы

У каждой таблицы присутствует пагинация, поиск и возможность изменения количества отображаемых элементов.

При нажатии на кнопку «Add New» в таблице узлов, откроется страница добавления нового узла, как показано на рисунке 6.6. Существует возможность выбора типа узла. Есть два типа узлов: «Команда» и «Страница». Если выбран тип «Команда», то в поле команды нужно вписать ее код, который предусмотрен обработчиком. Добавление новых обработчиков происходит только внесением изменений в программный код клиентской части web-приложения. При выборе «Страницы» в качестве типа команды, в поле значения команды необходимо ввести адрес страницы web-сайта.

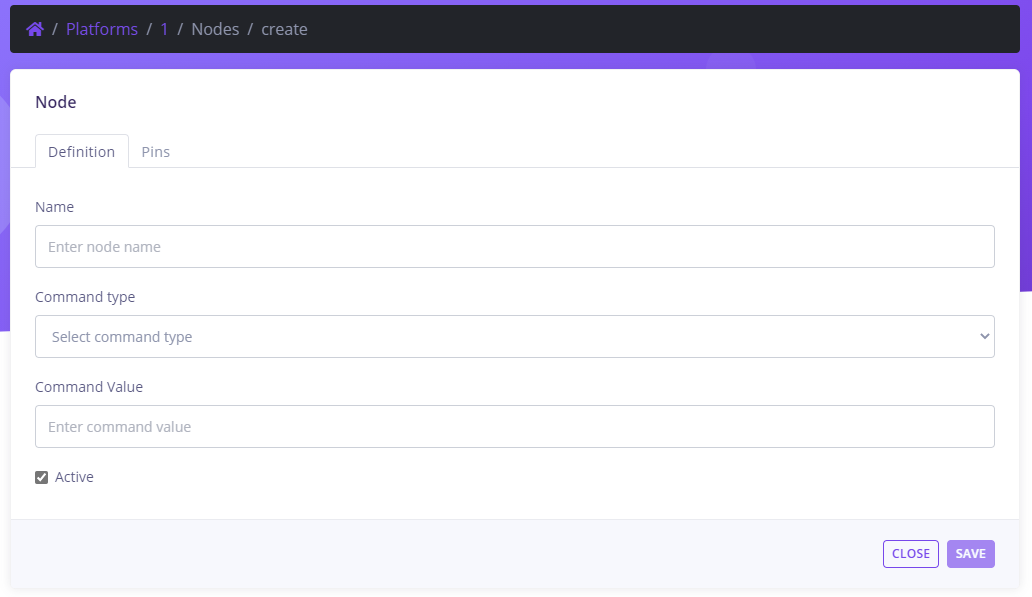


Рисунок 6.6 – Страница добавления узла

Если перейти на вкладку «Pins», то отобразится таблица со списком параметров данного узла. Изображение таблицы параметров узла показано на рисунке 6.7.

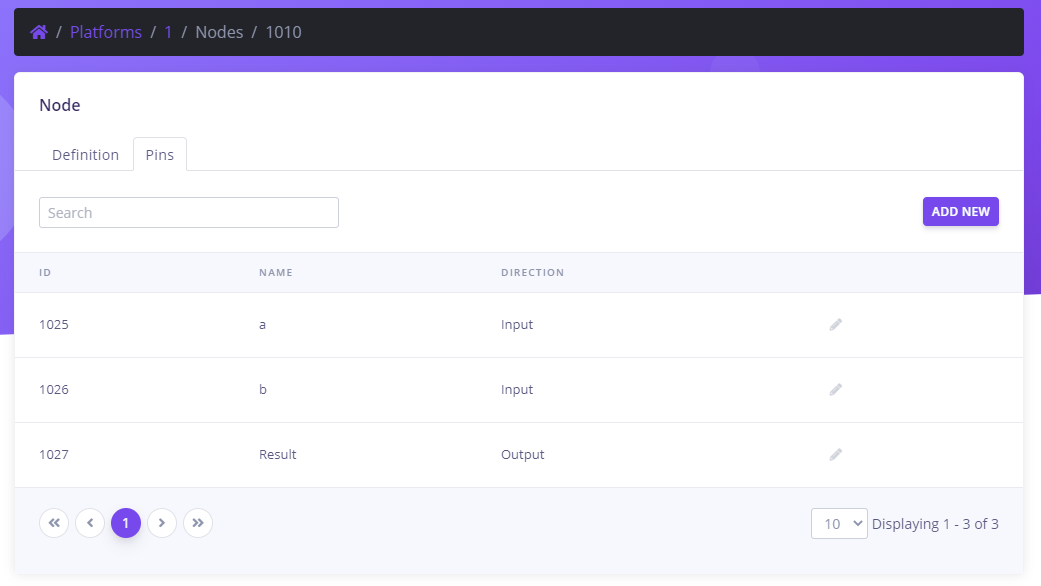


Рисунок 6.7 – Вкладка параметров на странице редактирования узла

При нажатии на кнопку «Edit» или «Add New» открывается страница редактирования параметра, которая показана на рисунке 6.8. Если снять галочку с поля «Is Public», то при использовании узла в конструкторе задач, к этому параметру нельзя будет присоединить линию связи.

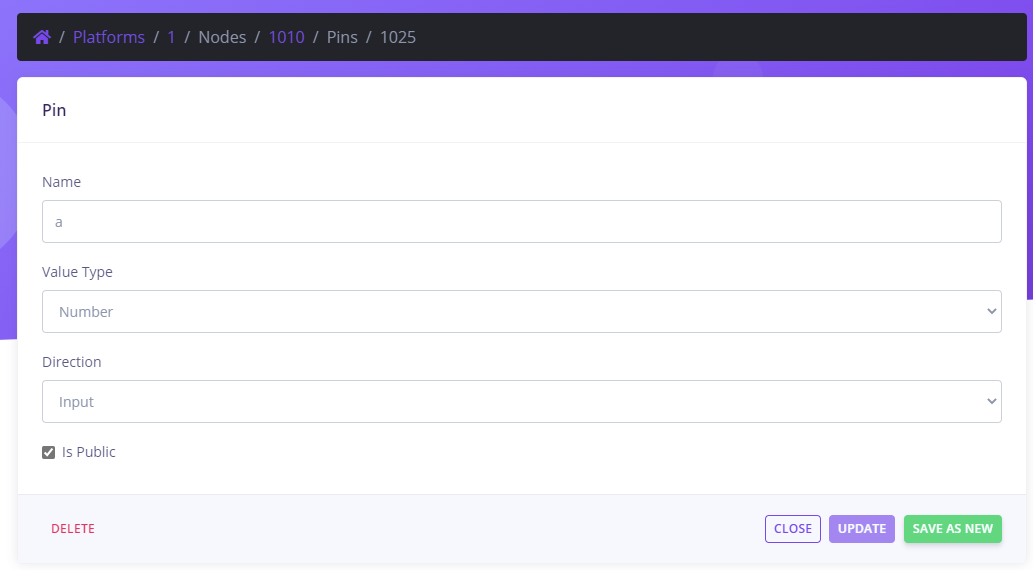


Рисунок 6.8 – Страница редактирования параметра узла

## Раздел конструирования задач

Данный раздел представляет собой конструктор, при помощи которого можно создавать задачи, путем комбинирования узлов и связывания их параметров. Конструктор можно увидеть на рисунке 6.9.

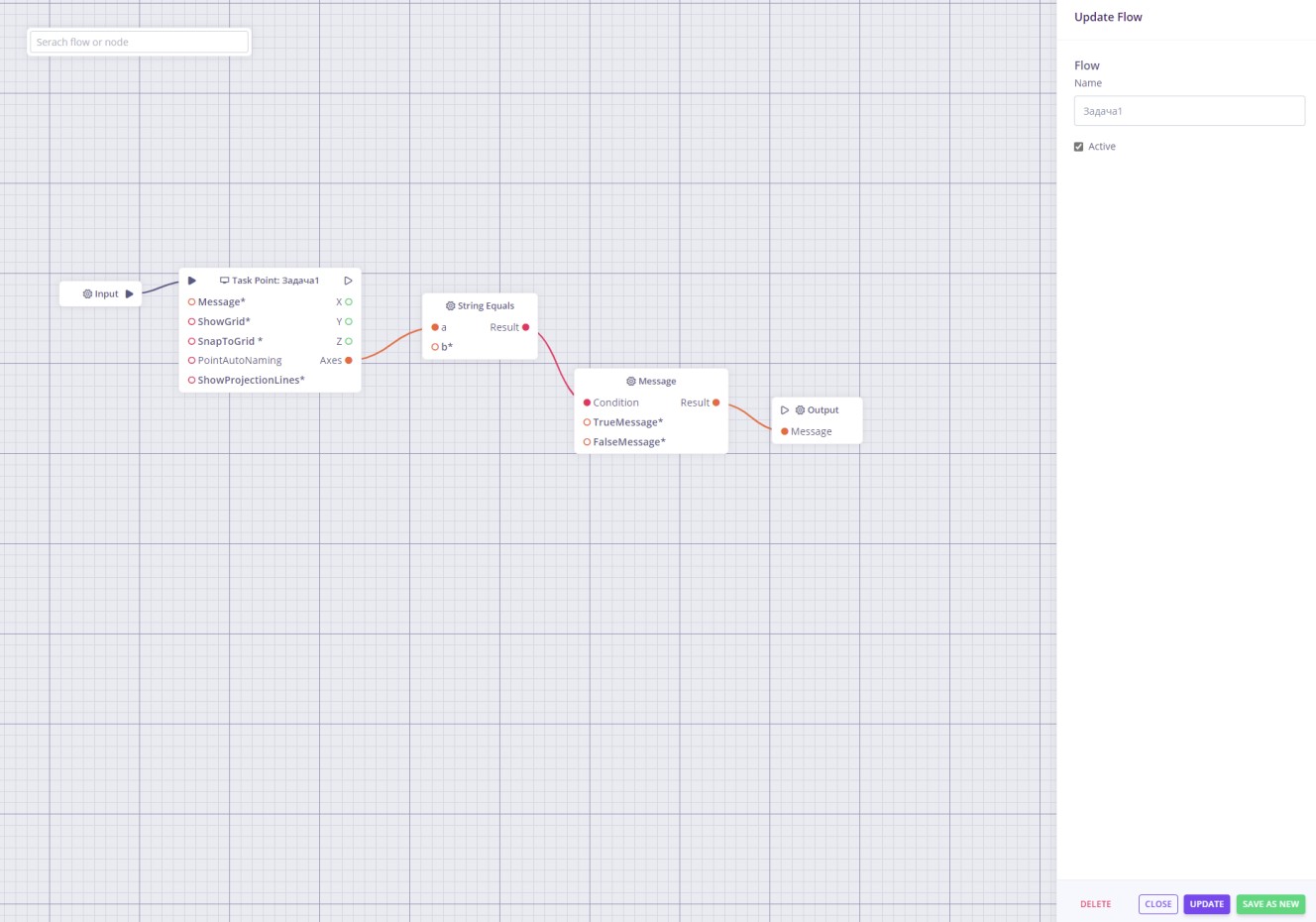


Рисунок 6.9 – Страница создания задачи

На странице отображаются следующие компоненты:

* «сетка» – рабочая область, на которой размещаются узлы;
* «меню параметров» – отображает параметры выделенного элемента;
* «список узлов» – представляет строку поиска узлов, которая разворачивается в список, если перевести на нее фокус. В развернутом состоянии отображает список доступных для добавления узлов. При нажатии на узел он добавляется на сетку. На рисунке 6.10 показан список узлов. При удерживании левой кнопки мыши на сетке и перемещении курсора можно изменить ее положение. Это удобно использовать в случаях, когда задача содержит большое количество блоков и не вмещается на пользовательском экране. Также при удерживании левой кнопки мыши на узле, и перемещении курсора узел будет также перемещаться, а вместе с ним и связи, относящиеся к данному узлу. При нажатии на параметр узла и перетаскивании его в противоположный параметр другого узла, а также при условии совпадения типов создается связь между узлами, которая отображается в виде кривой и имеет цвет соответствующего типа данных параметра.

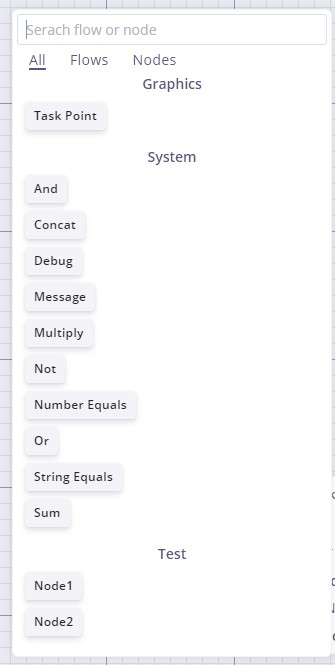


Рисунок 6.10 – Список доступных узлов

При выделении узла, в меню отображаются его входные и выходные параметры. Стоит отметить, что выходные параметры, а также входные, которые соединены линиями являются неактивными. У каждого параметра есть свой тип данных, согласно которому подбирается свой тип поля для ввода. На рисунке 6.12 можно увидеть меню параметров выбранного узла.

В данный момент поддерживаются 3 типа данных параметров:

* числовой тип данных – позволяет хранить целочисленные и дробные значения;
* текстовый тип – позволяет хранить строки с текстом;
* булевый тип – позволяет хранить значения «true» и «false».

Числовой тип данных отображается в виде текстового поля, в котором доступен ввод чисел. В представлении узла на сетке конструктора параметр данного типа имеет зеленый цвет. Текстовый тип данных отображается в виде текстового поля, и имеет оранжевый цвет в конструкторе. Булевый тип данных отображается в виде чек-бокса и имеет красный цвет в конструкторе. Соответствие цветов и полей для ввода типов данных показано на рисунке 6.11.

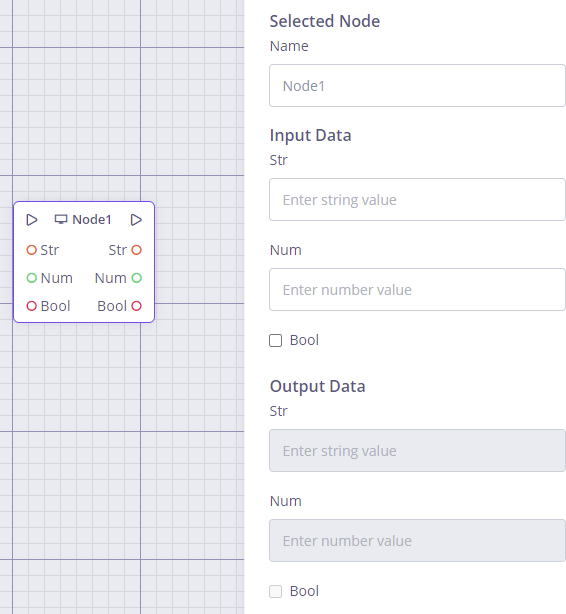


Рисунок 6.11 – Отображение цветов и полей для ввода типов данных

Таким образом обеспечивается организация строгой типизации входных и выходных параметров узлов, что значительно облегчает организацию передачи параметров между узлами.

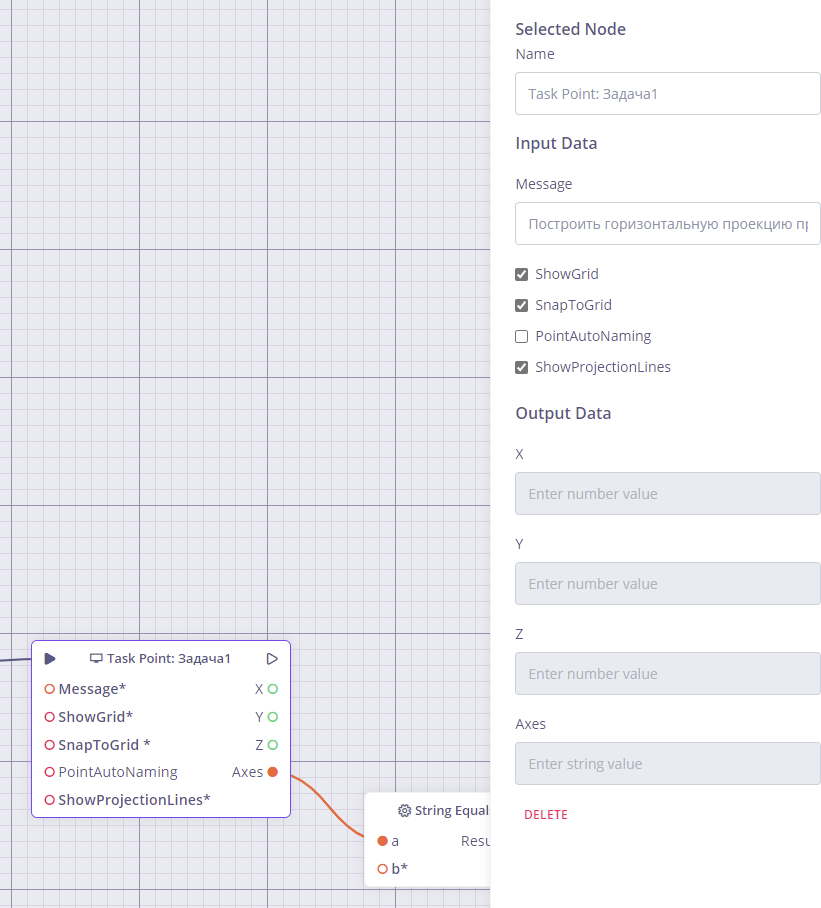


Рисунок 6.12 – Меню параметров выделенного узла

Стоит отметить, что у каждого узла в названии присутствует иконка, которая позволяет определить тип используемой команды в узле. На рисунке 6.13 показаны все возможные виды команд в узлах.

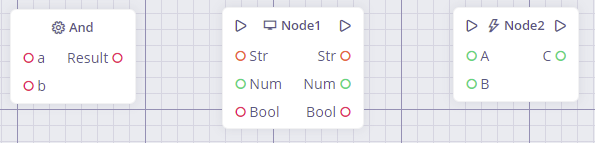


Рисунок 6.13 – Отображение узлов с разными командами

Таким образом в данный момент поддерживаются три типа команд:

* встроенные команды, которые имеют иконку в виде шестеренки и реализуются путем добавления обработчика команды в код приложения;
* внешние конструкторы, которые имеют иконку в виде дисплея, для таких команд разрабатываются отдельные web-сайты и затем интегрируются в текущее web- приложение;
* API команды, которые имеют иконку в виде молнии и представляют собой вызов API интерфейса.

## Раздел выполнения узлов и задач

Данный раздел предназначен для отладки узлов, а также запуска задач, созданных в конструкторе. На странице раздела отображается список задач в виде таблицы на вкладке «Flows» и список улов на вкладке «Nodes», как показано на рисунке 6.14.

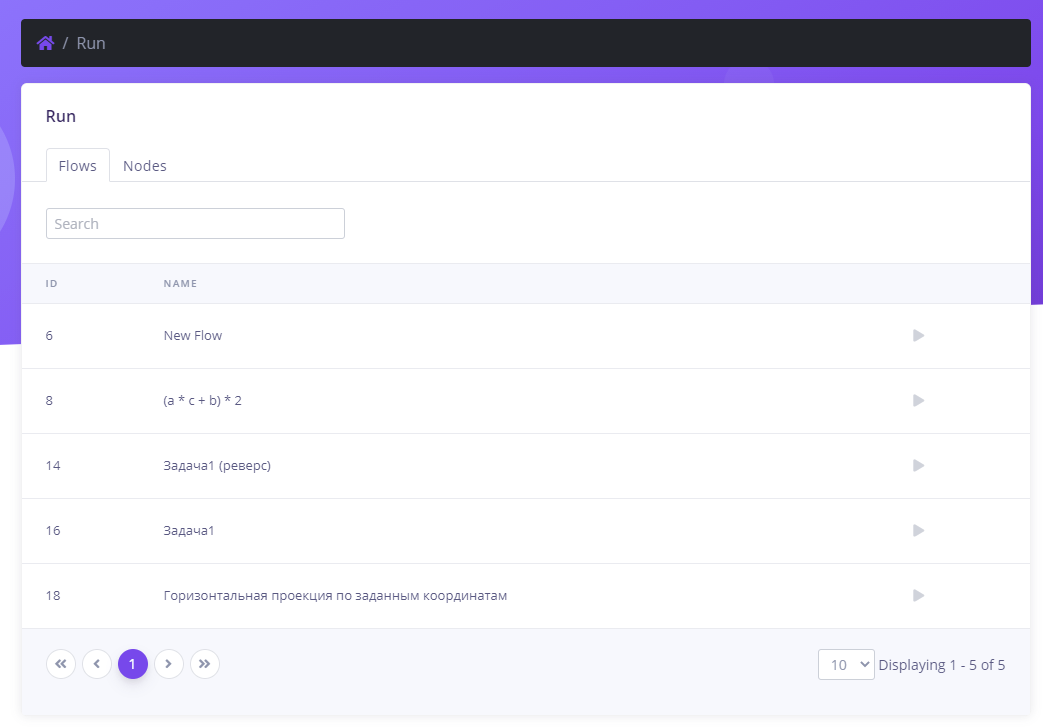


Рисунок 6.14 – Раздел выполнения узлов и задач

При нажатии на кнопку выполнения открывается страница выполнения задачи, как показано на рисунке 6.15. В верху страницы отображается название задачи, а также текущий узел. По центру отображаются входные параметры узла. В нижней части отображаются кнопки закрытия задачи, а также перехода к следующей задаче.

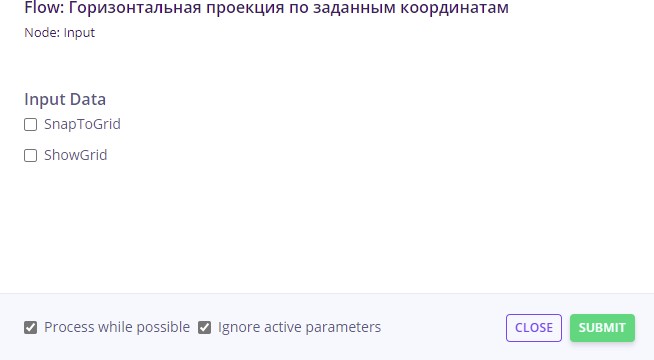


Рисунок 6.15 – Страница выполнения задачи

Если текущая задача имеет тип команды «Страница», отображается страница, которая была указана в качестве команды. Выглядит это как показано на рисунке 6.16.

Таким образом каждый узел выполняет свою команду и получает какой-то результат. Важным моментом является то, что выполняются только те узлы, у которых известны все параметры, если какой-либо параметр неизвестен, узел ожидает пока он заполнится.

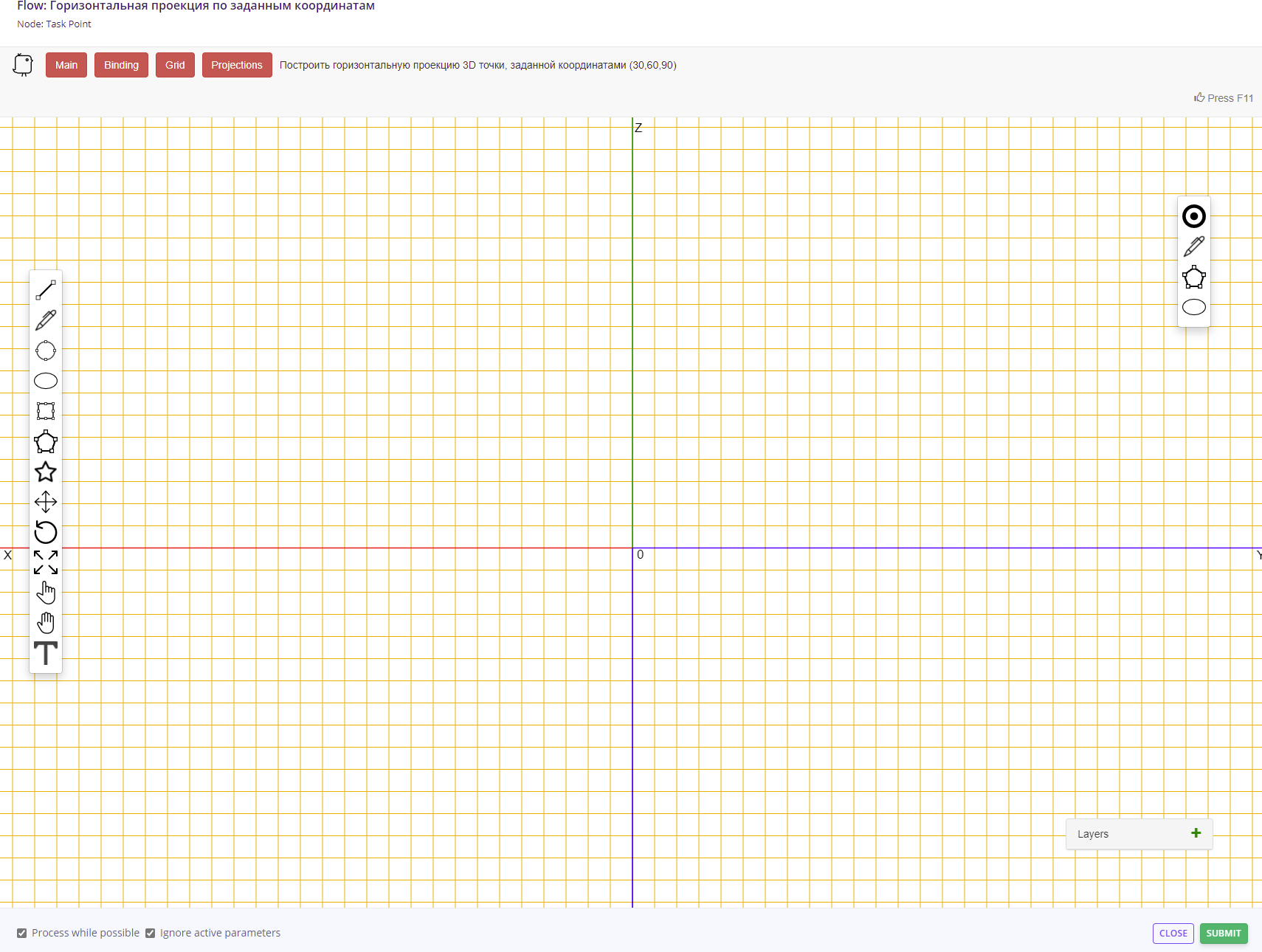


Рисунок 6.16 – Отображение узла с командой «Страница»

## Пример использования web-приложения

Согласно схеме общего алгоритма работы приложения, показанной на рисунке 6.1 можно привести пример работы с web-приложением. В примере демонстрируется добавление внешней системы для графического построения 3D точек, создание задачи с использованием данной системы и выполнение построенной задачи.

В первую очередь создадим платформу. Для этого переходим на главную страницу приложения, которая показана на рисунке 6.2 и нажимаем на карточку

«Platforms». Далее на открывшейся странице нажимаем кнопку «Add New» и переходим на страницу добавления платформы. Вводим данные и нажимаем на кнопку

«Save» для сохранения. В итоге получаем созданную платформу, изображение которой показано на рисунке 6.17.

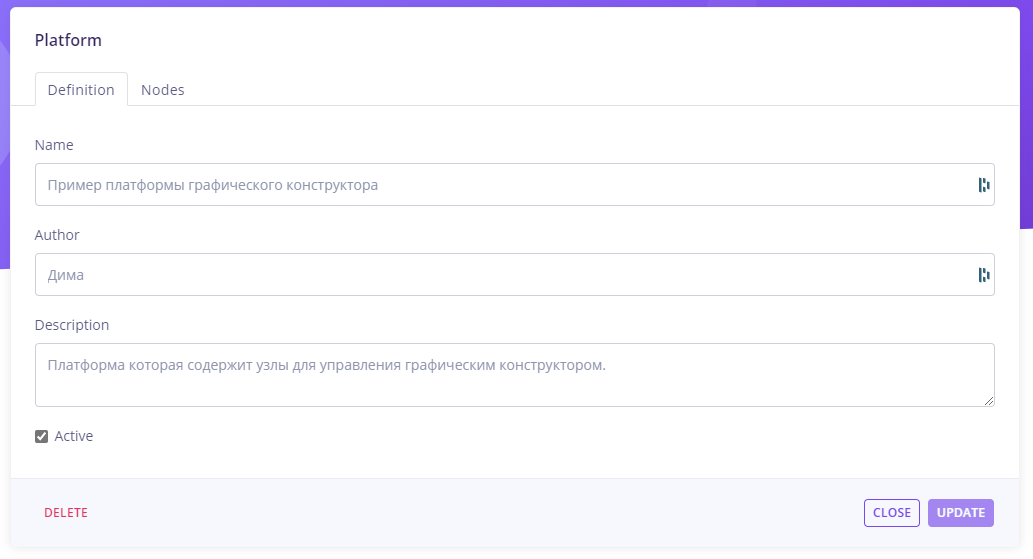


Рисунок 6.17 – Пример созданной платформы

Далее переходим на вкладку «Nodes» и нажимаем на кнопку «Add New». После этого откроется страница добавления узлов. Заполняем поле названия, в поле

«Command type» выбираем значение «Page», а в поле «Command» указываем адрес страницы графического конструктора, затем нажимаем кнопку «Save» для сохранения узла. Таким образом узел будет иметь вид, который показан на рисунке 6.18. Далее необходимо реализовать поведение внешнего графического конструктора (интегрировать его в текущее web-приложение). Это происходит путем использования кода, показанного на рисунке 4.31 и переопределения функций onSubmit и onReceived.

Конечная реализация поведения графического конструктора будет выглядеть как показано на рисунке 6.18.

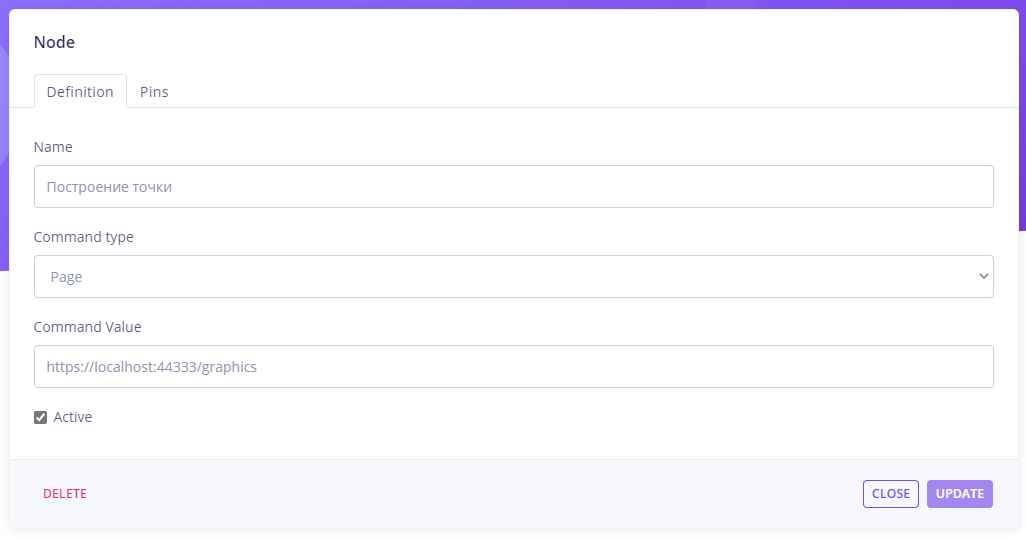


Рисунок 6.18 – Пример создания узла

Следующим этапом является добавление в узел всех использованных параметров при интеграции графического конструктора. Это делается путем нажатия на кнопку

«Add New» во вкладке «Pins» на странице редактирования узла. После этого открывается страница создания параметров. Там указывается название, тип данных и направление, которое определяет входной это или выходной параметр. Страницу добавления параметров можно увидеть на рисунке 6.8. Для более быстрого добавления параметров можно изменять текущий (редактируемый) параметр и вместо нажатия на кнопку «Save» нажимать на «Save as New». Таким образом редактируемый параметр останется не тронутым, а создастся новый с введенными данными. После настройки узла можно проверить его работоспособность. Для этого переходим на главную страницу и нажимаем на карточку «Runs». После перехода на страницу «Run», которая изображена на рисунке 4.27 переходим на вкладку «Nodes» и запускаем выбранный узел. После успешной проверки переходим на страницу конструктора, где собираем задачу из предоставленных узлов.

Таким образом получаем созданную задачу, которая проверяет построение горизонтальной проекции произвольно заданной точки. Схема задачи показана на рисунке 6.19.

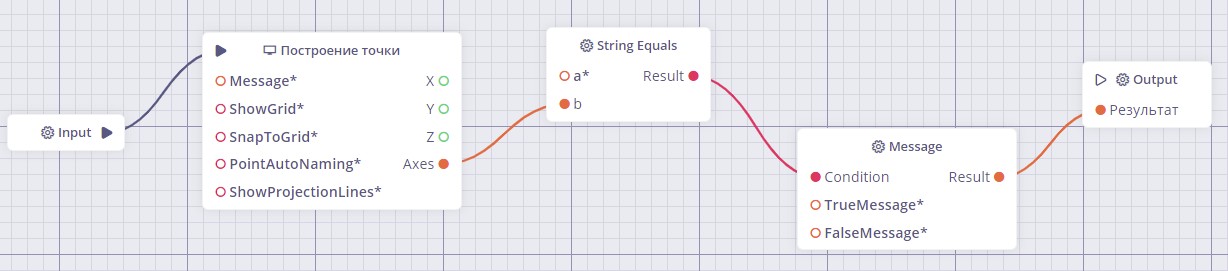


Рисунок 6.19 – Пример построения задачи

При настройке узла в задаче были заданы такие параметры как привязка к сетке, отображение сетки, наименование точек, а также отображение условия задачи. В качестве результата задача вернет текстовое сообщение об верном либо неверном решении. На рисунке 6.20 представлено переопределение функций onSubmit и onReceived в коде графического конструктора.

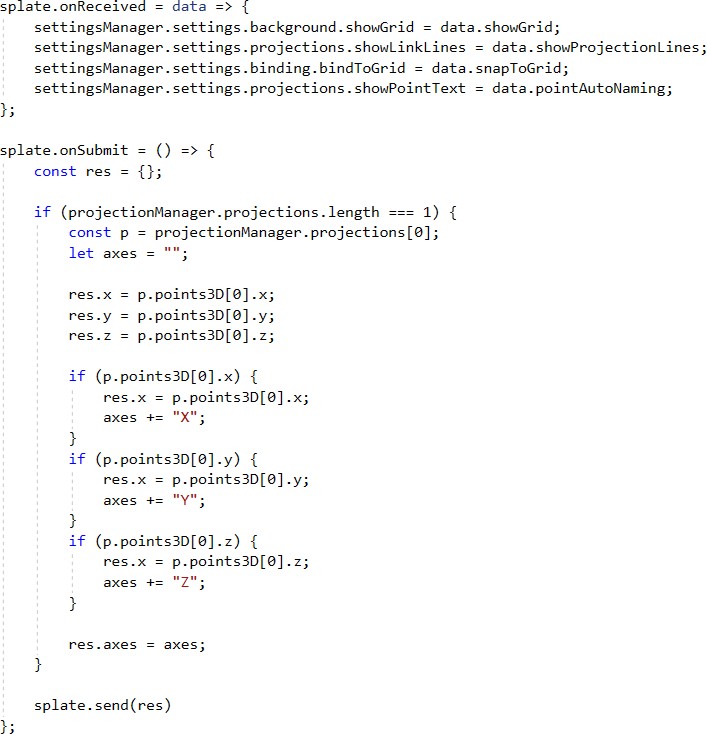


Рисунок 6.20 – Переопределение функций onSubmit и onReceived

Далее переходим на страницу «Runs» во вкладку «Flows» и запускаем созданную задачу. Таким образом открывается страница выполнения задачи, где предлагается построить проекцию точки. Страница содержит внутри себя встроенный графический конструктор, который показан на рисунке 6.21. Там же можно увидеть все настройки заданные при конструировании задачи.

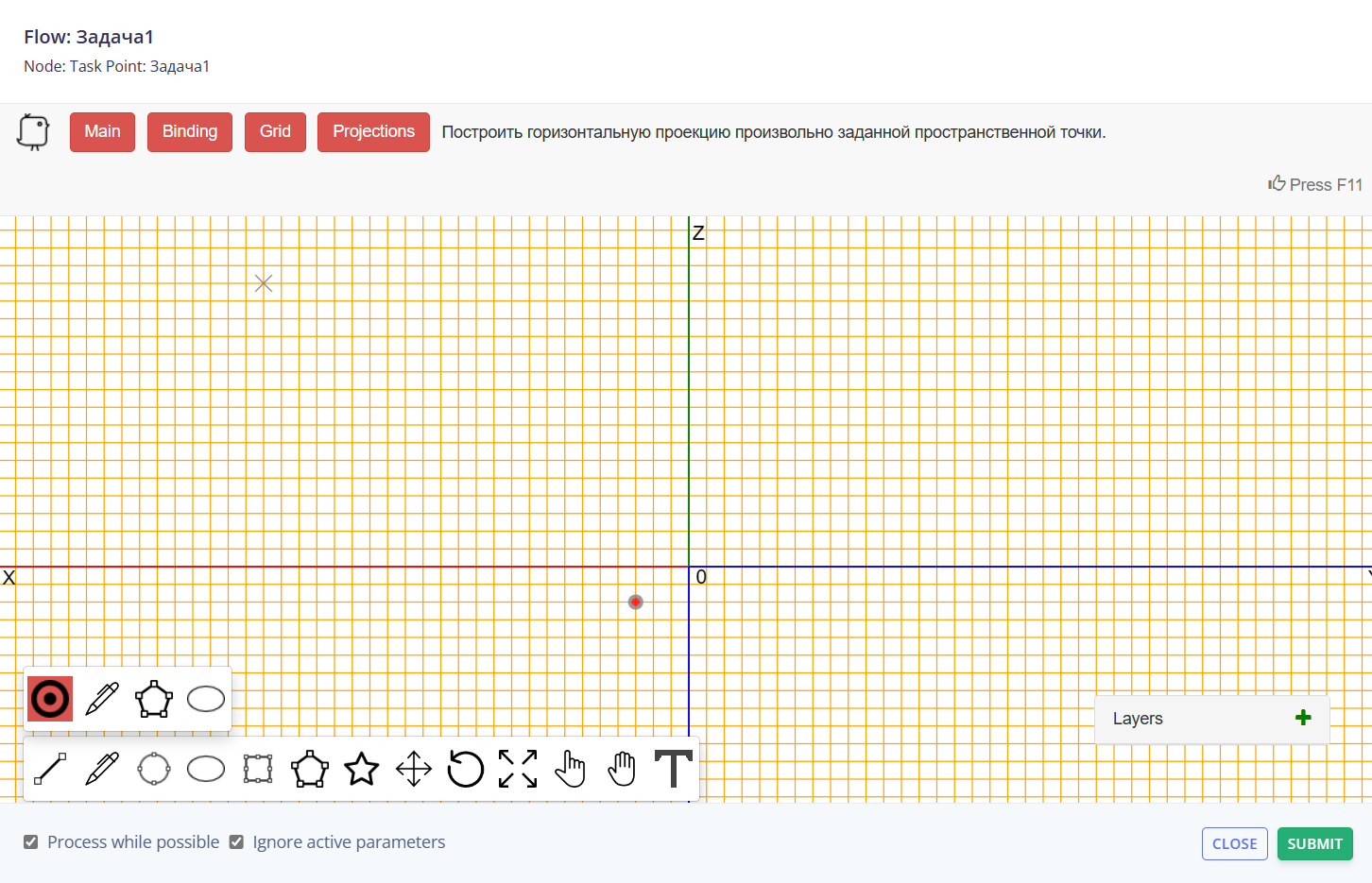


Рисунок 6.21 – Отображение узла графического конструктора

После выполнения задачи отображается поле с результатом, которое можно увидеть на рисунке 6.22.

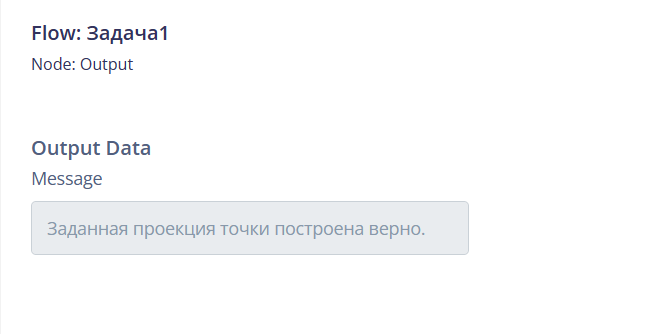


Рисунок 6.22 – Результат решения задачи

На основе приведенной выше задачи была реализована более сложная задача, схема которой продемонстрирована на рисунке 6.23.

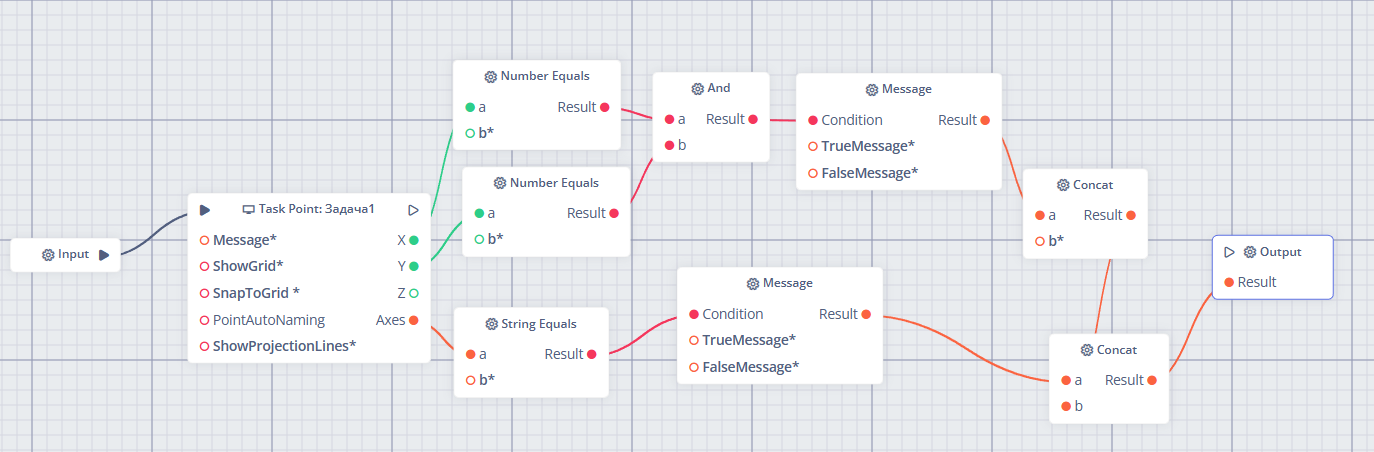


Рисунок 6.23 – Схема производной задачи

В данной задаче проверяется не только правильность выбранной плоскости для построения проекции точки, но также и координаты.

Процесс решения задачи изображен на рисунке 6.24.

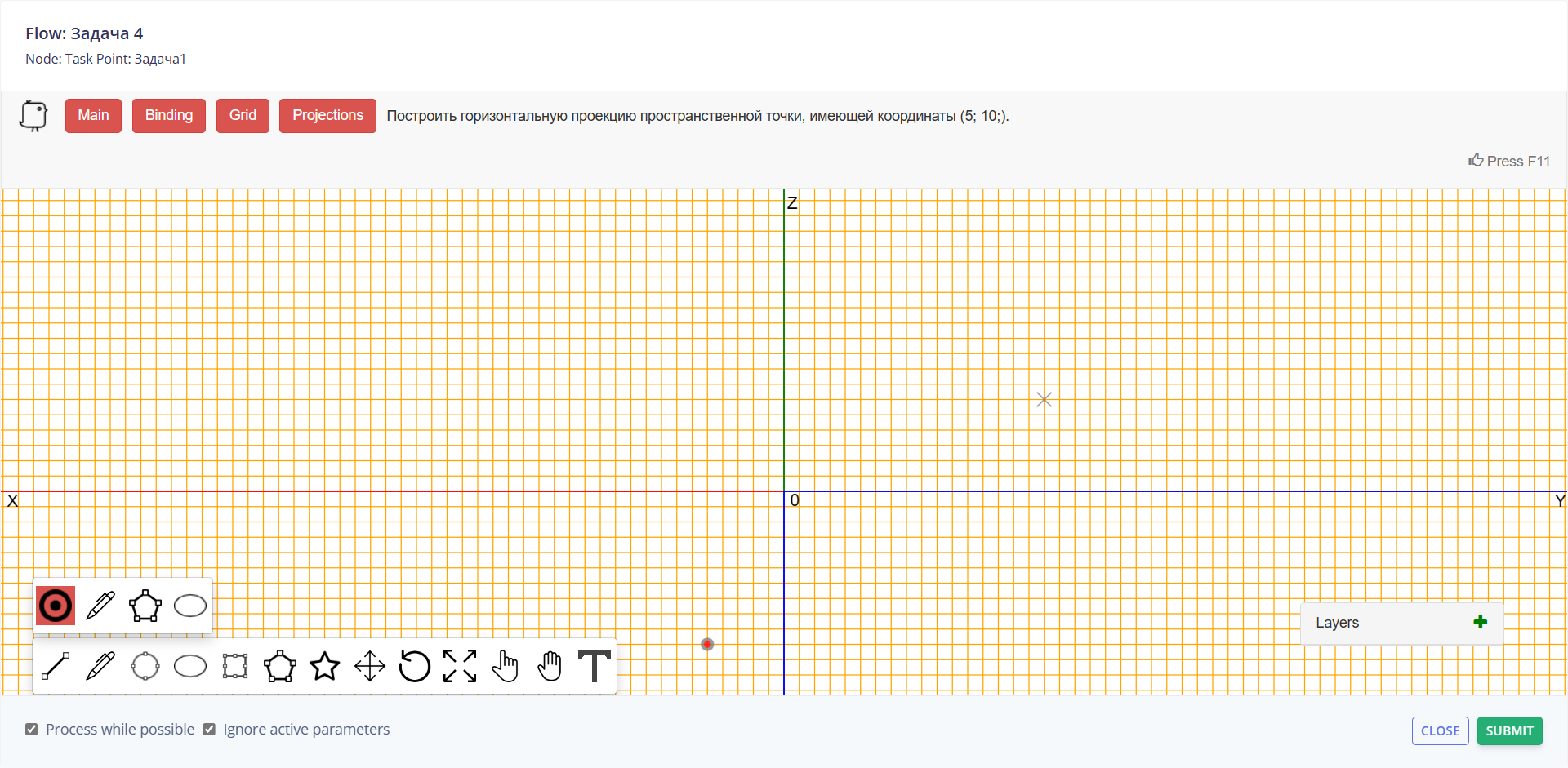


Рисунок 6.24 – Отображение узла графического редактора

На рисунке 6.25 изображен результат решения задачи.

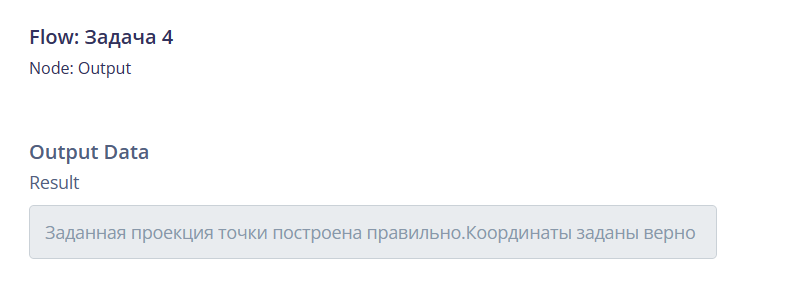


Рисунок 6.25 – Результат решения задачи

При использовании предыдущих примеров, а также с помощью написания нескольких аналогичных задач была реализована задача со следующим условием: построить горизонтальную и профильную проекции пространственной точки, заданной координатами (5; 10; 15). Блок-схема реализации данной задачи представлена на рисунке 6.26.

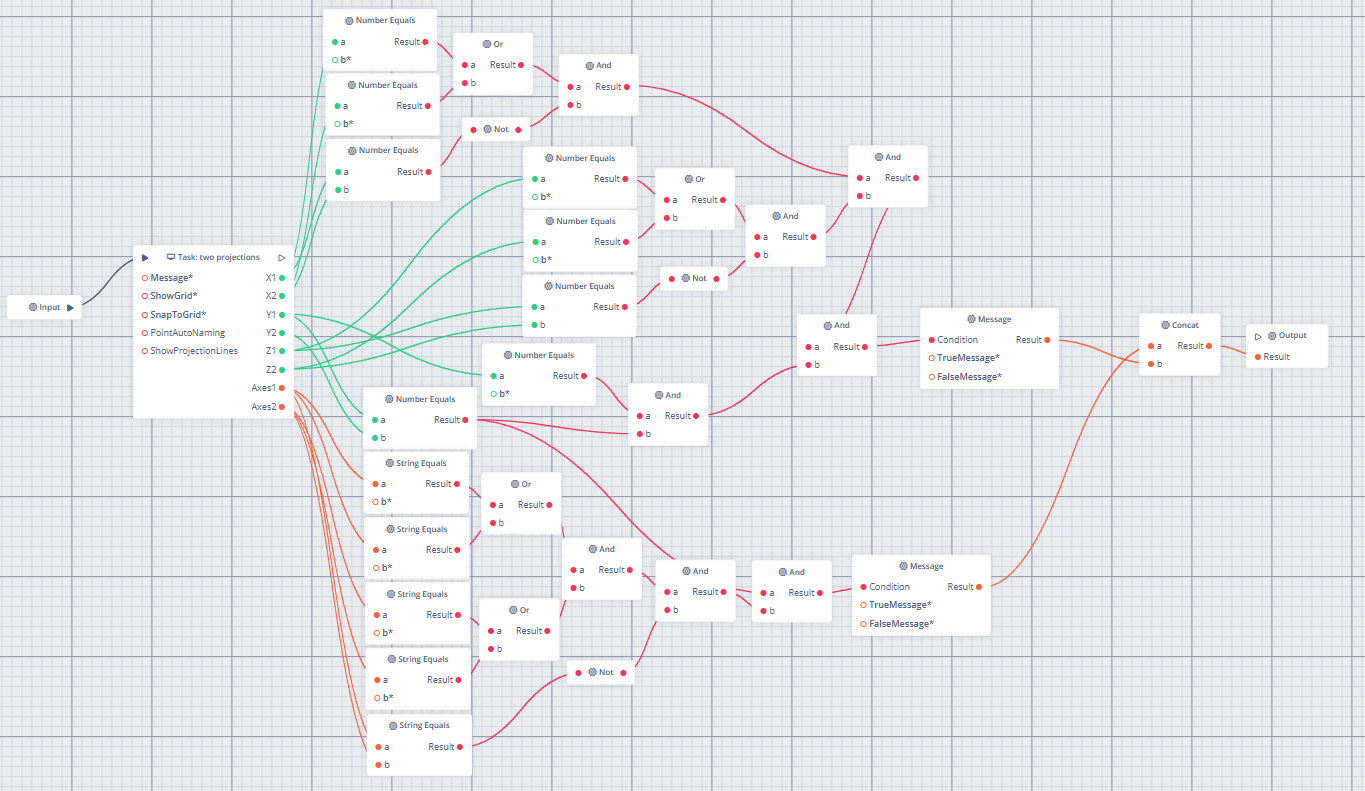


Рисунок 6.26 – Блок-схема реализации задачи

Процесс решения задачи изображен на рисунке 6.27.

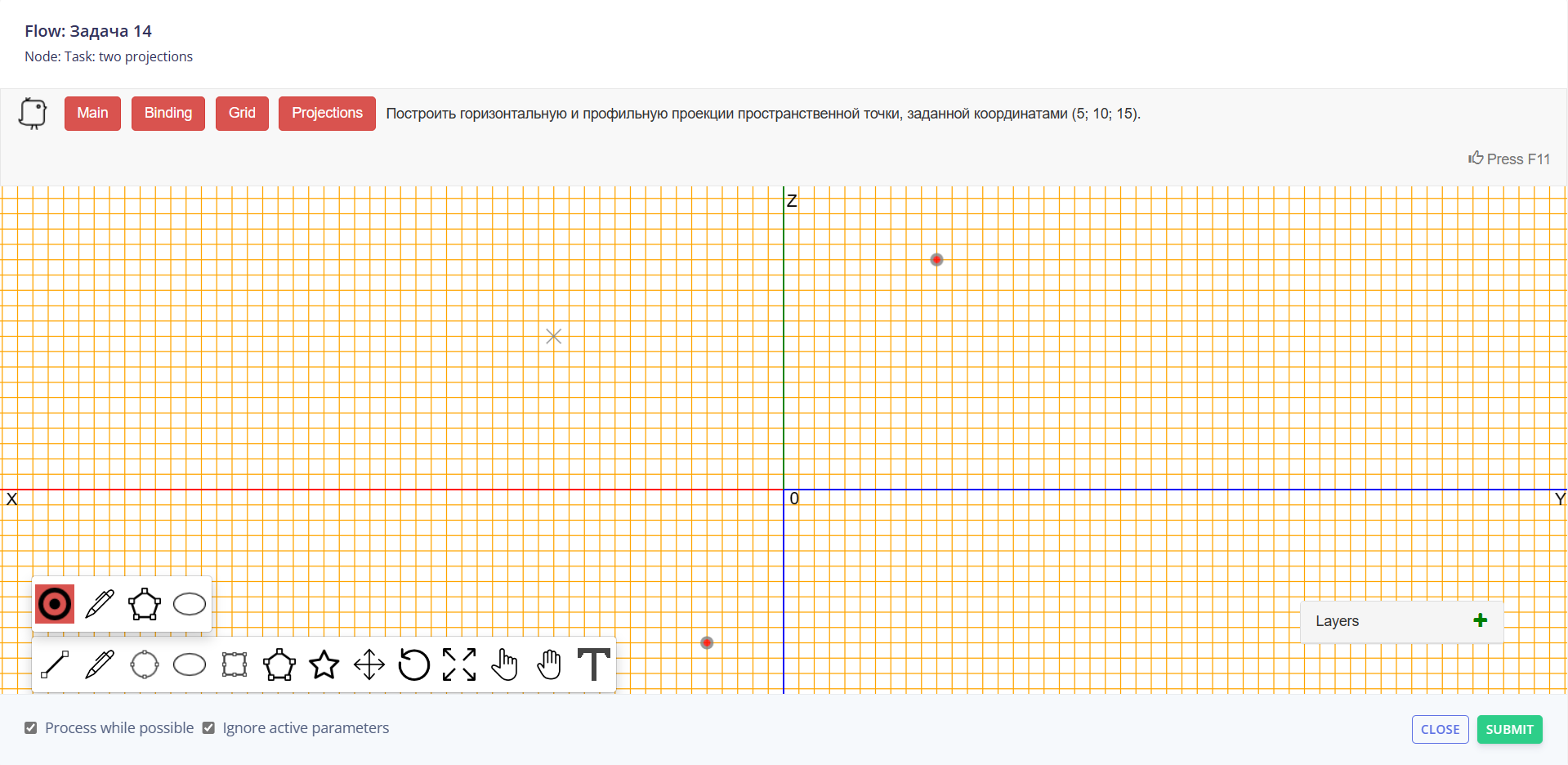


Рисунок 6.27 – Отображение узла графического редактора

На рисунке 6.28 изображен результат решения задачи.



Рисунок 6.28 – Результат решения задачи

Всего было реализовано и подготовлено к работе порядка 30 задач, полный список которых можно увидеть в таблице 6.1

Таблица 6.1 – Список реализованных задач

|  |  |  |
| --- | --- | --- |
| № задачи | Название задачи | Условие задачи |
| 1 | 2 | 3 |
|  | Задача1 | Построить горизонтальную проекцию произвольно заданной пространственной точки. |
|  | Задача1(reverse) | Для произвольно заданной пространственной точки построить проекцию, которая не является горизонтальной |
|  | Задача2 | Построить фронтальную проекцию произвольно заданной пространственной точки. |
|  | Задача2(reverse) | Для произвольно заданной пространственной точки построить проекцию, которая не является фронтальной. |
|  | Задача3 | Построить профильную проекцию произвольно заданной пространственной точки. |
|  | Задача3(reverse) | Для произвольно заданной пространственной точки построить проекцию, которая не является фронтальной. |
|  | Задача4 | Построить горизонтальную проекцию точки, заданную координатами (5;10). |
|  | Задача4(reverse) | Построить горизонтальную проекцию точки, координаты которой не имеют значения (5;10). |
|  | Задача5 | Построить фронтальную проекцию пространственной точки, имеющей координаты (10;15). |
|  | Задача5(reverse) | Построить фронтальную проекцию точки, координаты которой не имеют значения (10;15). |
|  | Задача6 | Построить профильную проекцию пространственной точки, имеющей координаты (5;15). |
|  | Задача6(reverse) | Построить профильную проекцию точки, координаты которой не имеют значения (5;15). |

Продолжение таблицы 6.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
|  | Задача7 | Построить горизонтальную проекцию пространственной точки, имеющей координаты (5;10;15). |
|  | Задача7(reverse) | Для пространственной точки, заданной координатами (5;10;15) построить проекцию, которая не является горизонтальной проекцией заданной точки. |
|  | Задача8 | Построить фронтальную проекцию пространственной точки, имеющей координаты (5;10;15). |
|  | Задача8(reverse) | Для пространственной точки, заданной координатами (5;10;15) построить проекцию, которая не является фронтальной проекцией заданной точки. |
|  | Задача9 | Построить профильную проекцию пространственной точки, имеющей координаты (5;10;15). |
|  | Задача9(reverse) | Для пространственной точки, заданной координатами (5;10;15) построить проекцию, которая не является профильной проекцией заданной точки. |
|  | Задача10 | Построить горизонтальную и фронтальную проекции произвольно заданной пространственной точки. |
|  | Задача11 | Построить горизонтальную и профильную проекции произвольно заданной пространственной точки. |
|  | Задача12 | Построить фронтальную и профильную проекции произвольно заданной пространственной точки. |
|  | Задача13 | Построить горизонтальную и фронтальную проекции пространственной точки, заданной координатами (5;10;15). |
|  | Задача13(reverse) | Построить горизонтальную и фронтальную проекции пространственной точки, координаты которой не имеют значения (5;10;15). |
|  | Задача14 | Построить горизонтальную и профильную проекции пространственной точки, заданной координатами (5;10;15). |
|  | Задача14(reverse) | Построить горизонтальную и профильную проекции пространственной точки, координаты которой не имеют значения (5;10;15). |
|  | Задача15 | Построить фронтальной и профильную проекции пространственной точки, заданной координатами (5;10;15). |
|  | Задача15(reverse) | Построить фронтальную и профильную проекции пространственной точки, координаты которой не имеют значения (5;10;15). |
|  | Задача16 | Построить три проекции произвольно заданной пространственной точки. |
|  | Задача17 | Построить три проекции пространственной точки, заданной координатами (5;10;15). |
|  | Задача17(reverse) | Построить три проекции пространственной точки, координаты которой не имеют значения (5;10;15). |

Таким образом была продемонстрирована работоспособность web-приложения на примере интеграции в систему внешнего интерфейса для конструирования графических задач. В результате разработки проекта было реализовано около 100 классов, а общий объем кода составляет до 9000 строк.

# ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

## Определение единовременных затрат на создание программного продукта (разработку модели)

При расчете экономической эффективности разработки программного обеспечения необходимо сопоставить затраты на решение задачи при ручном методе ее решения с затратами, связанными с ее автоматизацией. В том случае, если разрабатываемая задача внедряется взамен уже функционирующей или она представляет собой модификацию существующей задачи, необходимо осуществить сравнение затрат на создание и функционирование старой и новой задачи.

Единовременные капитальные затраты представляют собой цену программного продукта (ПП) или модели. Различают оптовую, отпускную и максимальную (сформированную доходным методом) цены. Все расчеты между покупателем и продавцом продукции, к числу которой относят и программные продукты (модели), производятся на основе отпускных цен (максимальная цена может быть приравнена к отпускной). В настоящее время в соответствии с законодательством РБ в отпускную цену наряду с оптовой ценой включается налог на добавленную стоимость.

Определяющим фактором цены разработки является трудоемкость создания ПП.

### Определение трудоемкости разработки ПП

Предусмотрено четыре степени новизны разрабатываемых задач:

* А – разработка задач, предусматривающая применение принципиально новых методов разработки, проведение научно-исследовательских работ (принципиально новые ПО, не имеющие подобных аналогов);
* Б – разработка типовых проектных решений, оригинальных задач и систем, не имеющих аналогов (принципиально новые ПО, не имеющие подобных аналогов);
* В – разработка проекта с использованием типовых проектных решений при условии их изменения; разработка проектов, имеющих аналогичные решения (ПО, являющиеся развитием определенного параметрического ряда ПО);
* Г – привязка типовых проектных решений (разработка ПО для ранее освоенных типов конфигурации ПК и ОС).

Сложность алгоритма представлена тремя группами:

1. алгоритмы оптимизации и моделирования систем и объектов;
2. алгоритмы учета, отчетности, статистики и поиска;
3. алгоритмы, реализующие стандартные методы решения, а также не предусматривающие применения сложных численных и логических методов.

В данном дипломном проекте ПП имеет степень новизны В и сложность алгоритма 1.

Трудоемкость разработки ПП может быть определена на основе типовых норм времени для программирования задач на ЭВМ. Она включает время на постановку задачи и время на программирование задачи и определяется по формуле:

|  |  |
| --- | --- |
|  | (7.1) |

где Тиа - трудоемкость подготовки описания задачи и исследования алгоритма решения;

Тбс - трудоемкость разработки блок-схемы алгоритма;

Тп - трудоемкость программирования по готовой блок-схеме;

Тотл -трудоемкость отладки программы на ЭВМ;

Тдр - трудоемкость подготовки документации по задаче в рукописи;

Тдо - трудоемкость редактирования, печати и оформления документации по задаче.

Составляющие приведенной формулы определяются, в свою очередь, через условное число операторов Q в разрабатываемом ПП по формуле:

|  |  |
| --- | --- |
|  | (7.2) |

где q − число операторов в программе (принято 3250);

С − коэффициент сложности программы (принято 1,26);

p − коэффициент коррекции программы в ходе ее разработки (принято 0,3).

Коэффициент сложности программы С характеризует относительную сложность программ задачи по отношению к так называемой типовой задаче, сложность которой принята за единицу. Значение коэффициента определяется из таблицы 7.1 на пересечении «группы сложности» и «степени новизны».

Коэффициент коррекции программ p характеризует увеличение объема работ за счет внесения изменений в алгоритм и программу, изменения состава и структуры информации, а также уточнений, вносимых разработчиком программы для улучшения ее качества без изменения постановки задачи. Значение p может быть принято равным 0,15...0,5 (принято 0,3).

Таблица 7.1 – Коэффициенты расчета трудоемкости

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Язык программирования | Группа сложности | Степень новизны | | | | | Коэффициент W |
| А | Б | В | Г |  | |
| Высокого уровня | 1 | 1,38 | 1,26 | 1,15 | 0,69 | 1,2 | |
| 2 | 1,30 | 1,19 | 1,08 | 0,65 | 1,35 | |
| 3 | 1,20 | 1,10 | 1,00 | 0,60 | 1,5 | |
| Низкого уровня | 1 | 1,58 | 1,45 | 1,32 | 0,79 | 1,2 | |
| 2 | 1,49 | 1,37 | 1,24 | 0,74 | 1,35 | |
| 3 | 1,38 | 1,26 | 1,15 | 0,69 | 1,5 | |

|  |  |
| --- | --- |
|  |  |

Составляющие трудоемкости разработки программы определятся по формулам:

|  |  |
| --- | --- |
|  | (7.3) |

|  |  |
| --- | --- |
|  | (7.4) |

|  |  |
| --- | --- |
|  | (7.5) |

|  |  |
| --- | --- |
|  | (7.6) |

|  |  |
| --- | --- |
|  | (7.7) |

|  |  |
| --- | --- |
| , | (7.8) |

где W – коэффициент увеличения затрат труда вследствие недостаточного или некачественного описания задачи (*W* = 1,2...1,5);

К – коэффициент квалификации разработчика алгоритмов и программ (при стаже работы до двух лет *К* = 0,8, при стаже от двух до трех лет *К* = 1,0, при стаже от трех до пяти лет *К* = 1,1...1,2, при стаже от пяти до семи лет *К* = 1,3...1,4, при стаже свыше семи лет *К* = 1,6...1,6).

(чел-час);

(чел-час);

(чел-час);

(чел-час);

(чел-час);

(чел-час).

Трудоемкость разработки ПП, следующая:

|  |  |
| --- | --- |
| Трз = 93,15 + 266,15 + 212,92 + 1064,6 + 354,86+ 266,1 = 2257,84 (чел-час). |  |

### Определение себестоимости создания ПП

Для определения себестоимости создания программного продукта необходимо определить:

1. Затраты на заработную плату разработчика по формуле:

|  |  |
| --- | --- |
|  | (7.9) |

где Трз – трудоемкость разработки программного продукта, чел-ч. (2257,84 чел/ч)

tчр – среднечасовая ставка работника, осуществлявшего разработку программного продукта, руб.;

kпр – коэффициент, учитывающий процент премий в организации-разработчике (принят 0,4);

а – коэффициент, учитывающий дополнительную заработную плату (0,15);

b – коэффициент, учитывающий отчисления от фонда заработной платы (0,346).

Среднечасовая ставка работника определяется исходя из Единой тарифной системы оплаты труда в Республике Беларусь по следующей формуле:

|  |  |
| --- | --- |
|  | (7.10) |

где 170 – среднее количество рабочих часов в месяце для 2023 года;

ТС – тарифная ставка работника, осуществлявшего создание программного продукта (принято 1250 руб).

Определим затраты на заработную плату разработчиков программы:

|  |  |
| --- | --- |
|  |  |
|  |  |

1. в себестоимость разработки веб-приложения включаются также затраты на отладку ПП в процессе его создания. Для определения их величины необходимо рассчитать стоимость машино-часа работы ЭВМ, на которой осуществлялась отладка. Данная величина соответствует величине арендной платы за один час работы ЭВМ.

Затраты на отладку программы определяются по формуле:

|  |  |
| --- | --- |
|  | (7.11) |

где Тотл – трудоемкость отладки программы, час (принято 12 ч);

Sмч – стоимость машино-часа работы ЭВМ, руб/ч.

Стоимость машино-часа работы ЭВМ определяется по формуле:

|  |  |
| --- | --- |
|  | (7.12) |

где Сэ - расходы на электроэнергию за час работы ЭВМ, руб;

Аэвм - годовая величина амортизационных отчислений на реновацию ЭВМ;

Рэвм - годовые затраты на ремонт и техническое обслуживание ЭВМ, руб;

Апл - годовая величина амортизационных отчислений на реновацию производственных площадей, занимаемых ЭВМ, руб;

Рпл - годовые затраты на ремонт и содержание производственных площадей, руб;

Рар - годовая величина арендных платежей за помещение, занимаемое ЭВМ, руб;

– амортизация нематериальных активов предприятия, руб;

Фэвм - годовой фонд времени работы ЭВМ, час.

Расходы на электроэнергию за час работы ЭВМ определяются по формуле:

|  |  |
| --- | --- |
|  | (7.13) |

где Чэл - среднечасовое потребление электроэнергии ЭВМ, кВт (0,135);

Цэ - стоимость 1 кВт-часа электроэнергии, руб (0,28666).

Годовой фонд времени работы ЭВМ определяется исходя из режима ее работы и может быть рассчитан по формуле:

|  |  |
| --- | --- |
|  | (7.14) |

где tсс – среднесуточная фактическая загрузка ЭВМ, час (принято 8);

Тсг – среднее количество дней работы ЭВМ в год (принято 250).

Годовая величина амортизационных отчислений на реновацию ЭВМ определяется по формуле:

|  |  |
| --- | --- |
|  | (7.15) |

где Цэвм - цена ЭВМ на момент ее выпуска, руб. (принято 1600 руб);

kу - коэффициент удорожания ЭВМ (принято 1);

kм - коэффициент, учитывающий затраты на транспортировку ЭВМ (1,05);

- норма амортизационных отчислений на ЭВМ, % (20);

- балансовая стоимость ЭВМ, руб.

Годовые затраты на ремонт и техническое обслуживание ЭВМ укрупненно могут быть определены по формуле:

|  |  |
| --- | --- |
|  | (7.16) |

где kро - коэффициент, учитывающий затраты на ремонт и техническое обслуживание ЭВМ, в том числе затраты на запчасти, зарплату ремонтного персонала и др. (kро = 0,13).

Годовая величина амортизационных отчислений на реновацию производственных площадей, занятых ЭВМ определяется по формуле:

|  |  |
| --- | --- |
|  | (7.17) |

где - балансовая стоимость площадей, руб.;

- норма амортизационных отчислений, % (1,2);

Sэвм - площадь, занимаемая ЭВМ, кв.м. (принято 1);

kд - коэффициент, учитывающий дополнительную площадь (kд = 3);

Цпл - цена 1 квадратного метра производственной площади, руб (1200).

Годовые затраты на ремонт и содержание производственных площадей укрупненно могут быть определены по формуле:

|  |  |
| --- | --- |
|  | (7.18) |

где kрэ - коэффициент, учитывающий затраты на ремонт и эксплуатацию производственных площадей (kрэ = 0,05).

Годовая величина арендных платежей за помещение, занимаемое ЭВМ, рассчитывается по формуле:

|  |  |
| --- | --- |
|  | (7.19) |

где Sэвм - площадь, занимаемая ЭВМ, кв.м (принято 1);

kд - коэффициент, учитывающий дополнительную площадь (kд = 3);

kар - ставка арендных платежей за помещение (принято 100);

kкомф – коэффициент комфортности помещения (принято 0,9);

kпов – повышающий коэффициент, учитывающий географическое размещение площади (принято 0,9).

Величина амортизационных отчислений нематериальных активов, подлежащая включению в себестоимость создания ПП пропорционально времени, приходящегося на его разработку, определяется по формуле:

|  |  |
| --- | --- |
|  | (7.20) |

где – норма амортизационных отчислений нематериальных активов, 20%;

– первоначальная стоимость нематериальных активов, зарегистрированных на предприятии, руб. (принято 1500 руб.).

|  |  |
| --- | --- |
|  |  |

Затраты на отладку программы:

|  |
| --- |
|  |

Себестоимость разработки ПП определяется по формуле:

|  |  |
| --- | --- |
|  | (7.21) |

где F - коэффициент накладных расходов проектной организации без учета эксплуатации ЭВМ (принято 1,15).

### Определение минимальной цены ПП

Минимальная (оптовая) цена складывается из себестоимости создания программного продукта и плановой прибыли на программу.

Оптовая цена ПП определяется по формуле:

|  |  |
| --- | --- |
|  | (7.22) |

где Пр - плановая прибыль на программу, руб.

Плановая прибыль на программу определяется по формуле:

|  |  |
| --- | --- |
|  | (7.23) |

где Спр - себестоимость программы;

Нп - норма прибыли проектной организации (принято 0,25).

Тогда:

|  |  |
| --- | --- |
|  |  |

## Определение ожидаемого прироста прибыли в результате внедрения

Внедрение ПП может обеспечить пользователю ожидаемый прирост прибыли за счет сокращения трудоемкости решения задачи, являющейся предметом автоматизации и, как результат, снижения текущих затрат, связанных с решением данной задачи.

В том случае, если внедряемый ПП заменяет ручной труд, то производится сопоставление текущих затрат, связанных с решением задачи в ручном режиме и автоматизированном. В том случае, если разрабатываемая задача внедряется взамен уже функционирующей или она представляет собой модификацию существующей задачи, необходимо осуществить сравнение затрат на создание и функционирование старой и новой задачи.

### Определение годовых эксплуатационных расходов при ручном решении задачи

Годовые эксплуатационные расходы при ручной обработке информации (ручном решении задачи) определяются по формуле:

|  |  |
| --- | --- |
|  | (7.24) |

где Тр - трудоемкость разового решения задачи вручную, чел-час (2);

к - периодичность решения задачи в течение года, раз/год (1000);

tчр - среднечасовая ставка работника, осуществляющего ручной расчет, руб;

kпр - коэффициент, учитывающий процент премий (0,3);

а - коэффициент, учитывающий дополнительную заработную плату (0,15);

b - коэффициент, учитывающий отчисления от фонда заработной платы (0,346).

|  |  |
| --- | --- |
|  | (7.25) |

где 170 – среднее нормативное количество рабочих часов в месяце для 2023 года;

ТС – тарифная ставка работника, осуществлявшего создание программного продукта (принято 650 руб).

Значение трудоемкости разового решения задачи вручную было выбрано равным 2 чел-час, т.к. в данное значение входит формулирование условий тестовой задачи, выбор алгоритма решения, построение графиков и самостоятельное решение тестовой задачи для получения эталона ответа, повторное решение тестовой задачи коллегами с целью удостовериться в правильности эталона и готовности задачи к использованию в учебном процессе. Периодичность решения задачи в течении года зависит непосредственно от нужд потребителя. Можно говорить о том, что периодичность решения задачи в год составляет порядка 1000 раз.

|  |
| --- |
|  |

### Определение годовых текущих затрат, связанных с эксплуатацией задачи

Для расчета годовых текущих затрат, связанных с эксплуатацией ПП, необходимо определить время решения данной задачи на ЭВМ. Данное время состоит из времени построения блок-схемы алгоритма решения тестовой задачи (15 минут), времени ввода исходных данных в ЭВМ (5 минут), времени вычислений (1 минута). В результате время решения задачи на ЭВМ будет равно:

|  |  |
| --- | --- |
|  |  |

На основе рассчитанного времени решения задачи может быть определена заработная плата пользователя данного ПП. Затраты на заработную плату пользователя ПП определяются по формуле:

|  |  |
| --- | --- |
|  | (7.26) |

где Тз - время решения задачи на ЭВМ, час (0,35);

к - периодичность решения задачи в течение года, раз/год (1000);

tчп, kпр, a, b определяются аналогично ставке работника, осуществляющего ручной расчет.

|  |  |
| --- | --- |
| (руб). |  |

В состав затрат, связанных с решением задачи, включаются также затраты, связанные с эксплуатацией ЭВМ.

Затраты на оплату аренды ЭВМ для решения задачи определяются по следующей формуле:

|  |  |
| --- | --- |
|  | (7.27) |

где Sмч - стоимость одного машино-часа работы ЭВМ, которая будет использоваться для решения задачи, руб.

Годовые текущие затраты, связанные с эксплуатацией задачи, определяются по формуле:

|  |  |
| --- | --- |
|  | (7.28) |

где Зп - затраты на заработную плату пользователя программы;

За - затраты на оплату аренды ЭВМ при решении задачи.

### Определение дополнительной прибыли пользователя за период использования ПП

Дополнительная прибыль пользователя за период использования ПП укрупненно может быть определен по формуле:

|  |  |
| --- | --- |
|  | (7.29) |

где Снп – коэффициент ставки налога на прибыль (принято 0,2).

|  |  |
| --- | --- |
|  |  |

### Определение максимальной, планируемой и отпускной цены

Для определения цены доходным методом необходимо привести экономию затрат пользователя за планируемый период использования ПП. Планируемая цена ПП (Цпл) должна находиться в интервале между минимальной (оптовой) и максимальной ценой. Планируемая цена была принята 55000 руб. в соответствии с функционалом и качеством на этапе сдачи ПП.

Отпускная цена программы определяется по формуле:

|  |  |
| --- | --- |
|  | (7.30) |

где НДС - ставка налога на добавленную стоимость (принято 0,2).

## Расчет показателей эффективности использования программного продукта

В дипломном проекте для оценки эффективности использования веб-приложения используется статический (абсолютный) метод оценки, который предполагает расчет следующих показателей:

Рентабельность авансированного капитала (Rак) – это годовой процент прибыли, который принесут инвестиции при реализации проекта, определяется по формуле:

|  |  |
| --- | --- |
|  | (7.31) |

где – суммарные инвестиционные затраты, руб.

Суммарные инвестиционные затраты на разработку и внедрения ПП определяются по формуле:

|  |  |
| --- | --- |
|  | (7.32) |

где Из – инвестиционные и приравненные к ним затраты;

Цпл – планируемая цена программы (принято 55000 руб).

Инвестиционные и приравненные к ним затраты определяются по формуле:

|  |  |
| --- | --- |
|  | (7.33) |

где - балансовая стоимость комплекта вычислительной техники, необходимого для решения задачи.

|  |
| --- |
|  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

Срок возврата инвестиций определяется по формуле:

|  |  |
| --- | --- |
|  | (7.34) |

где – сумма начисленной амортизации по активам предприятия, руб (700).

Годовой экономический эффект, полученный от внедрения в эксплуатацию веб-приложения определяется по формуле:

|  |  |
| --- | --- |
|  | (7.35) |

## Оценка конкурентоспособности объекта проектирования

Успешность реализации проекта напрямую зависит от конкурентоспособности объекта проектирования. Под конкурентоспособностью понимается комплекс потребительских и стоимостных (ценовых) характеристик товара, определяющих его успех на рынке.

Для оценки конкурентоспособности разрабатываемого ПП проводится анализ и сравнение разрабатываемого ПП с выбранным аналогом по техническим параметрам: функциональное назначение, основные технические и эксплуатационные показатели. Наименование и количество показателей, по которым проводится оценка конкурентоспособности выбирается в зависимости от конкретного объекта проектирования и согласовывается с руководителем проекта. Приводимые показатели должны всесторонне охарактеризовать объект проектирования.

Для оценки качества ПП, в соответствии со стандартом ISO/МЭК 9126:1991, выделяются шесть характеристик: функциональные возможности, надежность, практичность, эффективность, сопровождаемость, мобильность. Данные характеристики представлены на рисунке 7.1.

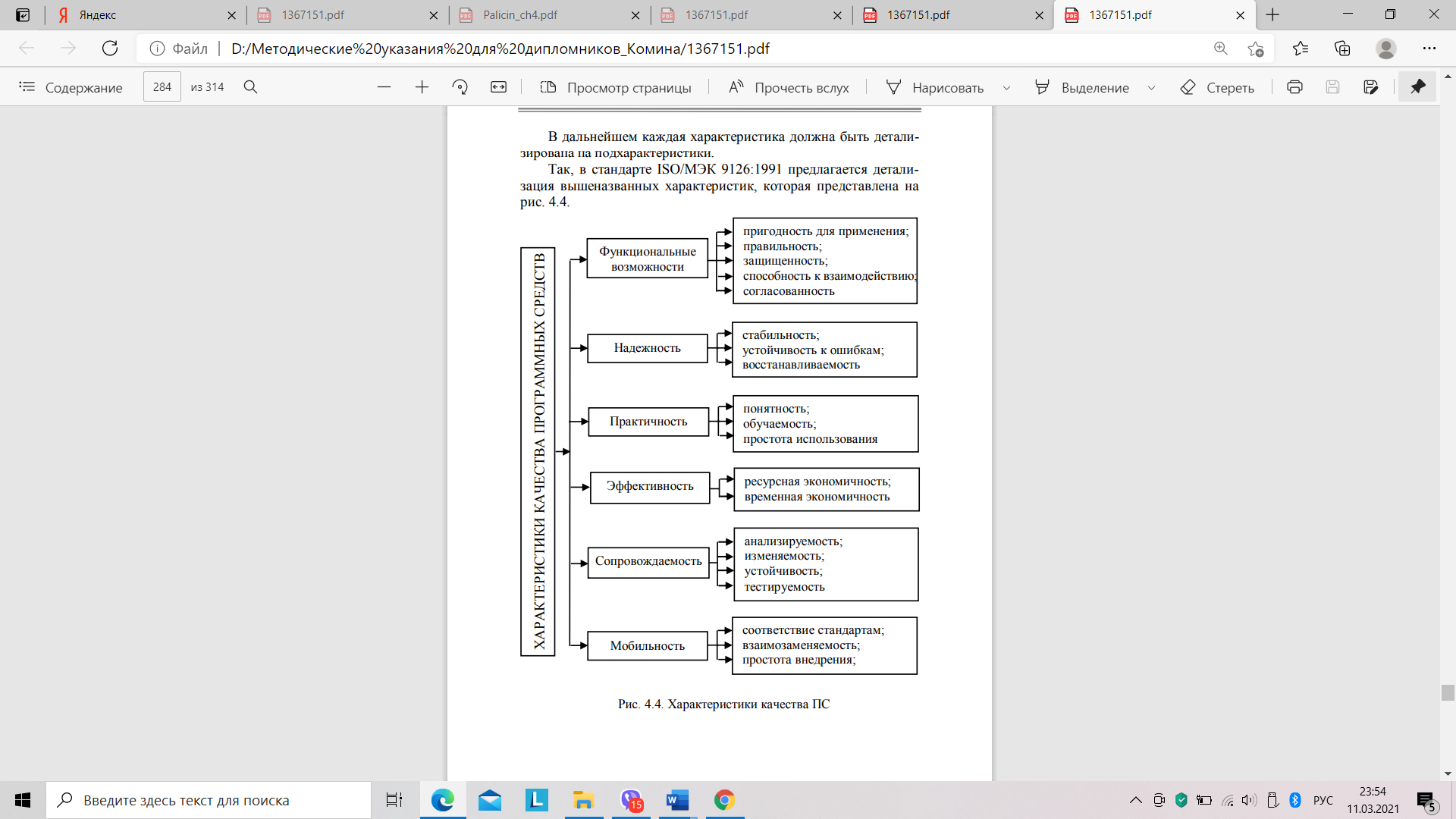


Рисунок 7.1 – Характеристика качества ПП

Функциональные возможности – набор атрибутов, относящихся к сути набора реализуемых в ПП функций и их конкретным свойствам (установленные или предполагаемые потребности).

Надежность – набор атрибутов, относящихся к способности ПО сохранять свой уровень качества функционирования при установленных условиях за установленный период времени.

Практичность – набор атрибутов, относящихся к объему работ, требуемых для использования и индивидуальной оценки такого использования определенным кругом пользователей.

Эффективность – набор атрибутов, относящихся к соотношению между уровнем качества функционирования программного обеспечения и объемом используемых ресурсов при установленном уровне качества.

Сопровождаемость – набор атрибутов, относящихся к объему работ, требуемых для проведения конкретных изменений (модификаций).

Мобильность – набор атрибутов, относящихся к способности ПО быть перенесенным из одного окружения в другое.

Далее осуществляется оценка соответствия нормативным показателям. В случае если объект не соответствует нормативным требованиям (например, требованиям патентной чистоты и т.п.), то он признается неконкурентоспособным, поскольку даже в случае превосходства по всем остальным показателям он не может быть реализован на рассматриваемом рынке в силу законодательных ограничений. Если нормативные требования соблюдаются, то проводится дальнейший анализ конкурентоспособности по экономическим параметрам, в частности по цене потребления.

Данные, которые формируются на основе проведенного исследования, используюся для расчета показателей конкурентоспособности разрабатываемого ПП, представлены в таблице 7.2.

Таблица 7.2 – Аналитическая таблица оценки конкурентоспособности объекта проектирования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование показателя | Весомость  показателя, ai | Вариант оценки, балл | | | |
| базовый | | проектный | |
| Технические параметры | | | | | |
| Функциональность | 0,3 | 5 | 7 | | |
| Новизна (соответствие современным требованиям) | 0,4 | 6 | 7 | | |
| Надежность (защита данных) | 0,2 | 6 | 6 | | |
| Скорость доступа к данным | 0,1 | 5 | 4 | | |
| Гибкость настройки | 0,2 | 5 | 8 | | |
| Удобство использования | 0,3 | 9 | 9 | | |
| Нормативные параметры | | | | | |
| Соответствие требованиям патентной чистоты | | 1 | | | 1 |
| Экономические параметры | | | | | |
| Суммарные инвестиционные затраты, руб. | | 61348,53 | | | 55294 |
| Среднегодовые эксплуатационные расходы, руб. | | 4951,57 | | | 3456,55 |
| Цена потребления (при сроке службы 3 года), руб. | | 14854,71 | | | 10369,65 |

Индивидуальные индексы по техническим показателям определяются по формуле:

|  |  |
| --- | --- |
|  | (7.36) |

где Pi, Pi0 – величина i-го технического показателя (если увеличение показателя ведет к улучшению качества, то используется первая из вышеприведенных формул, в обратном случае – вторая).

Групповой индекс по техническим показателям определяется по формуле:

|  |  |
| --- | --- |
|  | (7.37) |

где – единичный индекс по i-му техническому показателю;

– весомость i-го технического показателя, определяемая путем экспертных оценок (Σаi=1);

n – количество технических показателей, подлежащих оценке (6 показателей).

Групповой индекс по экономическим показателям определяется по формуле:

|  |  |
| --- | --- |
|  | (7.38) |

где Ц, Ц0 – цена соответственно проектируемого и базового варианта (уже существующего подобного аналога на рынке), руб.;

Ci, Ci0 –суммарные эксплуатационные (текущие) расходы, относящиеся к i-му году службы, соответственно оцениваемого и базового образцов;

Т – срок службы.

Расчет интегрального показателя конкурентоспособности ведется по формуле:

|  |  |
| --- | --- |
| где - групповой индекс по техническим показателям;  - групповой индекс по техническим показателям. | (7.39) |

Результаты расчета единичных и групповых показателей были сведены в таблицу 7.3.

Таблица 7.3 – Результаты расчета единичных и групповых показателей конкурентоспособности

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование  показателя | Весомость  показателя, ai | Единичный  показатель | Единичный показатель с учетом весомости |
| qi = Pi/ Pi0 | qi · ai |
| Технические параметры | | | |
| Функциональность | 0,3 | 7/5=1,4 | 0,42 |
| Новизна (соответствие современным требованиям) | 0,4 | 7/6=1,167 | 0,467 |
| Надежность (защита данных) | 0,2 | 6/6=1 | 0,2 |
| Скорость доступа к данным | 0,1 | 4/5=0,8 | 0,08 |
| Гибкость настройки | 0,2 | 8/5=1,6 | 0,32 |
| Удобство использования | 0,3 | 9/9=1 | 0,03 |
| Групповой показатель по техническим параметрам Iтех | | | 1,584 |
| Групповой показатель по нормативным паромерам Iнорм | | | 1 |
| Групповой показатель по экономическим параметрам Iэк | | | 0,861 |
| Комплексный (интегральный) показатель К | | | 1,839 |

К категории наиболее удобных способов оценки эксперты относят радар конкурентоспособности, который дает возможность сравнить и визуально представить широкий спектр ключевых свойств продукта.

По результатам проведенного расчета конкурентоспособности ПП с целью графической иллюстрации был построен радар конкурентоспособности, который продемонстрирован на рисунке 7.2.

Рисунок 7.2 – Радар конкурентоспособности ПП

Все технико-экономические показатели были сведены в таблицу 7.4.

Таблица 7.4 – Технико-экономические показатели проекта

|  |  |  |
| --- | --- | --- |
| Наименование показателя | Варианты | |
| Базовый | Проектный |
| 1. Трудоемкость решения задачи, час | 2 | 0,35 |
| 2. Периодичность решения задачи, раз/год | 1000 | 1000 |
| 3. Годовые текущие затраты, связанные с решением задачи, руб | 15373,74 | 2690,40 |
| 4. Планируемая цена программы, руб | - | 55000 |
| 5. Степень новизны программы | - | В |
| 6. Группа сложности алгоритма | - | 1 |
| 7. Экономия затрат на ЗП проектировщика за счет автоматизации труда, руб. | 12682,8 | |
| 8. Экономия отчислений в социальные фонды, руб. | 4388,24 | |
| 9. Дополнительная прибыль пользователя, руб |  | |
| 10. Рентабельность авансированного капитала, % | 17,24 % | |
| 11. Срок возврата инвестиций, лет | 5,4 лет | |
| 12. Годовой экономический эффект, руб. |  | |
| 13.Групповой показатель конкурентоспособности ПП по техническим параметрам | 1,584 | |
| 14.Групповой коэффициент конкурентоспособности ПП по экономическим параметрам | 0,861 | |
| 15. Комплексный (интегральный) показатель конкурентоспособности ПП | 1,839 | |

# ОХРАНА ТРУДА

## Производственная санитария, техника безопасности и пожарная профилактика

При работе с ПЭВМ пользователи могут подвергаться воздействию опасных и вредных производственных факторов, таких как повышенные уровни: электромагнитного, ультрафиолетового и инфракрасного излучения; статического электричества; запыленность воздуха рабочей зоны; повышенное или пониженное содержание аэроионов в воздухе рабочей зоны; повышенный или пониженный уровень освещенности рабочей зоны, содержание в воздухе рабочей зоны вредных веществ (оксид углерода, озон, аммиак, фенол, формальдегид); напряжение зрения, памяти, внимания; длительное статическое напряжение; монотонность труда; нерациональная организация рабочего места; эмоциональные перегрузки.

Работа пользователей ПЭВМ проводится в соответствии с Санитарными нормами и правилами «Требования при работе с видеодисплейными терминалами и электронно-вычислительными машинами» и Гигиеническим нормативом «Предельно допустимые уровни нормируемых параметров при работе с видеодисплейными терминалами и электронно-вычислительными машинами» (постановление Министерства здравоохранения от 28.06.2013 г. № 59), Гигиеническим нормативом «Показатели безопасности и безвредности факторов производственной среды и трудового процесса при работе с видеодисплейными терминалами и электронно-вычислительными машинами» (постановление Совета Министров Республики Беларусь от 25.01.2021 г. № 37) и Типовой инструкцией по охране труда при использовании в работе офисного оборудования, утвержденной постановлением Министерства труда и социальной защиты от 14.04.2021 № 25.

Площадь помещения на одного пользователя ПЭВМ на базе плоских дискретных экранов (жидкокристаллические, плазменные) составляет не менее 4,5 м2.

### Метеоусловия

Метеоусловия (или климатические) определяются в основном следующими факторами атмосферы: температурой, относительной влажностью, скоростью движения воздуха и тепловым излучением. Эти факторы определяют погоду (на открытом воздухе) или микроклимат на рабочих местах и в производственных помещениях. В производственных помещениях, в которых работа с использованием ПЭВМ является основной (операторские, расчетные, посты управления, залы вычислительной техники), обеспечиваются оптимальные параметры микроклимата для категории работ 1а и 1б, смотреть в таблице 8.1.

Таблица 8.1 – Оптимальные параметры микроклимата для помещений с ВДТ, ЭВМ и ПЭВМ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Период года | Категория работ | Температура воздуха, оС, не более | Относительная влажность воздуха, % | Скорость движения воздуха, м/с |
| Холодный | 1а | 22-24 | 40-60 | 0,1 |
| 1б | 21-23 | 40-60 | 0,1 |
| Теплый | la | 23-25 | 40-60 | 0,1 |
| 1б | 22-24 | 40-60 | 0,2 |

Работа с компьютером относится к категории 1а (работы, производимые сидя и сопровождающиеся незначительным физическим напряжением, при которых расход энергии составляет до 120 ккал/ч, т.е. до 139 Вт).

Интенсивность теплового излучения работающих от нагретых поверхностей технологического оборудования, осветительных приборов, инсоляции на постоянных рабочих местах не превышает значений, указанных в таблице 8.2.

Таблица 8.2– Предельно допустимые уровни интенсивности излучения в инфракрасном и видимом диапазоне излучения на расстоянии 0,5 м со стороны экрана ВДТ, ЭВМ и ПЭВМ

|  |  |  |  |
| --- | --- | --- | --- |
| Диапазоны длин волн | 400-760 нм | 760-1050 нм | свыше 1050 нм |
| Предельно допустимые уровни | 0,1 Вт/м2 | 0,05 Вт/м2 | 4,0 Вт/м2 |

Для создания нормальных метеорологических условий наиболее целесообразно уменьшить тепловыделения от самого источника — монитора, что предусматривается при разработке его конструкции.

В производственных помещениях для обеспечения необходимых показателей микроклимата предусмотрены системы отопления, вентиляции и кондиционирования воздуха.

### Вентиляция и отопление

Воздух рабочей зоны помещения соответствует санитарно-гигиеническим требованиям по содержанию вредных веществ и частиц пыли, приведенным в Санитарных нормах и правилах «Требованию к контролю воздуха рабочей зоны», Гигиеническом нормативе «Предельно допустимые концентрации вредных веществ в воздухе рабочей зоны», утвержденным постановлением Министерства здравоохранения от 10.10.2017 г. № 92.

В помещениях, проводится ежедневная влажная уборка и систематическое проветривание после каждого часа работы.

Уровни положительных и отрицательных аэроионов, а также коэффициент униполярности в воздухе всех помещений, где расположены ПЭВМ, соответствуют значениям, указанным в таблице 8.3.

Таблица 8.3 – Аэроионный состав воздуха в зоне дыхания производственных и общественных помещений

|  |  |  |
| --- | --- | --- |
| Уровни аэроионизации | Концентрация аэроионов,  ρ (ион/см3) | Коэффициент  униполярности, У |
| Минимально допустимые | ρ «+» ≥ 400 ρ «–» ≥ 600 | 0,4 < У< 1,0 |
| Максимально допустимые | ρ«+» < 50 000 ρ«–» < 50 000 |

Одним из мероприятий по оздоровлению воздушной среды является устройство вентиляции и отопления. Задачей вентиляции является обеспечение чистоты воздуха и параметров метеорологических условий на рабочих местах. Чистота воздушной среды достигается удалением загрязненного или нагретого воздуха из помещения и подачей в него свежего воздуха. Для поддержания нормального микроклимата необходим достаточный объем вентиляции, для чего предусматривается кондиционирование воздуха, осуществляющее поддержание постоянных параметров микроклимата в помещении независимо от наружных условий.

Параметры микроклимата поддерживаются в холодный период года за счет системы водяного отопления с нагревом воды до 100°С, а в теплый - за счет кондиционирования с параметрами, отвечающими требованиям СН 4.02.03-2019.

### Освещение

В помещении при использовании ПЭВМ предусмотрены естественное и искусственное освещение. Естественное освещение на рабочих местах осуществляется через световые проемы, ориентированные преимущественно на север, северо-восток, восток, запад или северо-запад и обеспечивает коэффициент естественной освещенности не ниже 1,5 %. Оконные проемы оборудованы регулируемыми устройствами типа жалюзи, занавесей.

Для отделки интерьера помещений используются материалы с коэффициентом отражения для потолка – 0,7- 0,8; для стен – 0,5- 0,6; для пола – 0,3- 0,5.

Искусственное освещение в помещениях осуществляется системой общего равномерного освещения. При работе с документами применяется система комбинированного освещения, а освещенность поверхности стола в зоне размещения рабочего документа составляет 300-500 люкс. Освещенность поверхности экрана не более 300 люкс согласно СН 2.04.03-2020. В качестве источников света применяем люминесцентные лампы типа ЛБ. Коэффициент запаса для осветительных установок общего освещения принимается равным 1,4, коэффициент пульсации – не более 5 %.

### Шум

Источниками шума в помещениях, оборудованных ПЭВМ, являются принтеры, множительная техника и оборудование для кондиционирования воздуха, в самих ПЭВМ — вентиляторы систем охлаждения и трансформаторы.

В таблице 8.4 приведены допустимые уровни шума, которые обеспечиваются за счет использования малошумного оборудования, применения звукопоглощающих материалов для облицовки помещений, а также различных звукопоглощающих устройств (перегородки и т. п.).

Таблица 8.4 – Предельно-допустимые уровни звука, эквивалентные уровни звука и уровни звукового давления в октавных полосах частот при работе с ВДТ, ЭВМ и ПЭВМ и периферийными устройствами

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Категория нормы  шума | Уровни звукового давления, дБ, в октавных полосах  со среднегеометрическими частотами, Гц | | | | | | | | | Уровни звука и  эквивалентные уровни звука, дБА |
| 31.5 | 63 | 125 | 250 | 500 | 1000 | 2000 | 4000 | 8000 |
| I | 86 | 71 | 61 | 54 | 49 | 45 | 42 | 40 | 38 | 50 |
| II | 93 | 79 | 70 | 63 | 58 | 55 | 52 | 50 | 49 | 60 |
| III | 96 | 83 | 74 | 68 | 63 | 60 | 57 | 55 | 54 | 65 |
| IV | 103 | 91 | 83 | 77 | 73 | 70 | 68 | 66 | 64 | 75 |

### Электробезопасность

Помещение вычислительного центра по степени опасности поражения электрическим током относится к помещениям без повышенной опасности.

Основные меры защиты от поражения током:

- изоляция и недоступность токоведущих частей;

- защитное заземление (R3 = 4 Ом ГОСТ 12.1.030 - 81).

Первая помощь при поражениях электрическим током состоит из двух этапов: освобождение пострадавшего от действия тока и оказание ему доврачебной медицинской помощи. После освобождения пострадавшего от действия электрического тока необходимо оценить его состояние. Во всех случаях поражения электрическим током необходимо вызвать врача независимо от состояния пострадавшего.

### Излучение

При работе с дисплеем ПЭВМ могут возникать следующие опасные факторы: электромагнитные и электростатические поля, ультрафиолетовое и инфракрасное излучение.

Уровни физических факторов на рабочих местах пользователей, создаваемые ПЭВМ и периферийными устройствами, не превышают предельно-допустимые уровни.

Допустимые уровни электромагнитных полей приведены в таблице 8.5.

Таблица 8.5 –*.*Предельно допустимые уровни электромагнитных полей от экранов ВДТ, ЭВМ и ПЭВМ

|  |  |  |
| --- | --- | --- |
| Наименование параметра | | Предельно-допустимые уровни |
| Напряженность электрического поля в диапазоне частот: | |  |
|  | 5 Гц-2 кГц | не более 25,0 В/м |
|  | 2-400 кГц | не более 2,5 В/м |
| Плотность магнитного потока магнитного поля в диапазоне частот: | |  |
|  | 5 Гц-2 кГц | не более 250 нТл |
|  | 2-400 кГц | не более 25 нТл |
| Напряженность электростатического поля | | не более 15 кВ/м |

Допустимые уровни электромагнитных полей при работе с ВДТ, ЭВМ, ПЭВМ приведены в таблице 8.6.

Таблица 8.6 – Предельно допустимые уровни электромагнитных полей при работе с ВДТ, ЭВМ, ПЭВМ от клавиатуры, системного блока, манипулятора «мышь», беспроводных системам передачи информации и иных периферийных устройств

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Диапазоны частот | 0,3-300  кГц | 0,3-3  МГц | 3-30  МГц | 30-300  МГц | 0,3-300  ГГц |
| Предельно допустимые уровни | 25 В/м | 15 В/м | 10 В/м | 3 В/м | 10 мкВт/см2 |

Предельно допустимые уровни интенсивности излучения в ультрафиолетовом диапазоне указаны в таблице 8.7.

Таблица 8.7 – Предельно допустимые уровни интенсивности излучения в ультрафиолетовом диапазоне на расстоянии 0,5 м со стороны экрана ВДТ, ЭВМ и ПЭВМ

|  |  |  |  |
| --- | --- | --- | --- |
| Диапазоны длин волн | 200-280 нм | 280-315 нм | 315-400 нм |
| Предельно допустимые уровни | не допускается | 0,0001 Вт/м2 | 0,1 Вт/м2 |

Наиболее эффективным и часто применяемым методом защиты от электромагнитных излучений является установка экранов. Экранируют либо источник излучения, либо рабочее место. Часто экран устанавливают непосредственно на монитор.

При работе монитора на экране кинескопа накапливается электростатический заряд, создающий электростатическое поле. При этом персонал, работающий с монитором, приобретают электростатический потенциал. Заметный вклад в общее электростатическое поле вносят электризующиеся от трения поверхности клавиатуры и мыши.

### Пожарная безопасность

По взрывопожарной и пожарной опасности помещения и здания для ЭВМ относятся к категории Д согласно ТКП 474-2013. Здания для ВЦ и части зданий другого назначения, в которых предусмотрено размещение ЭВМ, относятся к 2 степени огнестойкости согласно СН 2.02.05-2020.

Для предотвращения распространения огня во время пожара с одной части здания на другую устраивают противопожарные преграды в виде стен, перегородок, дверей, окон. Особое требование предъявляется к устройству и размещению кабельных коммуникаций.

Нормы первичных средств пожаротушения для вычислительных центров приведены в таблице 8.8.

Таблица 8.8 – Примерные нормы первичных средств пожаротушения для вычислительного центра

|  |  |  |  |
| --- | --- | --- | --- |
| Помещение | Площадь, м2 | Углекислотные огнетушители ручные | Порошковые огнетушители |
| Вычислительный центр | 100 | 1 | 1 |

Для ликвидации пожаров в начальной стадии применяются первичные средства пожаротушения: внутренние пожарные водопроводы, огнетушители типа ОВП-10, ОУ-2, асбестовые одеяла и др.

Эвакуация персонала вычислительного центра осуществляется через эвакуационные выходы. Количество и общая ширина эвакуационных выходов определяются в зависимости от максимального возможного числа эвакуирующихся предельно допустимого расстояния от наиболее удаленного места возможного пребывания людей до ближайшего эвакуационного выхода согласно СН 2.03.05-2020.

Расчетное время эвакуации устанавливается по реальному расчету времени движения одного или нескольких потоков людей через эвакуационные выходы из наиболее удаленных мест размещения людей. Необходимое время эвакуации устанавливается на основе данных о критической продолжительности пожара с учетом степени огнестойкости здания, категории производства по взрывной и пожарной опасности.

## Требования к помещениям для работы с ВДТ, ЭВМ и ПЭВМ

Помещения для эксплуатации ЭВМ и ПЭВМ имеют естественное и искусственное освещение.

Запрещается выполнение основной работы с использованием ЭВМ и ПЭВМ на постоянных рабочих местах без естественного освещения, если это не обусловлено технологическим процессом.

Естественное освещение на рабочих местах с ЭВМ и ПЭВМ осуществляется через световые проемы, ориентированные преимущественно на север, северо-восток, восток, запад или северо-запад и обеспечивает коэффициент естественной освещенности не ниже 1,5 %. Оконные проемы оборудованы регулируемыми устройствами типа жалюзи, занавесей, внешних козырьков и другое.

Площадь одного рабочего места для пользователей ЭВМ и ПЭВМ на базе электронно-лучевой трубки составляет не менее 6 м2. Минимальная площадь одного рабочего места для взрослых пользователей и обучающихся учреждений образования с использованием ЭВМ или ПЭВМ на базе ЭЛТ может составлять не менее 4,5 м2 при условиях: отсутствия на рабочем месте периферийных устройств (принтер, сканер и другое); продолжительности работы – не более 4 часов в день.

Площадь одного рабочего места для пользователей ЭВМ и ПЭВМ на базе плоских дискретных экранов (жидкокристаллические, плазменные и другое) составляет не менее 4,5 м2.

Помещения, где размещаются рабочие места с ЭВМ и ПЭВМ, оборудованы защитным заземлением (защитным занулением) в соответствии с техническими требованиями по эксплуатации. Запрещается размещать рабочие места с ЭВМ и ПЭВМ на расстоянии менее 10 м от силовых кабелей, вводов и высоковольтных трансформаторов.

Помещения, в которых для работы используются преимущественно ЭВМ и ПЭВМ (операторские, расчетные и т.п.), не граничат с помещениями, в которых уровни шума и вибрации превышают нормируемые значения для данной категории проводимых в них работ и их типа (механические цеха, мастерские и т.п.). Звукоизоляция ограждающих конструкций помещений с ЭВМ и ПЭВМ обеспечивает нормируемые параметры шума в них.

Помещения с ЭВМ и ПЭВМ оборудованы системами отопления, кондиционирования воздуха или эффективной приточно-вытяжной вентиляцией.

Полимерные материалы, используемые для внутренней отделки интерьера помещений с ЭВМ и ПЭВМ, соответствуют требованиям Санитарных норм и правил, устанавливающих критерии гигиенической безопасности полимерных и полимеросодержащих материалов, изделий и конструкций, применяемых в промышленном и гражданском строительстве.

Поверхность пола в помещениях эксплуатации ЭВМ и ПЭВМ ровная, без выбоин, нескользкой, удобной для очистки и влажной уборки, обладает антистатическими свойствами.

Для внутренней отделки интерьера помещений, где расположены ЭВМ и ПЭВМ, использованы диффузно отражающие материалы с коэффициентом отражения для потолка – 0,7-0,8; для стен – 0,5-0,6; для пола – 0,3-0,5.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта было разработано web-приложение «Интерактивный тренажер обучения построению проекций точек и отрезков на эпюре Монжа», был дан обзор аналогов, которые позволяют конструировать блок-схемы, имитировать процессы и организовывать проверку знаний.

При выполнении дипломного проекта была изучена предметная область, соответствующие технологии и инструменты для проектирования и разработки web- приложения.

Для разработанного web-приложения были использованы следующие технологии:

* React, Typescript, HTML, CSS, Argon React Design;
* C#, .NET Core, Entity Framework Core;
* Microsoft SQL Server.

В ходе разработки была спроектирована база данных с использованием СУБД Microsoft SQL Server, включающая 9 таблиц, которыми можно описать все сущности проекта.

В целом разработанный проект включает в себя 3 составных части:

* раздел управления платформами;
* раздел конструирования задач;
* раздел выполнения узлов и задач.

Первая часть проекта предоставляет следующие возможности:

1. просмотр, создание, редактирование, удаление платформ, узлов и параметров;
2. подключение сторонних web-страниц.

Вторая часть приложения предназначена для просмотра, создания, редактирования и удаление задач.

Третья часть позволяет просматривать, а также запускать созданные узлы и задачи.

В ходе тестирования приложение показало стабильную работу, как и в обычном режиме работы. Разработанное приложение имеет удобный и простой интерфейс, а также высокую скорость отклика. Кроме своей основной функции как тренажер обучения построению проекций точек на эпюре Монжа данное web-приложение может быть использовано в широком спектре направлений для конструирования задач и автоматизации процессов, а также может быть полезно для развития визуального программирования.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бизнес-процесс [Электронный ресурс] – Режим доступа: https://intuit.ru/studies/courses/1041/218/lecture/27273, свободный. - Загл. с экрана.
2. Indigo [Электронный ресурс] – Режим доступа: http://masters.donntu.org/2013/fknt/bulanaya/library/article8.html, – Загл. с экрана
3. UniTest [Электронный ресурс] – Режим доступа: http://masters.donntu.org/2013/fknt/bulanaya/library/article8.html, свободный. - Загл. с экрана.
4. C Sharp [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/C\_Sharp, свободный. - Загл. с экрана.
5. ASP .NET Core [Электронный ресурс] – Режим доступа: https://metanit.com/sharp/aspnet5/1.1.php, свободный. - Загл. с экрана.
6. SQL vs NoSQL [Электронный ресурс] – Режим доступа: https://www.thorntech.com/2019/03/sql-vs-nosql, свободный. - Загл. с экрана.
7. MS SQL Server [Электронный ресурс] – Режим доступа: https://metanit.com/sql/sqlserver/1.1.php, свободный. - Загл. с экрана.
8. React [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/React, свободный. - Загл. с экрана.
9. Обзор Typescript [Электронный ресурс] – Режим доступа: https://www.typescriptlang.org/, свободный. - Загл. с экрана.
10. HTML [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/HTML5, свободный. - Загл. с экрана.
11. CSS [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/CSS, свободный. - Загл. с экрана.
12. Argon Design System [Электронный ресурс] – Режим доступа: https://www.creative- tim.com/product/argon-design-system, свободный. - Загл. с экрана.
13. Web-приложение [Электронный ресурс] – Режим доступа: https://quality- lab.ru/blog/key-principles-of-web-testing/, свободный. - Загл. с экрана.
14. Диаграмма вариантов использования [Электронный ресурс] – Режим доступа: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl4/gl4.html, свободный. - Загл. с экрана.
15. Физическое тестирование [Электронный ресурс] – Режим доступа: https://www.appline.ru/services/testing/functionalnoe-testirovanie, свободный. - Загл. с экрана.
16. Критическое тестирование [Электронный ресурс] – Режим доступа: https://qaevolution.ru/testirovanie-po/vidy-testirovaniya-po/test-kriticheskogo-puti/, свободный. - Загл. с экрана.
17. Blueprint [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Blueprint, свободный. - Загл. с экрана.
18. Unreal Engine [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Unreal\_Engine, свободный. - Загл. с экрана.
19. Microsoft Visio [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft\_Visio, свободный. - Загл. с экрана.
20. Entity Framework [Электронный ресурс] – Режим доступа: https://metanit.com/sharp/entityframeworkcore, свободный. - Загл. с экрана.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Общая схема IDEF0** | | | | | | | | | | | |
|  |  |  |  |  | ДП-10702119/07-2023-01 | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № документа | Подпись | Дата |
| Разраб. | | Кисель |  |  | Web-приложение «Интерактивный тренажёр обучения построению проекций точек и отрезков на эпюре Монжа» | Лит | | | Лист | Листов |
| Руковод. | | Полозков |  |  |  | У |  | 1 | 10 |
| Консульт. | | Полозков |  |  | 1-40 05 01  БНТУ, г. Минск | | | | |
| Н.контр. | | Федосова |  |  |
|  | |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Диаграмма вариантов использования** | | | | | | | | | | |
|  |  |  |  |  | ДП-10702119/07-2023-02 | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № документа | Подпись | Дата |
| Разраб. | | Кисель |  |  | Web-приложение «Интерактивный тренажёр обучения построению проекций точек и отрезков на эпюре Монжа» | Лит | | | Лист | Листов |
| Руковод. | | Полозков |  |  |  | У |  | 2 | 10 |
| Консульт. | | Полозков |  |  | 1-40 05 01  БНТУ, г. Минск | | | | |
| Н.контр. | | Федосова |  |  |
|  | |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Схема базы данных** | | | | | | | | | | |
|  |  |  |  |  | ДП-10702119/07-2023-03 | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № документа | Подпись | Дата |
| Разраб. | | Кисель |  |  | Web-приложение «Интерактивный тренажёр обучения построению проекций точек и отрезков на эпюре Монжа» | Лит | | | Лист | Листов |
| Руковод. | | Полозков |  |  |  | У |  | 3 | 10 |
| Консульт. | | Полозков |  |  | 1-40 05 01  БНТУ, г. Минск | | | | |
| Н.контр. | | Федосова |  |  |
|  | |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Схема взаимодействия компонентов приложения** | | | | | | | | | | |
|  |  |  |  |  | ДП-10702119/07-2023-04 | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № документа | Подпись | Дата |
| Разраб. | | Кисель |  |  | Web-приложение «Интерактивный тренажёр обучения построению проекций точек и отрезков на эпюре Монжа» | Лит | | | Лист | Листов |
| Руковод. | | Полозков |  |  |  | У |  | 4 | 10 |
| Консульт. | | Полозков |  |  | 1-40 05 01  БНТУ, г. Минск | | | | |
| Н.контр. | | Федосова |  |  |
|  | |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Модель разработки REST API** | | | | | | | | | | |
|  |  |  |  |  | ДП-10702119/07-2023-05 | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № документа | Подпись | Дата |
| Разраб. | | Кисель |  |  | Web-приложение «Интерактивный тренажёр обучения построению проекций точек и отрезков на эпюре Монжа» | Лит | | | Лист | Листов |
| Руковод. | | Полозков |  |  |  | У |  | 5 | 10 |
| Консульт. | | Полозков |  |  | 1-40 05 01  БНТУ, г. Минск | | | | |
| Н.контр. | | Федосова |  |  |
|  | |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Схема работы SQL Server** | | | | | | | | | | |
|  |  |  |  |  | ДП-10702119/07-2023-06 | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № документа | Подпись | Дата |
| Разраб. | | Кисель |  |  | Web-приложение «Интерактивный тренажёр обучения построению проекций точек и отрезков на эпюре Монжа» | Лит | | | Лист | Листов |
| Руковод. | | Полозков |  |  |  | У |  | 6 | 10 |
| Консульт. | | Полозков |  |  | 1-40 05 01  БНТУ, г. Минск | | | | |
| Н.контр. | | Федосова |  |  |
|  | |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Общий алгоритм работы ПП** | | | | | | | | | | |
|  |  |  |  |  | ДП-10702119/07-2023-07 | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № документа | Подпись | Дата |
| Разраб. | | Кисель |  |  | Web-приложение «Интерактивный тренажёр обучения построению проекций точек и отрезков на эпюре Монжа» | Лит | | | Лист | Листов |
| Руковод. | | Полозков |  |  |  | У |  | 7 | 10 |
| Консульт. | | Полозков |  |  | 1-40 05 01  БНТУ, г. Минск | | | | |
| Н.контр. | | Федосова |  |  |
|  | |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Схема работы ASP.NET Core** | | | | | | | | | | |
|  |  |  |  |  | ДП-10702119/07-2023-08 | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № документа | Подпись | Дата |
| Разраб. | | Кисель |  |  | Web-приложение «Интерактивный тренажёр обучения построению проекций точек и отрезков на эпюре Монжа» | Лит | | | Лист | Листов |
| Руковод. | | Полозков |  |  |  | У |  | 8 | 10 |
| Консульт. | | Полозков |  |  | 1-40 05 01  БНТУ, г. Минск | | | | |
| Н.контр. | | Федосова |  |  |
|  | |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Схема работы Entity Framework Core**  Using EF Core in ASP.NET Core Web API for performing CRUD operations |  DotNetCurry | | | | | | | | | | |
|  |  |  |  |  | ДП-10702119/07-2023-09 | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № документа | Подпись | Дата |
| Разраб. | | Кисель |  |  | Web-приложение «Интерактивный тренажёр обучения построению проекций точек и отрезков на эпюре Монжа» | Лит | | | Лист | Листов |
| Руковод. | | Полозков |  |  |  | У |  | 9 | 10 |
| Консульт. | | Полозков |  |  | 1-40 05 01  БНТУ, г. Минск | | | | |
| Н.контр. | | Федосова |  |  |
|  | |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Схема микросервисной архитектуры** | | | | | | | | | | |
|  |  |  |  |  | ДП-10702119/07-2023-10 | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № документа | Подпись | Дата |
| Разраб. | | Кисель |  |  | Web-приложение «Интерактивный тренажёр обучения построению проекций точек и отрезков на эпюре Монжа» | Лит | | | Лист | Листов |
| Руковод. | | Полозков |  |  |  | У |  | 10 | 10 |
| Консульт. | | Полозков |  |  | 1-40 05 01  БНТУ, г. Минск | | | | |
| Н.контр. | | Федосова |  |  |
|  | |  |  |  |

# ПРИЛОЖЕНИЕ А

**Код Flow.tsx**

import React from "react";

import { FlowMenu } from "./components/Flow/FlowMenu"; import Draggable from "react-draggable";

import NodeList from "./components/Flow/FlowPanel/NodeList";

import ConnectorList from "./components/Flow/FlowPanel/ConnectorList"; import "./flows.scss";

import SearchList from "./components/Flow/FlowPanel/SearchList"; import {

ObservableValue, Observer, useObservable,

} from "../../utils/observable"; import {

IFlowNodeOutputModel, IPinValueOutputModel, IFlowOutputModel, IConnectorOutputModel,

} from "../models/output/flowOutput"; import {

FLOW\_INPUT\_NODE\_ID, FLOW\_OUTPUT\_NODE\_ID,

IPosition,

} from "../models/common"; import api from "../api";

import newGuid from "../../utils/guid";

import { PinDirection } from "../models/enums";

interface IFlowParams { id: number;

}

export interface IFlowSelection { flow?: number;

searchBox?: boolean; node?: INodeObservable;

connector?: IConnectorObservable;

}

export interface IFlowObservable { id: number;

name: ObservableValue<string>; isActive: ObservableValue<boolean>;

nodes: ObservableValue<INodeObservable[]>;

connectors: ObservableValue<ObservableValue<IConnectorObservable>[]>; position: IPosition;

}

export interface IConnectorObservable { key: string;

startPin: IPinObservable; endPin: IPinObservable;

}

export interface INodeObservable {

model: ObservableValue<IFlowNodeOutputModel>; connectors: ObservableValue<IConnectorObservable>[];

}

export interface IPinObservable { model: IPinValueOutputModel; node: INodeObservable;

}

export default function Flow({ id }: IFlowParams) { const flowPanelRef = React.createRef<HTMLDivElement>();

const draggableWrapperRef = React.createRef<HTMLDivElement>();

const selection = useObservable<IFlowSelection>({ flow: id }); const flow: IFlowObservable = {

id: id,

name: new ObservableValue(""), isActive: new ObservableValue(true),

nodes: new ObservableValue<INodeObservable[]>([]),

connectors: new ObservableValue<ObservableValue<IConnectorObservable>[]>( []

),

position: { x: 0, y: 0 },

};

const updateKeys = (nodes: IFlowNodeOutputModel[]) => { nodes.forEach((node) => {

node.key = newGuid();

node.inputPins.forEach((x) => (x.key = newGuid())); node.outputPins.forEach((x) => (x.key = newGuid()));

});

};

const onDrag = (data: IPosition) => { const tileX = data.x % 128;

const tileY = data.y % 128;

flow.position.x = Math.round(data.x); flow.position.y = Math.round(data.y);

if (draggableWrapperRef.current?.style)

draggableWrapperRef.current.style.transform = `translate(${-data.x}px,${- data.y}px)`;

if (flowPanelRef.current?.style) flowPanelRef.current.style.backgroundPosition = `${tileX}px ${tileY}px`;

};

const onAddNode = (id: number) => { api.flowNode.get(id, {

success: (response) => { const flowPanelSize = {

width: flowPanelRef.current?.clientWidth || 0, height: flowPanelRef.current?.clientHeight || 0,

};

updateKeys([response]);

response.x = response.x || flowPanelSize.width / 2 - flow.position.x; response.y = response.y || flowPanelSize.height / 2 - flow.position.y;

const node = { model: new ObservableValue(response), connectors: [] }; flow.nodes.value = [...flow.nodes.value, node];

selection.value = { node: node };

},

});

};

const onNodeSelect = (node: INodeObservable) => { selection.value = { node: node };

};

const onConnectorSelect = (connector: IConnectorObservable) => { selection.value = { connector: connector };

};

const onPinsConnect = (startPin: IPinObservable, endPin: IPinObservable) => { const connector = new ObservableValue<IConnectorObservable>({

key: newGuid(), startPin: startPin, endPin: endPin,

});

startPin.model.connections = startPin.model.connections

? [...startPin.model.connections, connector.value.key]

: [connector.value.key]; startPin.node.connectors.push(connector);

endPin.model.connections = endPin.model.connections

? [...endPin.model.connections, connector.value.key]

: [connector.value.key]; endPin.node.connectors.push(connector);

flow.connectors.value = [...flow.connectors.value, connector];

//TODO: Warning: Can't perform a React state update on an unmounted component.

This is a no-

op, but it indicates a memory leak in your application. To fix, cancel all subscrip tions and asynchronous tasks in the componentWillUnmount method.

startPin.node.model.notify(); endPin.node.model.notify();

};

const updateFlowObservable = (model: IFlowOutputModel) => { flow.id = model.id;

flow.name.value = model.name;

flow.nodes.value = model.nodes.map<INodeObservable>((x) => ({ model: new ObservableValue(x),

connectors: [],

}));

onDrag({ x: model.x, y: model.y });

selection.value = { flow: flow.id };

};

const loadFlow = () => { if (id === 0) {

api.flow.getTemplate(id, { success: (response) => {

const { width, height } = document

.getElementById("flow-panel")!

.getBoundingClientRect();

const x1 = Math.round(width \* 0.2); const x2 = Math.round(width \* 0.7); const y = Math.round(height \* 0.5);

response.nodes[0].x = x1; response.nodes[0].y = y; response.nodes[1].x = x2; response.nodes[1].y = y;

updateKeys(response.nodes); updateFlowObservable(response);

},

});

} else { api.flow.get(id, {

success: (response) => { updateKeys(response.nodes);

const connectors: ObservableValue<IConnectorObservable>[] = [];

response.aliases.forEach((x) => (x.id = -x.id)); response.connectors.push(

...response.aliases.flatMap((x) => x.pinValueIds.map<IConnectorOutputModel>((pinValueId) => {

const [startId, endId] = x.direction === PinDirection.Input

? [x.id, pinValueId]

: [pinValueId, x.id]; return {

id: 0,

startPinValueId: startId, endPinValueId: endId,

};

})

)

);

const inputNode = response.nodes.find(

(x) => x.nodeId === FLOW\_INPUT\_NODE\_ID

);

if (inputNode) {

inputNode.outputPins = response.aliases

.filter((x) => x.direction === PinDirection.Input)

.map<IPinValueOutputModel>((x) => ({ key: newGuid(),

id: x.id,

nodeId: inputNode.nodeId, flowNodeId: inputNode.id, name: x.name,

isPublic: true, pinId: 0, value: "",

valueType: x.valueType, direction: PinDirection.Output,

}));

}

const outputNode = response.nodes.find(

(x) => x.nodeId === FLOW\_OUTPUT\_NODE\_ID

);

if (outputNode) {

outputNode.inputPins = response.aliases

.filter((x) => x.direction === PinDirection.Output)

.map<IPinValueOutputModel>((x) => ({ key: newGuid(),

id: x.id,

nodeId: outputNode.nodeId, flowNodeId: outputNode.id, name: x.name,

isPublic: true, pinId: 0, value: "",

valueType: x.valueType, direction: PinDirection.Input,

}));

}

updateFlowObservable(response);

response.connectors.forEach((connector) => { const startNode = flow.nodes.value.find((x) =>

x.model.value.outputPins.some(

(p) => p.id === connector.startPinValueId

)

);

const startPin = startNode?.model.value.outputPins.find(

(x) => x.id === connector.startPinValueId

);

const endNode = flow.nodes.value.find((x) => x.model.value.inputPins.some(

(p) => p.id === connector.endPinValueId

)

);

const endPin = endNode?.model.value.inputPins.find(

(x) => x.id === connector.endPinValueId

);

if (startPin && startNode && endPin && endNode) {

const connector = new ObservableValue<IConnectorObservable>({ key: newGuid(),

startPin: { model: startPin, node: startNode }, endPin: { model: endPin, node: endNode },

});

startPin.connections = startPin.connections

? [...startPin.connections, connector.value.key]

: [connector.value.key]; startNode.connectors.push(connector);

endPin.connections = endPin.connections

? [...endPin.connections, connector.value.key]

: [connector.value.key]; endNode.connectors.push(connector);

connectors.push(connector);

}

});

flow.connectors.value = connectors; flow.nodes.value = [...flow.nodes.value];

},

});

}

};

loadFlow();

return (

<>

<div>

<Observer selection={selection}>

{(observer: { selection: IFlowSelection }) => { return (

<div className="d-flex constructor-root">

<div

id="flow-panel" ref={flowPanelRef}

className="w-100 flow-panel overflow-hidden" onContextMenu={(e) => e.preventDefault()}

>

<Draggable

defaultClassName="w-100 h-100" handle=".draggable-wrapper" cancel=".node, .search-panel" position={flow.position} onDrag={(\_e, data) => onDrag(data)}

onMouseDown={() => (selection.value = { flow: id })}

>

<div>

<div

ref={draggableWrapperRef}

className="draggable-wrapper w-100 h-100 position-absolute"

></div>

<ConnectorList connectors={flow.connectors} flowPosition={flow.position}

selectedConnector={selection.value.connector} onConnectorSelect={onConnectorSelect}

/>

<NodeList nodes={flow.nodes}

selectedNode={selection.value.node} onNodeSelect={onNodeSelect} onPinsConnect={onPinsConnect}

/>

</div>

</Draggable>

<SearchList isExpanded={observer.selection.searchBox}

onMouseDown={() => (selection.value = { searchBox: true })} onAddNode={onAddNode}

onAddFlow={(id) => console.log("Add flow:", id)}

/>

</div>

<FlowMenu flow={flow} selection={selection} />

</div>

);

}}

</Observer>

</div>

</>

);

}