Санкт-Петербургский государственный университет

Ивашева Валерия Михайловна

Выпускная квалификационная работа

Визуальная технология обработки медицинских изображений при помощи библиотеки MIRF на основе REAL.NET.

Уровень образования: бакалавриат

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

Основная образовательная программа *СВ.5006.2017 «Математическое обеспечение и администрирование информационных систем»*

Профиль *Системное программирование*

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

Рецензент:
разработчик Google LLC С.А. Мусатян

Санкт-Петербург
2021

Saint Petersburg State University

Valeria Ivasheva

Bachelor's Thesis

Visual technology for processing medical images using the MIRF library based on REAL.NET

Education level: bachelor

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

Programme *CB.5006.2017 "Software and Administration of Information Systems"*

Profile: *System Programming*

Scientific supervisor:
C.Sc., docent Y.V. Litvinov

Reviewer:
Software engineer "Google LLC, UK office" S.A. Musatyan

Saint Petersburg
2021

Оглавление

Введение

На сегодняшний день все чаще появляются различные алгоритмы на основе машинного обучения, решающие задачи, связанные с медициной. Одна из этих задач — это анализ и обработка медицинских данных.

На кафедре системного программирования СПбГУ разрабатывается библиотека MIRF [3], которая позволяет работать с медицинскими данными, в частности с изображениями. Для обеспечения гибкости в основе архитектуры данной библиотеки лежит архитектурный стиль «Pipes&Filters», поэтому обработка данных выполняется конвейерами, состоящими из независимых друг от друга блоков. На данный момент в библиотеке реализованы следующие алгоритмы: обработка ЭКГ-сигналов и распознавание различных заболеваний сердца, анализ МРТ-снимков головы на предмет рассеянного склероза, а также классификация внутричерепного кровоизлияния.

Изначально библиотека MIRF была ориентирована на медиков-исследователей с некоторыми навыками программирования. Поэтому для того, чтобы решить ту или иную задачу при помощи MIRF, медицинский специалист должен написать код на Kotlin, который задает конвейер для обработки медицинских данных. Умение писать даже простейший код на каком-нибудь из языков программирования ограничивает количество возможных пользователей, но теперь хочется расширить круг потенциальных пользователей проекта. Одним из подходов к пользовательскому программированию является визуальное программирование, при таком подходе пользователь задает программу, оперируя некоторыми графическими объектами. Программирование при помощи визуальных предметно-ориентированных языков более наглядно и проще для изучения в отличии от текстовых аналогов, поэтому часто используется в обучающих целях, например среда для программирования роботов TRIK Studio [10]. Таким образом для большей доступности использования данной библиотеки можно разработать специальный редактор, который позволит создавать конвейеры, используя визуальное программирование.

На кафедре системного программирования СПбГУ уже несколько лет разрабатывается среда для визуального программирования REAL.NET [4]. Изначально она создавалась, как десктопная система. Но со временем для удобства пользователей было решено сделать её веб-приложением, так появилась версия REAL.NET Web. На данный момент активно разрабатывается универсальный веб-редактор для работы с визуальными языками. Данная работа также может служить апробацией для редактора REAL.NET Web.

Одним из требований к данному редактору является удобство использования. Конечные пользователи — медицинские специалисты, которые не могут потратить много времени, чтобы научиться им пользоваться, они неизбежно будут допускать ошибки. Но если сама система будет их предупреждать и направлять, это в разы повысит удобство использования. На данный момент в десктопной версии REAL.NET реализована система проверки ограничений, которая предупреждает о частых ошибках и не допускает некорректных диаграмм. Необходимо адаптировать данную систему также для веб-версии среды для дальнейшего использования.

1. Постановка задачи

Целью работы является создание на основе платформы REAL.NET веб-редактора языка для обработки медицинских изображений при помощи библиотеки MIRF.

Для её выполнения были поставлены следующие задачи.

1. Разработать визуальный предметно-ориентированный язык для работы с библиотекой MIRF.
2. Спроектировать и реализовать пользовательский интерфейс для веб-редактора.
3. Реализовать генератор для преобразования графической модели в конвейер для обработки данных.
4. Адаптировать систему проверки ограничений для REAL.NET Web и расширить её для поддержки ограничений, специфичных для MIRF.
5. Провести апробацию веб-редактора.

2. Обзор

2.1. Обзор аналогов

При обзоре аналогов рассматривались редакторы для анализа медицинских данных, а также те, которые можно для этого использовать. Основным критерием отбора аналогов являлась ориентация на пользователей, которые не знакомы с программированием, или знакомы, но на самом примитивном уровне. Одна из основных целей обзора аналогов — это изучение интерфейсов редакторов и выявление полезных функций для дальнейшей реализации.

2.1.1. Northstar

Northstar [7] — интерактивная система для анализа данных, основная цель которой — дать возможность работать с данными более широкому кругу пользователей. Для достижения этой цели необходимо было сделать пользовательский интерфейс интуитивно понятным и интерактивным, чтобы охватить пользователей без опыта в компьютерных науках.

В редакторе справа есть боковая панель, на которой расположены вкладка datasets для выбора набора данных, на которых будет проводиться анализ, вкладка attributes содержит критерии, по которым можно фильтровать данные. Во вкладке operators есть уже реализованные операторы, которые позволяют использовать различные алгоритмы машинного обучения для анализа имеющих данных. На рисунке 1 реализован следующий сценарий: контейнер 1 отображает количество людей по критерию сердечной недостаточности, он связан также с контейнерами 2 и 3, которые отображают данные по полу и возрасту соответственно. Для того, чтобы отфильтровать количество людей с сердечной недостаточностью по возрасту и полу, необходимо выбрать интересующие значения на соответствующем контейнере. Конкретно на рисунке 1 в контейнере 1 отображается статистика женщин в возрасте от 40 до 100 лет, больных сердечной недостаточностью и нет.

Возможность настройки пользователями параметров и отображение их непосредственно на контейнере во время взаимодействия позволяет увидеть сценарий полностью.

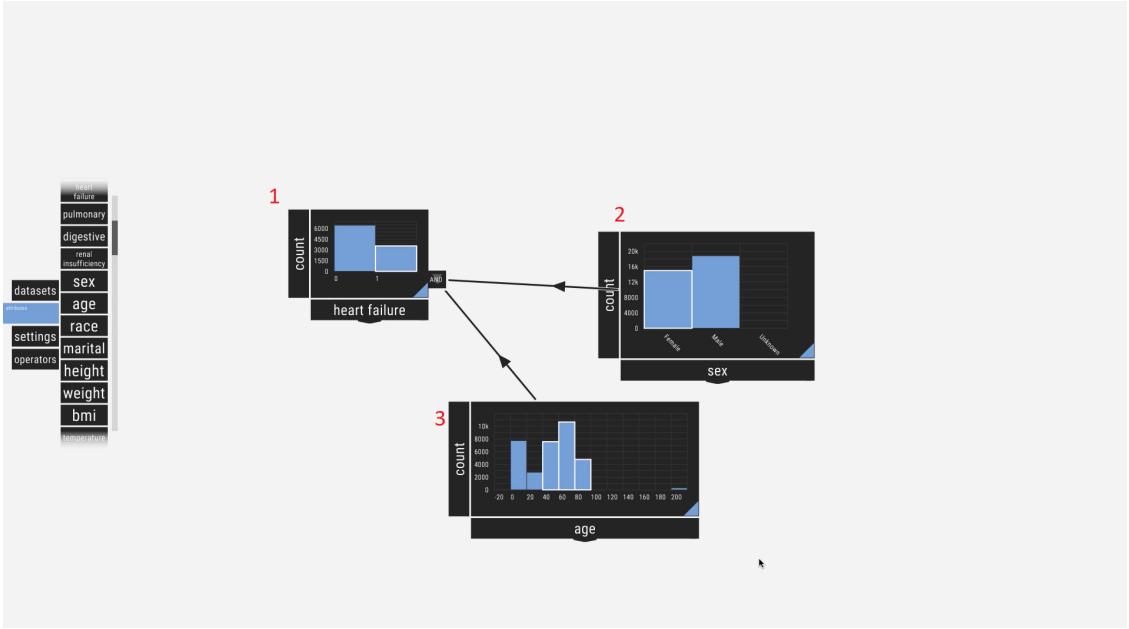

Рис. 1: Исследование влияния пола и возраста на сердечную недостаточность (из [12])

2.1.2. GraphMIC

GraphMIC [16] — кроссплатформенное приложение для обработки медицинских изображений с использованием библиотек ITK [6] и OpenCV [13]. Пользователь для обработки изображения создает конвейер, состоящий из функций данных библиотек. Визуальное программирование позволяет пользователям сосредоточиться на упорядочении и параметризации операций, а не на реализации эквивалентной функциональности изначально на C++. Помимо использования уже готовых функций у пользователя также есть возможность написать свою за счет встроенного интерпретатора Python.

На рисунке 2 изображен пользовательский интерфейс GraphMIC. Окно приложения разделено на три части: слева расположено боковое меню, в центре сцена для создания конвейеров, справа средство для

просмотра изображения, использующее библиотеку MITK [9]. Для создания узла используется текстовый быстрый поиск, в который пользователю необходимо ввести ключевые слова. Доступ к этому поиску происходит непосредственно на сцене задания конвейера через двойной клик левой кнопки мыши.

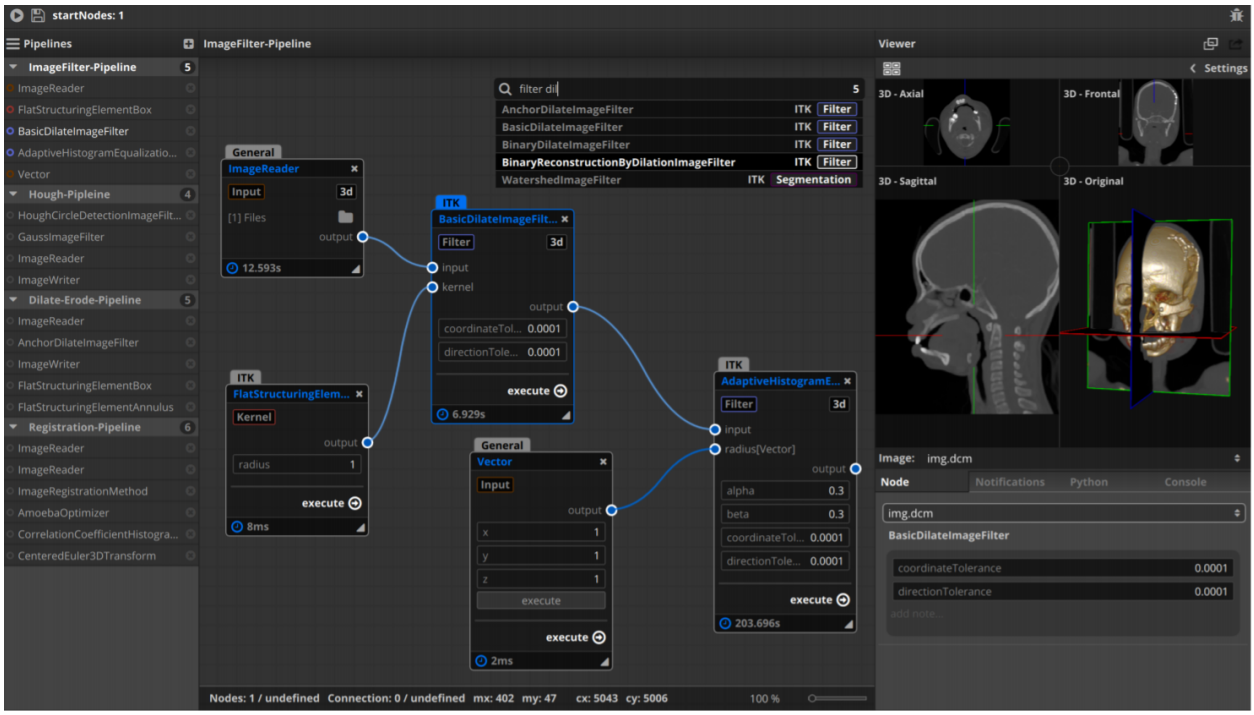

Рис. 2: Пользовательский интерфейс GraphMIC (из [16])

Визуализация изображения непосредственно в редакторе может быть удобной функцией для пользователя. На данный момент это не является первостепенной задачей, но может быть одним из дальнейших направлений работы.

2.2. Используемые технологии

2.2.1. REAL.NET

REAL.NET [19] — среда для быстрого создания предметно-ориентированных визуальных языков, разрабатываемая на кафедре системного программирования в СПбГУ. Для того чтобы задать визуальный язык, необходимо определить его метамодель, которая задает

все корректные конструкции языка. Изначально была реализована в виде десктопной версии на платформе .NET.

Репозиторий — ядро платформы REAL.NET, в котором хранятся модели и метамодели и интерфейс для редактирования моделей.

Для десктопной версии был реализован плагин проверки ограничений [17]. Для задания ограничений на язык A создается новая метамодель, которая содержит все элементы метамодели языка A, и также элементы проверки ограничений: логические операции и связи, специфические для данного языка ограничений. Ограничения на язык задаются пользователем визуально на отдельной панели. Плагин состоит из двух компонентов: первый отвечает за работу с графическим интерфейсом, а второй за проверку ограничений на корректность и проверку его выполнения.

REAL.NET Web — веб-версия редактора. Серверная часть реализована с использованием микросервисной архитектуры, она проиллюстрирована на рисунке 3. Репозиторий был выделен в отдельный микросервис, также есть микросервис хранилища, в котором находятся модели пользователей. Для более подробного описания серверной части см. [18]. Прототип клиентской части веб-редактора был реализован

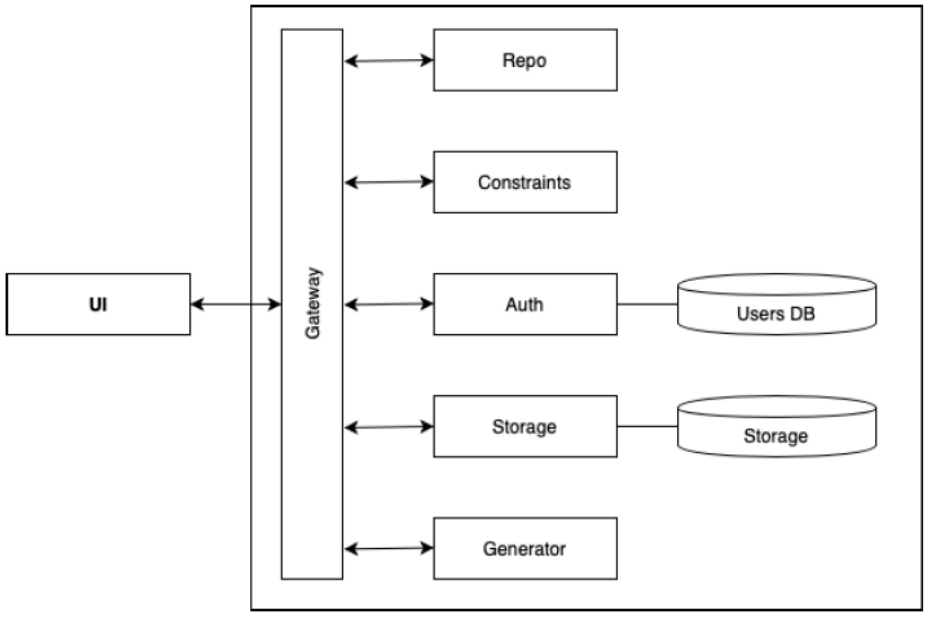

Рис. 3: Архитектура платформы REAL.NET Web

на платформе разработки веб-приложений ASP.NET с использованием паттерна Model-View-Controller. Основным языком разработки является TypeScript. На рисунке 4 изображен его пользовательский интерфейс. В редакторе была реализована работа с элементами: добавление на сцену при помощи перетаскивания, их удаление, а также возможность задавать связи между ними. На данный момент разрабатывается новая версия веб-редактора с использованием библиотеки React для пользовательского интерфейса и библиотеки React Flow для построения диаграмм.

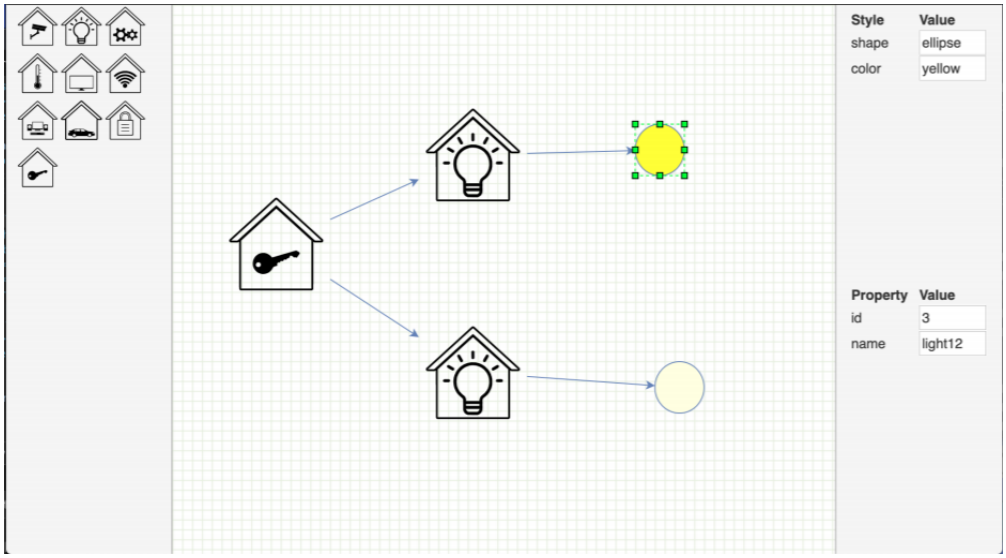

Рис. 4: Универсальный редактор REAL.NET Web

2.2.2. MIRF

MIRF [8] — это библиотека, написанная на языке Kotlin, с открытым исходным кодом для удобного создания приложений, обрабатывающих медицинские данные. Архитектура основана на Pipes&Filters, таким образом все компоненты, обрабатывающие данные, составляют конвейер.

Первая версия библиотеки имела монолитную архитектуру, была интегрирована с TensorFlow, также были реализованы утилиты для обработки медицинских форматов DICOM и NIfTI и генерация отчетов в формате PDF [11]. Так как со временем стало ясно, что для увеличения скорости обработки изображений необходимо больше вычислительных

ресурсов, появилась вторая версия, основанная на микросервисной архитектуре [20].

MIRF состоит из двух основных пакетов — Core и Features. Core содержит абстрактные классы и интерфейсы для блоков, фильтров и медицинских данных. Features содержит модули с различной функциональностью: доступ к репозиторию, инструменты для работы с различными конкретными медицинскими данными, генерация отчетов, инструменты для визуализации и сегментации изображений и др.

На данный момент конвейеры можно задавать двумя способами.

1. Непосредственно в коде, описав блоки и связи между ними.

2. Написав JSON-файл с конфигурацией. На рисунке 5 представлена конфигурация простейшего конвейера, который накладывает маску на DICOM-изображение и сохраняет результат в формате PDF.

```
[
  { "id": 0, "blockType" : "ReadDicomImageSeriesAlg", "children": [1, 2] },
  { "id": 1, "blockType" : "AddMaskAlg", "children": [3] },
  { "id": 2, "blockType" : "ConvertImagesToPdfAlg", "children": [4] },
  { "id": 3, "blockType" : "ConvertImagesToPdfAlg", "children": [4] },
  { "id": 4, "blockType" : "PrepareReportAlg", "children": [5] },
  { "id": 5, "blockType" : "SaveReportAlg", "children": [] }
]
```

Рис. 5: Конфигурация простейшего конвейера в формате JSON

2.3. Библиотека для визуализации конвейеров

Так как на момент написания работы редактор REAL.NET Web еще не был готов, было принято решение разработать свой редактор, чтобы уменьшить технические риски. Для визуализации конвейеров рассматривались следующие библиотеки.

- Rete.js [15] — фреймворк JavaScript для визуального программирования. Написан на TypeScript, хорошая документация. Состоит

из редактора, движка и компонент. Движок позволяет работать с конвейером в реальном времени и передавать данные последовательно через описанный пользователем граф. Компоненты — это шаблоны самих узлов. Редактор Rete используется для работы с узлами, реализованы базовые действия, такие как добавление и удаление узлов, а также связей между ними, для узлов поддерживаются множественные входы и выходы, а также можно добавить проверку на типы входных данных. Можно импортировать/экспортировать сценарий в JSON формате. Есть поддержка Vue и React.

- Drawflow [1] — библиотека для визуализации потоков, написанная на JavaScript. Определена работа с узлами на панели редактора (перетаскивание, удаление, добавление связей между ними), узлы также поддерживают множественные входы и выходы. Реализованы экспорт/импорт данных редактора. Существует поддержка компонентов Vue и Nuxt.

- React Flow [14] — React библиотека на TypeScript, используемая для создания диаграмм. Определена работа с узлами на панели редактора (перетаскивание, удаление, добавление связей между ними), узлы также поддерживают множественные входы и выходы. Есть возможность задавать валидацию соединения узлов, а также задавать пользовательские узлы.

Все библиотеки имеют возможность работы с узлами на панели редактора. Библиотека Drawflow имеет существенный недостаток по сравнению с остальными, в ней отсутствует интеграция с React. Из-за этого было решено отказаться от нее.

Основным преимуществом библиотеки Rete.js является наличие движка, но данная функция для редактора является излишней, так как вся работа с данными проходит непосредственно на сервере.

Была выбрана библиотека ReactFlow из-за наличия множества встроенных функций, а также она используется в новой версии редактора REAL.NET Web, что позволяет избежать излишних зависимостей.

3. Метамодель

Первое, что нужно сделать при разработке языка, — это понять, из каких сущностей будут состоять диаграммы. Так как библиотека MIRF состоит из алгоритмов и данных, было решено, что визуальный язык для библиотеки MIRF, следуя её идеологии, будет представлять вычисления в виде последовательности фильтров, через которые проходят медицинские данные. Но сами данные будут приходить извне. На момент написания работы разрабатывалась платформа автоматизации обработки медицинских данных, именно здесь и ведется вся работа с данными.

На рисунке 6 представлена метамодель языка для библиотеки MIRF. На данный момент пока добавлены только блоки для обработки ЭКГ-сигналов.

Корнем метамодели является AbstractMirfNode, от которого должны наследоваться все остальные члены метамодели. Атрибуты AbstractMirfNode:

- inputType содержит все возможные типы блоков входных данных;
- outputType содержит тип блока выходных данных.

Для того, чтобы иметь возможность работать с новыми алгоритмами после расширения библиотеки MIRF, необходимо всего лишь добавить те самые алгоритмы в метамодель, чтобы иметь возможность создавать конвейеры в редакторе. Имя не абстрактного элемента метамодели должно совпадать с типом блока, который он представляет. Вышеописанные атрибуты должны совпадать с именами элементов метамодели, которых они представляют.

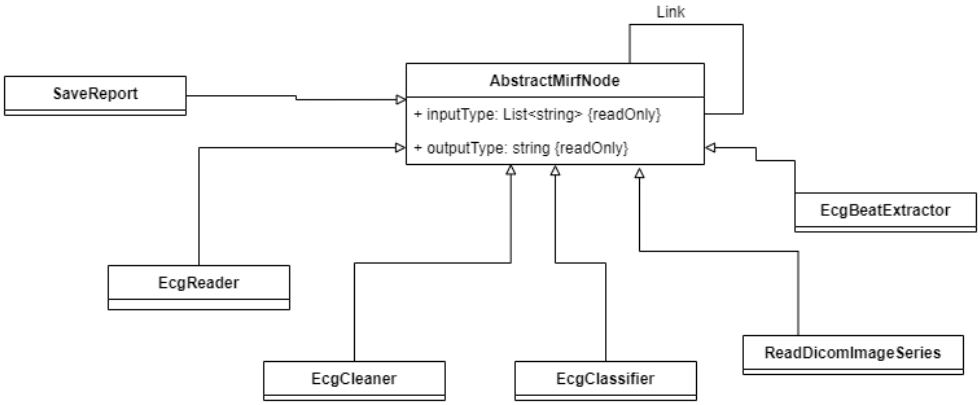

Рис. 6: Метамодель MIRF

4. Веб-редактор

Так как библиотека MIRF основана на Pipes&Filters и преобразование данных происходит последовательно по конвейеру, который задается пользователем, то основная цель редактора — предоставить пользователю возможность задавать этот самый конвейер.

4.1. Пользовательский интерфейс

Редактор состоит из двух частей:

- боковая панель, которая содержит элементы из метамодели, получаемые с сервера;
- сцена, на которой пользователь задает конвейер.

Для того, чтобы определить конвейер, необходимо также уметь задавать параметры для блоков. Рассматривались две возможности настройки каждого блока, справа как отдельное меню, которое появляется для каждого выбранного блока (реализовано в REAL.NET), или непосредственно на самом блоке, как в Northstar. Второй способ удобнее для пользователя, так как одновременно видны параметры каждого блока, и если что-то нужно уточнить, то не нужно тратить лишнее время на поиск и проверку отдельного блока. Был выбран и реализован второй вариант задания параметров.

На рисунке 7 изображен пользовательский интерфейс веб-редактора, на сцене которого задан конвейер для обработки и классификации сердечных ритмов на ЭКГ.

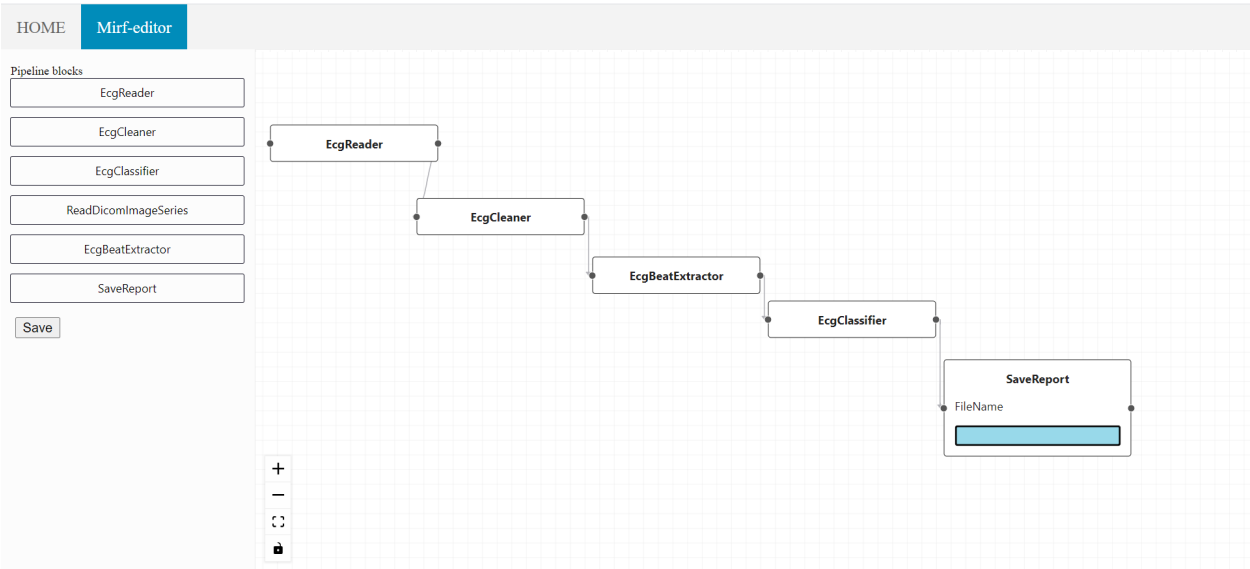

Рис. 7: Пользовательский интерфейс веб-редактора

4.2. Реализация

Было решено переиспользовать код универсального редактора REAL.NET Web. Поэтому задачи, связанные с отображением элементов на палитре, а также добавления и удаления их на сцену и связей между ними уже решены. Изначально разработка велась на языке TypeScript с использованием первой версии редактора REAL.NET Web. На данный момент существует новая версия REAL.NET Web на React, поэтому данный редактор было решено перенести на нее. Для построения диаграмм используется библиотека ReactFlow.

Для работы с сервером были добавлены методы, осуществляющие запросы к контроллерам сервиса репозитория для управления моделями и проверки на выполнение ограничений, и также к контроллерам сервиса генератора для генерации конфигурации.

В рамках работы также была реализована следующая функциональность пользовательского интерфейса.

- Была добавлена компонента MirfPipelineNode — узел, параметры которого задаются на нем самом, а не на боковой панели. На рисунке 8 изображен узел с двумя параметрами.
- Была добавлена проверка валидации соединения. Таким образом, если тип выходного блока не соответствует типу входного блока, то появляется предупреждение о несовпадении типов и ребро не создается.

Разработка ведется в репозитории [5].

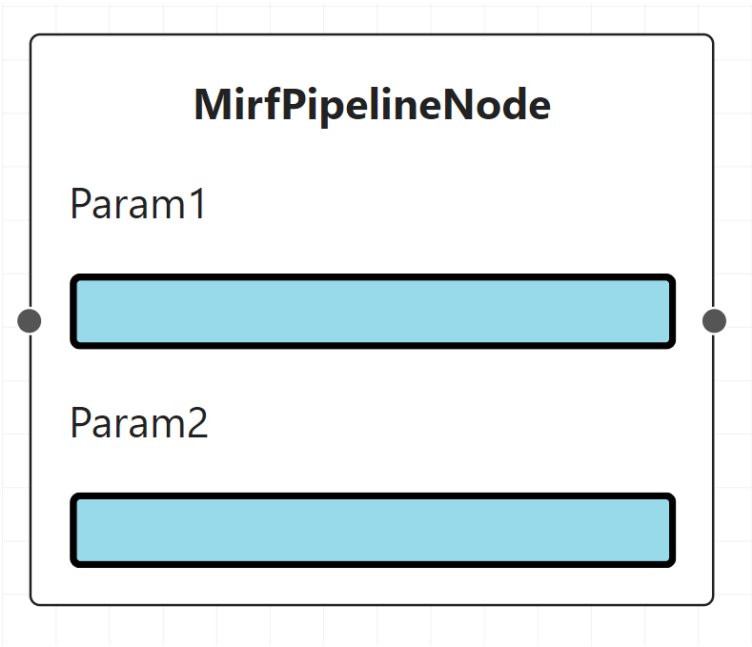

Рис. 8: Представление MirfPipelineNode

5. Интеграция REAL.NET и MIRF

5.1. Адаптация проверки ограничений для REAL.NET Web и MIRF

На рисунке 10 представлена архитектура модуля проверки ограничений для REAL.NET Web. Классы ConstraintsMatcher и ConstraintCheckingSystem были взяты из десктопной версии без изменений. Они соответствуют логической компоненте плагина и отвечают за проверку ограничений, задаваемую как в десктопной версии через метамодель языка, дополненной элементами языка ограничений (пока не реализовано в веб-редакторе REAL.NET). Класс StandartCheckingSystem отвечает за проверку ограничений, которая задана через стратегию сразу на сервере. Обе системы не исключают друг друга. Проверка в MirfCheckingStrategy состоит из двух условий:

- граф модели не содержит циклов;
- все входные и выходные типы данных корректны.

На данный момент проверка ограничений реализована на сервисе как часть микросервиса, отвечающего за работу с репозиторием, для того, чтобы избежать нагрузки сети. Разработка ведется в репозитории [2].

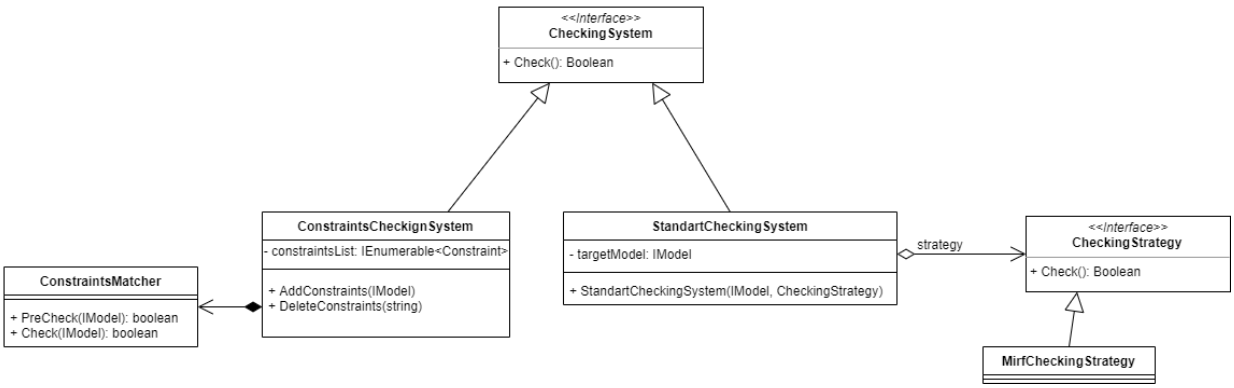

Рис. 9: Архитектура модуля проверки ограничений

5.2. Генератор

В ходе работы был добавлен генератор для конфигурации конвейера в виде микросервиса, написанного на ASP.NET Core. Параметром GET-запроса к серверу для генерации конфигурации является модель, которая отображает конвейер для обработки данных. Затем со стороны микросервиса посылается запрос на проверку соответствия модели всем стандартным ограничениям библиотеки. При условии выполнения данного пункта генерируется JSON-объект путем обхода всех вершин и ребер модели, полученной раннее из репозитория.

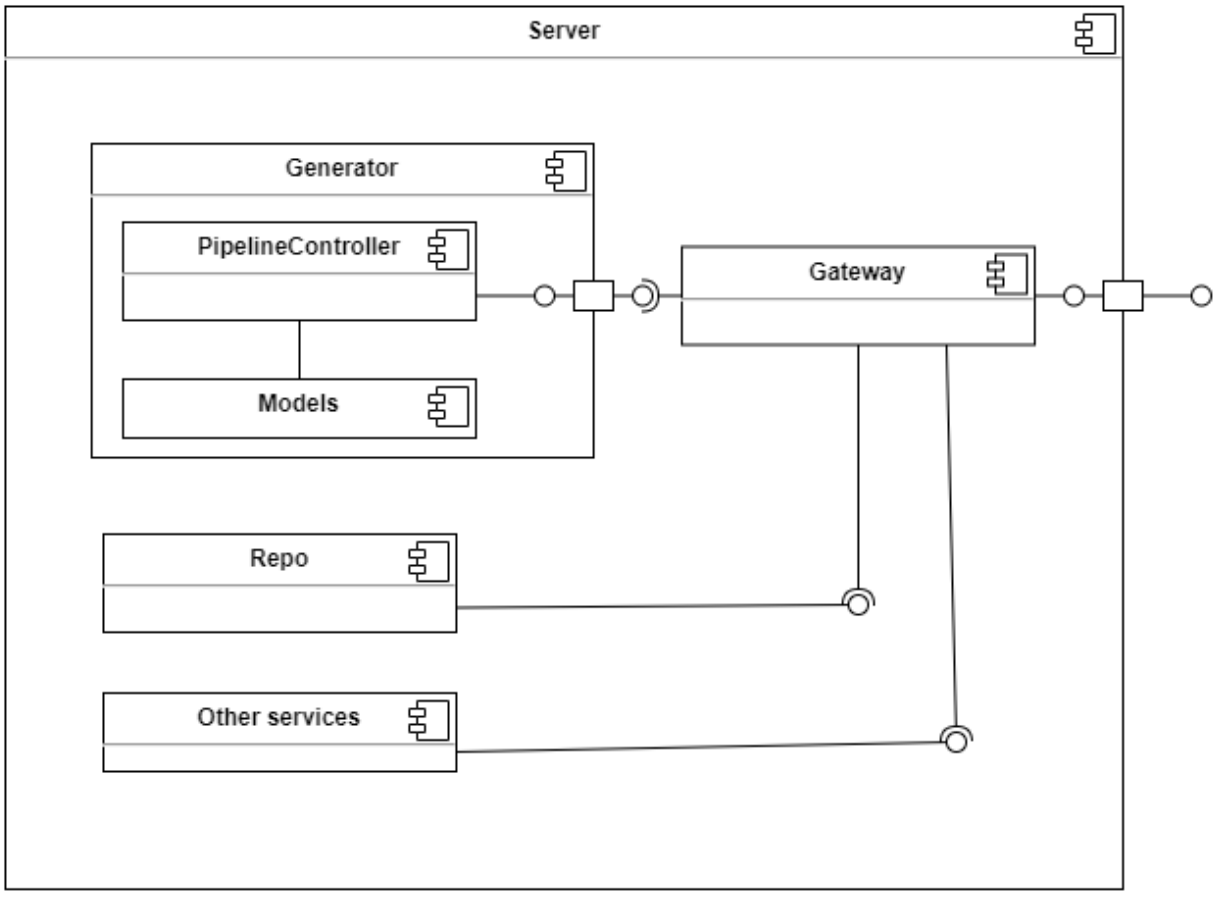

Рис. 10: Диаграмма компонентов сервера

6. Апробация прототипа

В апробации прототипа участвовали 3 студента медицинских факультетов старших курсов. Их попросили создать конвейер для классификации сердечных ритмов на ЭКГ и заполнить следующую анкету.

- Оценить удобство использования системы от 1 до 5.
- Отметить что понравилось, что не понравилось.
- Написать предложения по улучшению, что еще хотелось бы видеть.

В результате анкетирования средний балл удобства использования системы составил 4,33 балла. В целом анкетируемые готовы использовать данную систему. В плюсах отметили, что система проста в использовании, удобный и интуитивно понятный интерфейс, а также не требует много времени на обучение.

Также благодаря апробации были выявлены следующие минусы.

- Не работает соединение узлов в браузере Safari.
- Блеклый дизайн.
- Иногда не очень очевидно что с чем можно связывать.

Самым частым предложением было добавить возможность увидеть результат обработки сразу на редакторе.

Заключение

В рамках выпускной квалификационной работы были решены следующие задачи.

- Проведено исследование существующих аналогов.
- Разработан визуальный предметно-ориентированный язык для работы с библиотекой MIRF.
- Спроектирован пользовательский интерфейс и реализован веб-редактор для обработки медицинских изображений на языке TypeScript с использованием библиотеки React.
- Проведена адаптация системы проверки ограничений для REAL.NET Web, также система была расширена для поддержки ограничений, специфичных для MIRF.
- Реализован генератор для преобразования графической модели в конвейер для обработки данных.
- Проведена апробация прототипа, в ходе которой было отмечено удобство системы, а также выявлены некоторые недостатки системы.

Список литературы

[1] Drawflow. — URL: https://github.com/jerosoler/Drawflow (online; accessed: 12.03.2021).

[2] Github-репозиторий проверки ограничений. — URL: https://github.com/REAL-NET/web-editor-backend/tree/constraints-checking-system (online; accessed: 25.04.2021).

[3] Github-репозиторий проекта MIRF. — URL: https://github.com/MathAndMedLab/MIRF2 (online; accessed: 08.12.2020).

[4] Github-репозиторий проекта REAL.NET. — URL: https://github.com/REAL-NET (online; accessed: 08.12.2020).

[5] Github-репозиторий редактора. — URL: https://github.com/REAL-NET/web-editor-frontend/tree/mirf-editor (online; accessed: 25.04.2021).

[6] Insight Toolkit (ITK). — URL: https://itk.org/ (online; accessed: 14.12.2020).

[7] Kraska Tim. Northstar: an interactive data science system // Proceedings of the VLDB Endowment 11, 12 (2018). — 2018. — P. 2150–2164.

[8] MIRF 2.0 - A Framework for Distributed Medical Images Analysis / Alexandra Shvyrkova, Alexey Fefelov, Yurii Litvinov et al. // Proceedings of the Fifth Conference on Software Engineering and Information Management 2020 (SEIM 2020). — 2020. — URL: http://ceur-ws.org/Vol-2691/paper61.pdf (online; accessed: 15.12.2020).

[9] The Medical Imaging Interaction Toolkit (MITK). — URL: https://www.mitk.org/ (online; accessed: 14.12.2020).

[10] Mordvinov Dmitry, Litvinov Yurii, Bryksin Timofey. TRIK studio: Technical introduction // 2017 20th Conference of Open Innovations Association (FRUCT). — 2017. — P. 296–308.

[11] Musatian S., Lomakin A., Chizhova A. Medical images research framework // Fourth Conference on Software Engineering and Information Management (SEIM-2019). — 2019. — P. 60 – 66.

[12] Northstar. — URL: https://northstar.mit.edu/ (online; accessed: 07.12.2020).

[13] OpenCV (Open Source Computer Vision Library). — URL: https://opencv.org/ (online; accessed: 14.12.2020).

[14] React Flow. — URL: https://reactflow.dev/ (online; accessed: 12.03.2021).

[15] Rete.js. — URL: https://rete.js.org/ (online; accessed: 12.03.2021).

[16] Zehner Alexander, Szalo Alexander Eduard, Palm Christoph. GraphMIC: Easy Prototyping of Medical Image Computing Applications // Interactive Medical Image Computing (IMIC), Workshop at the Medical Image Computing and Computer Assisted Interventions (MICCAI 2015), 2015, Munich. — 2015. — P. 395 – 400.

[17] Алымова Д. А. Визуальный язык задания ограничений на модели в REAL.NET // сайт кафедры системного программирования СПбГУ. — URL: https://oops.math.spbu.ru/SE/diploma/2019/bmo/444-Alymova-report.pdf (online; accessed: 12.04.2021).

[18] Кидянкин М. В. Микросервисная архитектура DSM-платформы REAL.NET Web // сайт кафедры системного программирования СПбГУ. — URL: https://oops.math.spbu.ru/SE/YearlyProjects/vesna-2020/YearlyProjects/vesna-2020/mo-3rd-course/Kidyankin-report.pdf (online; accessed: 25.12.2020).

[19] Среда предметно-ориентированного визуального моделирования REAL.NET / Ю.В. Литвинов, Е.В. Кузьмина, И.Ю. Небогатиков, Д.А. Алымова // СПИСОК-2017. Материалы 7-й всероссийской

научной конференции по проблемам информатики. — 2017. — P. 80–89.

[20] Фефелов А. А. Архитектура фреймворка MIRF // сайт кафедры системного программирования СПбГУ. — URL: https://oops.math.spbu.ru/SE/diploma/2020/bmo/Fefelov-report.pdf (online; accessed: 14.12.2020).