Санкт-Петербургский государственный университет

БЕЛОКОННЫЙ Сергей Александрович

Выпускная квалификационная работа

Реинжиниринг веб-интерфейса профилировщика данных Desbordante

Уровень образования: бакалавриат

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

Основная образовательная программа *СВ.5162.2021 «Технологии программирования»*

Научный руководитель:
доцент кафедры информационно-аналитических систем, к. ф.-м. н., Михайлова Е. Г.

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

Рецензент:
Старший преподаватель кафедры ИАС, Смирнов К. К.

Санкт-Петербург
2025

Saint Petersburg State University

Sergey Belokonniy

Bachelor's Thesis

Reengineering the Desbordante data profiler web interface

Education level: bachelor

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

Programme *CB.5162.2021 "Programming Technologies"*

Scientific supervisor:
C.Sc, docent E.G. Mikhailova

Consultant:
Assistant G.A. Chernishev

Reviewer:
Senior lecturer K.K. Smirnov

Saint Petersburg
2025

Оглавление

Введение

В процессе разработки веб-приложения может происходить пересмотр и дополнение целей проекта. Важно предусмотреть возможность дальнейшего развития для того, чтобы успешно решать не только актуальные задачи, но и те, что могут возникнуть в будущем.

Для обеспечения устойчивости проекта к изменениям и новым требованиям, следует уделить особое внимание к его архитектуре. Модульная архитектура, логика приложения в которой разделена на отдельные компоненты, позволяет легко расширять функциональность приложения. Более того, такая архитектура уменьшает дублирование кода, поскольку устраняет необходимость в создании идентичных компонентов, благодаря возможности повторного использования уже существующих решений.

Desbordante [4] — высокопроизводительный профайлер данных, написанный на C++, одним из интерфейсов взаимодействия с которым выступает веб-приложение. Ядро проекта разрабатывается шесть лет, веб-приложение для него развивается уже три года. За это время было реализовано множество алгоритмов для более двадцати примитивов, паттернов, которые профайлер способен искать и валидировать. Изначально добавление функциональности примитива в веб-приложение требовало модификации кода ядра, в результате чего было реализованно всего лишь пять примитивов. В то время как ядро продолжало увеличивать свой функционал, веб-приложение столкнулось с трудностями, связанными с устаревшими технологиями и значительным техническим долгом. В следствии этого возможности веб-приложения уступают ядру, а добавление новой функциональности затруднительно.

В конце 2023 года у Desbordante появился интерфейс в виде библиотеки на языке Python, что упростило взаимодействие с его ядром. В связи с этим, а также стремлением предоставить все возможности Desbordante в веб-приложении, было принято решение о проведении реинжиниринга веб-интерфейса. Задачи по переносу существующей функциональности с возможностью дальнейшего горизонтального расширения стали основной темой данной работы.

1. Постановка задачи

Целью работы является разработка новой модульной архитектуры веб-интерфейса, модернизация существующих компонентов и внедрение новой функциональности. Для её выполнения были поставлены следующие задачи:

1. составить требования к новой версии веб-интерфейса;
2. обновить и заменить библиотеки для удовлетворения требованиям;
3. перенести страницы и функциональность в новый веб-интерфейс с применением выбранных библиотек с учётом требований;
4. провести апробацию новой версии веб-интерфейса.

2. Обзор

Для формулирования требований к новому веб-интерфейсу необходимо провести анализ существующих решений на рынке, а также оценить текущую реализацию с целью выявления её недостатков. В данном разделе рассматриваются аналогичные программные продукты и существующая версия веб-интерфейса, а также технологии, которые используются в новой реализации.

2.1. Обзор существующих решений

1. YData Profiling [15] — Python-библиотека с открытым исходным кодом, предоставляющая инструменты по анализу данных. Функциональность этой библиотеки ограничивается поверхностным анализом [23], а именно: нахождение примитивных статистик по колонкам (минимум, максимум, распределение значений), корреляцию данных между столбцами, а также визуализацией данных. Результат работы можно сохранить в виде HTML-файла. Требует локальной установки для получения результатов.

2. Metanome [27] — профайлер данных с открытым исходным кодом, написанный на языке Java. Этот профайлер поддерживает поиск некоторых закономерностей в данных, например, функциональные зависимости [2], порядковые зависимости [6], включающие зависимости [5] и многие другие. Поиск каждой зависимости возможен с помощью множества алгоритмов. Для работы с этим профайлером существует CLI, а также веб-интерфейс, написанный с помощью Angular. Требует локальной установки для получения результатов.

2.1.1. Выводы

В ходе исследования существующих решений был сделан вывод, что проекте YData Profiling хорошее представление результатов, но недостаточная глубина анализа. У Metanome всё наоборот: большая глубина анализа, но есть проблемы с представлением данных. Поэтому

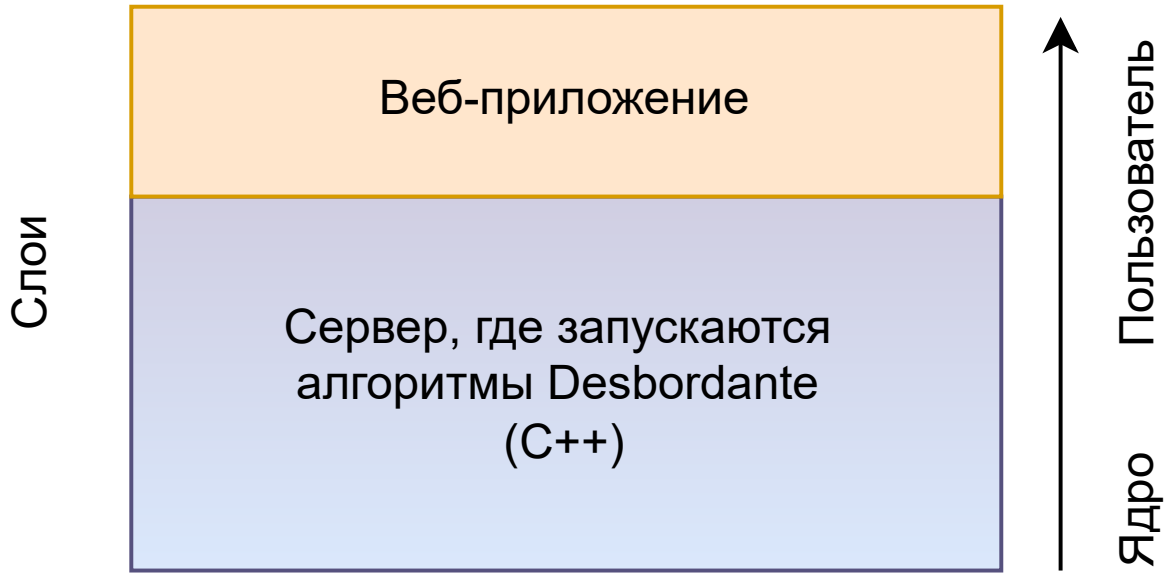


Рис. 1: Старая структура проекта веб-приложения

разработка продолжилась в DESBORDANTE, который обладает лучшими качествами из обоих проектов: есть хорошая глубина анализа и относительно хорошо интегрированный веб-интерфейс.

2.2. Обзор текущего решения

DESBORDANTE[1] — профайлер данных, основной код которого реализован на языке C++. Этот инструмент предназначен для обнаружения и верификации примитивов, паттернов, в наборах данных с использованием множества алгоритмов. DESBORDANTE предоставляет три интерфейса для взаимодействия: библиотека Python, для решения задач в коде, консольный интерфейс и веб-приложение. Библиотека Python и консольный интерфейс находятся в стадии активной разработки и обладают более широкой функциональностью по сравнению с веб-приложением.

В предыдущей версии веб-приложения для добавления новой функциональности требовалось обновлять не только веб-приложение, но и C++ код ядра. Также была проблема с тем, что код логики настройки примитивов и получения результата находился в двух файлах, из-за

[1] https://github.com/Desbordante (дата доступа: 8 мая 2025 г.).

чего разработка новых примитивов многими разработчиками была затруднена. Старую схему можно увидеть на Рисунке 1.

У Desbordante появился интерфейс в виде библиотеки на языке Python, с целью упрощения использования профайлера специалистами по анализу данных. Благодаря этому появилась возможность разделить код бекенда и ядра, избавиться от C++ кода в серверной части и упростить разработку бекенда. Таким образом стало легче делать быстрые изменения в веб-интерфейсе и бекенде.

Отдельной командой разработчиков разрабатывается новый бекенд[2] на языке Python, использующий библиотеку FastAPI для обработки запросов и Python интерфейс Desbordante для обработки данных. Этот проект призван упростить разработку бекенда с целью быстрого расширения функциональности.

В качестве образца функциональности профайлера были выбраны следующие примитивы для переноса в новый бекенд и веб-интерфейс:

- Ассоциативные правила (AR) [1] — правила, указывающие на вероятность появления одного элемента в наборе, при условии наличия другого элемента в этом же наборе. Вероятность вычисляется с помощью двух метрик: поддержки и достоверности.

- Метрические функциональные зависимости (MFD) [7] — зависимости, сопоставляющие набору атрибутов X другой набор атрибутов Y при условии, что метрика между каждым элементов в наборе атрибутов Y не больше некоторого параметра.

2.3. Обзор используемых технологий

До замены библиотек обновление проекта было невозможно из-за конфликтов зависимостей. Первоначально для кодогенерации запросов GraphQL использовалась библиотека `apollo-codegen` [9], однако её поддержка прекратилась в 2023 году[3]. В результате появления у Desbordante интерфейса в виде библиотеки на языке Python, был создан

[2] https://github.com/Desbordante/desbordante-server (дата доступа: 13 мая 2025 г.).
[3] https://github.com/apollographql/apollo-tooling (дата доступа: 13 мая 2025 г.).

новый сервер, который перешёл на использование OpenAPI. В связи с вышеуказанными обстоятельствами, в процессе реинжиниринга веб-приложения было принято решение перейти от использования GraphQL к OpenAPI.

Для организации запросов с сервером с помощью OpenAPI можно использовать следующие библиотеки:

- `@hey-api/openapi-ts` [10] — эта библиотека может генерировать богатые по функциональности API схемы и делать запросы к серверу, однако она уступает по таким параметрам, как размер и количество скачиваний.

- `openapi-fetch` [11] — в рамках этой библиотеки не предусмотрено генерирование схемы, для чего необходимо воспользоваться дополнительной библиотекой `openapi-typescript` [12], которая не включается в финальную сборку. Это позволяет `openapi-fetch` иметь меньший размер. Кроме того, у данной библиотеки большее количество скачиваний.

По вышеуказанным причинам выбор пал именно на `openapi-fetch`.

Для организации запросов и их глобального кеширования была выбрана библиотека TanStack Query [14], предоставляющая упрощенные механизмы для взаимодействия с сервером, включая инвалидацию и повторные запросы данных.

В проекте используюься следующие инструменты для автоматизации процессов

- Github Actions [25] — платформа, предназначенная для выполнения процессов на виртуальной машине в ответ на действия, совершаемые в репозитории. Она предоставляет возможность запуска как на машинах Github, так и на персональных виртуальных машинах. Одним из преимуществ данной платформы является обеспечение идентичной среды для каждого экземпляра запуска. Тем не менее, процессы инициализации виртуальной машины и

подготовки среды являются ресурсозатратными и требуют значительного времени для выполнения. С помощью Github Actions проект собирается и автоматически развёртывается на сервере.

- Git Hooks [17] — система, предназначенная для исполнения пользовательских скриптов при наступлении определённых событий — хуков (например, `git commit`). Для упрощения взаимодействия с хуками в проекте используется пакет husky [26], с помощью которого возможно осуществлять запуск скриптов, определённых в конфигурационном файле проекта, на любой машине. С помощью Git Hooks организована автоматическая проверка кода при коммитах.

Так же были обновлены библиотеки из предыдущей версии:

- NextJS [19] — основанный на ReactJS [22] фреймворк, предоставляет возможность создавать сильно оптимизированные по производительности сборки без необходимости в сложной настройке. Этот фреймворк поддерживает два вида роутера:
  - `pages router` [18] — предыдущий роутер в NextJS, обеспечивающий возможность навигации между страницами и удобное задание параметров в URL.
  - `app router` [16] — новый роутер в NextJS, не поддерживающий прямое задание параметров URL, однако предоставляющий более удобную файловую структуру для создания путей на веб-сайте и возможность интеграции страниц в шаблоны для повторного использования кода. Кроме того, данный роутер предлагает полезные инструменты для оптимизации поисковых систем (SEO). Поскольку разработчики NextJS рекомендуют использование `app router`, он был выбран для использования в веб-приложении.
- React Hook Form [20] — эта библиотека предоставляет возможности для создания контролируемых форм с валидацией.

- React Select [21] — эта библиотека используется как шаблон для создания селекторов в нужном стилем внутри проекта.

В процессе разработки также были использованы следующие инструменты:

- Strapi [30] — CMS (Content Managment System), которая используется для хранения информации о публикациях и о составе команды Desbordante.
- Fastapi [24] — Python-фреймворк для создания API сервера.
- Celery [13]— планировщик задач, позволяющий запускать задачи в воркерах.
- Rabbitmq [29] — брокер сообщений, который используется для передачи сообщений между частями бекенда.
- Minio [28] — хранилище файловых данных.

3. Требования к веб-интерфейсу

В ходе обсуждения проблем с командой бекенда и фронтенда, а также научным руководителем были собраны следующие требования:

1. Модульная архитектура — возможность расширения функционала без сильного изменения существующего кода.
2. Настройки взаимодействия с бекендом:
   - возможность авторизации;
   - взаимодействие с помощью спецификации OpenAPI;
   - подключение частей веб-приложения к серверу.
3. Перенос функциональности:
   - авторизация;
   - страница публикаций;
   - страница команды;
   - страница настройки примитива;
   - страницы вывода результатов примитивов AR и MFD.

4. Описание решения

Основной задачей текущей работы является создание платформы, функциональность которой можно дополнять примитивами.

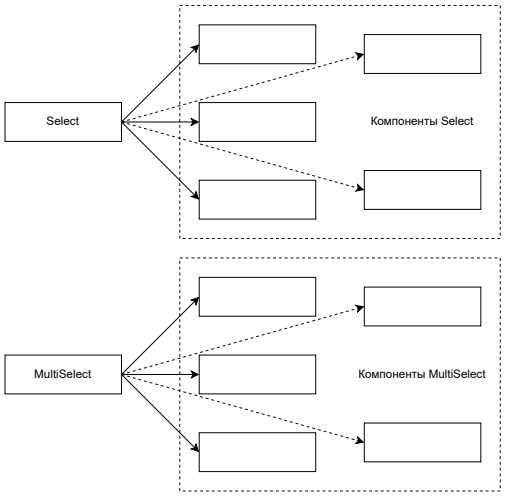
4.1. Обновление и замена устаревших библиотек

В процессе создания проекта все используемые пакеты были обновлены до актуальных версий, а фреймворк NextJS был обновлён с версии 13 до версии 14. В дополнение к этому был осуществлён переход с `pages router` на `app router`. Этот переход был выполнен с целью упрощения добавления новых страниц в веб-приложение и облегчения их разработки, поскольку весь код и стили сосредоточены в одном месте.

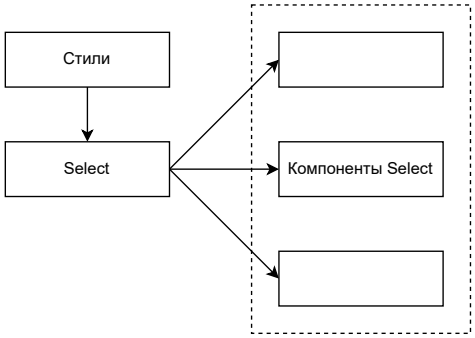
4.2. Перенос компонентов из прошлого проекта

В предыдущей версии веб-приложения каждое поле ввода данных в форме существовало в нескольких вариациях. В связи с этим были разработаны следующие компоненты, целью которых было разделение логики и устранение дублирования кода:

- Во всех компонентах ввода были убраны название поля, подсказка и сообщение об ошибке. Эти элементы были выделены в отдельный компонент — `FormField`.
- Компоненты, обёрнутые в контроллер, были удалены, и вместо них для управления вводом данных теперь применяется компонент `ControlledFormField`.
- Компоненты `Select` и `MultiSelect`, предназначенные для выбора одного или нескольких элементов, различались лишь незначительным набором внутренних компонентов и слегка отличающейся обработкой значений. Поскольку у этих компонентов было значительное количество общего кода, было принято решение об их объединении. На Рисунке 2а показана старая структура компонентов `Select` и `MultiSelect`, в которой дублировались внутренние



(a) Старая структура компонентов `Select` и `MultiSelect`



(b) Новая структура компонента `Select`

Рис. 2: Структура компонента `Select`

компоненты, отвечающие за стили. На Рисунке 2b показана новая структура `Select` в которой уменьшилось количество внутренних компонентов, а за стили стал отвечать набор параметров.

- Компоненты `NumberInput`, `NumberSlider` и `NumberRange`, предназначенные для ввода числовых данных, также содержали значительное количество общего кода. В связи с этим было принято решение об их объединении.

4.3. Архитектура и реимплементация

Диаграмма архитектуры веб-приложения показана на Рисунке 3. Далее будут рассмотрены отдельные компоненты, а также рассмотрены связи между ними.

4.3.1. Взаимодействие с бекендом и CMS

Для обеспечения типобезопасности, был реализован скрипт, который генерирует типы для параметров запросов и их ответов, после чего эти типы используются “клиентами”.

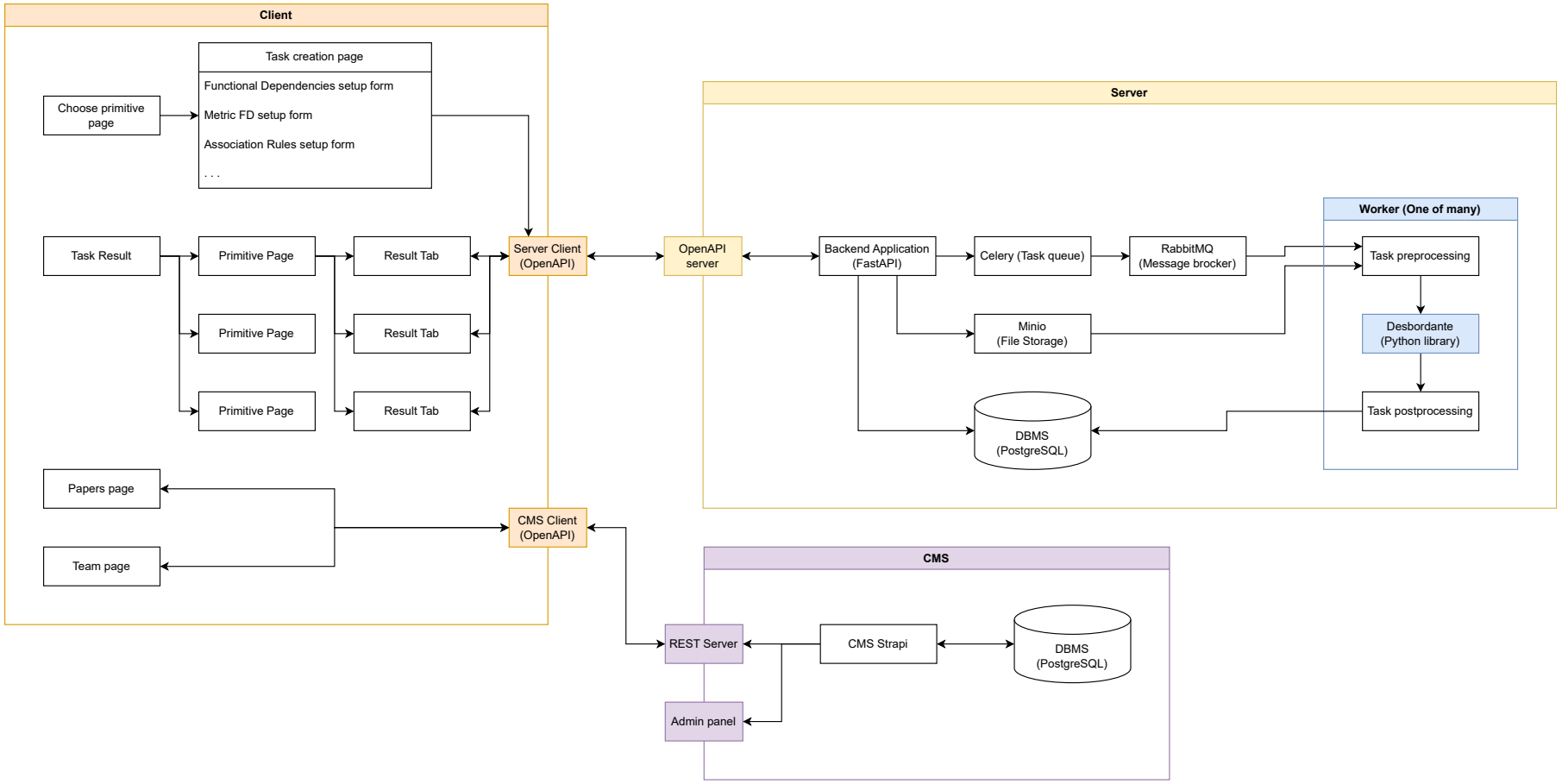
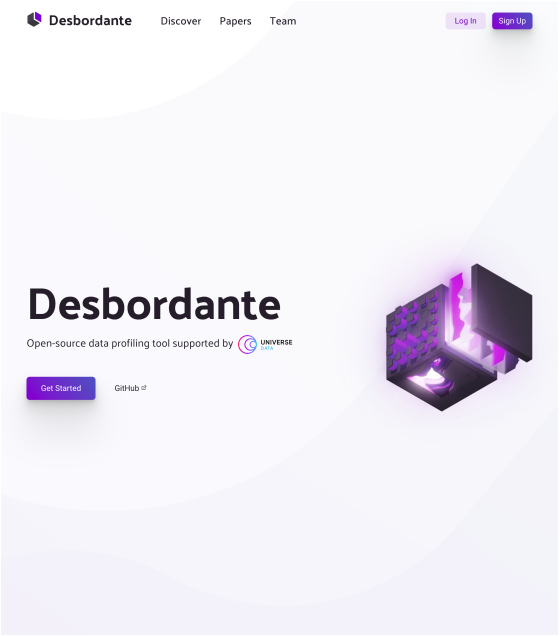
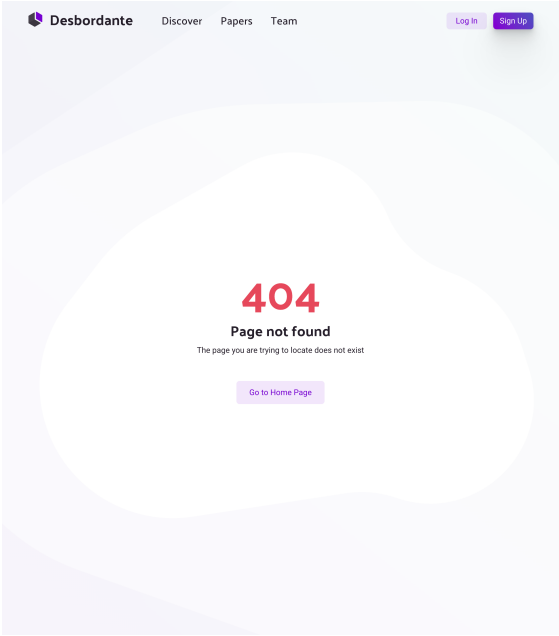


Рис. 3: Новая архитектура проекта веб-приложения



(a) Заглавная страница



(b) Страница “не найдено”

Рис. 4: Статичные страницы

Взаимодействие с бекендом происходит по спецификации OpenAPI с помощью “клиентов” — функций-обёрток, которые предлагают упрощённое создание запросов. Пример такой функции можно увидеть в Листинге 1. Такие “клиенты” позволяют абстрагироваться от API запросов, что даёт возможность сфокусироваться на данных.

Клиенты разделены по “сервисам”: сервер, аутентификации и CMS. Каждый из сервисов передаёт разные дополнительные параметры на сервер. Например, на эндпоинты аутентификации, для безопасности, не отправляется ничего, сервера отправляются аутентификационные cookie, а для CMS отправляется отдельный, секретный токен.

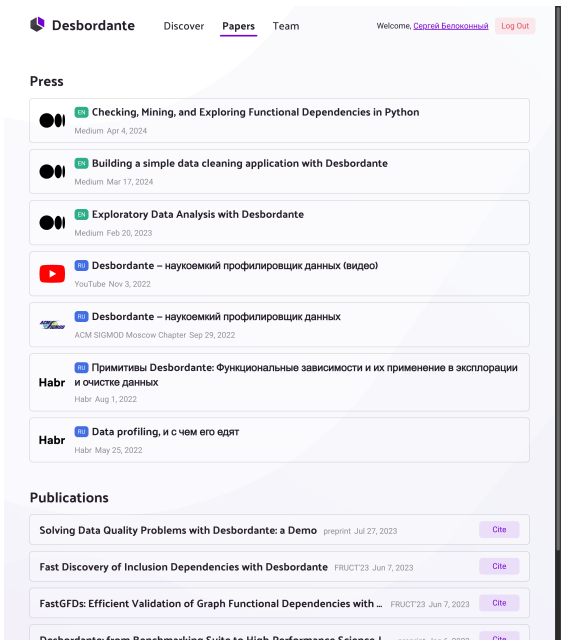
4.3.2. Статичные страницы

Эти страницы не меняются:

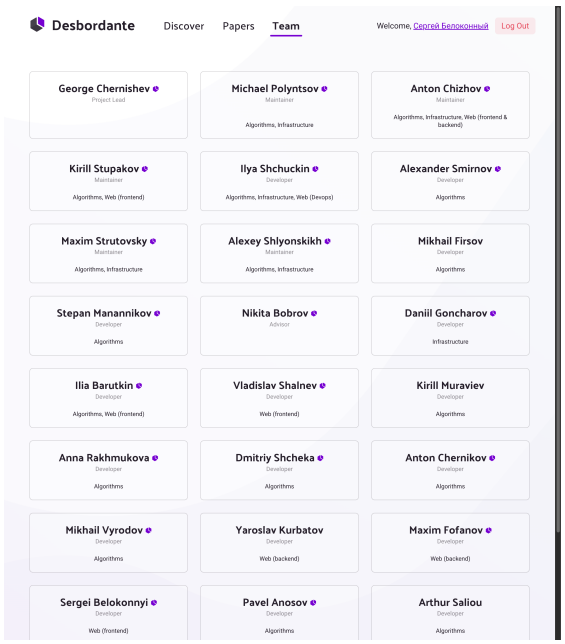
- Заглавная страница, показанная на Рисунке 4a.
- Страница “не найдено”, показанная на Рисунке 4b.

Листинг 1: Пример “клиента”.

```
export const serverFetchClient = createClient<paths>({
  baseUrl: baseUrl,
});

export const createQueryFn =
  <P extends ServicePaths<'/api', 'get'>>(
    path: P,
    params: ServiceParams<'/api', 'get', P>,
  ) =>
  async () => {
    const { data, error } =
        await serverFetchClient.GET(path, params);

    if (error) {
      throw error;
    }

    return data;
  };
```



(a) Страница публикаций



(b) Страница команды

Рис. 5: Страницы, генерируемые на сервере

4.3.3. Страницы публикаций и команды

Страницы публикаций и команды, в отличии от статичных страниц, содержат информацию, которая может изменяться, но эти страницы генерируются на сервере, а клиенту отдаётся готовая HTML страница. Так как информация на страницах меняется редко, они кешируются. Эти страницы показаны на Рисунке 5a и Рисунке 5b.

4.3.4. Страница настройки примитива

Страница настройки примитива разделена следующие модули:

- Точка входа в страницу.
- Макет страницы настройки примитива.
- Селектор пресетов.
- Отдельные формы для каждого из примитивов.

Далее будет рассказано подробнее про каждый из модулей.

Точка входа на страницу хранит в себе список компонентов форм для примитивов. На этой же странице происходит первоначальная проверка

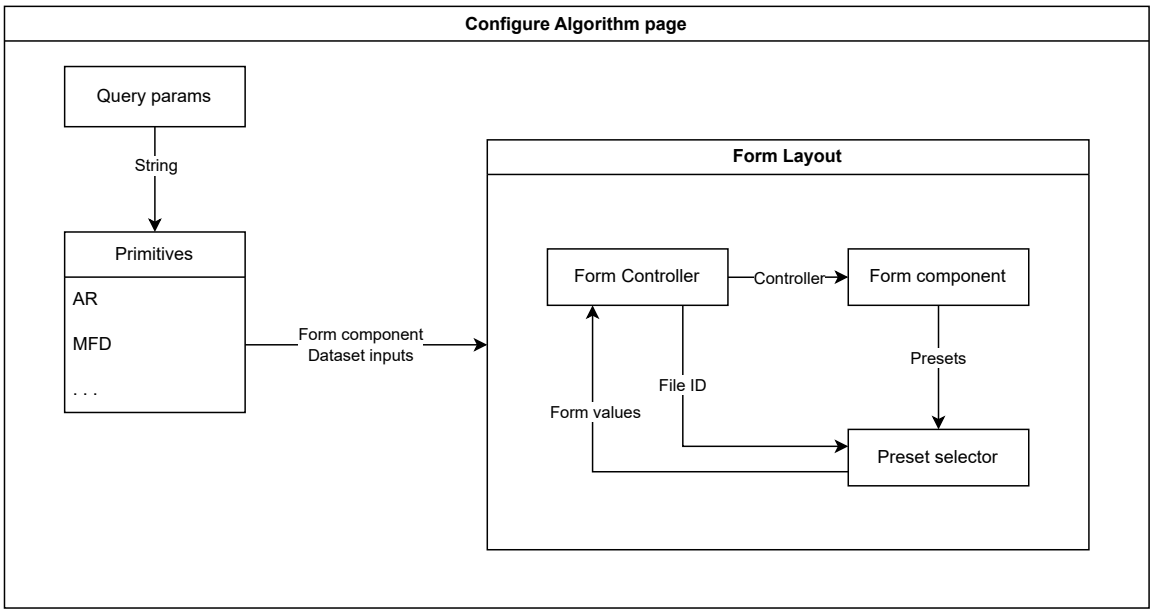


Рис. 6: Структура страницы настроек примитивов.

выбранного значения на то, что к нему можно сопоставить форму. На этой странице используется компонент макета формы, в которую передаётся компонент формы и набор полей для датасетов.

В макете “собирается” контент страницы настройки примитива. На этой странице происходит создание модели формы, создаются обработчики отправки формы, а также выводится компонент формы примитива и передаются пресеты в селектор пресетов.

В селекторе пресетов передаются ID файлов, которые выбрал пользователь, по этим ID запрашиваются имена файлов, после чего происходит фильтрация пресетов. При выборе пресета происходит автоматическое заполнение полей формы.

Форма примитива состоит из четырёх частей:

- Компоненты полей ввода.
- Функция очистки данных.
- Функций подготовки запроса к отправке.
- Пресеты для примитива.

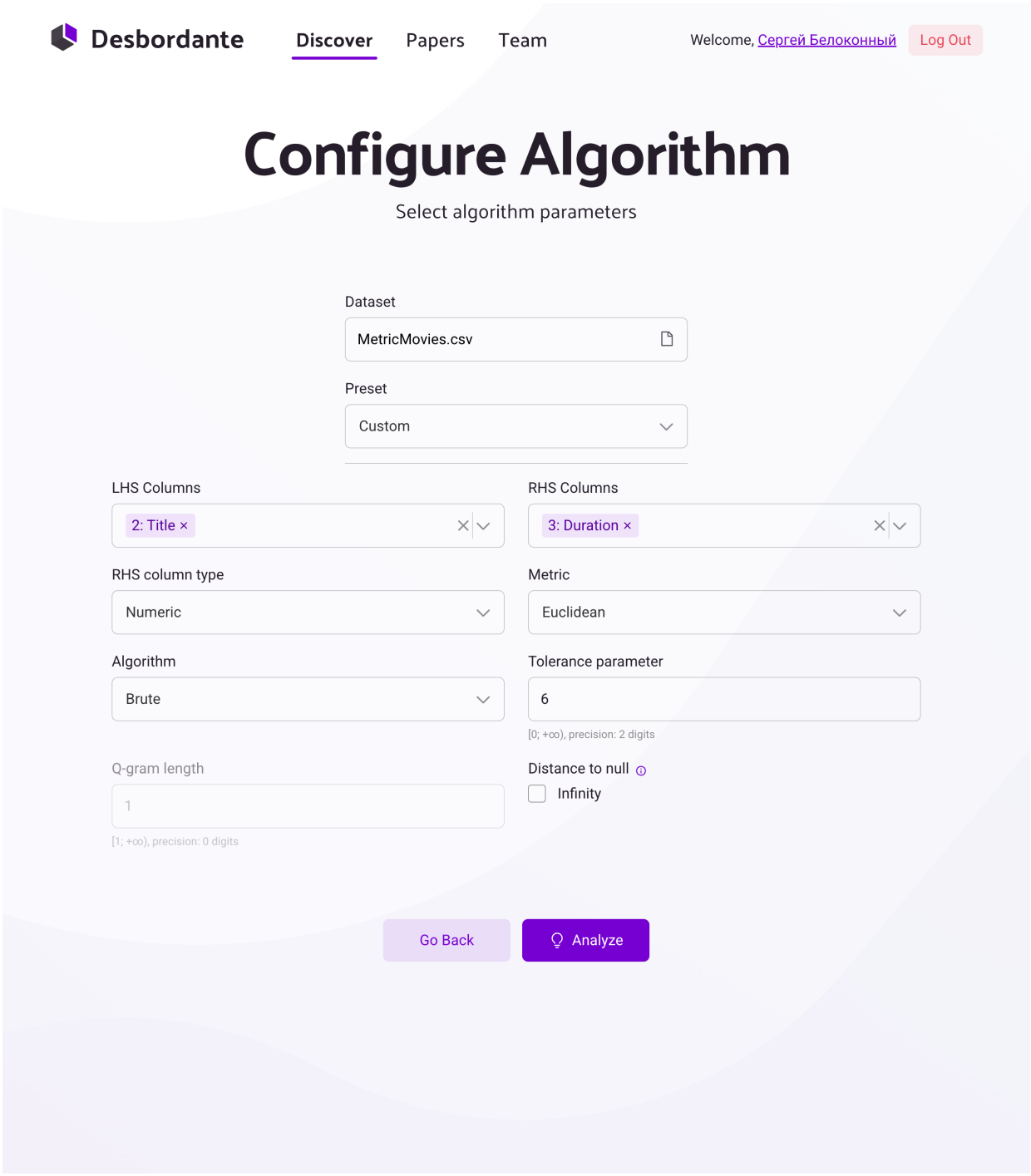


Рис. 7: Пример формы настройки примитива. В данном случае — MFD

Структуру страницы настроек примитивов можно посмотреть на Рисунке 6. Пример страницы можно увидеть на Рисунке 7.

4.3.5. Перенесённая функциональность

Для переноса функциональности было выбрано два примитива: AR и MFD.

Для обоих примитивов на фронтенде было сделано следующее:

- Перенесна форма настройки примитива.
- Перенесна страница результатов примитива.

Для ассоциативных правил на бекенде было сделано следующее:

- Создан API эндпоит для примитива.
- Написана сортировка по правым и левым столбцам, сортировка по достоверности.

Для метрических функциональных зависимостей на бекенде было сделано следующее:

- Создан API эндпоит для примитива.
- Написана сортировка по индексу строки в данных, индексу самой удалённой строки и по максимальному расстоянию.
- Написана фильтрация по кластеру.

4.4. Автоматизация

Для проверки стиля и типов кода используется Git Hooks, который вызывает HUSKY, который, в свою очередь, запускает скрипт проверки кода с помощью ESLint и Stylelint, а также проверку сообщения коммита на соответствие Conventional Commits.

Для проверки сборки и деплоя веб-приложения на сервер используется Github Actions.

5. Апробация

Апробация проводилась с помощью шкалы удобства использования системы (SUS) [3] и метода одного простого вопроса (SEQ) [8].

5.1. SUS

Шкала удобства использования системы — инструмент оценки уровня удобства взаимодействия пользователя с интерфейсом в общем. Эта шкала позволяет получить субъективные оценки по таким уровням как:

- Efficiency — эффективность.
- Effectiveness — результативность.
- Satisfaction — удовлетворённость.

Для вычисления результатов используется десять вопросов, на которые предлагается ответить по пятибальной шкале, где 1 — полное не соглание, а 5 — полное согласие:

1. Я хотел(а) бы часто пользоваться этой системой.
2. Я считаю, что система излишне сложная.
3. Я думаю, что система проста в использовании.
4. Я думаю, что мне понадобится поддержка специалиста, чтобы разобраться, как пользоваться системой.
5. Я думаю, что вся функциональность системы хорошо взаимосвязана.
6. Я думаю, что в системе слишком много несогласованности.
7. Мне кажется, что большинство людей быстро научатся пользоваться этой системой.
8. При использовании системы я счёл(сочла) её громоздкой.

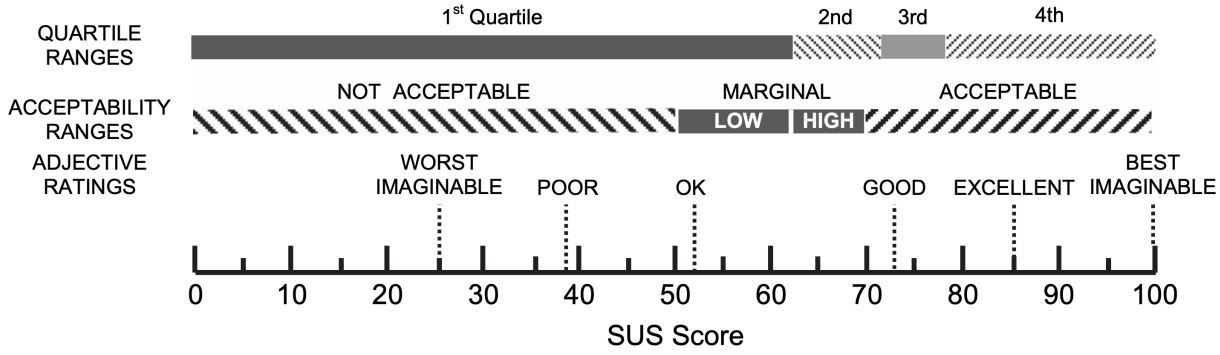


Рис. 8: Соответствие оценки SUS с квартильными диапазонами и диапазонами приемлемости. Рисунок взят из [3]

9. Я с уверенностью пользовался(пользовалась) этой системой.

10. Мне нужно было изучить дополнительные материалы, прежде чем я смог(смогла) приступить к работе с этой системой.

Для вычисления оценки от 0 до 100 используется следующая формула:

$$SUS = 2.5 \cdot (20 + \sum_{i=1,3,5,7,9} q_i - \sum_{i=2,4,6,8,10} q_i)$$

Итоговую оценку можно рассматривать в соответствии с Рисунком 8

5.2. SEQ

Метод одного простого вопроса — инструмент оценки сложности использования пользователя конкретных задач. Для этого пользователю предлагается набор сценариев, каждый из которых пользователь должен завершить и оценить по семибальной шкале, где 1 — очень сложно и 7 — очень просто. Оценкой в таком случае будет средним значением всех оценок.

Сценарии, которые были предложены пользователям:

1. Зарегистрироваться в системе.

2. Посмотреть участников команды.

3. Создать задачу и посмотреть её результаты.

Таблица 1: Результаты апробации.

Респондент	SUS	SEQ		
		q1	q2	q3
1	90	5	7	7
2	65	7	7	6
3	72.5	6	7	5
4	82.5	7	7	4
5	85	7	7	7
6	100	7	7	7
7	87.5	6	7	5
Среднее значение	83.2	6.4	7	5.8
			6.4	

5.3. Результаты

В опросах участвовало 7 человек. В итоге оценки оказались 83.2 балла для SUS, что соответствует оценке “GOOD” и 6.4 балла для SEQ, это означает, что тестирование пройдено успешно: процесс использования веб-приложения является эффективным и удобным. С результатами можно ознакомиться в Таблице 1.

Заключение

В результате работы были выполнены следующие задачи:

1. составлены требования к новой версии веб-интерфейса;
2. обновлены библиотеки;
3. перенесены страницы и функциональность в новый веб-интерфейс, а именно:
   - главная страница;
   - страница публикаций;
   - страница команды;
   - страница настройки примитива;
   - страницы результатов примитивов AR и MFD;
4. проведена апробация новой версии веб-интерфейса.

Ознакомится с проделанной работой можно по ссылке, изменения от пользователя с ником `pechenux`:
https://github.com/pechenux/desbordante-frontend

Список литературы

[1] Agrawal Rakesh, Srikant Ramakrishnan. Fast Algorithms for Mining Association Rules in Large Databases // Proceedings of the 20th International Conference on Very Large Data Bases. — VLDB '94. — San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1994. — P. 487–499.

[2] Approximate Discovery of Functional Dependencies for Large Datasets / Tobias Bleifuß, Susanne Bülow, Johannes Frohnhofen et al. // Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. — CIKM '16. — New York, NY, USA : Association for Computing Machinery, 2016. — P. 1803–1812. — URL: https://doi.org/10.1145/2983323.2983781.

[3] Bangor Aaron, Kortum Philip T., and James T. Miller. An Empirical Evaluation of the System Usability Scale // International Journal of Human–Computer Interaction. — 2008. — Vol. 24, no. 6. — P. 574–594. — https://doi.org/10.1080/10447310802205776.

[4] Desbordante: from benchmarking suite to high-performance science-intensive data profiler (preprint) / George A. Chernishev, Michael Polyntsov, Anton Chizhov et al. // CoRR. — 2023. — Vol. abs/2301.05965. — arXiv : 2301.05965.

[5] Inclusion Dependency Discovery: An Experimental Evaluation of Thirteen Algorithms / Falco Dürsch, Axel Stebner, Fabian Windheuser et al. // Proceedings of the 28th ACM International Conference on Information and Knowledge Management. — CIKM '19. — New York, NY, USA : Association for Computing Machinery, 2019. — P. 219–228. — URL: https://doi.org/10.1145/3357384.3357916.

[6] Langer Philipp, Naumann Felix. Efficient order dependency detection // The VLDB Journal. — 2016. — Apr. — Vol. 25, no. 2. — P. 223–241. — URL: https://doi.org/10.1007/s00778-015-0412-3.

[7] Metric Functional Dependencies / Nick Koudas, Avishek Saha, Divesh Srivastava, Suresh Venkatasubramanian // 2009 IEEE 25th International Conference on Data Engineering. — 2009. — P. 1275–1278.

[8] Sauro Jeff, Dumas Joseph S. Comparison of three one-question, post-task usability questionnaires // Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. — CHI '09. — New York, NY, USA : Association for Computing Machinery, 2009. — P. 1599–1608. — URL: https://doi.org/10.1145/1518701.1518946.

[9] Библиотека `apollo-codegen`. — URL: https://www.npmjs.com/package/apollo-codegen (дата обращения: 14 мая 2025 г.).

[10] Библиотека `@hey-api/openapi-ts`. — URL: https://www.npmjs.com/package/@hey-api/openapi-ts (дата обращения: 14 мая 2025 г.).

[11] Библиотека `openapi-fetch`. — URL: https://www.npmjs.com/package/openapi-fetch (дата обращения: 14 мая 2025 г.).

[12] Библиотека `openapi-typescript`. — URL: https://www.npmjs.com/package/openapi-typescript (дата обращения: 14 мая 2025 г.).

[13] Документация Celery. — URL: https://docs.celeryq.dev/en/stable/ (дата обращения: 14 мая 2025 г.).

[14] Документация TanStack Query. — URL: https://tanstack.com/query/v5/docs/framework/react/overview (дата обращения: 14 мая 2025 г.).

[15] Документация YData Profiling. — URL: https://docs.profiling.ydata.ai/latest/ (дата обращения: 14 мая 2025 г.).

[16] Документация app router. — URL: https://nextjs.org/docs/14/app/building-your-application (дата обращения: 14 мая 2025 г.).

[17] Документация git по Git Hooks. — URL: https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks (дата обращения: 14 мая 2025 г.).

[18] Документация pages router. — URL: https://nextjs.org/docs/14/pages/building-your-application (дата обращения: 14 мая 2025 г.).

[19] Официальный сайт NextJS. — URL: https://nextjs.org (дата обращения: 14 мая 2025 г.).

[20] Официальный сайт React Hook Form. — URL: https://react-hook-form.com/ (дата обращения: 14 мая 2025 г.).

[21] Официальный сайт React Select. — URL: https://react-select.com/home (дата обращения: 14 мая 2025 г.).

[22] Официальный сайт ReactJS. — URL: https://react.dev/ (дата обращения: 14 мая 2025 г.).

[23] П Б., Э Б. Практическая статистика для специалистов Data Science. Внесерийные книги. — БХВ-Петербург, 2018. — ISBN: 9785977539746. — URL: https://books.google.ru/books?id=l_6MDwAAQBAJ.

[24] Сайт Fastapi. — URL: https://fastapi.tiangolo.com/ (дата обращения: 14 мая 2025 г.).

[25] Сайт Github Actions. — URL: https://github.com/features/actions (дата обращения: 14 мая 2025 г.).

[26] Сайт Husky. — URL: https://typicode.github.io/husky/ (дата обращения: 14 мая 2025 г.).

[27] Сайт Metanome. — URL: https://hpi.de/naumann/projects/data-profiling-and-analytics/metanome-data-profiling.html (дата обращения: 14 мая 2025 г.).

[28] Сайт Minio. — URL: https://min.io/ (дата обращения: 14 мая 2025 г.).

[29] Сайт Rabbitmq. — URL: https://www.rabbitmq.com/ (дата обращения: 14 мая 2025 г.).

[30] Сайт Strapi. — URL: https://strapi.io/ (дата обращения: 14 мая 2025 г.).