Санкт-Петербургский государственный университет

Кафедра информационно-аналитических систем

Группа 20.Б09-мм

СТУПАКОВ Кирилл Валерьевич

Разработка клиентской части веб-приложения Desbordante

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
Ассистент кафедры ИАС Г.А. Чернышев

Санкт-Петербург
2022

Оглавление

1. Введение

1.1. Профилирование данных

Профилирование данных — процесс, заключающийся в нахождении закономерностей, исследовании характера распределения, оценке качества [5, 8]. В последние время появляется всё больше инструментов для сбора статистических данных, что влечёт за собой значительное увеличение размера датасетов. Создаются алгоритмы, решающие различные задачи профилирования посредством поиска зависимостей в данных.

1.2. Функциональные зависимости

Определение: Отношение R удовлетворяет функциональной зависимости $X \to Y \iff \forall t_1,\ t_2 \in R\ (t_1[X] = t_2[X]) \implies (t_1[Y] = t_2[Y])$. Другими словами, это отношение между двумя множествами столбцов в таблице, которое выполняется, если для любых двух строк значения всех столбцов из левой части зависимости совпадают, то должны совпадать и значения столбцов из правой части [15].

Зависимости интересны в первую очередь прикладным учёным, обладающим большими наборами данных. Зависимости в них — это закономерности, на основе которых можно формулировать гипотезы. Основная проблема заключается в том, что количество потенциально возможных зависимостей экспоненциально зависит от количества атрибутов в датасете, поэтому даже для не самых больших датасетов список найденных зависимостей может быть довольно длинный. Предстояло придумать способ наглядной визуализации этого списка, что позволило бы удобно с ним работать.

2. Постановка задачи

2.1. Цель работы

Целью работы является разработка клиентской части веб-приложения платформы Desbordante. Для её выполнения были поставлены следующие задачи:

1. найти способ наглядной визуализации найденных зависимостей;
2. выбрать подходящие технологии, библиотеки и фреймворки для реализации клиентской части;
3. реализовать минимальный рабочий продукт.

2.2. Требования к приложению

Приложение должно позволять пользователю:

1. загружать свои датасеты в виде .csv-таблицы;
2. менять конфигурацию загруженного файла:
    - наличие заголовка;
    - символ-разделитель.
3. выбирать датасет из небольшого списка предустановленных;
4. выбирать алгоритм и его конфигурацию:
    - для неточных алгоритмов:
        - максимальную длину левой части;
        - допустимую погрешность.
    - количество потоков;
5. изучать результаты работы соответствующего алгоритма (найденные зависимости);
6. получать доступ к результату алгоритма с другого устройства.

3. Обзор

3.1. Профилирование данных

3.1.1. Metanome

Metanome — совместный проект Hasso Plattner Institute (Германия) и Qatar Computing Research Institute — это открытая платформа для профилирования данных, реализованная на языке Java. Она содержит множество различных алгоритмов профилирования, в том числе алгоритмы поиска функциональных зависимостей. Платформа позволяет удобно запускать эти алгоритмы на необходимых наборах данных. Благодаря встроенному frontend-клиенту поддерживается управление через браузер.

3.1.2. Desbordante

Desbordante реализовывалась как альтернатива платформе Metanome. По сравнению с ней Desbordante работает быстрее и требует меньше памяти, в среднем в два раза [6]. Её разработка ведётся с июля 2020 года. На момент начала данной практики проект имел открытый исходный код на C++, реализовал пять алгоритмов поиска функциональных зависимостей, работал под Linux и имел консольный интерфейс.

3.2. Технологии

3.2.1. React

React — это JavaScript-библиотека для проектирования пользовательских интерфейсов, разработанная компанией Facebook [11]. Будучи самой популярной из всех доступных библиотек-альтернатив (Vue.js и Angular) [17], она обладает огромным сообществом пользователей, что позволяет быстро найти ответы на все интересующие вопросы. К основным достоинствам React и других фронтенд-библиотек можно отнести:

- возможность создания переиспользуемых компонентов;

- требуется меньше кода для реализации интерактивности;
- большая база готовых компонентов;
- простота тестирования за счёт существования специальных библиотек [12].

3.2.2. TypeScript

TypeScript — это «надстройка» над JavaScript [14]. Она решает многие проблемы этого языка, такие как:

- отсутствие строгой типизации;
- неудобство ООП за счёт отсутствия интерфейсов и наследования.

Наличие этих механизмов в языке позволяет IDE генерировать подсказки и находить примитивные ошибки во время написания кода.

3.2.3. D3.js

D3.js — это библиотека для манипуляции веб-страницей, используемая для визуализации данных [4]. Она является стандартом в данной области и даёт возможности:

- манипулировать HTML на основе данных;
- создавать интерактивные элементы, анимации и переходы;

Однако, большинство этих функций уже имеется в React, поэтому ответственность между данными библиотеками делится следующим образом:

- D3 занимается только подсчётом (позиция элементов, конвертация чисел в логарифмическую шкалу, интерполяция и пр.);
- React занимается отрисовкой элементов на основе расчётов D3.

3.2.4. Chart.js

Chart.js — это набор настраиваемых диаграмм, написанных на D3 [3]. Она предоставляет широкие возможности по настройке визуальной части диаграмм, а именно:

- возможность изменения параметров отдельных частей диаграммы при интеракции с пользователем;
- гибкая настройка параметров текста, таких как шрифт и начертание.

Среди альтернатив Chart.js рассматривались библиотеки Recharts и react-minimal-pie-chart [2]. Обе они проигрывают в возможностях кастомизации и сложности достижения требуемого вида диаграммы.

3.2.5. SCSS

SCSS — язык для написания стилей [13], компилирующийся в CSS и добавляющий некоторые полезные механизмы, такие как:

- более удобная работа с переменными;
- поддержка вложенных селекторов;
- функции.

Как альтернатива ему рассматривался LESS. Оба эти препроцессора имеют очень похожие возможности. SCSS, однако, более прост в использовании и имеет большую популярность [16].

3.2.6. Axios

Axios — это клиент для отправки HTTP-запросов [1]. Среди его преимуществ перед единственным аналогом Fetch [7] можно выделить:

- возможность получить прогресс загрузки файла;
- поддержку большего количества браузеров;

- возможность отмены запроса и установки таймаута;
- автоматическую трансформацию передаваемых и получаемых данных в JSON.

3.2.7. Jest

Jest — библиотека для unit-тестов [10]. Она позволяет:

- управлять тестами с помощью CLI;
- запускать тесты только для изменённых участков файла с помощью интерактивного режима.

Среди его альтернатив можно выделить Enzyme и Mocha [9]. Они обе предоставляют более широкий набор возможностей, чем Jest, однако эти возможности не будут использованы в рамках данной работы, из-за чего был сделан выбор в пользу более простой библиотеки Jest.

4. Реализация

4.1. Визуализация зависимостей

Для визуализации зависимостей была выбрана следующая стратегия:

1. для каждого атрибута считается взвешенная сумма

$$S(A) = \sum_f \frac{\text{isInLeftSide}(f, A)}{\text{leftSideLength}(f)}$$

$$\text{isInLeftSide}(f, A) = \begin{cases} 1, \text{если A присутствует в левой части } f; \\ 0, \text{иначе.} \end{cases}$$

$\text{leftSideLength}(f)$ — длина левой части зависимости f

2. по суммам строятся круговые диаграммы;

3. исходя из данных на диаграмме, пользователь выбирает атрибуты и просматривает только зависимости с их участием.

Мотивация выбора именно такой формулы состоит в том, что чем длиннее у зависимости левая часть, тем меньше её практическая ценность. Таким образом, «более ценные» зависимости будут сильнее остальных влиять на сумму, относящуюся к данному атрибуту.

4.2. Архитектура приложения

Структура веб-приложения изображена на Рис. 1.

4.2.1. Клиентская часть

Было решено разбить весь процесс использования приложения на четыре экрана:

1. домашняя страница, на которой, будет производиться выбор файла, алгоритма и его конфигурации;

2. экран с прогрессом загрузки датасета на сервер;
3. экран с результатом работы алгоритма, на котором находятся:
   (a) диаграммы с атрибутами;
   (b) список зависимостей.

Также должно присутствовать глобальное поле `taskId`, получаемое из URL-адреса и обеспечивающее независимую работу всех четырёх экранов. Это позволяет получить доступ к результатам анализа файла по ссылке.

4.2.2. Серверная часть

1. после получения `POST`-запроса с файлом и алгоритмом сервер добавляет запись о новом задании в БД;
2. посылается сообщение на сервер Apache Kafka (в определённый топик) о добавлении нового задания;
3. клиент в случае успешного добавления задания получает ответ от сервера, в котором содержится идентификатор нового задания `taskId` или сообщение об ошибке;
4. consumer обращается к серверу Kafka и получает новое задание, затем начинает выполнять его, по мере выполнения внося необходимые изменения в БД;
5. клиент обращается раз в 100 миллисекунд к серверу, узнавая статус выполнения задания. Когда оно будет выполнено, клиент получит информацию о выполненном задании и внесёт соответствующие изменения в интерфейс.

4.2.3. Преимущества архитектуры

Представленная архитектура имеет ряд преимуществ:

1. возможность горизонтальной масштабируемости;

2. чёткое разделение задач между компонентами.

4.3. Коммуникация с сервером

Работа с сервером предоставляется по схеме REST API. Интерфейс представляет следующие точки подключения:

- `GET`:
    1. `getAlgsInfo` содержит:
        - список поддерживаемых алгоритмов;
        - список параметров к ним;
        - максимальный поддерживаемый размер файла;
        - список поддерживаемых расширений файла.
    2. `getTaskInfo` принимает поле `taskId` и отдаёт:
        - имя анализируемого файла;
        - прогресс выполнения и название фазы;
        - найденные зависимости, если анализ закончился.
    3. `getSnippet` принимает `taskId` и возвращает представление анализируемого файла в виде двумерного массива.
- `POST`:
    1. `createTask` возвращает `taskId` созданного задания и принимает:
        - анализируемый файл;
        - название алгоритма и его конфигурацию.
- `DELETE`:
    1. `cancelTask` останавливает задачу с указанным `taskId`.

4.4. Тестирование

Был написан пакет из 52 тестов для валидаторов форм и правильной отрисовки компонентов. Также было проведено ручное тестирование UI и опробован ряд датасетов размером до 500 МБ.

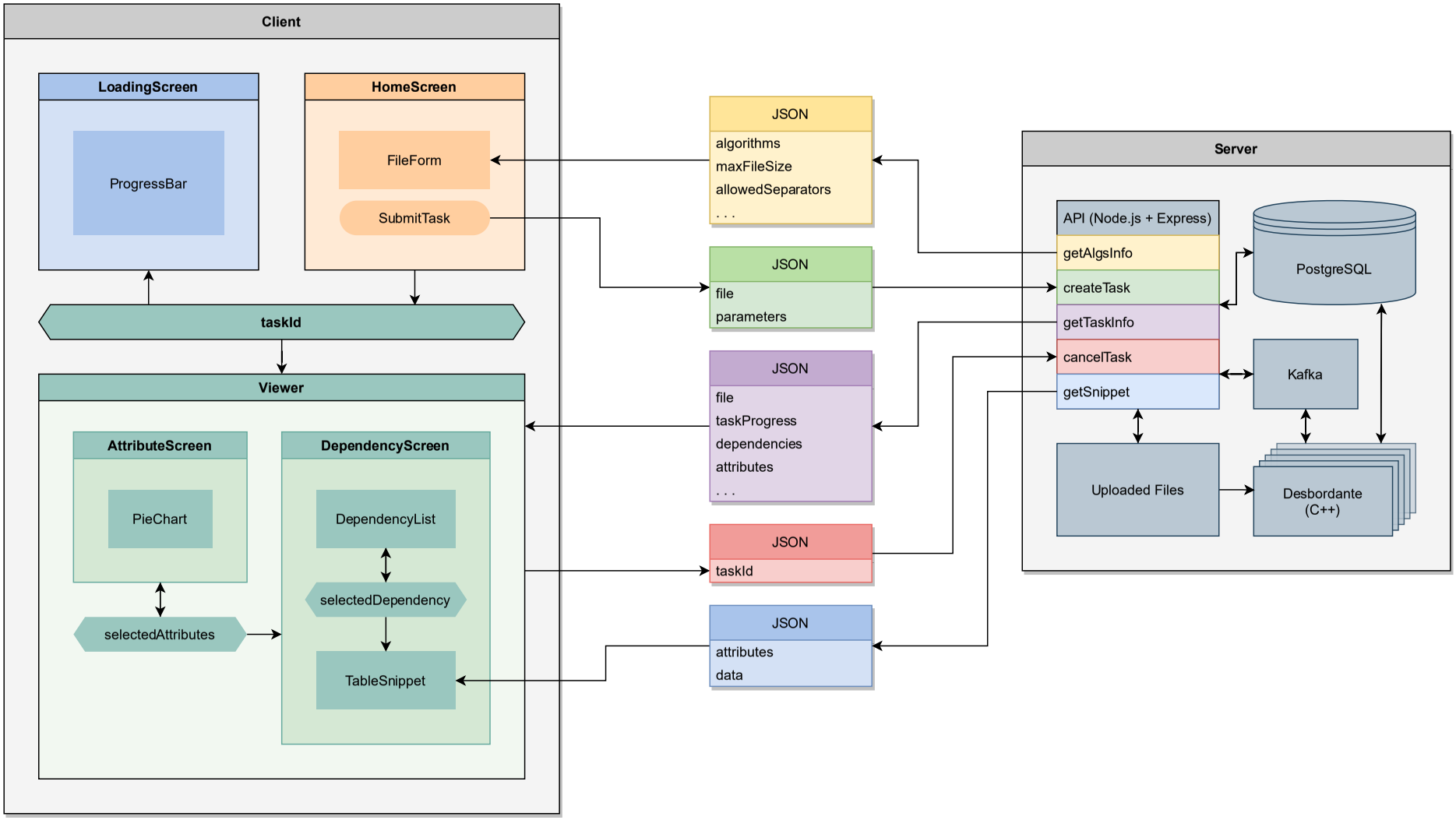

Рис. 1: Схема работы веб-приложения

5. Заключение

По итогам данной работы была реализована клиентская часть приложения платформы Desbordante, а именно:

1. найден способ наглядной визуализации найденных зависимостей;
2. выбраны подходящие технологии, библиотеки и фреймворки для реализации клиентской части;
3. получен минимальный рабочий продукт.

Ссылка на GitHub-репозиторий: vs9h/Desbordante (имя пользователя — kirill-stupakov)

Список литературы

[1] Axios Documentation. — URL: https://axios-http.com/docs/intro (online; accessed: 2021-10-29).

[2] Chart.js Alternatives. — URL: https://stackshare.io/js-chart/alternatives (online; accessed: 2021-11-20).

[3] Chart.js Documentation. — URL: https://www.chartjs.org/docs/latest (online; accessed: 2021-10-29).

[4] D3.js Documentation. — URL: https://github.com/d3/d3/wiki (online; accessed: 2021-10-29).

[5] Data Profiling / Ziawasch Abedjan, Lukasz Golab, Felix Naumann, Thorsten Papenbrock. — First edition. — Morgan & Claypool Publishers, 2018. — nov. — Vol. 10 of Synthesis Lectures on Data Management.

[6] Desbordante: a Framework for Exploring Limits of Dependency Discovery Algorithms / Maxim Strutovskiy, Nikita Bobrov, Kirill Smirnov, George Chernishev // 2021 29th Conference of Open Innovations Association (FRUCT). — 2021. — P. 344–354.

[7] Fetcch API Documentation. — URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch (online; accessed: 2021-11-21).

[8] Ilyas Ihab F., Chu Xu. Data Cleaning. — New York, NY, USA : Association for Computing Machinery, 2019. — ISBN: 978-1-4503-7152-0.

[9] Jest Alternatives. — URL: https://www.slant.co/options/12697/alternatives/~jest-alternatives (online; accessed: 2021-11-20).

[10] Jest Documentation. — URL: https://jestjs.io/docs/getting-started (online; accessed: 2021-10-29).

[11] React Documentation. — URL: https://reactjs.org/docs/getting-started.html (online; accessed: 2021-10-29).

[12] React Testing Library. — URL: https://testing-library.com (online; accessed: 2021-10-29).

[13] SCSS Documentation. — URL: https://sass-lang.com/documentation (online; accessed: 2021-10-29).

[14] TypeScript Documentation. — URL: https://www.typescriptlang.org/docs/handbook/intro.html (online; accessed: 2021-10-29).

[15] Введение в функциональные зависимости. — URL: https://habr.com/ru/company/JetBrains-education/blog/473882 (online; accessed: 2021-10-29).

[16] Сравнение SCSS и LESS. — URL: https://www.keycdn.com/blog/sass-vs-less (online; accessed: 2021-11-13).

[17] Статистика по загрузкам самых популярных фронтенд-библиотек. — URL: https://www.npmtrends.com/react-vs-vue-vs-@angular/core (online; accessed: 2021-10-29).