Санкт-Петербургский государственный университет

Группа 20Б.09-мм

БАРУТКИН Илья Дмитриевич

Разработка клиентской части веб-приложения Desbordante

Отчёт по учебной практике

Научный руководитель:
ассистент кафедры ИАС Г.А. Чернышев

Санкт-Петербург
2022

Оглавление

Введение

Профилирование данных [1] — это процесс извлечения метаданных из данных. Метаданные — это, к примеру, размер файла, время создания, авторство. Но это также и любые закономерности, сокрытые в данных. Закономерности, которые могут присутствовать в данных, описываются с помощью различных примитивов. Примитив — это некоторое описание правила, действующего над данными (или их частью), описанное математическими методами. Такими примитивами являются, к примеру, функциональная зависимость, ассоциативные правила, условные функциональные зависимости.

Отношение R удовлетворяет функциональной зависимости $X \to Y \iff \forall t_1, t_2 \in R \quad t_1[X] = t_2[X] \implies t_1[Y] = t_2[Y]$ где X, Y это наборы столбцов. 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 [11].

Условные функциональные зависимости [7] — это зависимости, которым разрешено выполняться только на определённом подмножестве множества атрибутов. Данное подмножество называется шаблоном.

Ассоциативные правила [12] — это заключения «если-то», которые помогают показать вероятность взаимосвязей между элементами данных в больших наборах.

Поскольку инструменты профилирования данных в основном нацелены на пользователей, эффективная визуализация результатов имеет первостепенное значение. Только тогда пользователи смогут интерпретировать результаты и реагировать на них [1]. Существуют инструменты профилирования с возможностью визуализации результатов, такие как Profiler [9], Metanome и Desbordante. Все они реализуют пользовательский интерфейс с помощью web-технологий. Desbordante — первый инструмент, предназначенный специально для высокопроизводительного поиска зависимостей. Он написан на C++, расширяем и имеет

полностью открытый исходный код [2]. Однако пользовательский интерфейс инструмента требовал, с точки зрения разработчиков и первых пользователей, доработки.

1. Постановка задачи

1.1. Цель работы

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

1. определить структуру страниц приложения с учетом изменений в дизайне
2. выбрать подходящие технологии для реализации клиентской части
3. сверстать макеты приложения
4. реализовать взаимодействие с серверной частью приложения и отображение получаемых данных с учетом изменений API сервера
5. проверить работоспособность приложения

1.2. Требования к приложению

Были поставлены следующие требования к приложению. Оно должно позволять пользователю

1. создавать аккаунт с подтверждением электронной почты
2. восстанавливать пароль
3. загружать и сохранять для дальнейшего использования датасеты в формате .csv
4. настраивать файл после загрузки
5. выбирать предустановленные датасеты
6. выбирать примитив для поиска на отдельной странице

- функциональные зависимости
- условные функциональные зависимости
- ассоциативные правила
- кластеры для поиска ошибок

7. выбирать и настраивать алгоритм на отдельной странице
8. просматривать результаты работы алгоритма в виде списка зависимостей или круговой диаграммы

2. Обзор существующих решений

2.1. Metanome

Metanome — совместный проект Hasso Plattner Institute 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.

2.2. Desbordante

Desbordante реализовывалась как альтернатива платформе Metanome. По сравнению с ней Desbordante работает быстрее и требует меньше памяти, в среднем в два раза [2]. Её разработка ведётся с июля 2020 года. В проекте Desbordante уже имеется веб-клиент, который имеет следующие недостатки:

- создание задачи происходит на одной странице с большим количеством настроек,
- нет возможности сохранить свои файлы для дальнейшего использования,
- сайт не работает в браузере Safari.

Кроме того, к моменту начала работы дизайнерами было разработано новое оформление интерфейса и элементов управления.

3. Реализация

3.1. Используемые технологии

3.1.1. React

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

- возможность создания переиспользуемых компонентов
- требуется меньше кода для реализации интерактивности
- большая база готовых компонентов
- простота тестирования за счёт существования специальных библиотек

3.1.2. Next.js

Next.js — фреймворк для проектирования React-приложений, дополнительно предоставляющий функционал маршрутизации, кеширования изображений, конфигурации webpack и отрисовки на стороне сервера (server-side rendering) [5].

3.1.3. GraphQL

GraphQL — это язык запросов для API, и серверная среда выполнения для запросов. Распространенной альтернативой взаимодействия с сервером является REST. Однако, в отличие от него GraphQL позволяет:

- агрегировать данные из разных компонентов пользовательского интерфейса,

- получать только необходимые поля в результатах одного запроса в разном контексте,
- обеспечивать представление данных, основанное на графах,
- обеспечивать статическую типизацию данных, получаемых по API.

В качестве клиента используется пакет Apollo [10], легко интегрируемый в React приложение и поддерживающий статическую типизацию ответов сервера.

3.1.4. TypeScript

TypeScript [6] — это расширения языка JavaScript, предоставляющий дополнительный слой. Этим слоем является система типов которая решает проблемы этого языка, такие как:

- отсутствие строгой типизации,
- отсутствие интерфейсов и наследования, из-за чего неудобно вести разработку в объектно-ориентированном стиле.

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

3.1.5. ESlint

ESlint [3] — это статический анализатор кода на языке JavaScript и TypeScript. Он необходим для проверки синтаксической корректности кода, обеспечения единого стиля кода и обнаружения нежелательных явлений в коде, таких как неиспользуемые переменные и опечатки.

3.2. Архитектура приложения

3.2.1. Страницы

Весь процесс использования приложения было решено разделить на следующие страницы:

- Домашняя страница
- Страницы создания задачи (Wizard)
  - Выбор примитива
  - Выбор и настройка датасета
  - Выбор параметров примитива
- Страницы с результатами
  - Визуализация зависимостей в виде круговых диаграмм
  - Визуализация зависимостей в виде списка
  - Предосмотр датасета с разбиением на страницы
  - Обзор настроек текущей задачи

Как страницы создания задачи, так и страницы с результатами имеют общие элементы разметки. Для сокращения количества кода и упрощения переиспользования общих элементов, были разработаны компоненты макета для этих групп страниц: WizardLayout и ReportsLayout соответственно.

3.2.2. Коммуникация с сервером

Взаимодействие с сервером происходит посредством языка GraphQL. Аргументы запросов к серверу и ответы типизированы и генерируются автоматически из схемы с серверной части с помощью GraphQL-клиента Apollo.

3.2.3. Контексты

Для того, чтобы получать некоторые глобальные данные в разных компонентах, мы использовали React-контексты:

- ErrorContext хранит в своем состоянии последнюю ошибку и предоставляет методы для ее показа, скрытия, изменения

- AuthContext предоставляет информацию о текущем пользователе

Благодаря использованию контекстов, мы инкапсулировали глобальное состояние и разделили логически связанные глобальные данные.

3.2.4. Формы с параметрами примитивов

Для создания форм было решено использовать наиболее популярный фреймворк react-hook-form. Так как в проекте используется TypeScript, то было решено применить его возможности по проверке типов при создания форм для того, чтобы уменьшить риск появления опечаток или ошибок при рефакторинге и последующей поддержке форм. Список полей ввода для каждого примитива определен в компоненте страницы в виде объекта следующего типа:

```
Record <MainPrimitiveType,
        Partial <Record <keyof AlgorithmProps, FormInput> > >
```

Тип MainPrimitiveType — это перечисление возможных примитивов, AlgorithmProps — объединение полей всех форм (FDForm & CFDForm & ARForm & TypoFDForm), а FormInput — специальный тип React-компонента, совместимый с фреймворком react-hook-form.

Объект определяется с помощью react-hook функции useMemo, из-за чего элементы формы не пересоздаются при повторной отрисовке (рендеринге), если только не был изменен алгоритм. Указанные в форме данные сохраняются в URL браузера и восстанавливаются из него, благодаря чему данные не теряются при переходе на другие страницы (например, на предыдущий этап создания задачи).

3.2.5. Server-side rendering

В приложении присутствуют страницы, где после первичной загрузки запрашиваются дополнительные данные по API с помощью AJAX. По время обработки AJAX-запроса страница выглядит пустой, что может вызывать дискомфорт для пользователя. Server-side rendering —

это функционал фреймворка Next.js, позволяющий производить отрисовку страниц на стороне сервера. С помощью него был значительно улучшен пользовательский опыт.

3.3. Верстка и стили

Для верстки страниц использован наиболее современный подход — разметка flexbox [8]. Он позволяет определять простой однонаправленный макет, позиционируя элементы как по направлению макета, так и перпендикулярно ему. На снимках экрана представлено сравнение страницы старого и нового приложения.

Рис. 1: Выбор файла в старой версии приложения

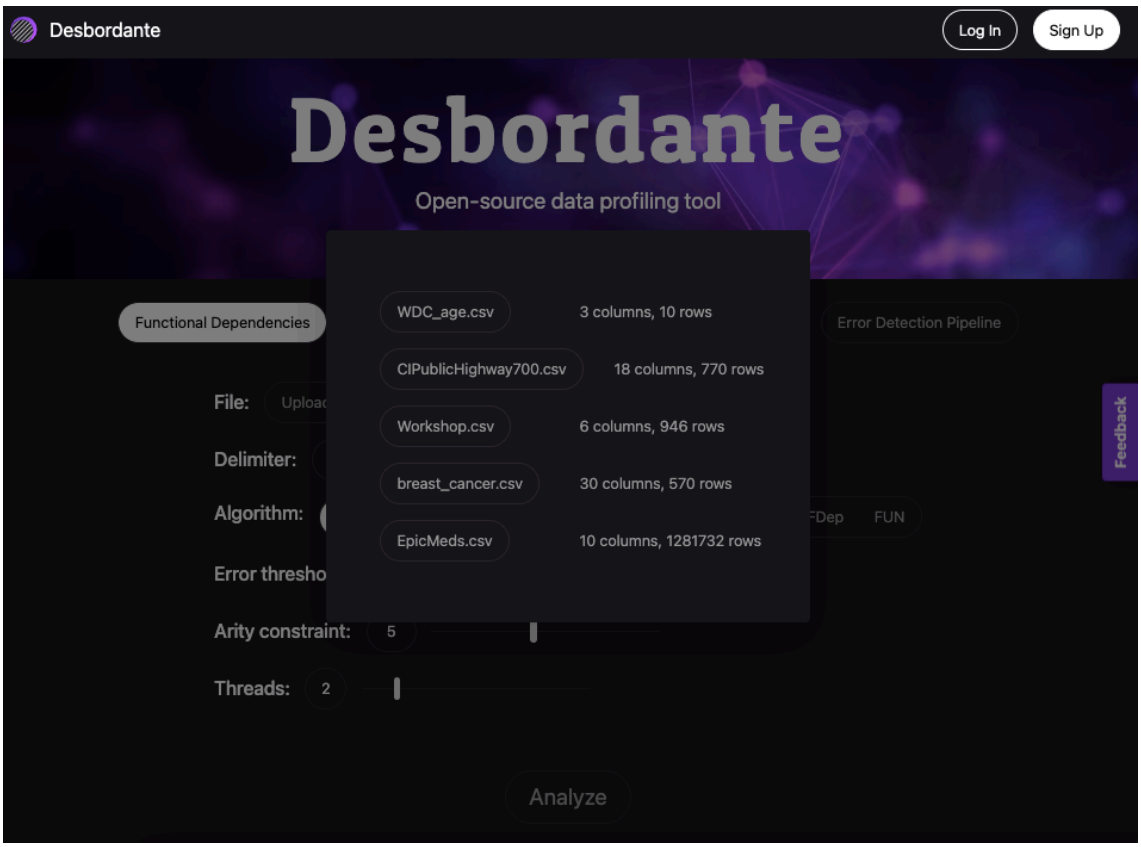
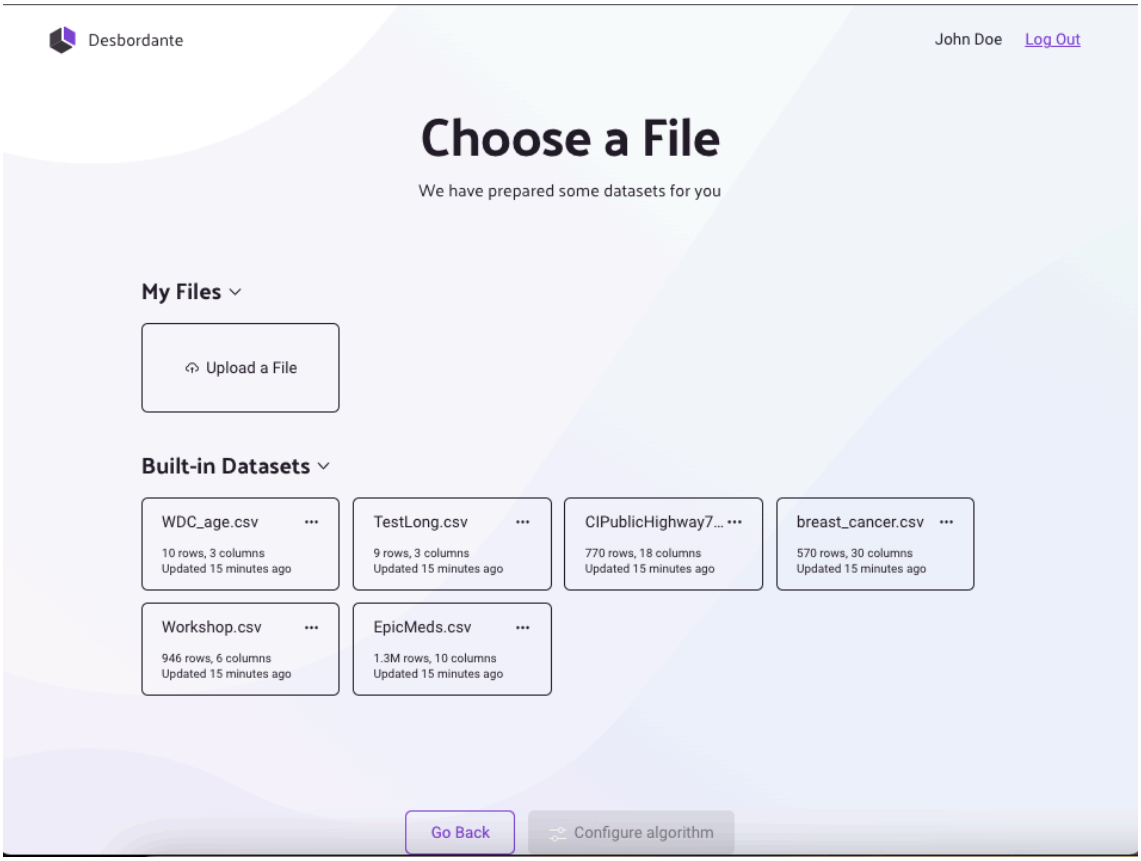


Рис. 2: Выбор файла в новой версии приложения

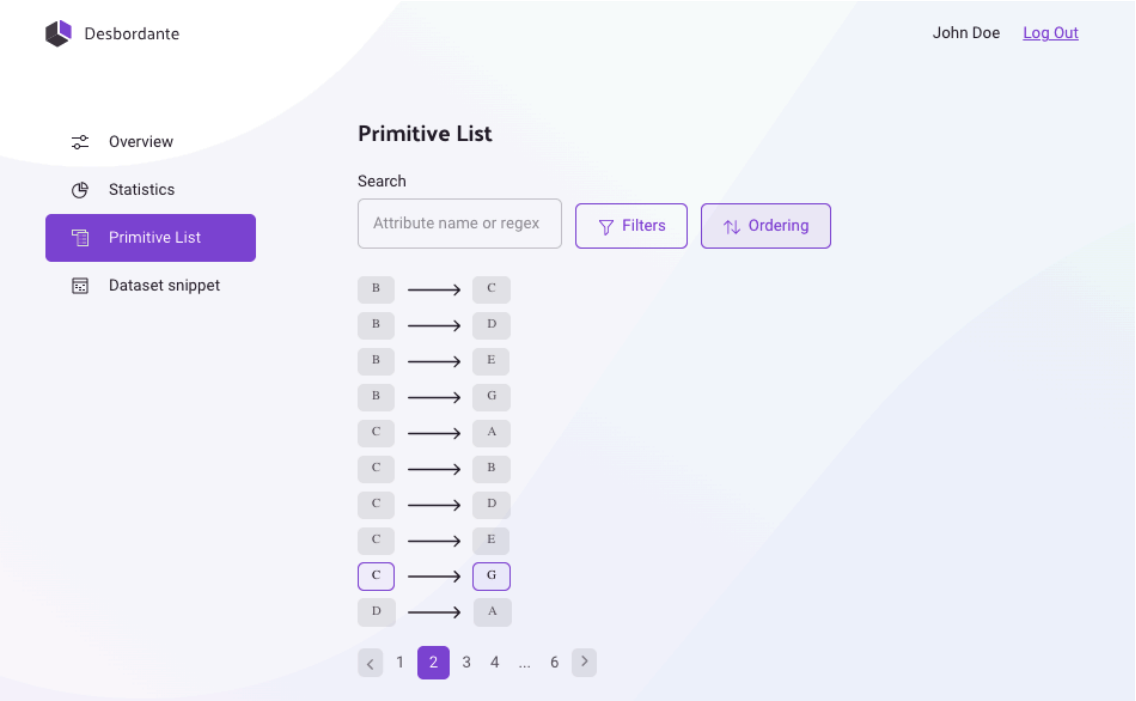


4. Тестирование

Было произведено ручное тестирование приложения. Проверен функционал регистрации пользователя и подтверждения электронной почты. Загружены тестовые csv-файлы посредством drag-and-drop. Проверена отзывчивость верстки при изменении размера экрана.

Функционал создания задачи был проверен для всех примитивов. Поиск функциональных зависимостей был произведен на датасетах с 30 колонками, 570 строками и 10 колонками, 1281732 строками с помощью алгоритмов Pyro, TaneX, FastFDs, FD mine, DFD, FDep, FUN. Поиск условных функциональных зависимостей был произведен на датасетах с 30 колонками, 570 строками и 6 колонками 946 строками с помощью алгоритма CTane.

Рис. 3: Просмотр списка найденных в тестовом датасете зависимостей



Заключение

В ходе данной работы были выполнены следующие задачи:

- определена структура страниц приложения с учетом изменений в дизайне
- выбраны подходящие технологии и обосновано их использование
- выполнена верстка главной страницы, страниц создания задачи и просмотра найденных зависимостей
- реализовано взаимодействие с серверной частью приложения
- проведена проверка работоспособности приложения

  Ссылка на GitHub-репозиторий https://github.com/vs9h/Desbordante/tree/webapp-redesign (имя пользователя — iliya-b).

Список литературы

[1] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. 2015. Profiling relational data: a survey. The VLDB Journal 24, 4 (August 2015), 557–581. https://doi.org/10.1007/s00778-015-0389-y

[2] 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.

[3] Eslint Documentation. — URL: https://eslint.org/docs/latest/ (online; accessed: 2022-09-22).

[4] React Documentation. — URL: https://reactjs.org/docs/getting-started.html (online; accessed: 2022-09-22).

[5] Next.js Documentation. — URL: https://nextjs.org/learn/basics/create-nextjs-app (online; accessed: 2022-09-22).

[6] TypeScript Documentation. — URL: https://www.typescriptlang.org/docs/handbook/intro.html (online; accessed: 2022-09-24).

[7] W. Fan, F. Geerts, L. V. S. Lakshmanan and M. Xiong, Discovering Conditional Functional Dependencies, 2009 IEEE 25th International Conference on Data Engineering, 2009, pp. 1231-1234, doi: 10.1109/ICDE.2009.208.

[8] MDN Web Documentation. — URL: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox (online; accessed: 2022-09-24).

[9] Sean Kandel, Ravi Parikh, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Profiler: Integrated statistical analysis and visualization for data quality assessment. In Proceedings of Advanced Visual Interfaces (AVI), pages 547–554, 2012.

[10] Apollo documentation. — URL: https://www.apollographql.com/docs/react/ (online; accessed: 2022-09-24).

[11] Thorsten Papenbrock, Jens Ehrlich, Jannik Marten, Tommy Neubert, Jan-Peer Rudolph, Martin Schönberg, Jakob Zwiener, and Felix Naumann. 2015. Functional dependency discovery: an experimental evaluation of seven algorithms. Proc. VLDB Endow. 8, 10 (June 2015), 1082–1093. https://doi.org/10.14778/2794367.2794377

[12] Charu C. Aggarwal. 2015. Data Mining: The Textbook. ISBN 3319141414. Springer Publishing Company, Incorporated.