

Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 21.Б15-мм

# Разработка клиентской части для примитива “статистики датасета” в веб-приложении Desbordante

***ШАЛЬНЕВ Владислав Александрович***

Отчёт по учебной практике  
в форме «Производственное задание»

Научный руководитель:  
ассистент кафедры ИАС Г. А. Чернышев

Санкт-Петербург  
2022

# Оглавление

<b>Введение</b>	<b>3</b>
<b>1. Постановка задачи</b>	<b>5</b>
1.1. Цель работы . . . . .	5
1.2. Требования к приложению . . . . .	5
<b>2. Обзор существующих решений</b>	<b>7</b>
<b>3. Реализация</b>	<b>8</b>
3.1. Используемые технологии . . . . .	8
3.2. Архитектура . . . . .	10
3.3. Модальная карточка . . . . .	12
3.4. Страница . . . . .	14
3.5. Server Side Rendering . . . . .	15
3.6. Взаимодействие с сервером . . . . .	16
3.7. Тестирование . . . . .	17
<b>Заключение</b>	<b>18</b>
<b>Список литературы</b>	<b>19</b>

# Введение

В современном мире огромное количество различных данных окружает нас повсеместно. Информация генерируется буквально на каждый клик, при пролистывании страницы сайта, просмотре видео и фотографий в социальных сетях, не говоря уже о датчиках смартфонов, ежедневно собирающих колоссальное количество сведений об устройстве. Эту информацию можно использовать. Так, музыкальные сервисы анализируют длительность прослушивания, временной и географический контекст, поисковую историю и другие атрибуты для создания профиля, описывающего индивидуальные предпочтения пользователя, и настройки системы рекомендаций так, чтобы человек как можно дольше оставался в сервисе. Другой пример — обучившиеся на датасетах медицинских обследований нейросети, способные по малейшему признаку выявить заболевание даже на ранних стадиях развития [16].

Следовательно, в сырой и неструктурированной информации могут быть скрыты закономерности, потенциально полезные для бизнеса и науки. У компаний и групп ученых появляются задачи, связанные с трудоемким поиском и извлечением таких закономерностей, их решением занимается область профилирования данных [1]. По определению, это процесс исследования данных, а также выявление и извлечение из них статистических характеристик или же метаданных. Если рассматривать файл, его метаданными являются, например, размер, название, даты создания и редактирования и авторство. В более широком смысле это понятие включает в себя любые неочевидные зависимости, такие как характер распределения, наличие выбросов или параметры выборки, то есть искомые закономерности.

Для наукоемкого профилирования разрабатываются специальные инструменты, которые занимаются поиском зависимостей, охарактеризованных с помощью различных примитивов — описаний правил, действующих над данными или какой-то их частью, заданных с помощью математических функций и методов. В контексте табличных данных можно рассмотреть как пример функциональную зависимость [15]. Бу-

дем говорить, что отношение удовлетворяет функциональной зависимости  $X \rightarrow Y$ , где  $X, Y$  — атрибуты в таблице, если для любых двух строк равенство значений по атрибуту  $X$  означает равенство значений по атрибуту  $Y$ . Подобных примитивов существует великое множество, каждый подкреплен теорией и предназначен для решения определенных задач.

Одним из профайлеров данных, созданных для поиска зависимостей в табличных данных посредством примитивов является Desbordante. Его ядро написано на C++ для обеспечения высокой производительности, он расширяем и имеет открытый исходный код [3]. Поскольку сервис предназначен для широкого круга людей, в который входят ученые из различных сфер, люди, работающие с финансами и аналитики данных, необходимо визуализировать результаты в понятном для неспециализированного пользователя виде. Для решения данной задачи, в качестве пользовательского интерфейса Desbordante использует веб-сервис. Помимо наукоемкого, Desbordante может осуществлять классическое профилирование данных. К нему относится поиск основных статистик пользовательского датасета, например, подсчет суммы, среднего значения, определение минимума и максимума для каждого столбца таблицы. Для этих целей в Desbordante существует отдельный примитив “статистики датасета”. В рамках учебной практики предстояло разработать клиентскую часть для данного примитива.

# 1. Постановка задачи

## 1.1. Цель работы

Целью работы является разработка страницы и модальной карточки основных статистик датасета для веб-приложения Desbordante. Для ее выполнения были поставлены следующие задачи:

1. реализовать компоненты, необходимые для разработки страницы и модальной карточки статистик;
2. сверстать макеты данного раздела приложения с помощью технологий, используемых командой Desbordante;
3. реализовать взаимодействие с серверной частью приложения;
4. проверить работоспособность реализованной части приложения и протестировать компоненты с помощью модульного тестирования.

## 1.2. Требования к приложению

К клиентской части разделов основных статистик датасета были поставлены следующие требования. Модальная карточка должна:

1. предоставлять пользователю возможность запуска поиска основных статистик для встроенных и загруженных датасетов;
2. давать возможность выбора количества процессорных потоков на этапе запуска обработки;
3. отображать сообщение, в случае, если поиск статистик для данного датасета недоступен;
4. оповещать пользователя, если во время обработки на сервере произошла ошибка;
5. осуществлять отображение результатов обработки:

- для всего датасета в целом (overview);
  - отдельно для каждой колонки.
6. предоставлять табличный режим отображения результатов для колонок;
  7. позволять пользователю выбирать колонку для отображения результатов из выпадающего списка;
  8. для каждой колонки в списке выводить определенный в процессе поиска для нее тип и параметр Categorical;
  9. иметь возможность перехода на отдельную страницу статистик для данного датасета.

Страница основных статистик датасета должна позволять пользователю:

1. просматривать результаты обработки:
  - для всего датасета (overview);
  - для всех колонок датасета одновременно, при этом изначально отображать только часть данных с возможностью раскрытия списка статистик по требованию пользователя;
2. переключать в табличный режим отображения:
  - общие характеристики датасета (overview);
  - результаты обработки каждой колонки.

В разделе общих для всего датасета статистик (overview) должны быть отображены все данные, присылаемые сервером в соответствующем поле JSON-объекта в формате “имя-значение”. Из результатов обработки для отдельной колонки необходимо визуализировать следующие характеристики: `name`, `index`, `type`, `distinct`, `isCategorical`, `count`, `avg`, `STD`, `skewness`, `kurtosis`, `min`, `max`, `sum`, `quantile25`, `quantile50`, `quantile75`.

## 2. Обзор существующих решений

Pandas Profiling — библиотека для создания статистического отчета о датафрейме (DataFrame) pandas [2]. Инструмент позволяет представлять для колонок датасета в виде интерактивного HTML отчета как минимум следующую информацию:

- выведенный тип, уникальные значения, пропущенные значения;
- квантильные статистики: минимум, максимум, квантили (quantile);
- описательные статистики: среднее значение (mean), моду (mode), стандартное отклонение, сумму, скошенность (skewness), коэффициент эксцесса (kurtosis);
- самые частые значения.

Дополнительно позволяет строить гистограммы, матрицы корреляции и точечные диаграммы взаимодействия признаков, а также отображает общую для датасета информацию: количество строк и столбцов, наличие дубликатов и размер занимаемой памяти.

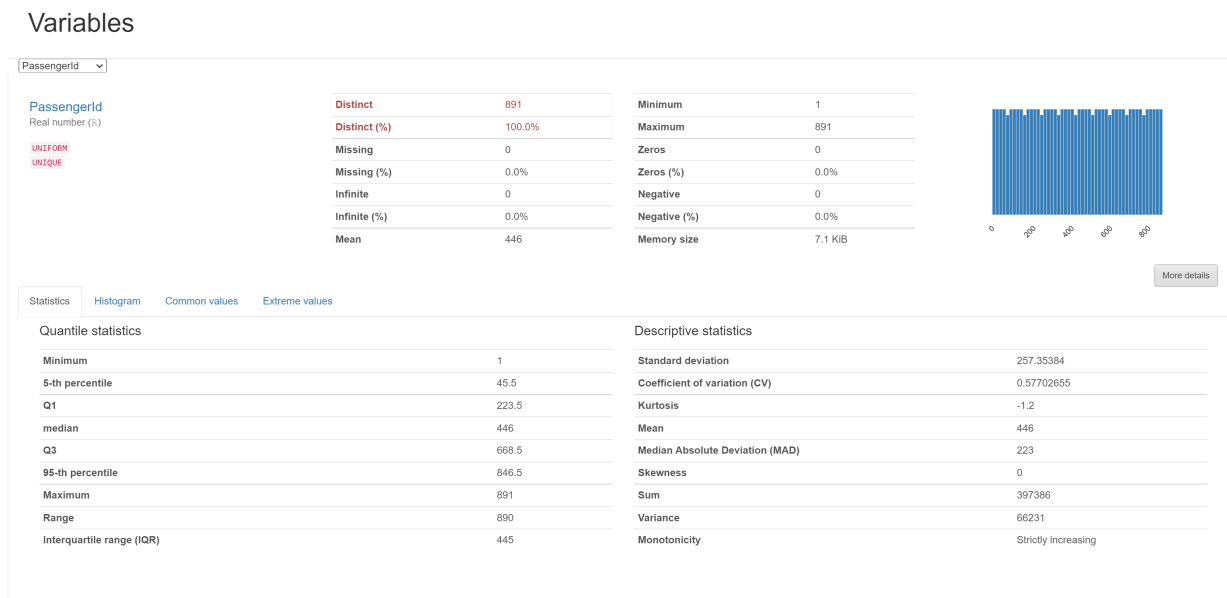


Рис. 1: Простые статистики колонки датасета, полученные с помощью pandas-profiling

## 3. Реализация

### 3.1. Используемые технологии

#### 3.1.1. React

React — это JavaScript-библиотека для создания пользовательских интерфейсов [10]. Входит в число самых распространенных веб-технологий [13], вследствие чего обладает огромным сообществом разработчиков, что позволяет быстро находить решения большинства проблем. К преимуществам React можно отнести:

- декларативный подход к описанию интерфейсов;
- создание переиспользуемых компонентов;
- большая база библиотек с готовыми компонентами;
- использование технологии виртуального DOM.

#### 3.1.2. Next.js

Next.js — гибкий React-фреймворк для проектирования веб-приложений, созданный компанией Vercel [9]. Как фреймворк, Next.js берет на себя конфигурацию React и сопутствующих инструментов сборки, предоставляя разработчику дополнительные возможности в виде маршрутизации, кеширования изображений, отрисовки на стороне сервера и оптимизации приложения.

#### 3.1.3. TypeScript

TypeScript — это надстройка над языком программирования JavaScript, разрабатывается компанией Microsoft [14]. Примеры добавляемых языком возможностей:

- аннотации типов и проверка типизации на этапе компиляции;
- вывод типов;



- переработанные классы, интерфейсы, механизм наследования из ООП;
- обобщенные типы и функции (generics).

Наличие данных механизмов способствует обнаружению некоторых видов ошибок на этапе написания кода, позволяет IDE генерировать подсказки, а также предоставляет возможность применения подходов объектно-ориентированного программирования.

#### **3.1.4. GraphQL и Apollo Client**

GraphQL — язык запросов для API и серверная среда выполнения запросов [5]. Является альтернативой традиционному REST API подходу. GraphQL обладает следующими преимуществами:

- облегчает агрегацию данных из нескольких источников;
- позволяет клиенту запрашивать только нужные ему данные, что улучшает производительность;
- обеспечивает строгую типизацию данных.

В качестве GraphQL-клиента используется Apollo Client, предоставляющий возможность интеграции с React с помощью соответствующей библиотеки. Умеет кешировать полученные с сервера данные, имеет поддержку TypeScript, позволяет типизировать ответы сервера, обрабатывать ошибки, а также использует современные возможности React (React Hooks).

#### **3.1.5. Sass**

Sass — компилируемый в CSS язык таблиц стилей. Позволяет использовать переменные, вложенные правила, миксины (mixins), функции и шаблоны [12]. Имеет полностью совместимый с CSS синтаксис SCSS.

### 3.1.6. ESLint

ESLint — инструмент для статического анализа кода, написанного на языке JavaScript [4]. Используется для поддержания качества и обеспечения единого стиля написания кода в проекте, обнаружения нежелательных паттернов в коде, например, неиспользуемых переменных и модулей. Настраивается с помощью конфигурационного файла, может обрабатывать TypeScript при подключении соответствующего плагина.

### 3.1.7. Jest и React Testing Library

Jest — фреймворк для тестирования JavaScript кода [8]. Предоставляет CLI интерфейс для управления тестами, умеет запускать тестирование параллельно, определять процент покрытия кода и составлять отчёт о тестировании в виде HTML страницы.

React Testing Library — библиотека для тестирования React-компонентов [11]. Предоставляет обширный набор инструментов для тестирования, например, имитирование клика и ввода текста с клавиатуры. Механизмы, поставляемые библиотекой, выполняют обращения к DOM максимально приближенно к тому, как это делал бы пользователь.

## 3.2. Архитектура

Структура модальной карточки и страницы статистик представлена на Рис. 2. На схеме отображено взаимодействие их внутренних компонентов, а также коммуникация с сервером.

При разработке использовался современный подход к созданию React-компонентов — функциональные компоненты. Такое название обусловлено тем, что они являются обычными JavaScript-функциями. Функциональные компоненты принимают один аргумент-объект **props**, который содержит параметры для отрисовки, а на выходе возвращают описание отображаемого компонентом интерфейса, выраженного с помощью JSX [7].

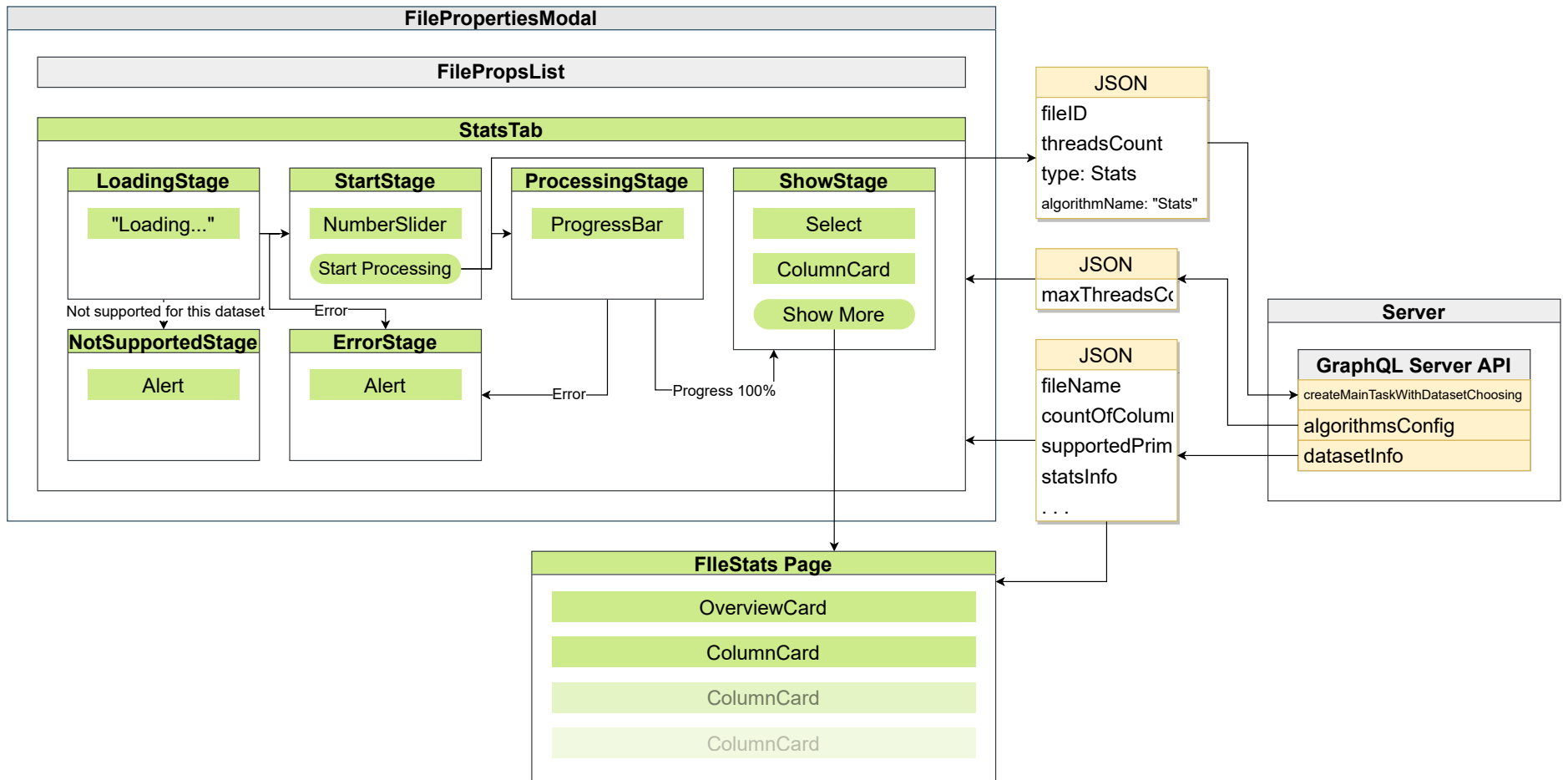


Рис. 2: Схема работы модальной карточки и страницы статистик

- — изменения не вносились
- — вносились изменения
- — разработаны с нуля в ходе учебной

### 3.3. Модальная карточка

Модальная карточка свойств датасета является точкой входа в раздел поиска статистик датасета. Было решено разделить её на следующие ключевые компоненты:

**StatsTab** — раздел модальной карточки, отвечающий за запуск, получение прогресса и отображение результатов поиска статистик датасета. На момент начала практики модальная карточка свойств датасета уже была реализована, но раздел статистик был заменен заглушкой. Данный компонент отвечает за переключение на клиенте текущего этапа обработки в зависимости от данных, возвращаемых сервером. В случае получения ошибки или недоступности поиска статистик для данного датасета отображает пользователю соответствующее сообщение. На этапе обработки запускает поллинг (polling) прогресса раз в 1000 мс.

Логика запросов к серверу, запуска обработки, а также переключения режимов поллинга выделена в отдельные пользовательские React-хуки — функции, позволяющие управлять состоянием и получать доступ к жизненному циклу компонента [6].

```
1 import { useEffect } from 'react';
2 import { StatsStage } from '@components/FilePropertiesModal/tabs/StatsTab';
3
4 export const usePollingControl = (
5   stage: StatsStage | null,
6   startPolling?: (pollInterval: number) => void,
7   stopPolling?: () => void
8 ) => {
9   useEffect(() => {
10     if (stage === StatsStage.Processing) startPolling?.(1000);
11
12     if (stage === StatsStage.Show) stopPolling?.();
13
14     return () => {
15       stopPolling?.();
16     };
17   }, [stage, startPolling, stopPolling]);
```

**Листинг 1:** хук `usePollingControl`, переключающий режим поллинга в зависимости от стадии обработки

Компонент **StatsTab** является основным в модальной карточке статистик, все нижеследующие компоненты используются внутри него.

**StartStage** — компонент начального этапа поиска статистик. Содержит ползунок и поле ввода для выбора количества потоков процессора, используемых при обработке на сервере, а также кнопку, вызывающую функцию начала поиска. Информация о максимальном числе процессорных потоков передаётся свыше из компонента **StatsTab** при помощи пропсов (props).

**ProcessingStage** соответствует стадии процесса поиска статистик. Отображает процент обработки с помощью прогресс-бара (progress-bar).

**ShowStage** занимается выводом результатов поиска статистик датасета. Изначально отображается таблица с общей информацией о датасете: количеством строк и столбцов таблицы, а также числом столбцов для каждого типа. С помощью выпадающего меню можно выбрать колонку для показа её статистик. Для того чтобы отобразить тип и категоричность (параметр `isCategorical`) колонки в выпадающем меню, необходимо было модифицировать компонент **Select**, имеющийся в проекте ещё до начала практики, благо библиотека **react-select**, с помощью которой он был реализован, позволяет использовать модифицированные компоненты меню и его опций. Компонент **ShowStage** также содержит кнопку, при нажатии на которую осуществляется переход на отдельную страницу статистик выбранного датасета с помощью предоставленного Next.js роутера (router).

Для показа статистик конкретной колонки используется компонент **ColumnCard**. Он имеет кнопку-переключатель для переключения между табличным и блочным режимами отображения данных. Так как табличный режим отображения реализован семантически правильно (с помощью HTML-тега `table`), он может быть полезен при переносе данных из веб-интерфейса в табличные редакторы. Блоки статистик также вынесены в отдельные компоненты и имеют несколько размеров, что позволяет сконцентрировать пользователя на важных данных. Компонент **ColumnCard** адаптивен, это значит, что он меняет своё поведение в зависимости от параметров устройства, в данном случае меняется на-

правление отрисовки блоков описательных и квантильных статистик, boolean-параметр `compact` также влияет на направление блоков и необходим для правильного отображения карточки колонки в модальном окне.

Компоненты `LoadingStage`, `ErrorStage` и `NotSupportedStage` необходимы для отображения статуса ожидания ответа от сервера, сообщений об ошибках, возникших в процессе обработки, и сообщения о недоступности поиска статистик для данного датасета соответственно.

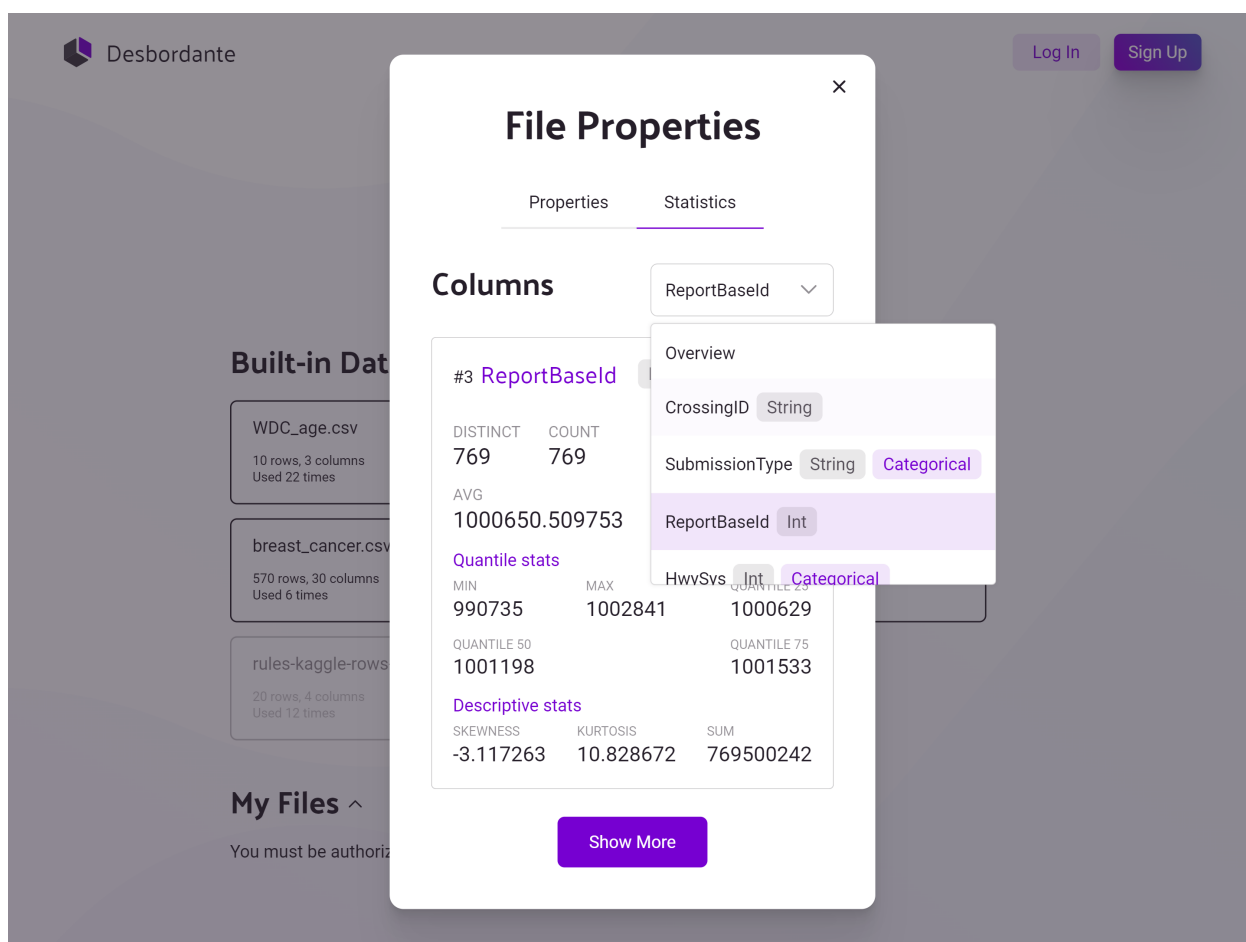


Рис. 3: Модальная карточка статистик с открытым меню выбора колонки

### 3.4. Страница

Отдельная страница статистик датасета дает пользователю возможность просматривать результаты поиска одновременно для всех колонок, так как они расположены в едином списке. Страница статистик

также использует компонент `ColumnCard` для отображения данных для колонки, но параметр `compact` при этом имеет значение по умолчанию — `false`. При таком значении параметра компонент карточки колонки позволяет пользователю раскрывать и скрывать детальную информацию в лице описательных и квантильных статистик. Идентификатор датасета для получения его статистик и последующего их отображения на странице берется из параметров URL.

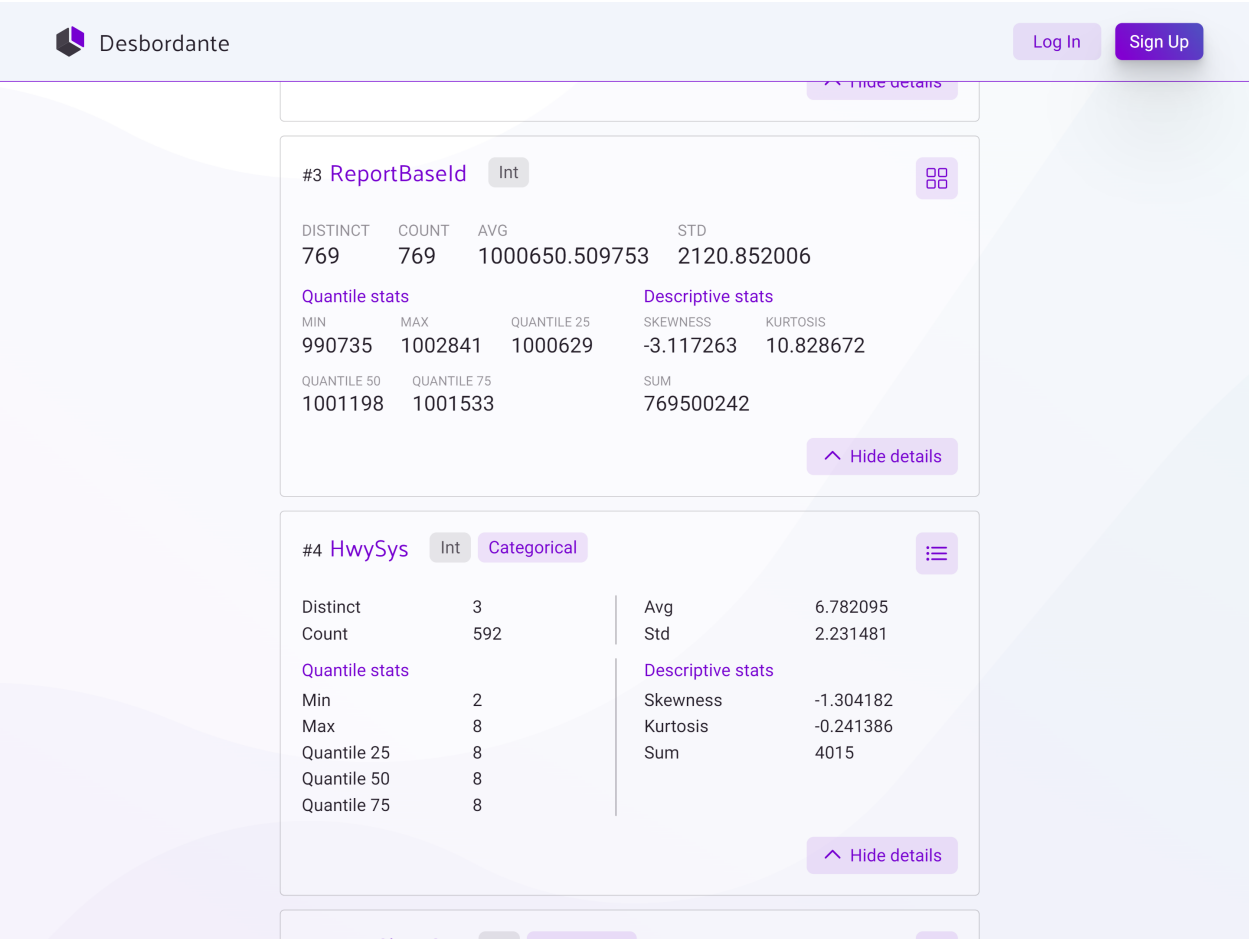


Рис. 4: Страница статистик датасета с карточками колонок в блочном и табличном режимах

### 3.5. Server Side Rendering

Одна из возможностей, предоставляемых Next.js, это Server Side Rendering (SSR). Server Side Rendering — это рендеринг страниц приложения на стороне сервера, он позволяет избавиться от ряда проблем,

имеющихся в обычных SPA (Single Page Application). В обычных React-приложениях отрисовка пользовательского интерфейса происходит целиком на стороне клиента с помощью JavaScript, изначально страница не содержит HTML, который отвечает за UI, поэтому поисковые движки видят веб-сайт как пустой. По этой причине сайт не может попасть на верхние позиции поисковых результатов, что негативно влияет на пользовательский трафик сервиса. Рендеринг на стороне сервера позволяет решить эту проблему, чтобы подключить SSR на страницу в Next.js достаточно экспортировать из файла страницы функцию `getServerSideProps`. Код, написанный внутри неё будет выполнен на стороне сервера Next.js. В рамках данной практики SSR используется на странице статистик.

### 3.6. Взаимодействие с сервером

Коммуникация с сервером приложения происходит посредством GraphQL-клиента Apollo. В GraphQL есть два основных типа взаимодействий. Первый — `query`, это обычный запрос, направленный на получение данных без каких-либо побочных эффектов. Использовались следующие запросы:

- `datasetInfo(fileID)` — получает данные о датасете по его ID, включая информацию о прогрессе и результатах поиска статистик;
- `algorithmsConfig()` — в контексте статистик необходим для получения максимального числа потоков.

Второй вид запросов — мутации (`mutations`), они нужны для выполнения каких-либо действий на сервере, например, для создания записи о новом пользователе или добавления датасета. Для раздела статистик использовалась лишь одна мутация — `createMainTaskWithDatasetChoosing(fileID, props)`, она нужна для создания на сервере задачи по поиску статистик для датасета, в объект `props` передается



число потоков для обработки, тип алгоритма в данном случае неизменен и имеет значение **Stats**.

Была также добавлена обработка ошибок GraphQL-клиента и показ их сообщений через всплывающие уведомления-тосты (toasts), реализованные при помощи библиотеки **react-toastify**.

### 3.7. Тестирование

Было проведено ручное тестирование модальной карточки и страницы статистик на встроенных и пользовательских датасетах. Проверен запуск обработки, отображение прогресса, просмотр результатов, включая выбор конкретной колонки через выпадающее меню, переключение режима отображения карточки колонки, раскрытие и скрывание детальной информации, а также отзывчивость верстки при изменении ширины экрана. Также было написано 11 юнит-тестов для проверки полей ввода и корректности отображения интерфейса. Проценты покрытия компонентов представлены на Рис. 5













File		Statements	Branches	Functions	Lines	
components/FilePropertiesModal/tabs/StatsTab		100%	29/29	94.11%	16/17	100%
components/FilePropertiesModal/tabs/StatsTab/hooks		100%	35/35	100%	5/5	100%
components/FilePropertiesModal/tabs/StatsTab/stages		100%	86/86	100%	10/10	100%
components/FileStats/Alert		100%	10/10	100%	3/3	100%
components/FileStats/Badge		100%	6/6	100%	1/1	100%
components/FileStats/ColumnCard		100%	18/18	100%	16/16	100%
components/FileStats/ModeButton		100%	9/9	100%	2/2	100%
components/FileStats/Paper		100%	6/6	100%	0/0	100%
components/FileStats/Progress		100%	6/6	100%	0/0	100%
components/FileStats/Statistic		100%	6/6	100%	2/2	100%
components/FileStats/StatsBlock		100%	12/12	100%	12/12	100%
components/FileStats/Table		100%	6/6	100%	0/0	100%
components/FileStats/hooks		100%	6/6	100%	0/0	100%

Рис. 5: Проценты покрытия компонентов тестами

# Заключение

По итогам семестровой практики была реализована клиентская часть поиска основных статистик для встроенных и загружаемых пользователем датасетов, а именно были выполнены следующие задачи:

1. разработаны и использованы компоненты, необходимые для страницы и модальной карточки статистик;
2. с помощью React, TypeScript и Sass сверстаны макеты данного раздела приложения;
3. реализовано взаимодействие с серверной частью приложения, включающее этапы запуска, отображения прогресса и результатов обработки;
4. проведена проверка работоспособности реализованной части приложения, компоненты протестированы с помощью unit-тестов.

Ссылка на GitHub-репозиторий: <https://github.com/vs9h/Desbordante> (имя пользователя — vladyoslav).

## Список литературы

- [1] Abedjan Ziawasch, Golab Lukasz, Naumann Felix. Profiling relational data: a survey // [The VLDB Journal](#). — 2015. — Aug. — Vol. 24, no. 4. — P. 557–581. — URL: <https://doi.org/10.1007/s00778-015-0389-y> (дата обращения: 2022-12-21).
- [2] Brugman Simon. pandas-profiling: Exploratory Data Analysis for Python. — 2022. — Version: 3.5.0. URL: <https://github.com/pandas-profiling/pandas-profiling> (дата обращения: 2022-12-21).
- [3] 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.
- [4] ESLint Documentation. — 2022. — URL: <https://eslint.org/docs/latest/> (дата обращения: 2022-12-21).
- [5] GraphQL Documentation. — 2022. — URL: <https://graphql.org/learn/> (дата обращения: 2022-12-21).
- [6] Hooks at a Glance. — 2022. — URL: <https://reactjs.org/docs/hooks-overview.html> (дата обращения: 2022-12-21).
- [7] Introducing JSX. — 2022. — URL: <https://reactjs.org/docs/introducing-jsx.html> (дата обращения: 2022-12-21).
- [8] Jest Documentation. — 2022. — URL: <https://jestjs.io/docs/getting-started> (дата обращения: 2022-12-21).
- [9] Next.js Documentation. — 2022. — URL: <https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs> (дата обращения: 2022-12-21).
- [10] React Documentation. — 2022. — URL: <https://reactjs.org/docs/getting-started.html> (дата обращения: 2022-12-21).

- [11] React Testing Library Documentation. — 2022. — URL: <https://testing-library.com/docs/react-testing-library/intro/> (дата обращения: 2022-12-21).
- [12] Sass Documentation. — 2022. — URL: <https://sass-lang.com/documentation/> (дата обращения: 2022-12-21).
- [13] Stack Overflow. Stack Overflow Developer Survey 2022. — 2022. — URL: <https://survey.stackoverflow.co/2022/#section-most-popular-technologies-web-frameworks-and-technologies> (дата обращения: 2021-11-21).
- [14] TypeScript Documentation. — 2022. — URL: <https://www.typescriptlang.org/docs/handbook/intro.html> (дата обращения: 2022-12-21).
- [15] Введение в функциональные зависимости. — 2022. — URL: <https://habr.com/ru/company/JetBrains-education/blog/473882/> (дата обращения: 2022-12-21).
- [16] Нейросеть обучили распознавать рак кожи. — 2022. — URL: <https://habr.com/ru/news/t/586482/> (дата обращения: 2022-12-21).