

Санкт-Петербургский государственный университет

Кафедра информационно-аналитических систем

Группа 21.Б08-мм

# Переработка конфигуратора настроек алгоритмов для упрощения создания форм

*Белоконный Сергей Александрович*

Отчёт по учебной практике  
в форме «Производственное задание»

Научный руководитель:  
Ассистент кафедры информационно-аналитических систем Г. А. Чернышев

Санкт-Петербург  
2023

# Оглавление

<b>Введение</b>	<b>3</b>
<b>1. Постановка задачи</b>	<b>4</b>
<b>2. Вводная информация</b>	<b>5</b>
2.1. DESBORDANTE . . . . .	5
2.2. Обзор технологий создания веб-приложений . . . . .	5
2.3. Обзор доступных библиотек для создания форм . . . . .	6
2.4. Веб-приложение DESBORDANTE . . . . .	7
<b>3. Недостатки предыдущего решения</b>	<b>9</b>
<b>4. Обзор решения</b>	<b>10</b>
<b>5. Практическая часть</b>	<b>12</b>
5.1. Реализация . . . . .	12
5.2. Примеры использования . . . . .	13
<b>6. Апробация</b>	<b>18</b>
<b>Заключение</b>	<b>19</b>
<b>Список литературы</b>	<b>20</b>

# Введение

Любая программа, которая принимает исходные значения от пользователя должна иметь интерфейс для общения с ним. Если программа имеет множество различных алгоритмов и множество возможных вариантов входных данных, то необходимо предусмотреть архитектурную возможность добавления и изменения интерфейсов для работы с программой.

Интерфейсом для работы с программой могут быть графические и текстовые меню, веб-сайты и формы. Без интерфейса пользователь не сможет настроить программу, а программа не сможет получить данные от пользователя.

DESBORDANTE [3] — профайлер данных, обрабатывающий данные с помощью примитивов. Каждый примитив — это паттерн, который проверяется на данных. У DESBORDANTE есть две версии: консольная и веб-версия, которые позволяют работать с этими примитивами. Для настройки примитивов в веб-версии DESBORDANTE используются формы, для каждого примитива — своя.

Предыдущая версия конфигуратора форм имела сложную архитектуру, не позволяющую легко добавлять новые формы, и не имела необходимого функционала для создания динамических форм. Поэтому было принято решение о переработке и расширении этой части, что и стало темой настоящей работы.

Новый конфигуратор должен быть высокоуровневым и предлагать простой интерфейс для работы. Он не должен заставлять разработчика думать об том, как конфигуратор работает изнутри. Данное изменение позволит легко расширять функционал веб-приложения не затрагивая обширные участки уже написанного кода. Наконец, разрабатывая новый конфигуратор в DESBORDANTE важно помнить об простоте добавления новых форм, о типобезопасности и об удобстве изменения существующих форм.

# 1. Постановка задачи

Целью работы является переработка конфигуратора форм для настройки примитивов и добавление возможности выбрать начальные значения (пресет) в веб-приложении DESBORDANTE<sup>1</sup>. Для её выполнения были поставлены следующие задачи:

1. разработать архитектуру конфигуратора форм;
2. разработать типы для компонентов конфигуратора форм с помощью языка TYPESCRIPT;
3. реализовать фабрику создания форм из конфигурационного файла;
  - создание набора компонентов для автоматического создания форм;
  - создание метода сборки формы из компонентов с помощью конфигурационного файла;
  - переработка логики отправки значений формы на сервер;
  - создание логики выбора значений по умолчанию из набора доступных;
4. перенос существующих форм в новый конфигуратор;
  - перенос значений по умолчанию;
  - перенос набора полей;
  - перенос логики работы полей;
  - перенос правил валидации полей;
5. реализовать выбор начальных параметров (пресетов).

---

<sup>1</sup><https://github.com/vs9h/Desbordante> (дата доступа: 31 мая 2023 г.).

## 2. Вводная информация

При работе с DESBORDANTE, пользователю требуется взаимодействовать с интерфейсом настройки алгоритмов. Для настройки этих алгоритмов, в веб-приложении DESBORDANTE используются формы. Формы предоставляют удобный способ ввода данных, необходимых для конфигурации, позволяющий пользователям эффективно взаимодействовать с DESBORDANTE и настраивать алгоритмы под свои нужды.

В прошлой работе [8] уже был приведен обзор технологий, используемых в веб-приложении DESBORDANTE, вкратце повторим его.

### 2.1. DESBORDANTE

DESBORDANTE<sup>2</sup> — профайлер данных, который может обнаруживать различные закономерности в табличных данных, выражаемые с помощью примитивов. Примитив — набор алгоритмов, с помощью которых проверяются закономерности. Ядро DESBORDANTE написано на C++ и работа с ним может вестись через консоль или через веб-приложение с простым в использовании интерфейсом. Больше информации можно найти в блоге UNIVERSE DATA [20, 2].

### 2.2. Обзор технологий создания веб-приложений

Для создания веб-приложений используется язык программирования JAVASCRIPT. Для облегчения разработки веб-приложений вместе с JAVASCRIPT может использоваться надстройка TYPESCRIPT, разработанная компанией MICROSOFT, которая добавляет строгую типизацию в JAVASCRIPT, и специальные фреймворки. Далее рассмотрим наиболее популярные из них:

- REACTJS — JAVASCRIPT-библиотека с открытым исходным кодом, разработанная компанией FACEBOOK, используемая для разработки пользовательских интерфейсов. Цель этой библиотеки —

---

<sup>2</sup><https://github.com/Mstrutov/Desbordante> (дата доступа: 31 мая 2023 г.).

предоставить высокую скорость разработки и масштабируемость. Часто REACTJS используют с другими библиотеками, такими как APOLLO GRAPHQL и NEXTJS. Последняя библиотека позволяет использовать отрисовку на стороне сервера (Server Side Rendering, SSR). Подробности можно узнать на главных страницах REACTJS [17], APOLLO GRAPHQL [12] и NEXTJS [14].

- ANGULAR — платформа для разработки веб-приложений на основе шаблона Модель-Представление-Контроллер (Model-View-Controller, MVC), созданная командой из GOOGLE написанная на TYPESCRIPT. ANGULAR был разработан с использованием идей декларативного программирования для создания пользовательских интерфейсов, так как разработчики убеждены в том, что этот подход лучше всего подходит для этих целей. Подробности можно узнать на официальном сайте [11].
- VUE.JS — JAVASCRIPT-фреймворк с открытым исходным кодом, разработанный для быстрого прототипирования пользовательских интерфейсов. Подробности можно узнать на официальном сайте [19].

Так как в проекте уже использовались REACTJS с TYPESCRIPT и NEXTJS, было решено использовать именно их.

## 2.3. Обзор доступных библиотек для создания форм

Для создания форм в веб-приложении REACTJS существует множество библиотек, облегчающих работу с компонентами формы. Рассмотрим наиболее популярные из них:

- YUP [10] — TYPESCRIPT-библиотека с открытым исходным кодом, используемая для разработки форм, анализа и валидации значений. Эта библиотека позволяет работать с формами как с схемами, что упрощает создание и поддержку форм.

- REACT-HOOK-FORM [16] — TYPESCRIPT-библиотека с открытым исходным кодом, используемая для разработки форм. Эта библиотека предлагает высокую производительность, малый размер и отсутствие дополнительных зависимостей.
- REDUX-FORM [18] — JAVASCRIPT-библиотека с открытым исходным кодом, используемая для разработки форм. Эта библиотека интегрируется с менеджером состояний Redux.
- REACT-FINAL-FORM [15] — JAVASCRIPT-библиотека с открытым исходным кодом от разработчика REDUX-FORM, используемая для разработки форм. Эта библиотека предлагает высокую производительность, малый размер и отсутствие дополнительных зависимостей также, как и REACT-HOOK-FORM, но вместе с этим имеет меньший функционал.
- FORMIK [13] — JAVASCRIPT-библиотека с открытым исходным кодом, используемая для разработки форм. Эта библиотека предлагает минимальный набор компонентов для создания формы.

Так как в проекте уже использовались REACT-HOOK-FORM, было решено продолжить использовать именно эту библиотеку.

## 2.4. Веб-приложение DESBORDANTE

Процесс создания и настройки задачи в DESBORDANTE состоит из нескольких шагов:

1. Выбор примитива — например:

- Функциональные зависимости (Functional dependencies) [6].
- Условные функциональные зависимости (Conditional Functional Dependencies) [4, 5].
- Ассоциативные правила (Association Rules) [1].
- Обнаружения ошибок в потоке данных (Error Detection Pipeline) [3].

- Метрические функциональные зависимости (Metric Functional Dependency) [7].
2. Выбор набора данных (датасета).
  3. Настройка примитива.
  4. Получение результатов работы примитива.

Каждый примитив может работать над некоторыми файлами и иметь несколько наборов значений по умолчанию.



### 3. Недостатки предыдущего решения

Предыдущая реализация конфигуратора форм имела ряд недостатков:

- Использование одного файла для описания типа поля, описания полей формы, логики полей и формирования формы.
- Сложность добавления новых и редактирования старых форм.
- Невозможность запрета отправки значений формы на сервер при наличии ошибок.
- Невозможность установки значений по умолчанию для отдельных файлов.

Единый файл для описания типа поля, описания полей формы, логики полей и формирования формы может создавать сложности при чтении и понимании кода, а также усложнять процесс добавления новых форм или внесения изменений в существующие формы. Одна большая файловая структура может быть менее организованной и более подверженной ошибкам.

Отсутствие возможности запретить отправку значений формы на сервер в случае наличия ошибок может быть проблемой для обеспечения корректной работы алгоритмов. Например, если некоторые данные не были введены.

Отсутствие возможности установки значений по умолчанию может помешать ознакомлению пользователя с программой. Возможность выбора начальных значений может улучшить пользовательский опыт и упростить заполнение формы.

## 4. Обзор решения

В разработке конфигулятора форм использовался шаблон строитель.

В разработке компонентов для создания формы настройки примитива использовался шаблон адаптер, позволяющий, при передаче данных одной и той же структуры, получать различные поля. Также использовались уже существующие в проекте наработки.

Созданные типы и компоненты необходимы для автоматического создания форм с помощью конфигулятора форм. Они описывают инструментарий, с помощью которого разработчик создаёт форму.

Список созданных типов:

- Тип списка значений по умолчанию (Defaults).
- Общий тип параметров поля формы, описывающий обязательные и опциональные параметры, которые должны быть у каждого поля (FormFieldProps).
- Расширения FormFieldProps дают более точное описание параметров, которые должно принимать каждое поле:
  - Скрытое поле (FormHiddenValueProps).
  - Нестандартное поле (FormCustomProps).
  - Числовой ввод (FormNumberInputProps).
  - Числовой ввод с шкалой (FormNumberSliderProps).
  - Выбор одного параметра из множества доступных вариантов (FormSelectProps).
  - Выбор нескольких параметров из множества доступных вариантов (FormMultiSelectProps).
- Типы для логики формы (FormHook и FormLogic).
- Тип объекта формы (Form).

Для описания формы создаётся список вида: название поля — начальное значение поля, которое может принимать типы `string`, `string[]`, `number`, `number[]` или `boolean`.

У этого списка определяется буквальный [9] тип, удовлетворяющий Defaults. На основании буквального типа составляется список конфигураций, в котором ключами являются названия полей, а значениями — параметры, удовлетворяющие расширениям `FormFieldProps`. Эти расширения, в свою очередь, удовлетворяют соответствующим начальным значениям формы. Доступные типы расширений описаны в списке выше.

Помимо этого, разработчику оставлена возможность определить то, как форма будет реагировать на введённые пользователем значения. Это делается путём создания метода, который принимает текущее состояние формы и функции, для изменения этой формы.

Список начальных значений, список конфигураций полей и логика, при наличии, собираются в один объект, и используются фабрикой для создания формы.

Список созданных компонентов:

- Числовой ввод.
- Числовой ввод с шкалой.
- Поле выбора одного параметра из множества доступных вариантов.
- Поле выбора нескольких параметров из множества доступных вариантов.
- Фабрика форм.
- Выбор набора параметров по умолчанию (пресета) в зависимости от датасета и выбранного примитива.

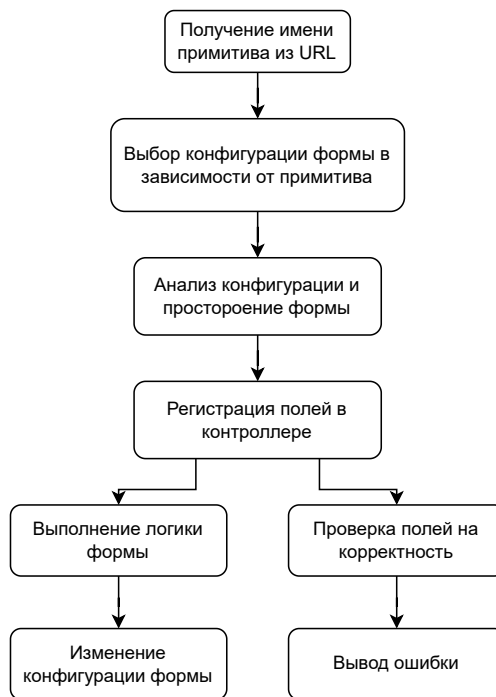


Рис. 1: Схема работы фабрики форм

## 5. Практическая часть

Основной целью настоящей курсовой работы разработка нового конфигуратора форм на основе фабрики форм и конфигурационных файлов. Фабрика форм принимает в качестве входных параметров имя примитива. На основании имени примитива фабрика собирает поля формы, обрабатывает изменения формы и подготавливает данные к отправке на сервер. На Рисунке 1 указана схема работы фабрики.

### 5.1. Реализация

В веб-приложении DESBORDANTE для получения результатов работы примитивов пользователь должен сделать следующую последовательность действий:

1. выбрать примитив;
2. выбрать датасет;
3. настроить примитив;

4. подождать, пока сервер обрабатывает запрос;
5. посмотреть результаты работы.

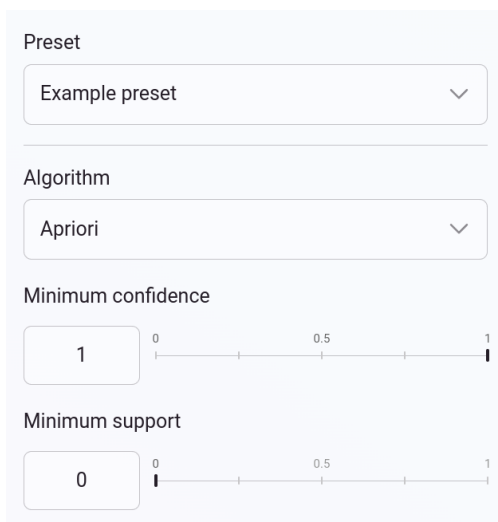
Каждый шаг в этой последовательности реализован как отдельная страница. При разработке были переделана страница настройки примитива.

Реализованные компоненты стараются поддерживать одинаковый набор входных параметров, что позволяет собирать форму из базовых полей, при этом не требуя отдельной логики для каждого отдельного поля. Такой подход к решению задачи позволяет крайне просто добавлять новые конфигурации форм не изменяя логику построения форм.

Написанная фабрика форм позволяет разработчику быстро создавать новые и обновлять старые формы без необходимости изменять большой файл в нескольких разных местах.

## 5.2. Примеры использования

В Листинге 1 приведёт пример конфигурации формы для примитива “Association Rules”. На Рисунке 2 показан результат работы фабрики форм над этой конфигурацией.



The image shows a configuration form for 'Association Rules'. It has a light blue background. The form contains the following elements:

- Preset:** A dropdown menu with 'Example preset' selected.
- Algorithm:** A dropdown menu with 'Apriori' selected.
- Minimum confidence:** A slider ranging from 0 to 1, with a value of 1 selected.
- Minimum support:** A slider ranging from 0 to 1, with a value of 0 selected.

Рис. 2: Пример формы для настройки примитива “Association Rules”

**Листинг 1: Пример конфигурации формы для настройки примитива “Association Rules”.**

```
const ARDefaults = {
  algorithmName: 'Apriori',
  minConfidence: 0,
  minSupportAR: 0,
} satisfies Defaults;

const ARFields = {
  algorithmName: {
    order: 0,
    type: 'select',
    label: 'Algorithm',
    isLoading: false,
    options: AROptions,
  },
  minConfidence: {
    order: 1,
    type: 'number_slider',
    label: 'Minimum_confidence',
    numberSliderProps: { min: 0, max: 1, step: 1e-4, size: 5 },
  },
  minSupportAR: {
    order: 2,
    type: 'number_slider',
    label: 'Minimum_support',
    numberSliderProps: { min: 0, max: 1, step: 1e-4, size: 5 },
  },
} satisfies FormFieldsProps<typeof ARDefaults>;
```

# Configure Algorithm

Select algorithm parameters

Preset

Default

LHS Columns

Select...

Cannot be empty

RHS Columns

Select...

Cannot be empty

RHS column type

Numeric

Metric

Euclidean

Algorithm

Brute

Tolerance parameter

1

Q-gram length

1

Distance to null

Infinity

Go Back Analyze

Рис. 3: Пример пустой формы для настройки примитива “Metric Dependency Verification”

# Configure Algorithm

Select algorithm parameters

Preset  
Custom

LHS Columns  
Select...  
Cannot be empty

RHS Columns  
1: Base1 × 8: Metric6 ×  
Columns must have one type

RHS column type  
Numeric

Metric  
Euclidean

Algorithm  
Brute

Tolerance parameter  
1

Q-gram length  
1

Distance to null  
Infinity

Go Back Analyze

**Input error**  
You need to correct the errors in the form.

Рис. 4: Пример ошибки отправки формы

Preset

Example preset

Example preset

Default

RHS Columns

3: Metric1 ×

(a) Пример выбора набора значений по умолчанию

Preset

Custom

LHS Columns

2: Base2 ×

RHS Columns

1: Base1 × 3: Metric1 ×

(b) Пример изменения значений по умолчанию

Рис. 5: Форма выбора значений по умолчанию



На Рисунке 3 показаны форма для настройки примитива “Metric Dependency Verification”, когда пользователь открывает страницу. Для некоторых файлов могут отсутствовать дополнительные наборы значений по умолчанию, поэтому выставляются значения, указанные в конфигурации формы.

- Первым полем пользователь может выбрать значения по умолчанию из набора предложенных.
- Ниже идут поля формы, которые описаны в конфигурационном файле.

На Рисунке 4 показаны вывод ошибок валидации полей и сообщение об ошибке отправки формы. Для каждого поля можно создать набор правил для валидации и выводить различные сообщения об ошибках.

На Рисунке 5а показывается пример выбора набора значений по умолчанию. Изначально выбирается первый доступный набор значений по умолчанию, однако у пользователя есть возможность выбрать значения, указанные в конфигурации формы.

Если пользователь изменил предустановленные значения формы, то форма покажет, что текущее состояние формы не является предустановленным. На Рисунке 5b показывается такая ситуация.

## 6. Апробация

Для оценки качества проделанной работы использовались метрики:

- типобезопасность кода;
- масштабируемость кода;
- чистота и читабельность.

Типобезопасность кода подтверждается тем, что статический анализатор языка `Typescript` не обнаружил ошибок.

Масштабируемость кода подтверждается тем, что в текущей реализации код различных форм разделён на несколько непересекающихся файлов.

В качестве доказательства чистоты и читабельности кода можно привести факт, что в нём отсутствуют неиспользуемые и избыточные конструкции. Так же это подтверждается отсутствием предупреждений линтеров, проверкой кода более опытными коллегами и принятым пул-реквестом (pull request).

Результаты апробации показали, что реализованная функциональность позволила достичь заявленных целей. Прделанная работа позволит другим разработчикам разрабатывать новый функционал более эффективно, так как им не придётся разбираться в подробностях реализации конфигуратора форм. В целом это поможет развитию проекта за счет ускорения процессов по добавлению новых примитивов, что в итоге положительно скажется на росте пользовательской базы.

# Заключение

В результате работы был переработан конфигуратор форм для настройки примитивов в веб-приложении DESBORDANTE и были выполнены следующие задачи:

- Разработана архитектура конфигулятора форм.
- Созданы типы для компонентов конфигулятора форм.
- Разработана фабрика создания форм.
- Существующие формы были переписаны с помощью конфигурационных файлов.
- Реализован выбор начальных параметров.

Подробнее о проделанной работе можно ознакомиться по ссылке:

<https://github.com/vs9h/Desbordante/pull/96>

## Список литературы

- [1] Borgelt Christian. [An Implementation of the FP-Growth Algorithm](#) // Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations. — OSDM '05. — New York, NY, USA : Association for Computing Machinery, 2005. — P. 1–5. — URL: <https://doi.org/10.1145/1133905.1133907>.
- [2] Data profiling, и с чем его едят / Георгий Чернышев, Максим Струтовский, Никита Бобров и др. — URL: <https://habr.com/ru/companies/unidata/articles/667636> (дата обращения: 31 мая 2023 г.).
- [3] 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](#).
- [4] [Discovering Conditional Functional Dependencies](#) / Wenfei Fan, Floris Geerts, Laks V. S. Lakshmanan, Ming Xiong // 2009 IEEE 25th International Conference on Data Engineering. — 2009. — P. 1231–1234. — URL: <https://doi.org/10.1109/ICDE.2009.208>.
- [5] [Discovering Conditional Functional Dependencies](#) / Wenfei Fan, Floris Geerts, Jianzhong Li, Ming Xiong // [IEEE Transactions on Knowledge and Data Engineering](#). — 2011. — Vol. 23, no. 5. — P. 683–698. — URL: <https://doi.org/10.1109/TKDE.2010.154>.
- [6] Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms / Thorsten Papenbrock, Jens Ehrlich, Jan-nik Marten et al. // [Proc. VLDB Endow.](#) — 2015. — jun. — Vol. 8, no. 10. — P. 1082–1093. — URL: <https://doi.org/10.14778/2794367.2794377>.
- [7] [Metric Functional Dependencies](#) / Nick Koudas, Avishek Saha, Di-

vesh Srivastava, Suresh Venkatasubramanian // 2009 IEEE 25th International Conference on Data Engineering. — 2009. — P. 1275–1278.

- [8] Вывод метрических функциональных зависимостей в веб-интерфейсе Desbordante. — URL: <https://github.com/Mstrutov/Desbordante/blob/17b792ae4e930c7f3dc2b84cece3d2b47cedb940/docs/papers/Frontend%20MFD%20-%20Sergey%20Belokonniy%20-%202022%20autumn.pdf> (дата обращения: 31 мая 2023 г.).
- [9] Документация TYPESCRIPT о буквальных типах. — URL: <https://www.typescriptlang.org/docs/handbook/2/everyday-types.html#literal-types> (дата обращения: 31 мая 2023 г.).
- [10] Официальный репозиторий Yup. — URL: <https://github.com/jquense/yup> (дата обращения: 31 мая 2023 г.).
- [11] Официальный сайт Angular. — URL: <https://angular.io> (дата обращения: 8 января 2023 г.).
- [12] Официальный сайт Apollo GraphQL. — URL: <https://www.apollographql.com> (дата обращения: 8 января 2023 г.).
- [13] Официальный сайт Formik. — URL: <https://formik.org/> (дата обращения: 31 мая 2023 г.).
- [14] Официальный сайт NextJS. — URL: <https://nextjs.org> (дата обращения: 8 января 2023 г.).
- [15] Официальный сайт React Final Form. — URL: <https://final-form.org/react> (дата обращения: 31 мая 2023 г.).
- [16] Официальный сайт React-Hook-Form. — URL: <https://react-hook-form.com/> (дата обращения: 31 мая 2023 г.).
- [17] Официальный сайт ReactJS. — URL: <https://www.graphql.org> (дата обращения: 8 января 2023 г.).

- [18] Официальный сайт Redux-Form. — URL: <https://redux-form.com> (дата обращения: 31 мая 2023 г.).
- [19] Официальный сайт VueJS. — URL: <https://vuejs.org> (дата обращения: 8 января 2023 г.).
- [20] Чернышев Георгий, Полынцов Михаил, Бобров Никита. Прimitives Desbordante: Функциональные зависимости и их применение в эксплорации и очистке данных. — URL: <https://habr.com/ru/companies/unidata/articles/679390> (дата обращения: 31 мая 2023 г.).