Санкт-Петербургский государственный университет

***Соловьёва Лиана-Юлия Викторовна***

Выпускная квалификационная работа

# Реимплементация существующих и добавление новых примитивов в веб-интерфейс 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:
Программист, НПО «Тайфун» Мухалева Н. В.

Санкт-Петербург
2025

Saint Petersburg State University

***Liana-Iuliia Soloveva***

Bachelor's Thesis

# Reimplementing existing and adding new primitives to the web version of Desbordante data profiler

Education level: bachelor

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

Programme *CB.5162.2020 "Programming Technologies"*

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

Consultant:
Assistant G.A. Chernishev

Reviewer:
Programmer, NPO "Typhoon" N.V. Mukhaleva

Saint Petersburg
2025

# Оглавление

# Введение

Desbordante [4] — это высокопроизводительный профилировщик данных, предназначенный для обнаружения и проверки различных закономерностей в данных, например, точных и приближенных функциональных зависимостей, ассоциативных правил, алгебраических ограничений и многого другого. Работа с данными основана на использовании примитивов, реализуемых соответствующими алгоритмами.

Ядро проекта[1] — алгоритмы профилирования, реализованные на языке C++. Для взаимодействия с пользователями Desbordante предлагает три интерфейса: консольное и веб-приложения[2], а также pip-пакет[3]. Консольное приложение и pip-пакет непрерывно развиваются, а веб-интерфейс не пополняется новыми примитивами, начиная с 2022 года, и на данный момент предлагает:

1. Поиск функциональных зависимостей (Functional Dependencies, **FD**) [7];
2. Поиск условных функциональных зависимостей (Conditional Functional Dependencies, **CFD**)[12];
3. Поиск ассоциативных правил (Association Rules, **AR**) [1];
4. Пайплайн по обнаружению ошибок (Error Detection Pipeline, **EDP**)[4] [14];
5. Верификация метрических зависимостей (Metric Dependency (**MFD**) **Verification**) [9].

Изначально веб-приложение разрабатывалось как концептуальный прототип, демонстрирующий базовую функциональность. За это время

[1] https://github.com/Desbordante/desbordante-core, (дата обращения: 1 мая 2025 г.)

[2] https://desbordante.unidata-platform.ru/ (дата обращения: 1 мая 2025 г.)

[3] https://pypi.org/project/desbordante/ (дата обращения: 1 мая 2025 г.)

[4] Не примитив, а демонстрация сложной функциональности, которую можно сделать с использованием примитивов. Попробовать: https://desbordante.streamlit.app/ (дата обращения: 1 мая 2025 г.), https://github.com/Desbordante/desbordante-core/tree/main/examples/expert (дата обращения: 1 мая 2025 г.)

количество разработанных примитивов существенно возросло и сейчас составляет более 20. Ожидается, что веб-интерфейс будет предоставлять возможность воспользоваться этим многообразием, поэтому необходимо начинать работу по расширению функциональности.

Одной из причин отсутствия развития веб-приложения является затруднительное добавление новой функциональности ввиду необходимости вносить изменения и в серверную, и в клиентскую части. Для ее устранения был разработан новый бэкенд на Python с использованием фреймворка FastAPI[5]. Он должен заменить серверную часть, реализованную так же, как и фронтенд-часть, на языке TypeScript. Подробнее о решении можно прочитать в работе бэкенд-разработчика проекта Вадима Якшигулова [40].

Появление нового бэкенда влечет необходимость изменений. На данный момент весь код веб-приложения находится в одном репозитории, однако новая серверная часть размещается в отдельном, а значит, нужно отделить и фронтенд-часть. Также изменился способ клиент-серверного взаимодействия: в текущей версии применяется язык GraphQL[6], но новая реализация использует более удобную для разработки спецификацию OpenAPI[7].

Помимо адаптации фронтенда к новому бэкенду, важно синхронизировать интерфейсы по наличию функциональности, то есть интегрировать новые примитивы в веб-приложение. В связи с этим может потребоваться изменение дизайна и переосмысление компонентов, которые потеряли актуальность или станут неудобными в использовании при добавлении новых примитивов.

Таким образом, реимплементация[8] фронтенд-части стала необходимым шагом для модернизации веб-приложения Desbordante.

[5] https://fastapi.tiangolo.com/ (дата обращения: 10 мая 2025 г.)

[6] https://graphql.org/learn/ (дата обращения: 10 мая 2025 г.)

[7] https://learn.openapis.org/ (дата обращения: 10 мая 2025 г.)

[8] Реимплементация [13] — это процесс повторной реализации уже существующего решения, алгоритма или системы, часто с целью улучшения, адаптации к новым условиям или изменения технологии. В программировании это может означать переработку кода, чтобы сделать его более эффективным или совместимым с другими системами.

# 1. Постановка задачи

Целью работы является реимплементация существующих и добавление новых примитивов в веб-интерфейс профилировщика данных Desbordante. Для её выполнения были поставлены следующие задачи:

1. проанализировать новые примитивы с целью выработать требования для их последующей интеграции в веб-приложение;
2. реимплементировать существующие примитивы;
3. разработать бэкенд для новых примитивов;
4. спроектировать и реализовать пользовательские интерфейсы для новых примитивов;
5. провести апробацию.

# 2. Обзор текущей версии фронтенд-части

Фронтенд-часть веб-интерфейса Desbordante разработана на языке TypeScript с использованием фреймворков React[9], Next.js[10].

Основное взаимодействие пользователя с веб-интерфейсом Desbordante заключается в профилировании данных. Этот процесс состоит из нескольких шагов, каждый из которых представлен отдельными страницами. Схематично это представлено на рис. 1, где стрелками обозначено, с какого шага на какой переходит пользователь.

## Выбор примитива

Сейчас соответствующая страница содержит пять примитивов, перечисленных во введении. При выборе определенного справа или снизу, в зависимости от расширения экрана, появляется его описание.

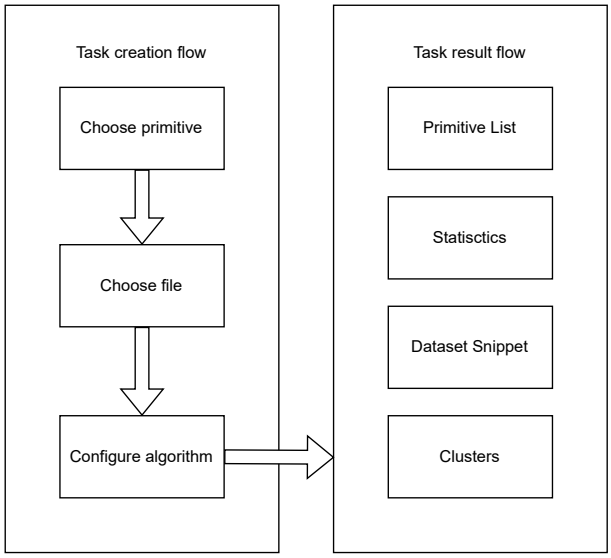



Рис. 1: Схема компонентов полной обработки задачи

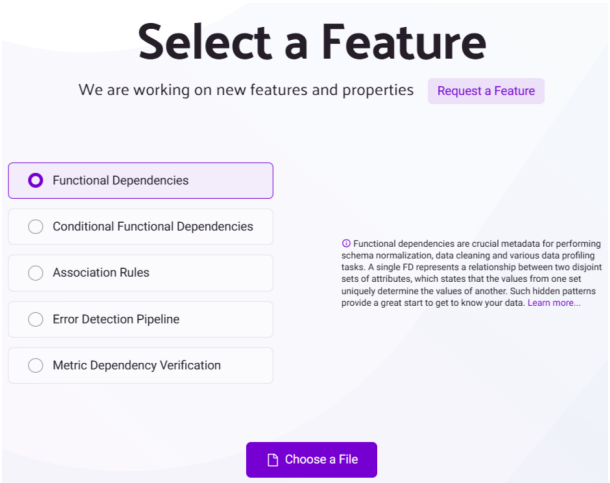



Рис. 2: Страница выбора примитива (исходная версия)

[9] https://react.dev/ (дата обращения: 10 мая 2025 г.)
[10] https://nextjs.org/ (дата обращения: 10 мая 2025 г.)

## Выбор файла

На данном этапе необходимо указать файл с данными, в котором будут искаться зависимости. Веб-приложение всегда предлагает несколько встроенных, однако, если пользователь авторизован, то он может выбрать ранее загруженный собственный датасет или добавить новый. Возможен выбор единственного файла.

## Конфигурирование алгоритма

На данном шаге пользователь должен указать параметры алгоритма, которые применяются для поиска зависимостей. В случае, когда один примитив реализуется несколькими алгоритмами, сначала нужно выбрать конкретный, а потом настроить его конфигурацию. Для упрощения пользовательской настройки добавлены пресеты (Preset) — предопределенные наборы значений параметров.

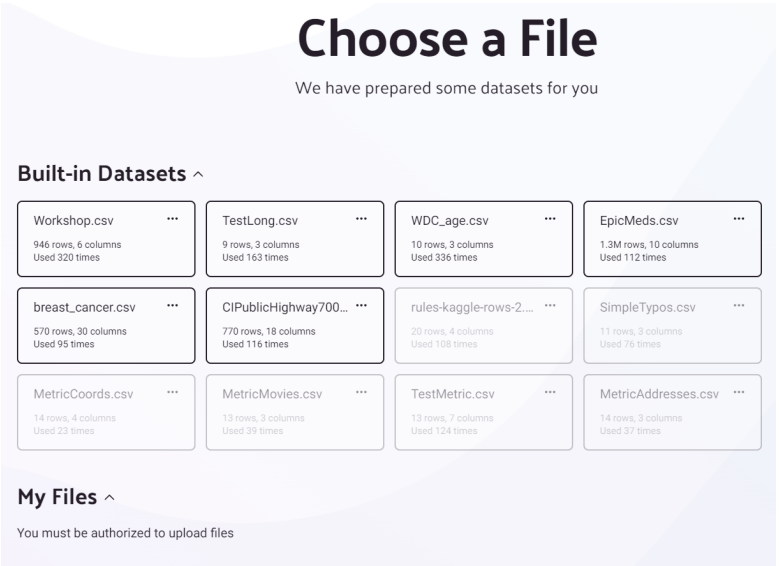



Рис. 3: Страница выбора файла (исходная версия)

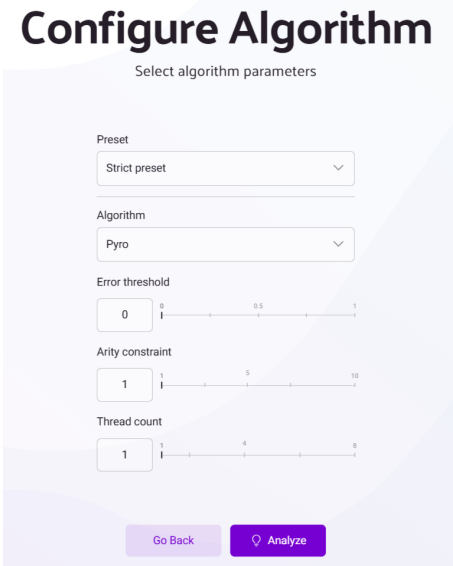



Рис. 4: Конфигурация алгоритма примитива FD

В данной работе этот шаг не рассматривается подробно, так как его переработкой занимается Белоконный Сергей.

## Вывод результата работы примитива

Данный этап является заключительным. Визуализируются найденные инстансы или кластеры — атомарные элементы результата. Часто страница результата состоит из нескольких вкладок, доступных из меню. Для каждого примитива этот набор может отличаться. Для примитивов поиска главная вкладка — Primitive List (рис. 5), для верификационных — Clusters (рис. 8). Для них реализованы поиск по названиям атрибутов, фильтрация по ключам (атрибутам, однозначно определяющим запись), сортировка (обычно по названиям атрибутов в левой и правой части) по возрастанию и убыванию, а также экспорт результата.

У FD и CFD есть еще одна вкладка — Statistics (рис. 10). Она отражает соотношение атрибутов в инстансах. Также она служит своеобразным фильтром: можно выбрать, какие показывать в левой части, а какие — в правой.

Для всех примитивов поиска актуален просмотр выбранного датасета (Dataset Snippet, рис. 6). При выборе определенного инстанса, подсвечиваются столбцы, соответствующие атрибутам из левой и правой частей.

С точки зрения архитектуры все вкладки реализованы как отдельные страницы, хотя логически это одна (для конкретного примитива), но с разным наполнением при разных условиях. Не существует отдельного файла для каждого примитива, который объединял бы соответствующие ему вкладки. Вместо этого данная информация хранится вместе с компонентом, который отвечает за расположение элементов на странице, в том числе меню и результата. Используется Pages Router[11]. Также существует специальный файл, соотносящий название примитива и путь до страницы результата, которую надо отобразить. Более того, для примитивов FD, CFD, AR, EDP существует единый файл вывода найденных зависимостей, так как, за исключением небольших различий (например, для EDP нет возможности импортировать результат, а для других есть), внешне результаты совпадает.

[11]Один из доступных в фреймворке Next.js вариантов маршрутизации. Подробнее https://nextjs.org/docs/pages (дата обращения: 10 мая 2025 г.)

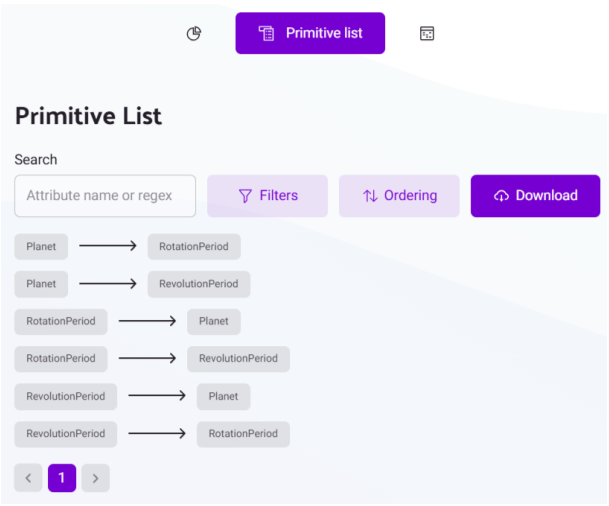



Рис. 5: Обнаруженные функциональные зависимости

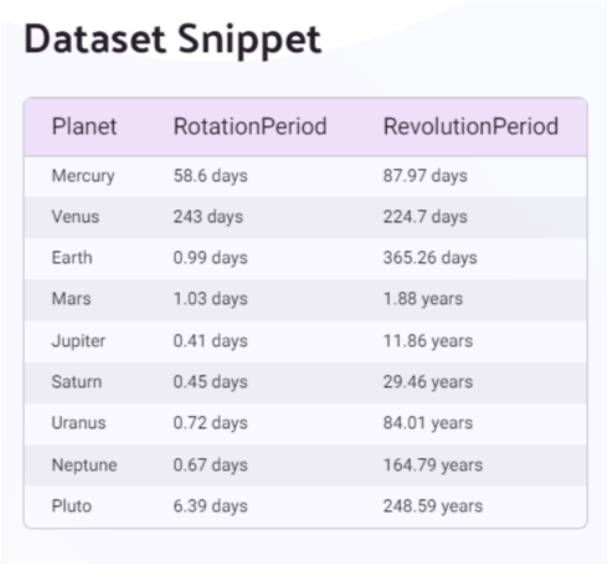



Рис. 6: Фрагмент датасета

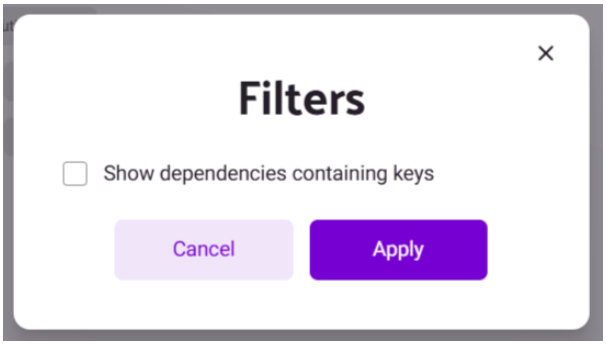



Рис. 7: Фильтрация

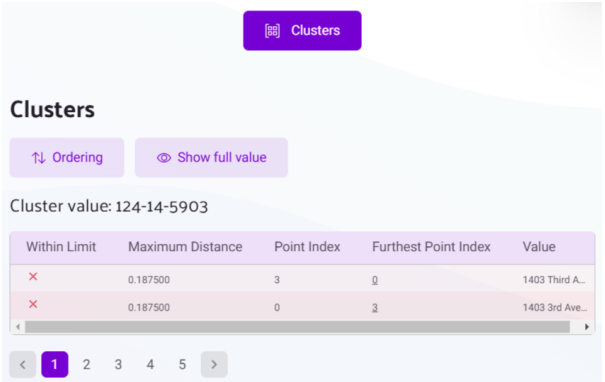



Рис. 8: Обнаруженные кластеры

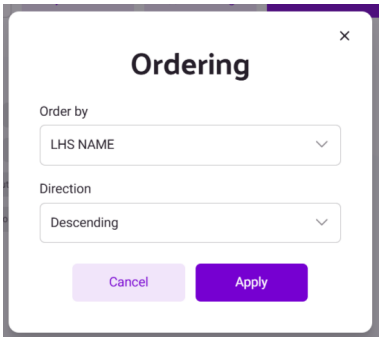



Рис. 9: Сортировка


Рис. 10: Статистика

С одной стороны, такая реализация удобна отсутствием дублирующего кода. С другой стороны, логически разные компоненты сильно взаимосвязаны, что является признаком плохой архитектуры и затрудняет понимание кода и модификацию отдельных элементов.

# 3. Обзор новой бэкенд-части

## Процесс обработки задачи

Прежде всего рассмотрим, как действует сервер в контексте работы примитивов.

Взаимодействие клиента и сервера начинается на странице настройки конфигурации. Клиент запрашивает доступные файлы, и сервер присылает список.

После того, как пользователь ввел все необходимые данные для настройки алгоритма, он отправляет POST-запрос с этой информацией. Сервер по названию примитива определяет, к какому классу обращаться, чтобы создать нужную задачу и впоследствии получить результат.

В каждую конфигурацию обязательно, помимо параметров для алгоритма, входит его название. По нему определяется, какой алгоритм импортировать из библиотеки desbordante.

Пока сервер обрабатывает полученные данные, пользователь видит страницу загрузки. В этот момент клиент посылает GET-запросы, пока сервер не подтвердит, что задача была выполнена (статус `COMPLETED`). После успешного завершения операции пользователю отображаются результаты вычислений, а сервер сохраняет их в базу данных для последующего использования при работе с данной конкретной задачей.

## Архитектура

Серверная часть системы подвергается постоянному развитию: изменяется архитектура, добавляются новые возможности, исправляются ошибки. В связи с этим в конце апреля фронтенд-разработчиками принято решение остановиться на последней на тот момент рабочей версии.

На основе рассмотренного жизненного цикла задачи можно сделать выводы о сегменте серверной архитектуры, который понадобится для добавления новых примитивов. Остальные компоненты остаются за рамками текущей работы. Подробнее об архитектуре нового бэкенда можно узнать в работах других студентов [22, 34].

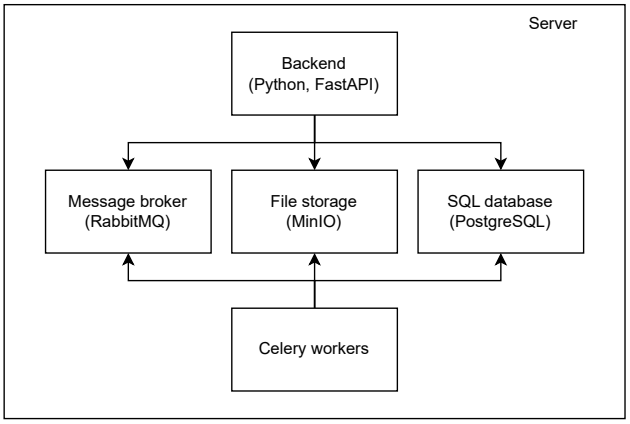



Рис. 11: Архитектура серверной части веб-приложения

| **Основная логика** | **Файловый модуль** | **Модуль задач** |
|---|---|---|
| app/ | file/ | task/ |
| ├── config/ | ├── dependencies.py | ├── schemas/ |
| ├── db/ | ├── models.py | │ ├── fd |
| ├── dependencies/ | ├── router.py | │ ├── base.py |
| ├── domain/ | ├── schemas.py | │ ├── schemas.py |
| │ ├── auth/ | ├── service.py | │ ├── types.py |
| │ ├── common/ | └── storage/ | ├── dependencies.py |
| │ ├── file/ | | ├── models.py |
| │ ├── storage/ | | ├── router.py |
| │ ├── task/ | | ├── service.py |
| │ ├── user/ | | └── utils.py |
| │ ├── worker/ | | |
| │ ├── exceptions/ | | |
| │ ├── models/ | | |
| │ ├── repository/ | | |
| │ ├── schemas/ | | |
| ├── migrations/ | | |
| └── scripts/ | | |

Внутренняя бизнес-логика бэкенда содержится в директории `app/domain`. Каждая поддиректория соотносится с определенным контекстом (задачи, файлы, пользователи и т.д.).

Рассмотрим директорию `task`, так как она является ключевой при

добавлении новых примитивов.

API-эндпоинты для обработки POST и GET-запросов по созданию задачи и получению результата содержит файл `router.py`.

В файле `utils.py` находится функция-определитель `match_task_by_primitive_name`.

Рассмотрим директорию `schemas`.

Она включает поддиректории `<primitive_short_name>` (короткое название примитива) с файлами, содержащими названия алгоритмов (`algo_name.py`), конфигурацию алгоритмов (`algo_config.py`), модель результата и функцию его вычисления (`task.py`). В частности, последний из перечисленных содержит функцию `match_algo_by_name`, которая упоминалась выше. Примеры моделей иллюстрируют листинги 1-3.

Рассмотрим директорию `file`. Её анализ инициировано наличием файловых операций в рассмотренном жизненном пути, что указывает на возможную роль в подготовке данных для обработки примитивами.

```python
class AcModel(BaseSchema):
    left_column: str
    right_column: str
    intervals: list[tuple[float, float]]
    outliers: list[int]

class BaseAcTaskModel(BaseSchema):
    primitive_name: Literal[PrimitiveName.AC]

class AcTaskResult(BaseAcTaskModel):
    operation: OperationType
    result: list[AcModel]
    table_header: list[str]
    count_results: int
```

**Листинг 1: Модель результата AC**

```python
class FdModel(BaseSchema):
    lhs: list[str]
    rhs: list[str]

class BaseFdTaskModel(BaseSchema):
    primitive_name: Literal[PrimitiveName.FD]

class FdTaskResult(BaseFdTaskModel):
    result: list[FdModel]
    table_header: list[str]
    count_results: int
```

**Листинг 2: Модель результата FD**

Аналогично рассмотренной директории `task`, файл `router.py` содержит API-эндпоинты. `models.py` — модели файлов (`FileBase` — базовая версия, и `FilePublic`, содержащая идентификатор файла, см. листинг 4), `service.py` отвечает за операции над файлами (получить, добавить).

```python
class AfdClusterModel(BaseSchema):
    num_distinct_rhs_values: int
    most_frequent_rhs_value_proportion: float
    rows: list[list[str]]

class AfdVerificationModel(BaseSchema):
    error: float
    num_error_clusters: int
    num_error_rows: int
    clusters: list[AfdClusterModel]
    table_header: list[str]
    lhs_rhs_indices: list[int]

class BaseAfdVerificationTaskModel(BaseSchema):
    primitive_name: Literal[PrimitiveName.AFD_VERIFICATION]

class AfdVerificationTaskResult(BaseAfdVerificationTaskModel):
    result: AfdVerificationModel
```

**Листинг 3: Модель результата AFD Validation**

```json
{
  "file_format": "csv",
  "name": "string",
  "byte_size": 1,
  "owner_id": 0,
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
}
```

**Листинг 4: Формат информации о датасете**

# 4. Обзор примитивов

Важной задачей данной работы является расширение возможностей веб-интерфейса профилировщика данных Desbordante путем интеграции новых примитивов. Для этой цели прежде всего необходимо проанализировать их, чтобы выработать требования, как визуализировать результат работы, какие параметры требуются алгоритмам, которые их реализуют, как они повлияют на текущую систему.

Для анализа рассмотрены статьи про перечисленные ниже примитивы, работы студентов, реализующих алгоритмы, Python-библиотека desbordante-stubs[12], предлагающая описания к инструментам одноименной библиотеки desbordante.

В данной работе рассмотрен только один из существующих примитивов: Функциональные зависимости (FD). Реимплементацией остальных занимается Белоконный Сергей.

В данном разделе проанализированы следующие примитивы:

1. Поиск вероятностных функциональных зависимостей (Probabilistic Functional Dependencies, **PFD**) [16];
2. Поиск приближенных функциональных зависимостей (Approximate Functional Dependencies, **AFD**) [11];
3. Верификация приближенных функциональных зависимостей (Approximate Functional Dependencies (**AFD**) **Verification**) [8];
4. Поиск дифференциальных зависимостей (Differential Dependencies, **DD**) [15];
5. Поиск сопоставляющих зависимостей (Matching Dependencies, **MD**) [5];
6. Поиск приближенных запрещающих ограничений с метрикой $g_1$ (Approximate Denial Constraints, with $g_1$ metric, **ADC**) [6];

[12] https://pypi.org/project/desbordante-stubs/ (дата обращения: 10 мая 2025 г.)

7. Поиск алгебраических ограничений (Algebraic Constraints, **AC**) [3];

8. Поиск численных ассоциативных правил (Numerical Association Rules, **NAR**)[10].

## 4.1. FD

Функциональная зависимость (ФЗ) $X \to Y$ означает, что значения набора атрибутов $X$ однозначно определяют значения атрибута $Y$. То есть для любых двух кортежей $t_1,\ t_2$ в отношении $R$ выполняется:

$$t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y].$$

В Desbordante данный примитив представляется 11 алгоритмами: AID-FD [24], DFD [28, 27], Depminer [20], EulerFD, FDep [25], FUN, FastFDs [23], FdMine [39], HyFD, Pyro [32, 30], Tane [31].

У каждого из них свой набор принимаемых параметров, но один присутствует во всех:

- Ограничение левой части (Arity constraint) — максимальное количество столбцов в левой части. Значением является целое неотрицательное число, где 0 означает отсутствие ограничений.

Для алгоритмов **AID-FD** и **FDep** это единственный параметр. Остальные примитивы также принимают

- Равенство Null-объектов (Is null equal null) — указывает, можно ли рассматривать два разных объекта типа Null как одинаковые. Принимает логические значения (true/false).

Для алгоритмов **Depminer**, **FUN**, **FdMine** и **Tane** на этом список параметров заканчивается.

Помимо уже перечисленных, существуют еще два параметра:

- Количество потоков (Thread count) — обеспечение параллельной работы алгоритма. Значением является целое неотрицательное

число, где 0 означает использование всех возможных потоков в вычислительной машине.

- Сид (Seed) — число, используемое алгоритмом для инициализации генератора псевдослучайных чисел, необходимое для воспроизводимости. Значением является целое неотрицательное число.

В качестве третьего параметра первый из них принимают алгоритмы **DFD**, **FastFDs** и **HyFD**, а второй — **EulerFD**. Алгоритм **Pyro** принимает оба (то есть все 4 указанных выше параметра).

## 4.2. PFD

Пусть $\overline{X}$ — набор атрибутов и $A$ — конкретный атрибут в отношении $R$. Тогда вероятностная функциональная зависимость обозначается $\overline{X} \to^p A$, где $p$ — вероятность того, что выполняется функциональная зависимость $\overline{X} \to A$.

Для данного примитива в Desbordante реализован алгоритм **PFDTane** [17], который, помимо параметров, описанных для Tane в пункте 4.1, дополнительно принимает:

1. Порог ошибки (Error threshold) — доля строк, удовлетворяющих ограничениям на левую часть, но нарушающих ФЗ. Значением является вещественное число из интервала [0, 1], где 0 означает нахождение точных функциональных зависимостей.

2. Мера ошибки (Error measure) — способ вычисления величины ошибки. Значением является один из встроенных вариантов: $per\_value$, $per\_tuple$.

## 4.3. AFD

Приближенные функциональные зависимости схожи с точными, однако результат может быть найден с некоторой погрешностью.

Для этой цели Desbordante предлагает два алгоритма: **Pyro** и **Tane** [33]. Помимо перечисленных в пункте 4.1 для примитива AFD есть возможность указывать дополнительные параметры.

Общий:

- Порог ошибки (Error threshold) — доля строк, удовлетворяющих ограничениям на левую часть, но нарушающих ФЗ. Значением является вещественное число из интервала [0, 1], где 0 означает нахождение точных функциональных зависимостей.

Специфичный только для Tane:

- Мера ошибки (Error measure) — способ вычисления величины ошибки. Значением является один из встроенных вариантов: $g_1$, $pdep$, $\tau$, $\mu^+$, $\rho$.

## 4.4. AFD Verification

Примитив AFD Verification проверяет, выполняется ли указанная ФЗ, а если нет, то находит, с какой погрешностью.

Для данной цели в Desbordante реализован алгоритм **FD Verifier**, который принимает следующие параметры:

1. Индексы атрибутов левой части (LHS) проверяемой ФЗ.
2. Индексы атрибутов правой части (RHS) проверяемой ФЗ.
3. Равенство Null-объектов (Is null equal null) — указывает, можно ли рассматривать два разных объекта типа Null как одинаковые. Принимает логические значения (true/false).

## 4.5. DD

При заданном отношении $R$ дифференциальная зависимость имеет вид $\varphi_L[X] \to \varphi_R[Y]$, где $\varphi_L[X]$ и $\varphi_R[Y]$ являются дифференциальными функциями, задающими ограничения на расстояния по атрибутам $X$ и

$Y$ из $R$ соответственно. Это значит, что для любых двух кортежей из $R$, если их различия в значениях по атрибутам $X$ соответствуют дифференциальной функции $\varphi_L[X]$ (т.е. ограничениям расстояния для $X$), то их различия в значениях для $Y$ должны соответствовать дифференциальной функции $\varphi_R[Y]$.

Для обнаружения дифференциальных зависимостей в Desbordante реализован алгоритм **SPLIT** [26]. Входными данными для него служат следующие параметры:

1. Количество строк (Number of rows) — ограничение на выборку первых N строк таблицы, используемых алгоритмом. Значением является целое неотрицательное число, не превышающее количество строк в датасете, где 0 означает использование всех строк.

2. Количество столбцов (Number of columns) — ограничение на выборку первых N столбцов таблицы, используемых алгоритмом. Значением является целое неотрицательное число, не превышающее количество строк в датасете, где 0 означает использование всех колонок.

Помимо основного датасета требуется также таблица различий (Difference table) — файл с пределами допустимых значений для каждого столбца.

## 4.6. MD

Сопоставляющие зависимости представляют собой обобщение ФЗ, однако соотносят похожие (по некоторой мере), а не одинаковые элементы.

Для данного примитива в Desbordante реализован алгоритм **HyMD** [36, 35]. Он находит набор границ принятия решений для всех MD, которых достаточно, чтобы выявить зависимости, удовлетворяющие некоторым требованиям (критериям интересности) и выполняющиеся на данных.

Алгоритм принимает следующие параметры:

1. Минимальная поддержка (Minimum support) — минимальное количество пар, для которых выполняется левая часть. Значением является целое неотрицательное число.

2. Максимальная мощность (Maximum cardinality) — максимальное количество ненулевых границ решения в левой части выданной в ответе зависимости. Значением является целое число не меньше -1, где нижняя граница означает отсутствие ограничений.

3. Определение уровня (Level definition) — внутренний параметр алгоритма, ограничивающий область анализа в решётке зависимостей. Значением является одна из двух строк: cardinality (только зависимости с одинаковой мощностью), lattice (ограничение отсутствует).

4. Количество потоков (Thread count) — обеспечение параллельной работы алгоритма. Значением является целое неотрицательное число, где 0 означает использование всех возможных потоков в вычислительной машине.

5. Сопоставления столбцов (Column matches, далее CMs) — тройка (метрика и два столбца). Также можно указывать дополнительные настройки. Подробнее:

   (a) Мера сходства (Metrics) — алгоритм измерения сходства между значениями двух столбцов. Значением является пользовательская метрика в формате python-кода или одна из встроенных: Левенштейн (Levenshtein), Жаккард (Jaccard), Монж-Элкан (Monge-Elkan), нормированное расстояние между числами (Number Normalized Difference), нормированное расстояние между датами (Date Normalized Difference), наибольшая общая подпоследовательность (LCS), равенство (Equality).

   (b) Два столбца (Column #1, Column #2) — названия колонок,значения которых сравниваются.

(c) Функция обработки колонок (Column functions) — преобразование значений перед сравниванием. Значением является пользовательский python-код, встроенных вариантов не предусмотрено, по умолчанию None.

(d) Минимальное сходство (Minimum similarity) — минимальный порог сходства, ниже которого совпадение считается равным 0. Недоступно при выбранной мере «Равенство».

(e) Максимальное количество границ (Bound number limit) — максимальное число границ для заданного CM в левой части. Значением является целое неотрицательное число, где 0 означает отсутствие ограничений. Недоступно при выбранной мере «Равенство».

6. Нужно ли отсекать пересечения (Prune nondisjoint) — режим отсечения зависимостей, где одно и то же сопоставление столбцов (Column match, далее CM) колонок участвует и в левой, и в правой части. Принимает логическое значение (true/false).

На вход примитиву может подаваться как один датасет, так и два (Left и Right tables). Если используются две таблицы, то при задании сопоставления столбцов первый столбец берется из левой, а второй — из правой.

## 4.7. ADC

Неформальное определение DC гласит: «Для всех пар разных строк в таблице никогда не должно выполняться какое-либо условие». Формально: $\forall s, t \in R, s \neq t : (p_1 \wedge ... \wedge p_m)$, где $p_i$ — предикаты вида $column_i\ op\ column_j$, где $op \in \{>, <, \leq, \geq, =, \neq\}$.

Иногда допустимо, что значение DC может быть приближенным, то есть небольшое количество пар строк может нарушить его. Для этого используется метрика $g_1$, которая проверяет, какая доля из всех пар строк нарушает DC, и если эта доля ниже выбранного порогового значения, результат считается «достаточно допустимым».

Для поиска DC в Desbordante реализован алгоритм **FastADC** [21], который принимает следующие параметры:

1. Пороговое значение (Evidence threshold) задает долю пар строк, которые должны удовлетворять DC, чтобы результат считался допустимым. Значение является вещественное число в интервале [0, 1].

2. Длина сегмента (Shard length) предназначается для разбиения набора данных на сегменты для распараллеливания. Значением является целое число от 0 до количества строк в таблице включительно, где 0 означает, что разбиение отсутствует, поэтому весь набор данных обрабатывается одновременно.

3. Минимальная доля общих значений между двумя столбцами (Minimum shared value). Если для некоторых двух атрибутов значение меньше, чем данный параметр, то алгоритм не будет проверять между собой данные столбцы на равенство ($=$, $\neq$). Значением является вещественное число в интервале [0, 1].

4. Порог сравнения средних значений (Comparable threshold) — пороговое значения для определения, будет ли алгоритм применять неравенства ($<$, $\leq$, $>$, $\geq$). Если для некоторых двух столбцов отношение средних значений меньше, чем данный параметр, то алгоритм не будет сравнивать между собой данные столбцы. Значением является вещественное число в интервале [0, 1].

5. Разрешение пересечения атрибутов (Allow cross columns) — параметр, определяющий могут ли быть ограничения-отрицания между разными атрибутами. Принимает логические значения (true/false). Если false, ограничения строятся только в пределах одного столбца (например, A $\neq$ A’ для дубликатов).

## 4.8. AC

Алгебраические ограничения — это отношения вида $a_1 \oplus a_2 \in I$, где $a_1,\ a_2$ — атрибуты численного типа, операция $\oplus \in \{+,\ -,\ *,\ /\},\ I$ — интервал (bump), возможно, вырожденный.

Для поиска алгебраических ограничений в Desbordante реализован алгоритм **BHUNT** [37, 38], который принимает следующие параметры:

1. Максимальное количество интервалов (Bumps limit), которые будут рассматриваться. Значением является целое неотрицательное число, где 0 означает отсутствие ограничений, и алгоритм может находить любое количество интервалов.

2. Сид (Seed) — случайное начальное число для выборки данных. Нужен для воспроизводимости результатов, то есть если его изменить, может получиться другая выборка и, как следствие, другие интервалы. Значением является целое неотрицательное число.

3. Максимальное количество итераций (Iterations limit) при поиске интервалов. Чем больше значение, тем дольше работает алгоритм, но тем точнее могут быть результаты. Значением является целое число не меньше 1.

4. Нечеткость (Fuzziness) — допустимая доля выбросов (аномальных записей) в данных, то есть какая доля может не попадать в найденные интервалы. Чем меньше значение, тем строже алгоритм. Значение является вещественное число в интервале (0, 1].

5. Вероятность (P fuzz) того, что доля выбросов не превысит fuzziness. Чем ближе к 1, тем строже отбор. Значение является вещественное число в интервале (0, 1].

6. Вес (Weight) — параметр, влияющий на длину и количество интервалов. Значение является вещественное число в интервале (0, 1]. Ближе к 0 — алгоритм будет выдавать много коротких интервалов, ближе к 1 — мало длинных интервалов.

7. Бинарная операция (Operation), которая применяется к данным при поиске зависимостей. Значением является одна из четырех операций: сложение, вычитание, умножение, деление.

## 4.9. NAR

Численные ассоциативные правила (NAR) являются расширением традиционных ассоциативных правил (AR). Отличие состоит в работе с числовыми и категориальными данными (например, сколько единиц товара было приобретено) вместо двоичных (например, был ли приобретен товар или нет), что делает NAR более гибким для обнаружения взаимосвязей в наборах данных с числовыми данными. Более формально NAR можно описать так: пусть $A = \{A_1,\ A_2,\ ...,\ A_n\}$, $B = \{B_1,\ B_2,\ ...,\ B_m\}$ — атрибуты отношения, $I = \{I_1,\ I_2,\ ...,\ I_n\}$, $J = \{J_1,\ J_2,\ ...,\ J_n\}$ — множества значений, которые могут принимать соответствующие атрибуты, $p$ — достоверность (confidence). Тогда $A \in I \to B \in J$ с вероятностью $p$.

Для обнаружения NAR в Desbordante реализован алгоритм **DES** [18, 19], основывающийся на методе дифференциальной эволюции[13]. Входными данными служат следующие параметры:

1. Минимальная поддержка (Minimum support) — доля кортежей, удовлетворяющих ограничениям на левую и правую часть среди всех кортежей в таблице. Значением является вещественное число в интервале [0, 1].

2. Минимальная достоверность (Minimum confidence) — доля кортежей, удовлетворяющих ограничениям на левую и правую часть среди кортежей, удовлетворяющих лишь условию левой части. Значением является вещественное число в интервале [0, 1].

3. Размер популяции (Population size) — число индивидов в популяции в течение исполнения алгоритма. Значением является целое неотрицательное число.

[13] https://en.wikipedia.org/wiki/Differential_evolution (дата обращения: 1 мая 2025 г.)

4. Максимальное число вычислений параметра приспособленности (Maximum fitness evaluations). Значением является целое неотрицательное число.

5. Вероятность кроссовера (Crossover probability) — вероятность мутировать гену в новом индивиде. Значением является вещественное число в интервале [0, 1].

6. Дифференциальный масштаб (Differential scale) — размах мутаций. Значением является вещественное число, строго большее 0.

7. Дифференциальная стратегия (Differential strategy) — это стратегия поведения алгоритма DES. В разных стратегиях по разному получаются новые индивиды. На текущий момент имплементирована только одна — Rand1Bin, поэтому данный параметр не принимается алгоритмом.

## 4.10. Выводы

С учетом количества примитивов, которые предстоит добавить, текущий дизайн страницы примитивов становится неудобным ввиду отсутствия поиска и малой информативности (на экране помещается не более десяти примитивов при разрешении экрана $1920 \times 1200$ и масштабе страницы в браузере 80%).

Описания примитивов, а также названия метрик приближенных примитивов содержат математические обозначения, поэтому важно корректно их отображать. Например, удобнее для понимания в научном обществе, если опции будут отображаться не как текст (g1, pdep, tau, mu+, rho), а как стандартные символы ($g_1,\ pdep,\ \tau,\ \mu^+,\ \rho$).

В конфигурации DD требуется использование двух таблиц. Однако текущая реализация выбора файла такое не поддерживает.

Примитив MD может работать как на одном, так и на двух датасетах. Поэтому помимо возможности выбирать несколько файлов, нужно предусмотреть случай, когда пользователь выбрал Right table, но потом изменил решение.

В четырех примитивах встречаются параметры, значения в которых зависят от выбранного датасета:

1. AFD (Shard length — требуется количество строк);
2. MD (Column matches — требуются названия столбцов);
3. DD (Num columns, Num rows — требуется количество столбцов и строк соответственно);
4. AFD Verification (LHS, RHS — требуются названия столбцов).

Кроме того, существуют параметры, зависимые от других параметров. В случае FD и AFD, которые представлены несколькими алгоритмами, от выбранного зависит набор параметров. В случае MD при настройке CM от выбора метрики зависит количество настроек. Из этого следует, что надо соответствующим образом выводить поля ввода для них.

Реализация алгоритма позволяет для конфигурации отдельного CM указывать собственную меру сходства, а также функцию для преобразования значений столбцов перед вычислением, однако в веб-приложении отсутствует возможность вводить код. На данный момент её добавление не предусмотрено, поэтому данные опции указать не получится.

При проектировании страниц с результатами работы примитивов важно учитывать, какую информацию они несут. Примитивы FD, AFD, PFD, DD, MD, NAR трактуются как «Из некоторой информации *следует* некоторая другая информация». А значит, можно попробовать использовать компонент для вывода зависимостей, реализованный ранее. Однако он позволяет отображать только названия атрибутов и достоверность (confidence) при наличии, но в DD и NAR также фигурируют их значения, а в MD — два столбца и метрика (CM). Примитив ADC трактуется совершенно иначе (см. пункт 4.7), поэтому требует создания нового компонента. Фронтенд-часть для AC и AFD Verification была реализована ранее [29], поэтому для них не нужно проектировать дизайн. Однако следует учесть, что за прошедшее время в конфигурации AFD Verification появился третий параметр — «Равенство нулю».

Добавление перечисленных выше примитивов требует изменения не только фронтенд-части. Рассматриваемая версия бэкенда не содержит полного набора данных, в том числе количество строк, столбцов, факт наличия заголовка и сам заголовок, необходимых для настройки конфигурации некоторых примитивов (см. пункт 5.3).

# 5. Описание решения

Учитывая необходимость разработки новых интерфейсных страниц и модификации существующих, предварительно были созданы соответствующие макеты. Проектирование осуществлялось с использованием онлайн-редактора Figma[14].

## 5.1. Реимплементация страницы выбора примитива

Новый дизайн (см. рис. 12) включает в себя поле для поиска примитива по названию, модальное окно для фильтрации, а также карточки примитивов с названием, тегами и двумя видимыми строчками описания, которое можно открыть полностью в модальном окне по нажатию на кнопку «Show more».

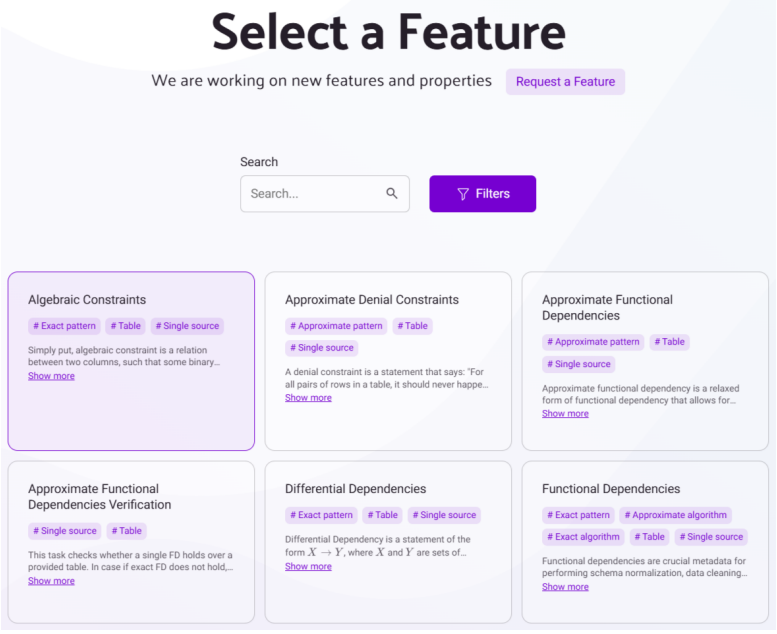



Рис. 12: Страница выбора примитива (настоящая версия)

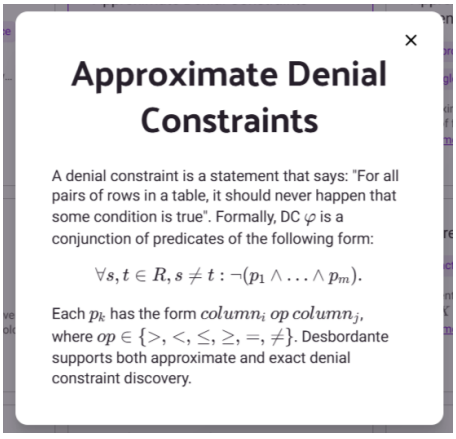



Рис. 13: Описание примитива ADC

Поиск и фильтрация по тегам реализованы на стороне фронтенда, поскольку видимая пользователю информация (названия, теги, описание) находится там. Надо отметить, что данных не так много, поэтому это не скажется на производительности.

[14] https://www.figma.com/design/SpgmZRz27y8D2fnMKGzU30/Desbordante

Чтобы отфильтровать примитивы по интересующим тегам, пользователь должен нажать на кнопку «Filters» и в открывшемся модальном окне выбрать один или несколько вариантов из выпадающего списка. После этого необходимо подтвердить свой выбор нажатием на кнопку «Apply», в ином случае ничего не произойдет.

На момент написания данной работы список тегов следующий:

- По количеству датасетов
    1. Единственный (Single source)
    2. Несколько (Multi source)
- По типу данных
    1. Таблица (Table)
    2. Транзакционная таблица (Transactional)
    3. Граф (Graph)
- По типу паттерна
    1. Точный (Exact pattern)
    2. Приближенный (Approximate pattern)
- По типу алгоритмов
    1. Точный (Exact algorithm)
    2. Приближенный (Approximate algorithm)

Различие в типе алгоритма и типе паттерна в том, что приблизительный паттерн настраивается с указанием допустимого процента ошибки, и при некоторой конфигурации может быть точным, что неверно для приблизительного алгоритма, который использует различные эвристики для нахождения результата.

Для отображения специальных математических символов была интегрирована поддержка KaTeX[15] посредством библиотеки react-katex[16].

[15] https://en.wikipedia.org/wiki/KaTeX (дата обращения: 1 мая 2025 г.)

[16] https://www.npmjs.com/package/react-katex, (дата обращения: 1 мая 2025 г.)

Она обеспечивает корректную визуализацию формул в соответствии с принятой в научной литературе нотацией, что облегчает читаемость формул.

## 5.2. Реимплементация выбора файла с данными

В текущем решении выбор файла и задание конфигурации алгоритма представляются отдельными шагами. В данной работе эти две стадии объединились в одну, и файл с данными воспринимается как параметр.

Реализация нового решения (см. рис. 14) включает создание поля ввода, доступного только для чтения. По нажатию на него открывается модальное окно, содержимое которого визуально повторяет прошлое решение. Одно поле ввода позволяет выбрать один файл.

Отличием от предыдущего решения стала кнопка «Unselect», отменяющая выбор.

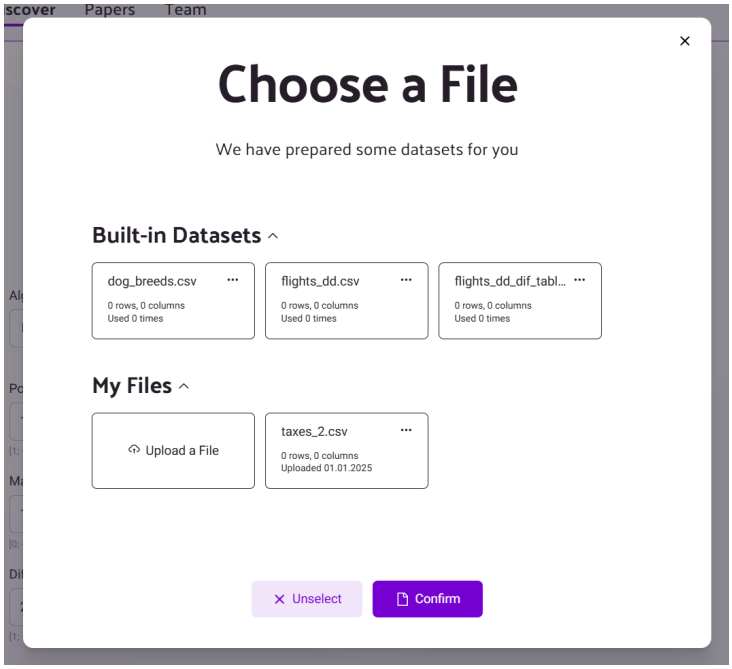



Рис. 14: Модальное окно для выбора файла

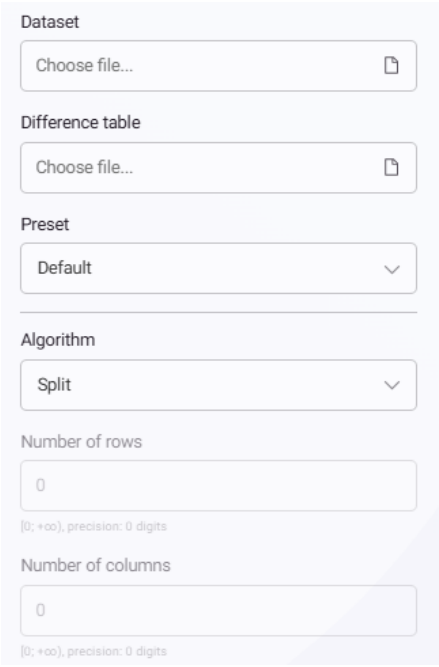



Рис. 15: Конфигурация примитива DD

## 5.3. Страница конфигурации алгоритма для новых примитивов

В разделе 4 были перечислены параметры, которые принимают соответствующие примитивам алгоритмы. Компоненты для их ввода пользователем уже были реализованы ранее: ввод числа, слайдер для выбора числа из заданного диапазона, выбор опций из выпадающего списка, в том числе множественный, переключатель выбора определенной опции (checkbox). Также в данной работе был реализован выбор файла.

### Новые поля ввода для MD

Идея наивной реализации, состоящая в добавлении сразу всех необходимых полей по нажатию на специальную кнопку, не была пригодной, так как тогда экран был бы визуально перегружен. Решением стало создание нового поля (см. рис. 16), которое отображает, сколько CMs пользователь уже выбрал, а по нажатию на него открывается модальное окно (см. рис. 17). В нем содержится столько полей, сколько указано в значении первого поля ввода, каждое из которых содержит информацию о метрике и выбранных столбцах. По нажатию на конкретное поле открывается другое модальное окно (см. рис. 18), которое состоит из трех полей с выбором из выпадающего списка (метрика и два столбца) и двух полей с вводом числа (максимальное количество границ и минимальное сходство).

Чтобы добавить еще одно CM, надо нажать на кнопку «Add column match». Тогда появится пустое поле ввода, и пока пользователь не настроит новое CM, кнопка добавления будет заблокирована. В случае, если модальное окно было закрыто прежде, чем заполнилось пустое поле, то оно не будет считаться за еще одно CM и при следующем открытии модального окна его не будет.

Если пользователю потребуется изменить некоторое CM, то он может нажать на соответствующее поле и изменить выбор параметров. Там же он может его удалить нажатием на кнопку «Delete».

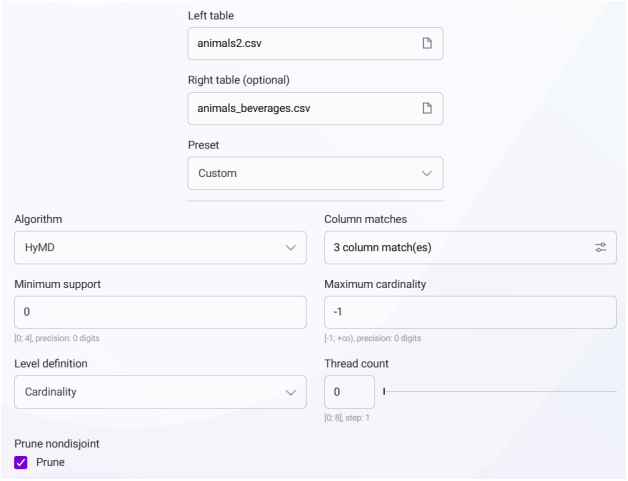



Рис. 16: Страница конфигурации для примитива MD

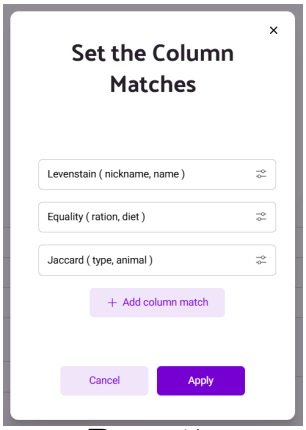



Рис. 17: Модальное окно для CMs

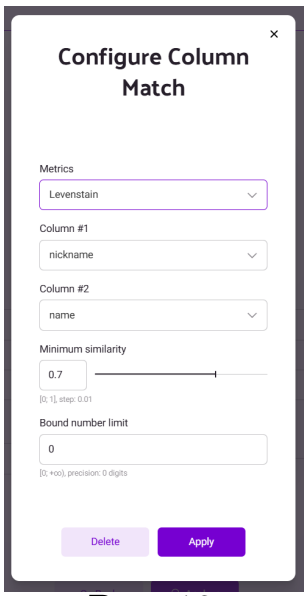



Рис. 18: Настройки конкретного CM

### Зависимые параметры

Тогда пока файл не выбран, соответствующие им поля ввода недоступны для взаимодействия (см. рис. 15).

После выбора датасета сервер присылает информацию о нем, например, заголовок, количество строк и столбцов. В случае изменения датасета выбранные настройки сбрасываются.

В случае FD и AFD, которые представлены несколькими алгоритмами, в зависимости от выбранного меняется набор полей ввода для соответствующих параметров. В случае MD при настройке CM при выборе метрики «Equality» для взаимодействия недоступны поля «Minimum similarity» и «Bound number limit», поскольку соответствующие им опции не предусмотрены в алгоритме.

## 5.4. Реимплементация вывода результата работы примитива

Для каждого примитива создан свой файл, в котором на страницу вывода результата передается список нужных вкладок, которые отобра-

жаются в виде меню. По нажатию на кнопку меню, меняется url-адрес, и в ответ на это событие открывается соответствующая вкладка.

Такая реализация обусловлена использованием App Router[17]. Подробнее про это можно узнать в выпускной квалификационной работе Белоконного Сергея.

### Ревизия и модернизация компонентов

Было принято решение убрать функцию просмотра выбранного датасета, так как данная информация не является результатом работы примитива, но добавляет сложность в веб-приложение за счет излишней функциональности.

Также в новое решение не вошла вкладка «Статистика». Для большого количества атрибутов (больше 8) такое представление неудобно за счет появляющейся секции «Other». Для небольшого (3-5) — не имеет смысла ввиду наглядности результата. Кроме того, так как функциональные зависимости в левой части могут содержать несколько атрибутов, то не всегда очевидно, как интерпретировать результат.

Контекст фильтрации отсылает к следующему преобразованию.

С одной стороны, текущая реализация фильтрации требует дополнительных ресурсов для вычисления, для новых примитивов не имеет смысла, а для старых потеряла актуальность.

С другой стороны, пользовательский ввод при поиске по названию атрибутов можно воспринимать как фильтр. Это наблюдение послужило основанием для разработки новой фильтрации. Ее реализация так же включает в себя модальное окно, открывающееся по нажатию на кнопку «Filter», но в котором названия атрибутов и другие опции в зависимости от примитива (см. пункт 5.5) выбираются из выпадающего списка, при этом реализован множественный выбор. Чтобы применить выбранные опции, нужно нажать на кнопку «Apply», иначе список результатов не изменится.

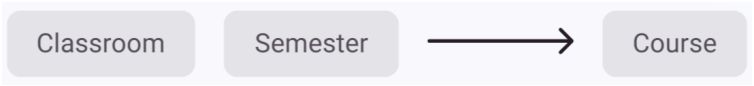
[17]Один из доступных в фреймворке Next.js вариантов маршрутизации. Подробнее https://nextjs.org/docs/app (дата обращения: 10 мая 2025 г.)

## 5.5. Страница вывода результата новых примитивов

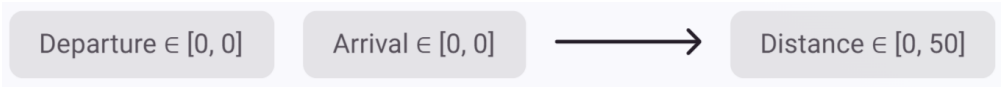
Далее будет использоваться понятие «инстанс» как атомарный элемент результата.

Для визуализации результатов работы AFD, PFD, DD, MD, NAR использован компонент, реализованный ранее для FD, однако с модификацией, заключающейся в расширении списка принимаемых параметров. Для каждого примитива опционально передать свою форматирующую функцию, которая позволит правильно отобразить итоговый результат, в противном случае выведутся только названия атрибутов.

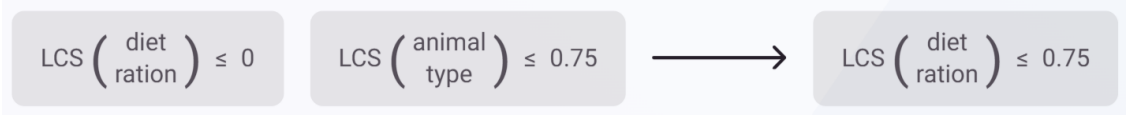
Дизайн компонента для вывода инстансов примитива ADC основывается на определении DC (см. пункт 4.7).



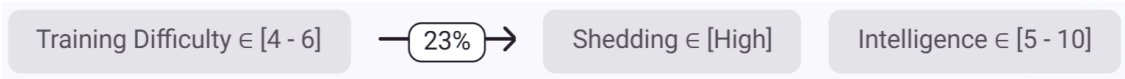

(a) FD, AFD, PFD




(b) DD




(c) MD




(d) NAR


(e) AC


(f) ADC

Рис. 19: Примеры инстансов результатов работы примитивов

Уже упоминалось, что результаты для примитивов AC и AFD Verification были спроектированы ранее. Однако тогда для них не было бэкенда ни на TypeScript, ни на Python, а первая версия pip-пакета появилась только к окончанию разработки фронтенд-части (декабрь 2023). Поэтому вывод, какой функционал будет добавлен, частично строился на предположениях разработчиков. К сожалению, при реализации данных примитивов в текущей работе выяснилось, что для гистограмм AC не удастся получить необходимую информацию из pip-пакета Desbordante, а значит, эту вкладку невозможно реализовать. У AFD Verification предполагалось, что можно будет узнать наиболее частое значение правой части, а не только его долю относительно всех значений. Однако такую информацию также не удалось получить.

### Фильтрация, сортировка и пагинация

Для анализа полученного результата его можно сортировать и фильтровать. Эти действия, как и само вычисление зависимостей, происходят на сервере. Реализация данных функций также является частью текущей работы.

В текущей версии результат также можно экспортировать, однако в новой эта функциональность не реализована, так как эта задача относится к компетенции бэкенд-разработчиков.

Для большинства примитивов (FD, AFD, PFD, DD, NAR, AC, ADC) фильтрация происходит по названиям атрибутов, которые можно выбрать из выпадающего списка (несколько опций). У MD тоже есть фильтрация по названиям атрибутов, но она происходит только по правой части ввиду того, что в левой всегда будут присутствовать все указанные в конфигурации алгоритма CMs. Это можно изменить, убрав из левой части CM с нулевыми границами (кнопка «Visibility»). Также присутствует дополнительный параметр для фильтрации — метрика. У AFD Verification ввиду кардинального отличия результатов (не выводятся атрибуты напрямую) фильтрация (кнопка «Visibility») заключается в сокрытии столбцов, отличных от левой и правой части проверяемой зависимости.

Чтобы обеспечить возможность фильтрации по атрибутам, сервер вместе с результатом работы алгоритма присылает заголовок выбранного датасета. В случае, если при загрузке файла было указано, что таблица не имеет заголовка, сервер формирует массив из чисел, соответствующих нумерации столбцов.

Параметры сортировки между примитивами различаются значительнее, чем фильтрации. Однако тут также для большинства примитивов (FD, AFD, PFD, DD, MD, NAR) сортировка заключается в упорядочивании по конкатенации строк левой и правой частей (LHS и RHS Names). В примитиве NAR также присутствует сортировка по достоверности (confidence). В примитивах AC и ADC нет понятия левой и правой частей, однако там сохраняется сортировка по конкатенации строк (Attributes names). Помимо этого в ADC происходит сортировка по количеству конъюнктов в инстансе (Number of conjuncts), а в AC по количеству интервалов (Number of intervals) и исключений (Number of outliers). AFD Verification, как и в случае с фильтрацией, отличается от других примитивов. Параметрами сортировки кластеров являются:

- Доля наиболее частого значения правой части по отношению ко всем значениям (Most frequent RHS value proportion);
- Количество различных значений правой части (Distinct RHS values);
- Размер (Size).

Также можно выбрать, в каком порядке сортировать: по возрастанию (Ascending) или по убыванию (Descending).

Для объемных исходных данных результат работы алгоритма может быть большим. На этот случай на сервере реализована пагинация, выводящая 6-10 инстансов на странице (в зависимости от примитива).

Таким образом, результатом проведенной работы по реимплементации примитива FD и добавлению новых примитивов в фронтенд-часть стала архитектура на рис. 21.

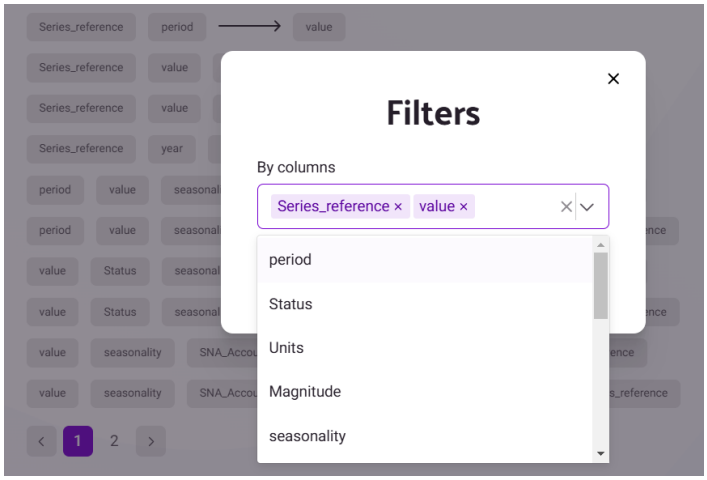



Рис. 20: Модальное окно для новой фильтрации

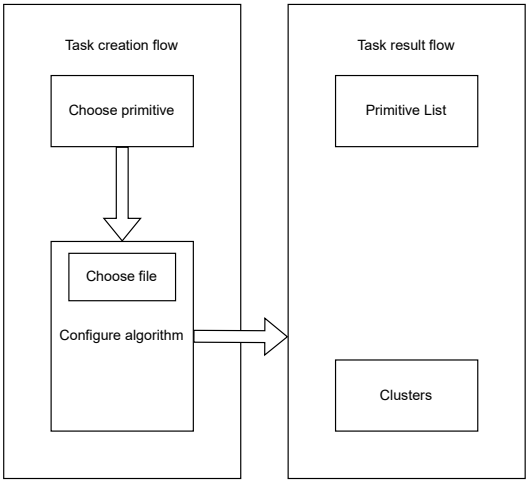



Рис. 21: Новая структура полной обработки задачи

## 5.6. Реализация бэкенда для новых примитивов

В качестве примера бэкенд-разработчиками проекта реализован поиск функциональных зависимостей (создание задачи и получение результата). Однако не представлены фильтрация, сортировка и пагинация для найденных инстансов. Поэтому настоящая работа включает также реализацию перечисленных функций.

В процессе разработки указанного функционала были созданы дополнительные файлы `sort.py` и `filter.py` для каждого примитива. Каждый из них содержит опции сортировки и фильтрации соответственно, а также функции, их реализовывающие. Изменены параметры запроса. Если первоначально клиент посылал GET-запросы только с идентификатором задачи, то теперь в запросе присутствуют также опции для фильтрации (`filter_options` и `filter_params`) и сортировки (`sort_option` и `sort_direction`).

Аналогично, как и при создании задачи, сервер по названию примитива определяет, какой класс потребуется для выполнения текущего действия. Функции `match_filter_by_primitive_name` и `match_sorter_by_primitive_name` так же находятся в файле `utils.py`. При интеграции нового примитива в веб-приложение так же, как в случае с классами задачи, конфигурации и результата, нужно сопоставить название с классами сортировки и фильтра, а также добавить в Union-

тип классы их опций. Указанные действия проводятся в тех же файлах.

Новая файловая структура для некоторого примитива:

```
<primitive_short_name>/
├── algo_config.py
├── algo_name.py
├── filter.py
├── sort.py
└── task.py
```

Пагинация реализована следующим образом. Клиент вместе с параметрами сортировки и фильтрации отправляет в запросе `pagination_offset` и `pagination_limit`, которые отвечают за то, сколько нужно пропустить результатов и сколько выдать. Несмотря на то, что модели не однообразны (см. листинги 1 и 2), все примитивы поиска возвращают результат в виде списка. У AFD Verification (см. листинг 3) более сложная структура, которую не удалось бы преобразовать без добавления лишней информации в каждый кластер, что является нежелательным способом. Поэтому на данный момент решено реализовать пагинацию его результата на стороне клиента.

Кроме того, в пункте 4 упоминалась недостаточность хранимой на сервере информации о файлах. Работа над этим велась со стороны бэкенд-разработчика, однако из-за ошибок вовремя справиться не удалось. Поэтому в рамках данной работы согласовано оперативно реализовать временное решение. Оно заключается в создании дополнительного пути для GET-эндпоинта, чтобы разделить рабочее решение и имитатор функциональности.

Клиент запрашивает информацию о датасетах с указанными идентификаторами. Сервер получает их из хранилища, вычисляет нужные характеристики с помощью библиотеки pandas[18] и вместе с хранящейся информацией отправляет ответ. При этом результат вычислений никуда не сохраняется.

[18] https://pandas.pydata.org/

# 6. Апробация

Для апробации выбрано три методологии. Каждый из них ориентирован на различные аспекты оценки пользовательского опыта

Так как Desbordante, в узком смысле, инструмент для аналитики данных, то респонедентами стали люди, имеющие отношение к Data Science, а именно студенты и выпускники кафедры информационно-аналитических систем, ML-инженеры и аналитики. Кроме того, поскольку данная работа сосредоточена на веб-интерфейсе, то также опрашивались веб-программисты. Всего в апробации приняло участие 10 человек.

## 6.1. A/B-тестирование

Значимые изменения были внесены в интерфейс страниц для выбора примитива и выбора файла. Поэтому следовало сравнить прошлое и новое решение.

A/B-тестирование (сплит-тестирование) заключается в сравнении двух вариантов, которые показывают пользователям, а они решают, что им нравится больше. В качестве вариантов выступали развернутая ранее версия[19] и развернутое в качестве демонстрации новое веб-приложение[20], часть которого разработана в рамках данной работы.

Пользователям было предложено пройти пользовательский путь для получения результата примитива FD, так как он присутствует в обоих интерфейсах.

A/B-тестирование содержит следующие вопросы:

1. Какой способ представления примитивов кажется удобнее?
2. Какой способ выбора файла кажется удобнее?
3. Какой способ поиска зависимостей по названию примитива кажется удобнее?

[19] https://desbordante.unidata-platform.ru (дата обращения: 8 мая 2025 г.)
[20] https://desbordemo.store, (дата обращения: 8 мая 2025 г.)

Результаты иллюстрирует гистограмма на рис. 22. На ней видно, что во всех случаях новый интерфейс пользователи оценивают как более удобный.

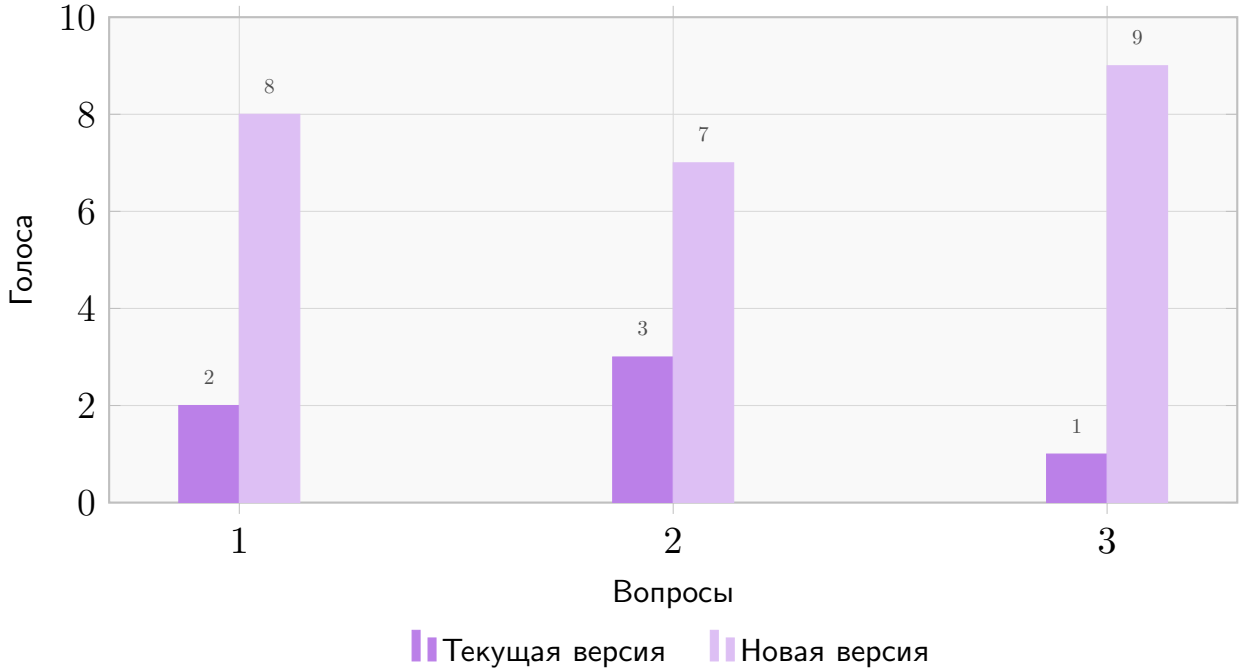



Рис. 22: Результаты A/B-тестирования для веб-интерфейса

## 6.2. SEQ

Следующим этапом был метод одного простого вопроса (Single Ease Question, SEQ). Респондентам предлагалось выполнить некоторое действие и оценить, насколько просто это было осуществить.

Данный метод дает понять разработчикам, насколько интуитивно понятна новая функциональность.

Ключевые принципы интерпретации:

- 1-3 — Критические сложности.
- 4 — Нейтрально.
- 5-7 — Удобно/очень легко.

SEQ-тестирование содержит следующие вопросы:

1. Найти примитив который одновременно помечен тегом «Approximate pattern» и содержит в названии подстроку «Constraints».

2. Найти и выбрать примитив который одновременно помечен тегами «Single Source» и «Multi Source».

3. Выбрать два файла (Left table и Right table), а затем убрать опциональный.

4. Настроить 4 произвольных Column match'а, затем удалить любой из них. Обязательное условие: 4 разных метрики и 4 разных пар столбцов.

5. На странице с результатом убрать элементы в левых частях с нулевыми границами.

6. Оставить только зависимости, где в правой части присутствует метрика «Jaccard».

Пояснение: шаги нужно выполнять поочередно. На первом должен найтись ADC, а на втором MD.

Результаты иллюстрирует гистограмма на рис. 23. Проблемные зоны (1-3 балла) изображены оттенками серого, а позитивные (4-7 балла) — сиреневого.

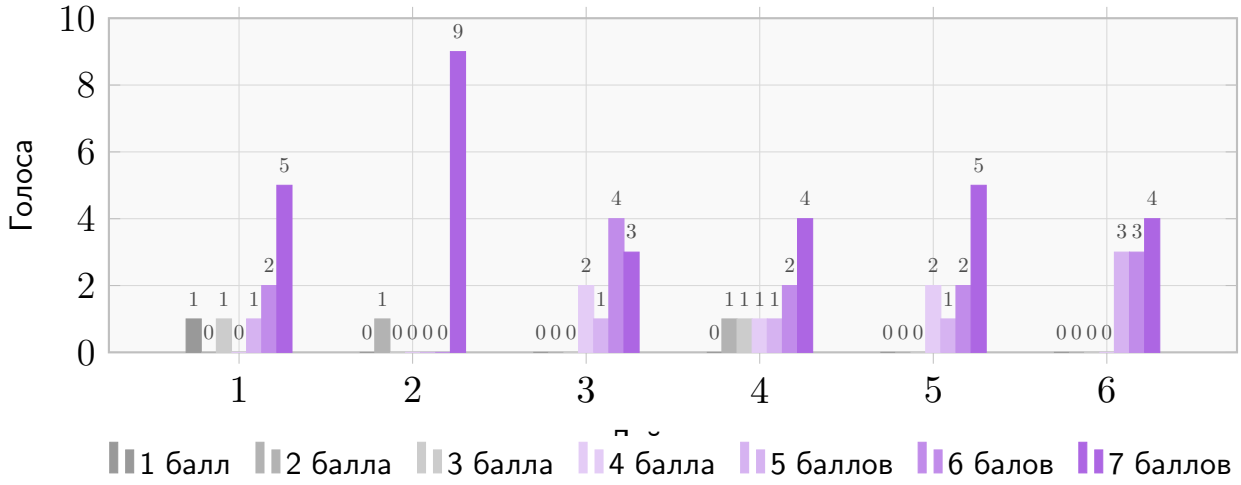



Рис. 23: Результаты SEQ-тестирования для веб-интерфейса

Как можно видеть на рис. 24, среднее значение для каждого действия превышает 5, что говорит об удобстве выполнения.

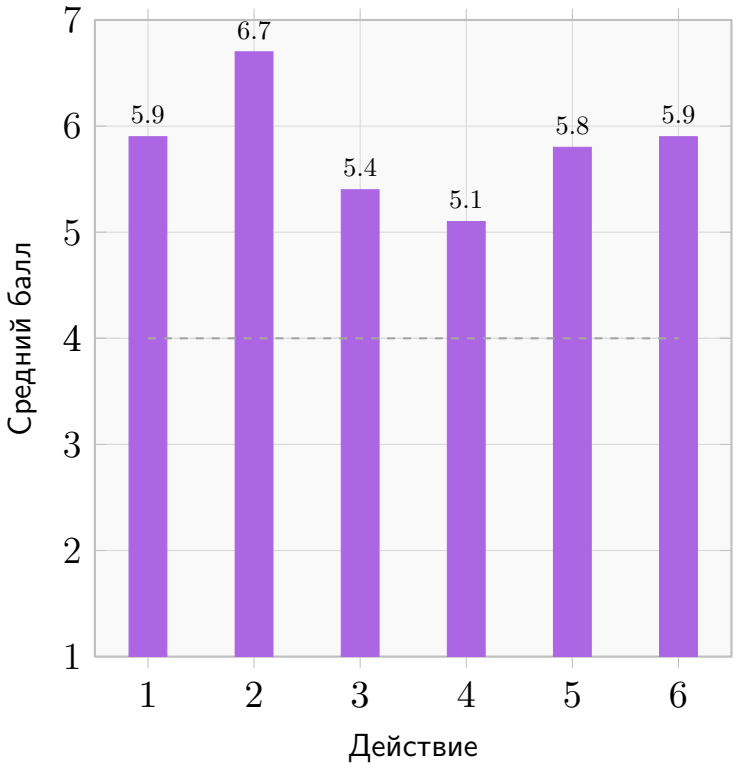



Рис. 24: Средние оценки SEQ-тестирования для 10 пользователей. Пунктирная линия показывает нейтральный уровень (4 балла)

## 6.3. SUS

После того, как пользователи поработали с системой, им предлагается ответить на вопросы, направленные на анализ общего восприятия опыта взаимодействия.

Для этого выбрана шкала удобства использования (System Usability Scale, SUS).

Пользователям предлагается оценить степень согласия с 10 утверждениями по 5-бальной шкале, где 1 — полностью не согласен, а 5 — полностью согласен.

Утверждения для SUS-опроса:

1. Я думаю, что мне понравится часто пользоваться этой системой.
2. Я считаю веб-интерфейс излишне сложным.

3. Я считаю систему простой в использовании.
4. Мне понадобится помощь специалиста, чтобы пользоваться этой системой.
5. Функции в этой системе хорошо согласуются.
6. В системе слишком много противоречий.
7. Я могу представить, что большинство людей быстро научатся пользоваться этой системой.
8. Я считаю систему неудобной.
9. Я чувствовал(а) себя уверенно, пользуясь системой.
10. Мне нужно было изучить дополнительную информацию, прежде чем начать работать с системой.

Для каждого респондента полученные результаты обрабатываются по следующей схеме:

- Для вопросов 1, 3, 5, 7, 9 (прямые) вычесть из полученного значения 1.
- Для вопросов 2, 4, 6, 8, 10 (обратные) вычесть полученное значение из 5.
- Вычислить сумму преобразованных значений.
- Умножить полученную сумму на 2.5.

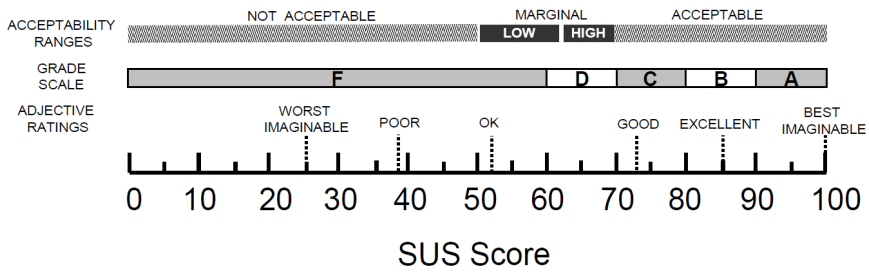



Рис. 25: Градация результатов SUS

| № | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 5 | 4 | 3 | 5 | 3 | 4 | 4 | 5 | 5 |
| 2 | 1 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |
| 3 | 3 | 5 | 3 | 3 | 5 | 3 | 4 | 3 | 4 | 5 |
| 4 | 3 | 1 | 2 | 3 | 1 | 2 | 1 | 3 | 4 | 1 |
| 5 | 4 | 5 | 5 | 4 | 5 | 5 | 5 | 4 | 4 | 5 |
| 6 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 1 |
| 7 | 4 | 4 | 4 | 3 | 5 | 4 | 5 | 4 | 5 | 4 |
| 8 | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 1 |
| 9 | 2 | 5 | 3 | 4 | 5 | 4 | 5 | 3 | 4 | 5 |
| 10 | 3 | 2 | 1 | 1 | 1 | 4 | 1 | 3 | 5 | 1 |
| **Итог** | 67.5 | 92.5 | 82.5 | 72.5 | 97.5 | 67.5 | 92.5 | 67.5 | 72.5 | 97.5 |

Таблица 1: Результаты SUS-опроса для веб-интерфейса

Результаты иллюстрирует таблица 1.

Средний результат проведенного опроса составляет 81, что отвечает оценке «B», согласно шкале на рис. 25 [2]. Средним показателем считается 68 баллов.

Таким образом, апробация показала, что новый интерфейс спроектирован удачно с точки зрения удобства использования.

# Заключение

В настоящей работе были выполнены следующие задачи:

1. реимплементирован существующий примитив по поиску функциональных зависимостей;
2. проанализированы новые примитивы и выработаны требования для их последующей интеграции в веб-приложение;
3. спроектированы и реализованы пользовательские интерфейсы с учетом добавления новых примитивов;
4. разработан бэкенд для новых примитивов;
5. проведена апробация.

Код работы можно найти в GitHub-репозиториях веб-приложения Desbordante[21].

[21] Фронтенд-часть: https://github.com/Pechenux/desbordante-frontend.
Бэкенд-часть: https://github.com/Desbordante/desbordante-server/tree/dev-old-tasks.
Никнейм: LiaSolo.

# Список литературы


[1] Agrawal Rakesh, Srikant Ramakrishnan. Fast Algorithms for Mining Association Rules in Large Databases // VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile / Ed. by Jorge B. Bocca, Matthias Jarke, Carlo Zaniolo. — Morgan Kaufmann, 1994. — P. 487–499. — URL: http://www.vldb.org/conf/1994/P487.PDF.

[2] 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.

[3] Brown Paul G., Hass Peter J. BHUNT: Automatic Discovery of Fuzzy Algebraic Constraints in Relational Data // VLDB 2003: Proceedings of the 29th International Conference on Very Large Data Bases. — VLDB Endowment, 2003. — P. 668–679. — URL: https://www.vldb.org/conf/2003/papers/S20P03.pdf.

[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] Efficient Discovery of Matching Dependencies / Philipp Schirmer, Thorsten Papenbrock, Ioannis Koumarelas, Felix Naumann // ACM Transactions on Database Systems. — 2020. — Vol. 45, no. 3. — P. 13:1–13:33. — URL: https://dl.acm.org/doi/10.1145/3392778.

[6] Fast Approximate Denial Constraint Discovery / Renjie Xiao, Zijing Tan, Haojin Wang, Shuai Ma // Proceedings of the VLDB Endowment. — 2022. — Vol. 16, no. 2. — P. 269–281. — URL: https://doi.org/10.14778/3565816.3565828.

[7] Functional Dependency Discovery: An Experimental Evaluation of Seven Algorithms / Thorsten Papenbrock, Jens Ehrlich, Jannik Marten et al. // Proc. VLDB Endow. — 2015. — Vol. 8, no. 10. — P. 1082–1093. — URL: http://www.vldb.org/pvldb/vol8/p1082-papenbrock.pdf.

[8] Kruse Sebastian, Naumann Felix. Efficient Discovery of Approximate Dependencies // Proc. VLDB Endow. — 2018. — mar. — Vol. 11, no. 7. — P. 759–772. — URL: https://www.vldb.org/pvldb/vol11/p759-kruse.pdf.

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

[10] Numerical Association Rule Mining: A Systematic Literature Review / Minakshi Kaushik, Rahul Sharma, Iztok Fister Jr., Dirk Draheim // arXiv preprint. — 2023. — July. — Vol. 1, no. 1. — P. 50. — arXiv:2307.00662. URL: https://arxiv.org/abs/2307.00662.

[11] Parciak Marcel et al. Measuring Approximate Functional Dependencies: A Comparative Study // 2024 IEEE 40th International Conference on Data Engineering (ICDE). — Utrecht, Netherlands, 2024. — P. 3505–3518. — 2312.06296.

[12] Rammelaere Joeri, Geerts Floris. Revisiting Conditional Functional Dependency Discovery: Splitting the "C" from the "FD" // Machine Learning and Knowledge Discovery in Databases / Ed. by Michele Berlingerio, Francesco Bonchi, Thomas Gärtner et al. — Cham : Springer International Publishing, 2019. — P. 552–568.

[13] Sneed Harry, Verhoef Chris. Re-implementing a legacy system // Journal of Systems and Software. — 2019. — Vol. 155. — P. 162–184. — URL: https://www.sciencedirect.com/science/article/pii/S0164121219301050.

[14] Chernishev George, Polyntsov Michael, Chizhov Anton et al. Solving Data Quality Problems with Desbordante: a Demo. — 2023. — 2307.14935.

[15] Song Shaoxu, Chen Lei. Differential Dependencies: Reasoning and Discovery // ACM Transactions on Database Systems. — 2011. — Vol. 36, no. 3. — P. 16:1–16:41. — URL: https://sxsong.github.io/doc/11tods.pdf.

[16] Wang Daisy Zhe et al. Functional Dependency Generation and Applications in Pay-As-You-Go Data Integration Systems // WebDB 2009. — 2009. — URL: http://webdb09.cse.buffalo.edu/papers/Paper18/webdb09.pdf.

[17] Баруткин И. Д. Реализация алгоритма проверки вероятностных функциональных зависимостей в профилировщике данных Desbordante. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/PFD%20-%20Ilia%20Barutkin%20-%20BA%20thesis.pdf (дата обращения: 1 мая 2025 г.).

[18] Волгушев И. Р. Интеграция эволюционного алгоритма поиска численных ассоциативных правил в профайлер Desbordante. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/DES%20Integration%20-%20Ivan%20Volgushev%20-%202024%20spring.pdf (дата обращения: 1 мая 2025 г.).

[19] Волгушев И. Р. Улучшение кода алгоритма по поиску численных ассоциативных правил в Desbordante. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/DES%20Refactoring%20-%20Ivan%20Volgushev%20-%202024%20autumn.pdf (дата обращения: 1 мая 2025 г.).

[20] Гайсин Э. Р. Реализация алгоритма поиска функциональных зависимостей Dep-Miner в системе Desbordante. — URL: https://github.com/Desbordante/desbordante-core/blob/main/

docs/papers/Dep-Miner%20-%20Eduard%20Gaisin%20-%202021%20spring.pdf (дата обращения: 1 мая 2025 г.).

[21] Морозко И. Д. Реализация алгоритма поиска запрещающих ограничений в платформе Desbordante. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/FastADC%20-%20Ivan%20Morozko%20-%202024%20spring.pdf (дата обращения: 1 мая 2025 г.).

[22] Нурмухаметов Р. Проектирование и реализация архитектуры нового веб-сервера Desbordante. — 2024. — URL:

[23] Полынцов М. А. Реализация алгоритма FastFDs в платформе Desbordante. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/FastFDs%20-%20Michael%20Polyntsov%20-%202021%20autumn.pdf (дата обращения: 1 мая 2025 г.).

[24] Рахмукова А. И. Реализация алгоритма поиска функциональных зависимостей AID-FD. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/AID-FD%20-%20Anna%20Rakhmukova%20-%202022%20spring.pdf (дата обращения: 1 мая 2025 г.).

[25] Салью А. К. Высокопроизводительный поиск функциональных зависимостей в данных. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/FDep%20-%20Arthur%20Saliou%20-%202021%20spring.pdf (дата обращения: 1 мая 2025 г.).

[26] Синельников М. А. Реализация алгоритма поиска дифференциальных зависимостей в рамках платформы Desbordante. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/SPLIT%20-%20Michael%20Sinelnikov%20-%202023%20autumn.pdf (дата обращения: 1 мая 2025 г.).

[27] Смирнов А. А. Оптимизация алгоритма DFD поиска функциональных зависимостей в рамках платформы Desbordante. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/DFD%20Optimization%20-%20Smirnov%20Alexandr%20-%202021%20autumn.pdf (дата обращения: 1 мая 2025 г.).

[28] Смирнов А. А. Реализация алгоритма DFD поиска функциональных зависимостей. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/DFD%20-%20Alexandr%20Smirnov%20-%202021%20spring.pdf (дата обращения: 1 мая 2025 г.).

[29] Соловьёва Л. В. Добавление новых примитивов в веб-версию Desbordante. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/Frontend%20AC%20AFD%20-%20Solovyova%20Liana-Yulia%20-%202023%20autumn.pdf (дата обращения: 1 мая 2025 г.).

[30] Струтовский М. А. Реализация алгоритмов поиска функциональных зависимостей в базах данных. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/Comparison%20-%20Maxim%20Strutovsky%20-%202021%20autumn.pdf (дата обращения: 1 мая 2025 г.).

[31] Струтовский М. А. Реализация алгоритмов поиска функциональных зависимостей на языке программирования C++. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/TANE%20-%20Maxim%20Strutovsky%20-%202019%20autumn.pdf (дата обращения: 1 мая 2025 г.).

[32] Струтовский М. А. Реализация современных алгоритмов для поиска зависимостей в базах данных. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/Pyro%20-%20Maxim%20Strutovsky%20-%202020%20spring.pdf (дата обращения: 1 мая 2025 г.).

[33] Шалашнов Е. Д. Реализация мер приближенных функциональных зависимостей в проекте DESBORDANTE. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/Tane%20AFD%20measures%20-%20Egor%20Shalashnov%20-%20spring%202024.pdf (дата обращения: 1 мая 2025 г.).

[34] Шальнев В. А. Автоматизация сборки компонентов и развертывания инфраструктуры приложения Desbordante. — 2024. — URL:

[35] Шлёнских А. А. Обзор сопоставляющих зависимостей и алгоритма их поиска HYMD. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/HyMD%20review%20-%20Shlyonskikh%20Alexey%20-%202023%20spring.pdf (дата обращения: 1 мая 2025 г.).

[36] Шлёнских А. А. Реализация алгоритма HYMD на C++. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/HyMD%20Initial%20Implementation%20-%20Shlyonskikh%20Alexey%20-%202023%20autumn.pdf (дата обращения: 1 мая 2025 г.).

[37] Щека Д. В. Реализация алгоритма BHUNT для поиска нечётких алгебраических ограничений. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/Algebraic%20Constraints%20-%20Dmitriy%20Shcheka%20-%202022%20spring.pdf (дата обращения: 1 мая 2025 г.).

[38] Щека Д. В. Реализация поиска аномалий в данных с помощью алгоритма поиска алгебраических ограничений BHUNT. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/Algebraic%20Constraints%20Exceptions%20-%20Dmitriy%20Shcheka%20-%202022%20autumn.pdf (дата обращения: 1 мая 2025 г.).

[39] Щукин И. В. Реализация алгоритма поиска функциональных зависимостей FD_Mine. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/FD_Mine%20-%20Ilya%20Shchuckin%20-%202021%20spring.pdf (дата обращения: 1 мая 2025 г.).

[40] Якшигулов В. Н. Реализация MVP нового веб-сервера Desbordante. — URL: https://github.com/Desbordante/desbordante-core/blob/main/docs/papers/Backendreimplementation-YakshigulovVadim-2023spring.pdf (дата обращения: 1 января 2025 г.).