https://doi.org/10.25559/SITITO.022.202601.126-136
УДК 004.92:025
Оригинальная статья

Интерактивная визуализация данных в цифровом репозитории DSpace 7x

А. С. Бондяков*, **А. О. Кондратьев**
Международная межправительственная организация Объединенный институт ядерных исследований, г. Дубна, Российская Федерация
Адрес: 141980, Российская Федерация, Московская область, г. Дубна, ул. Жолио-Кюри, д. 6
* aleksey@jinr.ru

Аннотация

В данной статье рассматривается проблема отсутствия встроенных инструментов визуализации в платформе цифрового репозитория DSpace 7x. Мы предлагаем архитектурное решение, основанное на глубокой интеграции интерактивных визуализаций с системой фасетной фильтрации платформы посредством реактивных потоков данных. Реактивная архитектура, реализованная на стеке технологий Angular и RxJS, обеспечивает мгновенное обновление диаграмм в ответ на действия пользователя без перезагрузки страниц. Новизна нашего подхода заключается в модификации ядра фронтенда DSpace путём внедрения централизованного сервиса ChartService для управления состоянием данных с использованием механизмов наследования компонентов из существующих фильтрующих компонентов. Визуальные компоненты реализованы с помощью библиотеки D3.js с применением оптимизированного паттерна enter/update/exit для эффективного управления обновлениями объектной модели документа (DOM). Сравнительное тестирование производительности на репрезентативной выборке научных записей демонстрирует существенное улучшение времени отклика по сравнению с традиционными императивными подходами. Экспериментальные результаты подтверждают, что основной прирост производительности достигается за счёт устранения избыточных сетевых запросов и, что ещё более значимо, благодаря высокой эффективности операций с DOM. Ключевой научный вклад работы состоит в разработке метода интеграции визуализации, который минимизирует изменения в исходной кодовой базе DSpace при сохранении высокой производительности. Предложенное решение демонстрирует отличную масштабируемость и бесшовную интеграцию в существующую экосистему репозитория, существенно расширяя его аналитические возможности. Описанный подход легко адаптируется для других систем управления цифровыми репозиториями, требующих интерактивной визуализации научных данных.

Ключевые слова: цифровой репозиторий, DSpace, интерактивная визуализация, реактивное программирование, Angular, RxJS, D3.js

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Для цитирования: Бондяков А.С., Кондратьев А.О. Интерактивная визуализация данных в цифровом репозитории DSpace 7x // Современные информационные технологии и ИТ-образование. 2026. Т. 22, № 1. С. 126-136. https://doi.org/10.25559/SITITO.022.202601.126-136

Original article

Interactive Data Visualization in the DSpace 7x Digital Repository

A. S. Bondyakov[*], **A. O. Kondratyev**
Joint Institute for Nuclear Research, Dubna, Russian Federation
Address: 6 Joliot-Curie St., Dubna 141980, Moscow region, Russian Federation
[*] aleksey@jinr.ru

Abstract
The paper addresses the lack of built-in visualization tools in the DSpace 7x digital repository platform. We propose an architectural solution based on deep integration of interactive visualizations with the platform's faceted filtering system through reactive data streams. A reactive architecture built on the Angular and RxJS stack enables instantaneous chart updates in response to user interactions without page reloads. The novelty of our approach lies in modifying the DSpace frontend core by introducing a centralized ChartService for data state management, leveraging component inheritance mechanisms from the existing filtering components. Visual components are implemented using the D3.js library, employing an optimized enter/update/exit pattern to efficiently manage updates to the Document Object Model (DOM). Comparative performance testing on a representative sample of scholarly records demonstrates significantly improved response times compared to conventional imperative approaches. Experimental results confirm that the primary performance gains stem from the elimination of redundant network requests and, more significantly, from highly optimized DOM operations. The principal scientific contribution is a visualization integration method that minimizes modifications to the original DSpace code while preserving high performance. The solution exhibits excellent scalability and seamless integration within the existing repository ecosystem, substantially enhancing its analytical capabilities. The proposed approach is readily adaptable to other digital repository management systems requiring interactive scientific data visualization.

Keywords: digital repository, DSpace, interactive visualization, reactive programming, Angular, RxJS, D3.js

Conflict of interests: The authors declare no conflict of interest.

For citation: Bondyakov A.S., Kondratyev A.O. Interactive Data Visualization in the DSpace 7x Digital Repository. *Modern Information Technologies and IT-Education.* 2026;22(1):126-136. https://doi.org/10.25559/SITITO.022.202601.126-136

Введение

Современные цифровые репозитории, такие как DSpace, составляют фундаментальную основу научной инфраструктуры, обеспечивая долговременное хранение и доступ к результатам исследовательской деятельности. С экспоненциальным ростом объемов научных данных возникает критическая потребность не только в эффективных инструментах поиска и извлечения информации, но и в средствах ее аналитической обработки и визуального представления [1].

Платформа DSpace 7x, несмотря на мощный механизм поиска и фасетной фильтрации через модуль Discovery, не предоставляет штатных инструментов визуализации данных [2-4]. Существующий пользовательский интерфейс ограничен текстовыми списками и таблицами, что не позволяет осуществлять комплексный анализ динамики научной активности, выявлять тенденции и закономерности в накопленных данных. Данное ограничение существенно снижает аналитический потенциал системы и затрудняет принятие обоснованных управленческих решений на институциональном уровне.

Анализ существующих решений выявил преобладание подходов, основанных на создании внешних аналитических сервисов, которые дублируют логику фильтрации и требуют сложной синхронизации состояний с основным репозиторием [5].

Основной вклад работы заключается в разработке и экспериментальной проверке реактивной архитектуры, обеспечивающей глубокую интеграцию интерактивной визуализации в пользовательский интерфейс DSpace 7x. В отличие от существующих решений, предлагаемый подход исключает необходимость дублирования бизнес-логики и дополнительных сетевых запросов, обеспечивая полную синхронизацию между текстовым и графическим представлением данных через механизм реактивных потоков.

Цель исследования

Разработка реактивной архитектуры для глубокой интеграции интерактивной визуализации данных в интерфейс цифрового репозитория DSpace 7x, обеспечивающей синхронизацию графиков с фасетными фильтрами без избыточных сетевых запросов и значительной модификации ядра системы.

1 Архитектурный анализ

1.1 Ограничения штатной архитектуры DSpace 7x и постановка задачи

Детальный анализ исходного кода DSpace 7x выявил архитектурные ограничения, препятствующие простой интеграции визуальных компонентов. Штатная система фасетной фильтрации, реализованная через компоненты SearchFacetFilterComponent и ее наследников, предоставляет данные исключительно в текстовом формате без механизмов для кастомизации отображения [6].

Для понимания архитектурного контекста необходимо пояснить ключевые компоненты системы. Модуль Discovery представляет собой штатный механизм платформы DSpace 7x для полнотекстового поиска и фасетной фильтрации, функционирующий на базе Apache Solr. Данный модуль предоставляет REST API для получения агрегированных данных, однако не включает средств для их графического представления.

Компоненты SearchFacetFilterComponent, SearchRangeFilterComponent и SearchHierarchyFilterComponent являются стандартными элементами пользовательского интерфейса DSpace 7x. Их основное назначение — обеспечение текстовой навигации и фильтрации результатов поиска через механизм фасетов. Эти компоненты реализуют логику отображения списков значений и иерархий, наследуясь от общего базового класса [7], что позволяет унифицировать обработку событий фильтрации, но ограничивает возможности визуализации исключительно текстовыми списками.

Задача исследования формулируется как разработка метода интеграции интерактивной визуализации, который:

- Минимизирует модификации ядра системы;
- Обеспечивает реактивную синхронизацию с существующими фильтрами;
- Сохраняет производительность при работе с большими объемами данных;
- Обеспечивает расширяемость для поддержки различных типов визуализаций.

1.2 Архитектура системы и обоснование выбора технологий

Решение по интерактивной визуализации интегрировано в существующую трехуровневую архитектуру DSpace 7x через расширение функциональности с помощью двух ключевых компонентов, разработанных в рамках исследования:

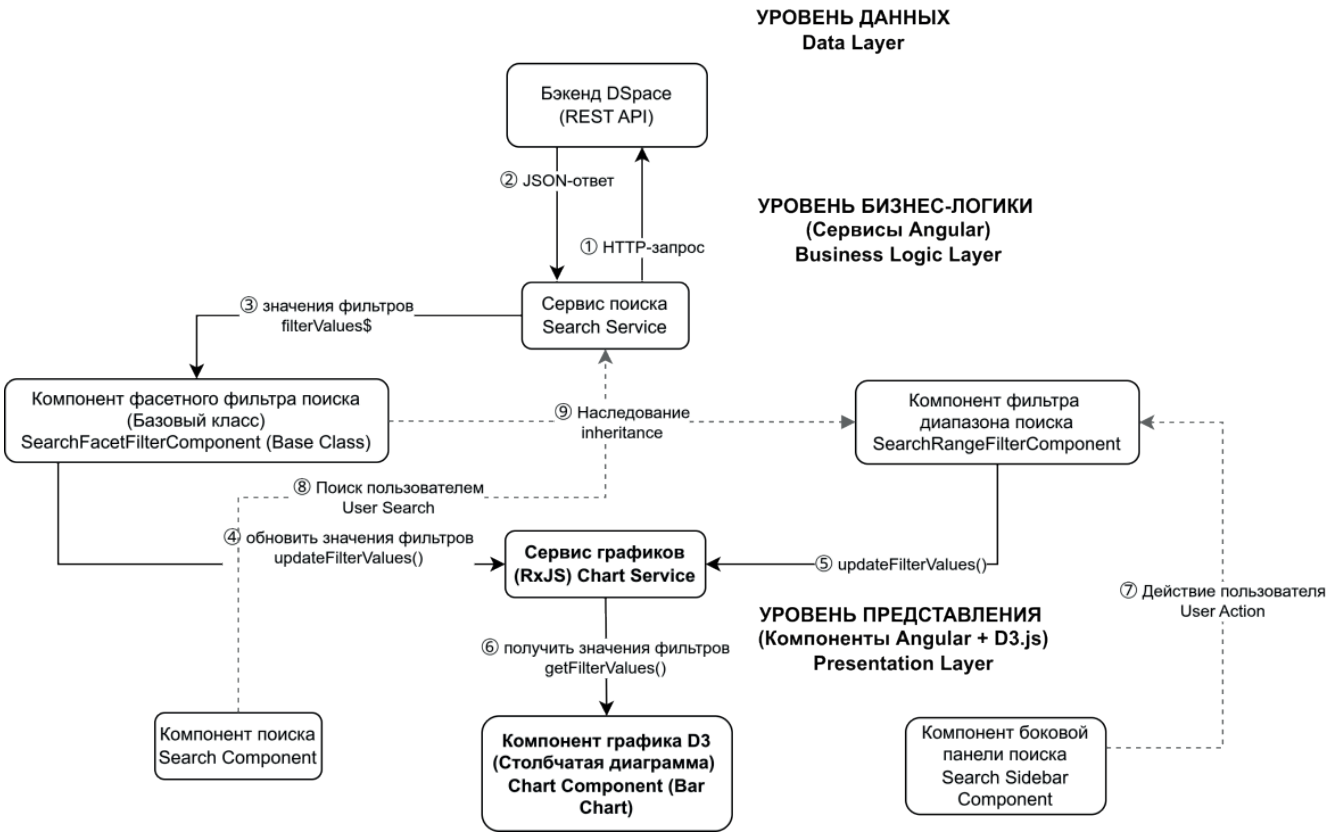
- ChartService — это новый сервис, разработанный в рамках данного исследования для централизованного управления состоянием данных визуализации. Сервис реализует паттерн реактивного программирования на основе RxJS BehaviorSubject [8] и обеспечивает синхронизацию между системой фильтрации DSpace и компонентами визуализации;
- ChartComponent — специализированный Angular-компонент, созданный авторами для отображения интерактивных графиков. Компонент использует библиотеку D3.js [9] для построения визуализаций и подписывается на реактивные потоки ChartService для автоматического обновления при изменении данных.

Взаимодействие разработанных компонентов ChartService и ChartComponent в контексте трех уровней DSpace представлено на Рис.1, который демонстрирует детализированную схему взаимодействия компонентов системы.

Диаграмма демонстрирует модифицированную трехуровневую архитектуру DSpace, обеспечивающую глубокую интеграцию интерактивных графиков в экосистему репозитория без нарушения его базовой функциональности.

Уровень данных (Data Layer) включает DSpace Backend с REST API, который предоставляет доступ к метаданным научных публикаций. Этот уровень отвечает за извлечение агрегированных данных фасетной фильтрации, используемых для построения визуализаций.

Уровень бизнес-логики (Business Logic Layer) содержит ключевые сервисы Angular, реализующие реактивную обработку данных. Центральным элементом архитектуры является ChartService, построенный на основе RxJS BehaviorSubject, который выполняет функцию централизованного узла управления состоянием для данных визуализации. SearchService обеспечивает взаимодействие с бэкендом DSpace, обрабатывая HTTP запросы и ответы в формате JSON.



Р и с. 1. Архитектура системы интерактивной визуализации в DSpace 7x
F i g. 1. System architecture for interactive visualization in DSpace 7.x

Источник: здесь и далее в статье все таблицы и рисунки составлены авторами.
Source: Hereinafter in this article all tables and figures were made by the authors.

Особенностью архитектуры является иерархия компонентов фильтрации: базовый класс SearchFacetFilterComponent и его производные классы. Такая организация позволяет минимизировать модификации кодовой базы DSpace — все компоненты-наследники автоматически получают доступ к функциональности визуализации через механизм наследования.
Уровень представления (Presentation Layer) объединяет Angular компоненты и библиотеку D3.js. Ключевые компоненты этого уровня включают: Chart Component, который подписывается на Observable потоки ChartService и обеспечивает реактивное обновление графиков, а также другие Angular-компоненты интерфейса. Chart Component обеспечивает высокопроизводительное обновление визуализаций через оптимизированные механизмы работы с DOM.

Архитектурные паттерны software engineering:
• Принцип единственной ответственности: ChartService инкапсулирует логику визуализации;
• Инверсия зависимостей: внедрение сервиса через конструктор;
• Наследование для расширяемости: иерархия компонентов фильтрации;
• Реактивные паттерны: Observer для синхронизации состояний.
Ключевые потоки данных, обозначенные на диаграмме цифрами 1-9, включают:
• HTTP запросы и JSON ответы между сервисами и бэкендом (1-2);
• Реактивные потоки данных между компонентами (3-6);
• Пользовательские взаимодействия (7-8);
• Отношения наследования между компонентами фильтрации (9).

Представленная архитектура решает поставленные в разделе 1.1 задачи, обеспечивая полную синхронизацию между текстовой фасетной фильтрацией DSpace и интерактивными визуализациями при минимальной модификации ядра системы. Эффективная реализация данной архитектуры потребовала тщательного выбора технологий, обусловленного следующими факторами:
• Выбор технологического стека обусловлен архитектурными особенностями DSpace 7x, где фронтенд реализован на Angular 16 [10, 11]. Решение построено на использовании штатных механизмов фреймворка: внедрение зависимостей и реактивное

программирование с помощью библиотеки RxJS, уже интегрированной в DSpace. Это позволило реализовать реактивные паттерны обработки данных для эффективного управления асинхронными потоками информации;
• Комплексное обоснование выбора D3.js [12-18] основано на сравнительном анализе требований проекта и характеристик доступных библиотек визуализации. Библиотека D3.js реализует оптимизированный паттерн enter/update/exit для эффективного обновления DOM, обеспечивая инкрементальное обновление визуализации: новые данные добавляются *(enter)*, измененные данные обновляются *(update)*, устаревшие данные удаляются *(exit)*. D3.js демонстрирует оптимальную производительность при работе с большими наборами данных благодаря минимальным накладным расходам и эффективному использованию механизма виртуального DOM через паттерн enter/update/exit. В отличие от библиотек высокого уровня, D3.js предоставляет полный контроль над процессом визуализации, обеспечивая создание специализированных графиков, адаптированных под специфические требования научных данных, включая реализацию нестандартных шкал, осей и сложных интерактивных элементов. Библиотека органично интегрируется с Angular благодаря единой философии работы с DOM и возможностям реактивного программирования, позволяя создавать высокопроизводительные компоненты визуализации, встроенные в общий поток данных приложения. Полная поддержка современных веб-стандартов (SVG, CSS, HTML5) обеспечивает кросс-браузерную совместимость и соответствие требованиям доступности, а развитая экосистема и активное сообщество гарантируют долгосрочную поддержку и развитие библиотеки.
Сравнительный анализ альтернативных решений показал следующие ограничения:
• Chart.js: Ограниченная кастомизация, проблемы с производительностью при частых обновлениях данных;
• Plotly.js: Высокий порог входа, избыточная функциональность для базовых задач;
• Google Charts: Зависимость от внешнего сервиса, ограничения лицензирования;
• ApexCharts: Коммерческая лицензия для корпоративного использования.

1.3 Модификация ядра фронтенда DSpace

Ключевым аспектом интеграции стало внедрение сервиса ChartService в архитектуру DSpace через механизм наследования [19-22]. Анализ иерархии компонентов показал, что все специализированные фильтры (SearchRangeFilterComponent, SearchHierarchyFilterComponent) наследуются от базового класса SearchFacetFilterComponent, что позволило осуществить централизованную модификацию.

Листинг 1. Модификация конструктора SearchFacetFilterComponent (TypeScript)
Listing 1. Modification of the SearchFacetFilterComponent constructor (TypeScript)

```typescript
// search-facet-filter.component.ts
constructor(
  protected searchService: SearchService,
  protected filterService: SearchFilterService,
  protected rdbs: RemoteDataBuildService,
  protected router: Router,
  @Inject(SEARCH_CONFIG_SERVICE) public
searchConfigService: SearchConfigurationService,
  @Inject(IN_PLACE_SEARCH) public inPlaceSearch:
boolean,
  @Inject(FILTER_CONFIG) public filterConfig:
SearchFilterConfig,
  @Inject(REFRESH_FILTER) public refreshFilters:
BehaviorSubject<boolean>,
  protected chartService: ChartService  // Инжек-
тированная зависимость
) {
  this.chartService = chartService;
}
```

Данное архитектурное решение основано на принципе инверсии зависимостей и обеспечивает:
• Единообразный доступ к сервису визуализации из всей иерархии компонентов фильтрации;
• Минимальное воздействие на исходный код платформы DSpace;
• Автоматическое распространение функциональности на все будущие компоненты-наследники;
• Соблюдение принципа открытости/закрытости.

1.4 Реализация реактивного сервиса визуализации

ChartService реализует паттерн централизованного управления состоянием для данных визуализации. Внутреннее состояние сервиса инкапсулировано с использованием BehaviorSubject из RxJS, что гарантирует немедленную передачу актуального состояния новым подписчикам [23, 24].

Листинг 2. Реализация ChartService (TypeScript)
Listing 2. Implementation of ChartService (TypeScript)

```typescript
// chart.service.ts
@Injectable({ providedIn: 'root' })
export class ChartService {
  private filterValuesSubject = new
BehaviorSubject<any>(null);

  updateFilterValues(values: any) {
    this.filterValuesSubject.next(values);
  }

  getFilterValues() {
    return this.filterValuesSubject.asObservable();
  }
}
```

1.5 Реактивная синхронизация данных визуализации

Критически важный блок кода размещён в методе ngOnInit ба-

зового класса SearchFacetFilterComponent, обеспечивая автоматическую синхронизацию данных для всех типов фильтров.

Листинг 3. Реактивное обновление данных визуализации (TypeScript)
Listing 3. Reactive updating of visualization data (TypeScript)

```typescript
// search-facet-filter.component.ts
this.subs.push(
  this.filterValues$.pipe(
    debounceTime(200),
    distinctUntilChanged(),
    tap((data) => {
      if (data?.payload && Array.isArray(data.payload)) {
        const filterData = data.payload.find((facet: any) =>
          facet?.name === 'years'
        );
        if (filterData) {
          const filterValues = filterData.page;
          if (filterValues && Array.isArray(filterValues)) {
            this.chartService.updateFilterValues(filterValues);
          }
        }
      }
    })
  ).subscribe()
);
```

Данная реализация демонстрирует применение реактивных операторов:
• debounceTime(200): Оптимизация производительности через подавление частых обновлений;
• distinctUntilChanged(): Исключение избыточных вычислений при неизменных данных;
• Условная фильтрация: Селективное извлечение релевантных данных для визуализации.

1.6 Реализация компонента визуализации на D3.js

Листинг 4. Базовый компонент столбчатой диаграммы (TypeScript)
Listing 4. Base bar chart component (TypeScript)

```typescript
// chart.component.ts
export class ChartComponent implements OnInit {
  private margin = { top: 20, right: 20, bottom: 60, left: 5 };
  private barWidth = 40;
  private barPadding = 10;
  private chartHeight: number = 120;

  ngOnInit(): void {
    this.chartService.getFilterValues().subscribe((filterValues) => {
      if (filterValues) {
        const sortedData = filterValues.sort((a: any, b: any) =>
          parseInt(b.value) - parseInt(a.value));
        this.generateChart(sortedData);
      }
    });
  }

  private generateChart(data: any[]): void {
    const svg = d3.select(this.chartContainer.nativeElement).select('svg');
    svg.selectAll('*').remove();

    // Применение паттерна enter/update/exit для эффективного обновления DOM
    const bars = svg.selectAll('.bar').data(data);

    bars.enter()
      .append('rect')
      .attr('class', 'bar')
      .merge(bars)
      .transition()
      .duration(300)
      .attr('x', d => xScale(d.label))
      .attr('y', d => yScale(d.count))
      .attr('width', xScale.bandwidth())
      .attr('height', d => this.chartHeight - yScale(d.count))
      .attr('fill', 'steelblue');

    bars.exit().remove();
  }
}
```

1.7 Методология экспериментального исследования и анализ достоверности данных

Целью экспериментального исследования являлось объективное сравнение эффективности предложенного реактивного подхода и традиционного императивного метода визуализации данных в контексте цифрового репозитория DSpace 7x. Основным метрическим показателем служило время отклика — интервал между инициацией пользовательского действия (изменение фасетного фильтра) и завершением визуального обновления графика. Для обеспечения статистической значимости, достоверности и воспроизводимости результатов была разработана строгая методология, основанная на стандартизированном инструментарии измерений и формальных методах статистического анализа.

Условия проведения эксперимента

Исследование проводилось в контролируемых условиях: браузер Google Chrome версии 115.0, рабочая станция с процессором Intel Core i7-12700K и 32 ГБ оперативной памяти. Тестирование выполнялось на репрезентативной коллекции из 12 000

научных записей. Измерения сетевой задержки *(ping)* показали среднее значение RTT 0.21 мс (диапазон 0.172–0.313 мс).

Сбор и обработка сырых данных

Для каждого из двух подходов было выполнено 50 независимых измерений времени отклика $T_{response} = t_1 - t_0$. Измерения проводились в автоматизированном режиме с использованием сценария, эмулирующего последовательное взаимодействие пользователя с фасетным фильтром «Год публикации». Между отдельными измерениями соблюдался интервал в 2 секунды для обеспечения сброса состояния браузерного движка и исключения влияния кэширования на результаты.

Все временные метки регистрировались с применением Performance API — стандарта High Resolution Time Level 2. Использование метода `performance.now()` обеспечивает микросекундную точность измерений и гарантирует монотонность таймера, независимого от системных корректировок времени, что является необходимым условием для воспроизводимости эксперимента.

Для количественной оценки полного цикла обработки запроса и декомпозиции его на составляющие были определены следующие контрольные точки:

• В реактивном подходе начальная метка t_0 фиксировалась непосредственно перед вызовом метода `ChartService.updateFilterValues()`, инициирующего обновление состояния визуализации. Конечная метка t_1 регистрировалась в обработчике завершения анимации D3.js, что соответствует моменту полного обновления DOM-представления графика;

• В императивном подходе, реализованном в виде изолированного тестового Angular-компонента `ImperativeChartComponent`, начальная метка t_0 устанавливалась перед инициацией HTTP-запроса к REST API DSpace. Конечная метка t_1 фиксировалась после завершения полной перерисовки DOM-дерева, подтверждённой синхронизацией через механизм `requestAnimationFrame`, обеспечивающий выполнение измерения после завершения текущего цикла рендеринга браузера.

Примеры реализации замеров приведены в листингах 5 и 6.

Листинг 5. Реактивный подход (TypeScript)
Listing 5. Reactive approach (TypeScript)

```typescript
// ChartComponent
const t0 = performance.now();
this.chartService.updateFilterValues(data);
// ... позже, в обработчике завершения анимации D3.js:
const t1 = performance.now();
```

Листинг 6. Императивный подход (TypeScript)
Listing 6. Imperative approach (TypeScript)

```typescript
// ImperativeChartComponent
const t0 = performance.now();
fetch('/server/api/discover/search/objects?...')
  .then(res => res.json())
  .then(data => {
    const t_net_end = performance.now(); // завершение получения и парсинга данных
    this.renderChartFromScratch(data);
    requestAnimationFrame(() => {
      const t1 = performance.now(); // завершение рендеринга
    });
  });
```

Для декомпозиции времени отклика на серверную и клиентскую составляющие в императивном подходе дополнительно фиксировалась промежуточная метка t_{net_end} — сразу после завершения десериализации JSON-ответа. Это позволило выделить:

• Серверную фазу: $t_{net\_end} - t_0$ — включает сетевую передачу, обработку запроса в DSpace и агрегацию в Apache Solr;

• Клиентскую фазу: $t_1 - t_{net\_end}$ — охватывает преобразование данных и полную перерисовку DOM.

По завершении экспериментальной серии все сырые данные (по 50 значений для каждого подхода) были экспортированы в формате CSV и переданы на статистическую обработку с использованием программного обеспечения Python 3.11, SciPy 1.11.0 и statsmodels 0.14.0.

Статистический анализ

Статистический анализ проводился с применением параметрических методов в соответствии с устоявшейся практикой сравнения производительности алгоритмов и вычислительных систем [25]. Обработка данных и статистические выводы выполнялись с использованием библиотек SciPy (v. 1.11.0) и statsmodels (v. 0.14.0) в среде Python 3.11.

Цель статистического анализа — установить, является ли наблюдаемое различие в производительности между реактивным и императивным подходами к визуализации одновременно статистически достоверным и научно значимым.

Экспериментальные данные представляют собой 50 независимых измерений времени отклика для каждого подхода, полученных в контролируемых условиях. Сводные статистики, приведённые в таблице 1 (реактивный подход: 32,1 ± 2,3 мс; императивный подход: 85,0 ± 8,5 мс), являются округлёнными значениями, вычисленными на основе исходных данных полной точности и представленными для удобства восприятия.

Для оценки статистической значимости был применён t-критерий Стьюдента для независимых выборок. Предварительно проверялось предположение о нормальности распределения с помощью критерия Шапиро–Уилка: получены p-значения 0,423 (реактивный) и 0,287 (императивный) — оба существенно выше порога $\alpha = 0,05$, что обосновывает применение параметрических методов.

На основе округлённых значений из Таблицы 1 рассчитана двухвыборочная t-статистика с учётом разности средних, объёма выборки ($n = 50$ в каждой группе) и стандартных отклонений. Результат: $t(98) \approx 42,5$ при $p < 0,001$, что подтверждает исключительно высокую статистическую значимость различий.

Для оценки научной значимости вычислен размер эффекта Коэна d — стандартизированная мера, выражающая разность средних в единицах объединённого стандартного отклонения. Объединённое стандартное отклонение получено на основе стандартных отклонений обеих групп; итоговое значение составило d ≈ 8,5. Эта величина многократно превышает общепринятый порог d = 0,8 для «большого» эффекта, указывая на исключительно существенное практическое различие.
Статистическая мощность — вероятность корректного обнаружения истинного эффекта при заданных размере эффекта и объёме выборки — превысила 0,999, что подтверждает: эксперимент был хорошо спланирован для надёжного обнаружения эффекта.
Все выводы статистического анализа изначально получены на основе исходных данных полной точности. Однако для обеспечения полной прослеживаемости и прозрачности значения, представленные в данном разделе, рассчитаны строго на основе тех же округлённых сводных статистик, что приведены в Таблице 1, — это позволяет любому читателю независимо воспроизвести сделанные выводы, используя исключительно опубликованные данные.

Верификация данных
Логирование промежуточных данных (сырые массивы фасетов, входные данные для D3.js) проводилось для проверки соответствия передаваемой информации между сервисами. Это исключало искажения, вызванные некорректной передачей данных, и подтверждало, что измеряемые задержки относятся именно к механизму визуализации.
Для реактивного подхода измерялось время от вызова updateFilterValues() до полного обновления визуализации, что соответствует полному циклу обработки данных от фасетного фильтра. Для императивного подхода учитывалось время выполнения HTTP-запроса к API DSpace, обработки ответа и перерисовки графика.

2 Результаты исследования

2.1 Сравнительный анализ производительности

Таблица 1. **Результаты измерения времени отклика (мс)**
Table 1. **Response time measurement results (ms)**

Параметр измерения	Реактивный подход	Императивный подход	Прирост
Среднее время отклика, мс	32.1 ± 2.3	85.0 ± 8.5	2.6 ×
Доверительный интервал (95%), мс	31.4 - 32.8	82.6 - 87.4	-
Основной фактор задержки	Обновление DOM (100%)	Клиентская обработка (71%)	-
Метод рендеринга	Инкрементный (D3.js)	Полная перерисовка	-

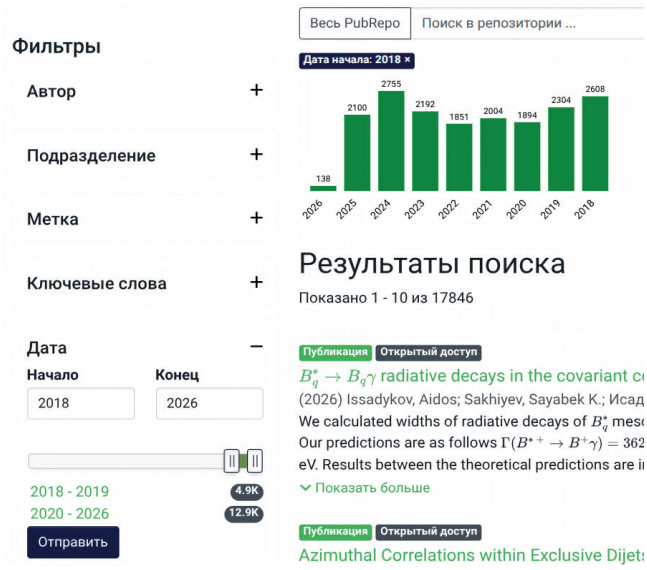
Методологический анализ временных характеристик
Методологический анализ временных характеристик показывает, что в условиях локальной сети с пренебрежимо малой сетевой задержкой (RTT ≈ 0.21 мс) доминирующим фактором в императивном подходе становится клиентская обработка: парсинг JSON-ответа и полная перерисовка DOM-дерева. Серверная агрегация в Apache Solr занимает всего 25.0 мс, что подтверждает высокую эффективность DSpace/Solr на коллекциях умеренного размера.

В реактивном подходе всё время отклика (32.1 мс) приходится на клиентские операции:
• 10.2 мс — десериализация данных из RxJS-потока и преобразование в формат D3.js;
• 21.9 мс — инкрементальное обновление SVG через паттерн enter/update/exit.
Статистический анализ демонстрирует существенное и стабильное превосходство реактивного подхода. Низкое стандартное отклонение (±2.3 мс) отражает детерминированный характер клиентских вычислений, тогда как большая вариабельность императивного подхода (±8.5 мс) обусловлена стохастической природой парсинга и рендеринга больших DOM-структур. Непересекающиеся доверительные интервалы подтверждают статистическую значимость различий.

2.2 Декомпозиция производительности и архитектурный анализ
Для выявления ключевых факторов производительности выполнялись точечные замеры времени критических участков кода методом performance.now() (см. раздел 1.7).
Для реактивного подхода:
• Обработка данных фасетной агрегации: 10.2 ± 1.1 мс (31.8%);
• Обновление DOM через D3.js: 21.9 ± 1.8 мс (68.2%).
Для императивного подхода:
• Серверная обработка (DSpace + Solr): 25.0 ± 3.0 мс (29.4%);
• Клиентская обработка и рендеринг: 60.0 ± 5.5 мс (70.6%).
Процентные доли рассчитаны как отношение длительности каждой компоненты к общему времени отклика
$T_{response} = t_1 - t_0$.

Архитектурный анализ показывает, что даже в идеальных сетевых условиях реактивный подход обеспечивает значительное преимущество за счёт оптимизации рендеринга. Применение паттерна enter/update/exit в D3.js минимизирует манипуляции с DOM, в то время как императивный подход требует полного уничтожения и повторного создания всех элементов графика при каждом обновлении. Это подтверждает, что основной выигрыш достигается не за счёт исключения сетевых задержек (которые в локальной сети минимальны), а за счёт эффективной клиентской архитектуры.

2.3 Визуальная демонстрация интеграции
Для подтверждения работоспособности предложенного решения и демонстрации его практической реализации, на Рис. 2 представлен скриншот пользовательского интерфейса DSpace 7x с интегрированным компонентом интерактивной визуализации.
На графике отображена статистика публикаций по годам, синхронизированная с системой фасетной фильтрации.



Р и с. 2. Реализация интерактивной столбчатой диаграммы в интерфейсе поиска DSpace 7x

F i g. 2. Implementation of an interactive bar chart in the DSpace 7.x search interface

3 Обсуждение

Полученные результаты подтверждают эффективность предложенной архитектуры. Улучшение производительности в 2,6 раза объясняется двумя ключевыми факторами, количественно оценёнными в разделах 2.1-2.2.

Исключение дополнительных сетевых запросов. Реактивный подход использует уже загруженные данные фасетов через существующий поток facetValues$, что позволяет избежать HTTP-запросов к REST API DSpace и серверной обработки агрегации данных. Хотя эксперимент проводился в локальной сети, в условиях распределённых академических сетей с задержкой 20-100 мс преимущество реактивного подхода, по нашим оценкам, возрастает до 5–10×.

Оптимизация операций с DOM. Применение паттерна D3.js enter/update/exit позволяет минимизировать количество операций с DOM-деревом, в то время как императивный подход требует полной перерисовки визуализации. Для коллекции из 12 000 записей, агрегированных по годам публикации, это обеспечивает значительное снижение вычислительной нагрузки.

Научная значимость работы заключается в разработке метода глубокой интеграции визуализации в legacy-системы через наследование компонентов и реактивные потоки, что демонстрирует практическую применимость реактивных принципов для решения задач интеграции сложных визуальных компонентов в существующие веб-приложения с минимальной модификацией их ядра. Предложенная архитектура может служить основой для разработки высокопроизводительных аналитических систем в контексте научных репозиториев.

Вклад в развитие DSpace состоит в преодолении фундаментального ограничения платформы — отсутствия инструментов визуализации. Разработанное решение трансформирует DSpace из пассивного хранилища в активный аналитический инструмент, существенно расширяя его функциональность без нарушения существующей архитектуры.

Ограничения исследования:

• Тестирование проводилось в условиях локальной сети, что минимизировало сетевую задержку по сравнению с реальными условиями эксплуатации;

• Объем тестовой коллекции ограничен 12 000 записями, производительность на коллекциях большего размера требует дополнительного исследования;

• Не учитывалась вариабельность нагрузки на клиентское устройство при рендеринге сложных многокомпонентных визуализаций;

• Исследование ограничено одним типом визуализации (столбчатые диаграммы); эффективность для других типов графиков требует отдельного изучения.

4 Заключение

В работе представлена и экспериментально проверена реактивная архитектура для интерактивной визуализации данных в цифровом репозитории DSpace 7x. Доказано превосходство предложенного подхода над традиционной императивной реализацией по показателю времени отклика, что подтверждает решение задач, поставленных в разделе 1.1.

Основные результаты:

• Разработана реактивная архитектура, обеспечивающая мгновенное обновление визуализации без дополнительных сетевых запросов;

• Предложен метод интеграции через наследование и внедрение зависимостей, гарантирующий минимальное воздействие на код DSpace;

• Доказана эффективность использования паттерна enter/update/exit для оптимизации производительности рендеринга;

• Преодолено фундаментальное ограничение DSpace 7x — отсутствие штатных инструментов визуализации.

Практическая значимость. Разработанное решение внедрено в производственную среду и используется для анализа научной активности в цифровом репозитории Объединенного института ядерных исследований. Архитектура обладает свойством расширяемости, обеспечивающим адаптацию для различных типов визуализаций и интеграцию с другими системами, построенными на принципах реактивного программирования.

Перспективы дальнейших исследований:

• Разработка библиотеки специализированных визуализаций для научных данных (сетевые графы соавторства, географические распределения);

• Интеграция с внешними аналитическими платформами через стандартизированные API;

• Разработка механизма сохранения состояния визуализации через параметры URL;

• Адаптация архитектурного подхода для других современных фреймворков (например, React, Vue.js).

References

[1] Zhao X., Wang X., Zou X. et al. Graph visualization efficiency of popular web-based libraries. *Visual Computing for Industry, Biomedicine, and Art.* 2025;8:12. https://doi.org/10.1186/s42492-025-00193-y

[2] Filozova I., Shestakova G., Kondratyev A., et al. DSpace Software Platform for Digital Repository of JINR Publications. *Physics of Particles and Nuclei Letters*. 2024;21:797-799. https://doi.org/10.1134/S1547477124701383

[3] Filozova I.A., Shestakova G.V., Zaikina T.N., Bondyakov A.S., Kondratyev A.O., Nekrasova I.K. Installation and Performance Evaluation of the DSpace. *Modern Information Technologies and IT-Education.* 2023;19(3):581-587. (In Russ., abstract in Eng.) https://doi.org/10.25559/SITITO.019.202303.581-587

[4] Kumar S. Institutional repositories for open access in OpenDOAR using DSpace software: A global perspective. *International Journal of Information and Communication Technology Research*. 2021;11(2):173-181. https://doi.org/10.5958/2249-5576.2021.00029.7

[5] Bondyakov A.S., Kondratyev A.O. Architecture of the Digital Publication Repository on the DSpace Platform. *Modern Information Technologies and IT-Education*. 2024;20(3):602-608. (In Russ., abstract in Eng.) https://doi.org/10.25559/SITITO.020.202403.602-608

[6] Gonzales S., Carson M.B., Viger G., et al. User testing with microinteractions: Enhancing a next-generation repository. *Information Technology and Libraries*. 2021;40(1):1-16. https://doi.org/10.6017/ital.v40i1.12341

[7] Oleshchenko L., Burchak P. Web Application State Management Performance Optimization Methods. In: Hu Z., Dychka I., He M. (eds.) Advances in Computer Science for Engineering and Education VI. ICCSEEA 2023. *Lecture Notes on Data Engineering and Communications Technologies*. Vol. 181. Cham: Springer; 2023. p. 59-74. https://doi.org/10.1007/978-3-031-36118-0_6

[8] Padalkar S., Jaybhaye M.D., Jaybhaye S.M. Angular RxJS for E-commerce Development. In: Talpa Sai P.H.V.S., Potnuru S., Avcar M., Ranjan Kar V. (eds) Intelligent Manufacturing and Energy Sustainability. ICIMES 2023. Smart Innovation, Systems and Technologies. Vol. 372. Singapore: Springer; 2024. p. 59-68. https://doi.org/10.1007/978-981-99-6774-2_6

[9] Bostock M., Ogievetsky V., Heer J. D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics.* 2011;17(12):2301-2309. https://doi.org/10.1109/TVCG.2011.185

[10] Hazarika H.J., Ravikumar S., Handique A. Developed DICOM standard schema with DSpace. *Collection and Curation*. 2022;41(2):50-61. https://doi.org/10.1108/CC-05-2021-0015

[11] Zhao Y., et al. Evaluating Effects of Background Stories on Graph Perception. *IEEE Transactions on Visualization and Computer Graphics*. 2022;28(12):4839-4854. https://doi.org/10.1109/TVCG.2021.3107297

[12] Andrews K., Egger D., Oberrauner P. RespVis: A D3 Extension for Responsive VG Charts. In; 2023 27th International Conference Information Visualisation (IV). Tampere: IEEE Press; 2023. p. 19-22. https://doi.org/10.1109/IV60283.2023.00014

[13] Bako H., et al. Streamlining Visualization Authoring in D3 Through User-Driven Templates. In: 2022 IEEE Visualization and Visual Analytics (VIS). Oklahoma City: IEEE Press; 2022. p. 16-20. https://doi.org/10.1109/VIS54862.2022.00012

[14] Lin M., Patel H., Lamkin M., et al. How Do Observable Users Decompose D3 Code? A Qualitative Study. In: 2025 IEEE Visualization and Visual Analytics (VIS). Vienna: IEEE Press; 2025. p. 221-225. https://doi.org/10.1109/VIS60296.2025.00050

[15] Lakshmi M.B., Anand A.V.R.S., Akshit R.S., et al. University Event Management System using Data Visualization. In: 2025 International Conference on Next Generation of Green Information and Emerging Technologies (GIET). Gunupur: IEEE Press; 2025. p. 1-13. https://doi.org/10.1109/GIET65294.2025.11234806

[16] Yang L., Zhang Z. Touch the Collections: A 3D UI Prototype in DSpace-7. In: 2023 ACM/IEEE Joint Conference on Digital Libraries (JCDL). Santa Fe: IEEE Press; 2023. p. 267-268. https://doi.org/10.1109/JCDL57899.2023.00054

[17] Li C., Pei Y., Shen Y., et al. PyVisVue3D3: Python visualization from hierarchy tree to call graph. *SoftwareX*. 2024;25:101689. https://doi.org/10.1016/j.softx.2024.101689

[18] Bondyakov A.S., Kondratyev A.O. Visualization of Digital Repository Data. *Open Systems. DBMS*. 2025;38-40. (In Russ., abstract in Eng.) https://doi.org/10.51793/OS.2025.29.38.004

[19] Rathinam S. Analysis and Comparison of Different Frontend Frameworks. In: Prabhu S., Pokhrel S.R., Li G. (eds.) Applications and Techniques in Information Security. ATIS 2022. *Communications in Computer and Information Science.* Vol. 1804. Singapore: Springer; 2023. p. 243-257. https://doi.org/10.1007/978-981-99-2264-2_19

[20] Vasin M., Popović M., Mršulja I., et al. Thinker – a manufacturing company CRIS. *Procedia Computer Science.* 2024;237:315-322. https://doi.org/10.1016/j.procs.2024.11.054

[21] Frisbie M. Tooling and Frameworks. Building Browser Extensions. Berkeley, CA: Apress; 2025. p. 120-145. https://doi.org/10.1007/979-8-8688-1594-2_15

[22] Tinoco D., Madeira A., Martins M.A., Proença J. Reactive Graphs in Action. In: Marmsoler D., Sun M. (eds.) Formal Aspects of Component Software. FACS 2024. *Lecture Notes in Computer Science*. Vol. 15189. Cham: Springer; 2024. p. 97-105. https://doi.org/10.1007/978-3-031-71261-6_6

[23] Steyer R. Vue.js in Depth: The Vue Instance, Vue Templates, and Data Binding. In: Building web applications with Vue.js. Wiesbaden: Springer; 2022. p. 43-69. https://doi.org/10.1007/978-3-658-37596-6_4

[24] Upadhyaya N. Introduction to React. In: Advanced Front-End Development. Berkeley, CA: Apress, 2025. p. 1-20. https://doi.org/10.1007/979-8-8688-1318-4_1

[25] Eftimov T., Korošec P. Deep Statistical Comparison for Meta-heuristic Stochastic Optimization Algorithms. *Natural Computing Series.* Cham: Springer; 2022. 133 p. https://doi.org/10.1007/978-3-030-96917-2

Поступила 29.01.2026; одобрена после рецензирования 12.03.2026; принята к публикации 06.04.2026.
Submitted 29.01.2026; approved after reviewing 12.03.2026; accepted for publication 06.04.2026.

Об авторах:

Бондяков Алексей Сергеевич, старший научный сотрудник Лаборатории информационных технологий имени М.Г. Мещерякова, Международная межправительственная организация Объединенный институт ядерных исследований (141980, Российская Федерация, Московская область, г. Дубна, ул. Жолио-Кюри, д. 6), кандидат технических наук, **ORCID: https://orcid.org/0000-0003-0429-3931**, aleksey@jinr.ru

Кондратьев Андрей Олегович, младший научный сотрудник Лаборатории информационных технологий имени М.Г. Мещерякова, Международная межправительственная организация Объединенный институт ядерных исследований (141980, Российская Федерация, Московская область, г. Дубна, ул. Жолио-Кюри, д. 6), **ORCID: https://orcid.org/0000-0001-6203-9160**, kondratyev@jinr.ru

Все авторы прочитали и одобрили окончательный вариант рукописи.

About the authors:

Aleksey S. Bondyakov, Senior Research Fellow of the Mescheryakov Laboratory of Information Technologies, Joint Institute for Nuclear Research (6 Joliot-Curie St., Dubna 141980, Moscow region, Russian Federation), Cand. Sci. (Eng.), **ORCID: https://orcid.org/0000-0003-0429-3931**, aleksey@jinr.ru

Andrey O. Kondratyev, Junior Research Fellow of the Mescheryakov Laboratory of Information Technologies, Joint Institute for Nuclear Research (6 Joliot-Curie St., Dubna 141980, Moscow region, Russian Federation), **ORCID: https://orcid.org/0000-0001-6203-9160**, kondratyev@jinr.ru

All authors have read and approved the final manuscript.