**Исследования и разработки в области новых информационных технологий и их приложений**

**https://doi.org/10.25559/SITITO.021.202502.251-259**
**УДК 004.056.53**

# Фронтенд цифрового репозитория публикаций на платформе DSpace

**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

Оригинальная статья

## Аннотация

Данная статья посвящена анализу архитектуры фронтенда на базе Angular для цифрового репозитория публикаций на платформе DSpace. Рассматриваются ключевые принципы построения современного веб-интерфейса, обеспечивающего высокую производительность, масштабируемость и интуитивную навигацию. Авторы подробно описывают модульную структуру приложения, компонентный подход, двунаправленное связывание данных и использование реактивного программирования с RxJS, что позволяет эффективно управлять асинхронными операциями и потоками данных. Особое внимание уделено интеграции Angular с DSpace REST API, ленивой загрузке модулей, организации сервисов, маршрутизации и работе с формами. В работе представлены фрагменты кода, иллюстрирующие реализацию поиска и отображения публикаций, а также блок-схема архитектуры фронтенда, наглядно демонстрирующая взаимодействие ключевых элементов. Проведено тестирование производительности в облачной инфраструктуре ОИЯИ с использованием метрик Core Web Vitals, что подтверждает высокое качество пользовательского опыта. Также выполнено сравнение Angular с React и Vue.js, обосновывающее выбор Angular как официального фронтенда DSpace. Рассмотрены ограничения и перспективы развития, включая переход к Progressive Web App, внедрение серверного рендеринга с Angular Universal и интеграцию с искусственным интеллектом для улучшения поиска. Статья предназначена для разработчиков и специалистов по цифровым репозиториям, стремящихся к оптимизации архитектуры и повышению эффективности научных платформ. Анализ архитектуры позволяет глубже понять принципы построения сложных веб-приложений, обеспечить устойчивость системы и упростить поддержку. Особое значение имеет использование RxJS для обработки асинхронных запросов, что критично при работе с большими объемами данных.

**Ключевые слова:** цифровой репозиторий, архитектура Angular, фронтенд, веб-приложение

**Конфликт интересов:** авторы заявляют об отсутствии конфликта интересов.

**Для цитирования:** Бондяков А. С., Кондратьев А. О. Фронтенд цифрового репозитория публикаций на платформе DSpace // Современные информационные технологии и ИТ-образование. 2025. Т. 21, № 2. С. 251-259. https://doi.org/10.25559/SITITO.021.202502.251-259

**Research and Development in the Field of New IT and Their Applications**

# Frontend of a Digital Publication Repository on the DSpace Platform

**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

Original article

## Abstract

This article presents an analysis of the Angular-based frontend architecture for a digital publication repository on the DSpace platform. The study focuses on the key principles of building a modern web interface that ensures high performance, scalability, and intuitive navigation. The authors provide a detailed description of the application's modular structure, component-based approach, two-way data binding, and the use of reactive programming with RxJS, which together enable efficient management of asynchronous operations and data streams. Particular attention is given to the integration of Angular with the DSpace REST API, lazy loading of modules, organization of services, routing, and form handling. The paper includes code snippets illustrating the implementation of publication search and display functionality, as well as a block diagram of the frontend architecture that clearly demonstrates the interaction of core components. Performance testing was conducted within the cloud infrastructure of the JINR using Core Web Vitals metrics, confirming a high level of user experience quality. A comparative analysis of Angular with React and Vue.js is also provided, justifying the selection of Angular as the official frontend for DSpace. The article discusses current limitations and future development prospects, including the transition to Progressive Web Apps, implementation of server-side rendering via Angular Universal, and integration of artificial intelligence to enhance search capabilities. The paper is intended for developers and specialists working with digital repositories who aim to optimize architecture and improve the efficiency of scientific platforms. The architectural analysis facilitates a deeper understanding of complex web application design, enhances system stability, and simplifies maintenance. Special emphasis is placed on the use of RxJS for handling asynchronous requests, which is critical when working with large volumes of data.

**Keywords:** digital repository, Angular architecture, frontend, web application

**Conflict of interests:** The authors declares no conflict of interest.

**For citation:** Bondyakov A.S., Kondratyev A.O. Frontend of a Digital Publication Repository on the DSpace Platform. *Modern Information Technologies and IT-Education*. 2025;21(2):251-259. https://doi.org/10.25559/SITITO.021.202502.251-259

## Введение

С принятием Angular в качестве официального фронтенда с версии 7 DSpace перешёл к современной архитектуре одностраничного приложения. Этот шаг стал стратегическим решением международной команды разработчиков, направленным на повышение производительности, улучшение пользовательского опыта и обеспечение современной, масштабируемой архитектуры.

DSpace – это платформа для создания цифровых репозиториев, широко используемая университетами, научными учреждениями и архивами для управления и хранения научных публикаций. Внедрение Angular позволило значительно повысить функциональность интерфейса за счёт динамической загрузки данных, интуитивной навигации и эффективной обработки больших объёмов информации.

Angular построен на TypeScript – языке со статической типизацией, разработанном Microsoft. Это обеспечивает надёжность и поддерживаемость кода в крупных проектах. Перед выполнением в браузере код компилируется в стандартный JavaScript, гарантируя кросс-браузерную совместимость. Фреймворк поддерживает модульность, повторное использование компонентов и двухстороннее связывание данных, что делает его особенно подходящим для сложных приложений.

Хотя в веб-разработке популярны и другие фреймворки, такие как React и Vue.js, выбор Angular для DSpace обусловлен его зрелой экосистемой и строгой архитектурой. В отличие от React, требующего сторонних решений для маршрутизации и управления состоянием, и Vue.js, ориентированного на гибкость и быструю разработку, Angular предоставляет ключевые инструменты «из коробки»: встроенную маршрутизацию, управление формами, HTTP-клиент и механизм инъекции зависимостей. Особую роль играет интеграция с RxJS, позволяющая эффективно управлять асинхронными потоками данных от DSpace REST API [1-10].

Сравнение Angular, React и Vue.js в контексте DSpace представлено в Таблице 1.

Как видно из анализа, Angular превосходит альтернативы по степени интеграции с платформой, уровню структурирования кода и поддержке в научной среде. Официальная поддержка Google, наличие долгосрочных версий (LTS) и строгая типизация делают Angular особенно привлекательным для академических систем, таких как DSpace, где важны стабильность, безопасность и долгосрочная поддержка.

Таблица 1. **Сравнение Angular, React и Vue.js в контексте платформы DSpace**
Table 1. **Comparison of Angular, React, and Vue.js in the context of the DSpace platform**

| Функциональная характеристика | Angular | React | Vue.js |
|---|---|---|---|
| Официальная поддержка DSpace | Да (с версии 7 как стандартный фронтенд) | Нет | Нет |
| Язык программирования | TypeScript (обязательно) | JavaScript/TypeScript (опционально) | JavaScript/TypeScript (опционально) |
| Встроенная маршрутизация | Да (RouterModule) | Нет (требуется React Router) | Да (Vue Router) |
| Управление формами | Да (Reactive и Template-driven формы) | Нет (через библиотеки: Formik, Hook Form) | Да (встроенные директивы v-model) |
| Механизм инъекции зависимостей | Да (встроенный) | Нет (через контекст или сторонние решения) | Ограниченный (через provide/inject) |
| Работа с асинхронными данными | RxJS (встроен) | **Hooks** (useEffect, useReducer), библиотеки (react-query) | **Hooks** (Composition API), библиотеки (Pinia, Vue Query) |
| HTTP-клиент | Встроенный (HttpClient) | Нет (fetch, axios, сторонние) | Нет (fetch, axios) |
| Система сборки и CLI | Встроенный Angular CLI | Через Create React App или Vite | Через Vue CLI или Vite |
| Поддержка PWA и SSR | Да (Angular Universal, PWA-пакеты) | Да (Next.js для SSR, ручная настройка PWA) | Да (Nuxt.js для SSR, Vite PWA) |
| Степень структурирования | Высокая (строгая архитектура) | Средняя (гибкость, требует дисциплины) | Средняя (гибкость с возможностью структурирования) |
| Документация и поддержка в научной среде | Официальная поддержка DSpace, Google, LTS-версии | Широкое комьюнити, но нет интеграции с DSpace | Хорошая документация, но не используется в DSpace |

*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.

## Цель исследования

Цель исследования данной статьи – анализ архитектуры фронтенда на базе Angular для цифровых репозиториев на платформе DSpace, с акцентом на структуру компонентов и их взаимодействие. Изучение данной архитектуры позволит разработчикам глубже понять принципы работы фронтенда, интегрированного с DSpace, и создавать более эффективные и инновационные приложения для обеспечения удобного взаимодействия с научным контентом.

## Архитектура фронтенда DSpace на базе Angular

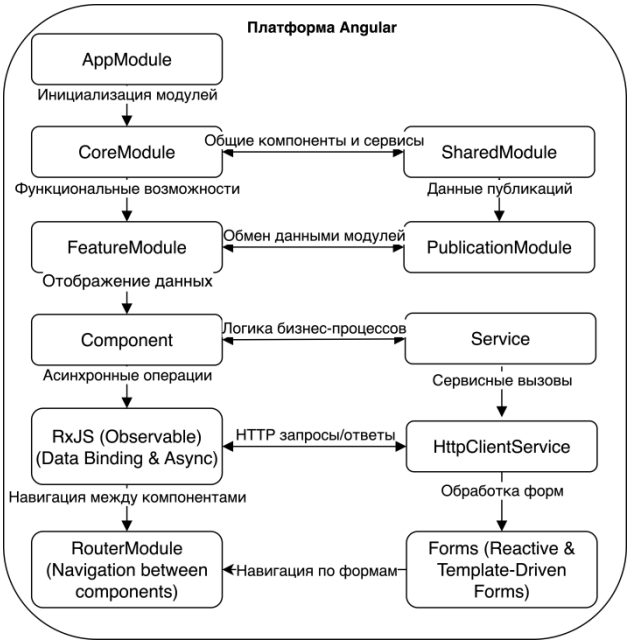
Одной из ключевых особенностей Angular является модульная структура, позволяющая разделять приложение на независимые компоненты и модули. Это упрощает расширение функциональности и поддержку кода, особенно в крупных репозиториях с большим объёмом данных и сложной структурой интерфейса. Двустороннее связывание данных *(two-way data binding)* автоматически синхронизирует изменения между пользовательским интерфейсом и моделью данных: например, при вводе поискового запроса или изменении фильтров данные мгновенно отображаются в интерфейсе, а действия пользователя – передаются в логику приложения. Это значительно упрощает работу с динамическим контентом, таким как научные публикации, и улучшает взаимодействие с пользователем.

Ещё одной важной особенностью является реактивное программирование на основе библиотеки RxJS *(Reactive Extensions for JavaScript)*, которая позволяет эффективно управлять асинхронными потоками данных – такими как HTTP-запросы, события интерфейса или действия пользователя. RxJS использует объекты типа Observable («наблюдаемые»), которые позволяют отслеживать поступление данных во времени, фильтровать, комбинировать и отменять операции. Это особенно важно при реализации сложных сценариев, таких как параллельный поиск публикаций, загрузка метаданных или синхронизация состояния. Благодаря RxJS интерфейс обновляется в реальном времени без перезагрузки страницы, что критично для приложений с высокой нагрузкой, а код остаётся декларативным и легко поддерживаемым [11-19].

Angular также поддерживает механизм ленивой загрузки *(lazy loading)*, при котором модули и компоненты загружаются только при необходимости – например, при переходе пользователя на соответствующую страницу. Это снижает объём данных, передаваемых при первоначальной загрузке приложения, ускоряет запуск и оптимизирует использование ресурсов, что особенно важно для крупных репозиториев с обширной функциональностью.

Для взаимодействия с серверной частью Angular предоставляет встроенные инструменты работы с API *(Application Programming Interface)* – стандартизированным интерфейсом, определяющим правила обмена данными между клиентом и сервером. Вместо прямого доступа к внутренней логике системы, обмен происходит через запросы и ответы, что позволяет абстрагироваться от деталей реализации. В DSpace API используется для получения, обновления и удаления информации о публикациях, сообществах и метаданных. Это обеспечивает гибкость, безопасность и упрощает интеграцию с другими системами.

В архитектуре фронтенда DSpace ключевые элементы – модули, компоненты, сервисы, маршрутизация и реактивное программирование – организованы в единую, масштабируемую структуру. Их взаимодействие показано на Рис. 1.




Р и с.1. Архитектура взаимодействия ключевых элементов фронтенда DSpace

F i g. 1. The architecture of interaction between key elements of the DSpace front end

• AppModule – корневой модуль приложения, который инициализирует все необходимые зависимости. В нём импортируются другие ключевые модули, такие как CoreModule и SharedModule, настраивается маршрутизация (RouterModule) и взаимодействие с сервером (HttpClientModule). Все компоненты, сервисы и базовые настройки приложения регистрируются в этом модуле, что делает его центральной точкой входа и конфигурации всей системы.

• CoreModule – модуль, предназначенный для хранения глобальных сервисов, используемых на протяжении всей сессии пользователя. Например, сервис AuthService отвечает за управление авторизацией, а LoggingService – за сбор и обработку системных событий. Эти сервисы становятся доступны во всех частях приложения благодаря

механизму внедрения зависимостей *(Dependency Injection)*, который позволяет автоматически передавать нужные объекты в компоненты и другие сервисы без необходимости создавать их вручную.

• SharedModule содержит компоненты, директивы и пайпы, которые используются в нескольких модулях. К ним относятся, например, навигационная панель, модальные окна, кнопки или пагинация. Использование SharedModule предотвращает дублирование кода и обеспечивает единообразие интерфейса, упрощая повторное использование элементов в разных разделах приложения.

• FeatureModule – отдельные модули, отвечающие за конкретную функциональность. Например, PublicationModule может включать компоненты для отображения списка публикаций, страницы деталей, формы поиска и связанные сервисы. Такие модули могут быть настроены на ленивую загрузку, что позволяет загружать их только при переходе пользователя в соответствующий раздел, что повышает производительность.

• Component (компоненты) – основные строительные блоки пользовательского интерфейса. Каждый компонент управляет своим собственным представлением (HTML-шаблоном) и логикой (TypeScript-кодом). Компоненты взаимодействуют с сервисами для получения данных, например, через HTTP-запросы или подписки на изменения состояния, и отображают их в интерфейсе.

• Services (сервисы) – классы, инкапсулирующие бизнес-логику и работу с данными. Они используются для обмена информацией между компонентами и для взаимодействия с серверной частью системы. Сервисы часто используют библиотеку RxJS для обработки асинхронных операций. Например, HTTP-запросы выполняются через встроенный HttpClient, который возвращает объект типа Observable. Компоненты подписываются на этот поток данных и автоматически обновляют интерфейс, как только данные становятся доступны.

• RouterModule – модуль Angular, управляющий переходами между страницами и компонентами. В DSpace он используется для навигации между списками публикаций, страницами детальной информации, формами поиска и другими разделами. Маршруты могут быть настроены так, чтобы соответствующие модули загружались только при необходимости (ленивая загрузка), что существенно улучшает производительность, особенно в приложениях с большим количеством страниц.

• RxJS (Reactive Extensions for JavaScript) – библиотека для реактивного программирования, используемая в Angular для работы с асинхронными событиями. В DSpace она применяется для обработки данных, поступающих от сервера. Например, сервис PublicationService отправляет запрос к API через HttpClient, получает Observable и передаёт его компоненту PublicationListComponent. Тот подписывается на поток и обновляет интерфейс при получении данных. RxJS также позволяет обрабатывать ошибки, комбинировать несколько потоков, фильтровать и преобразовывать данные, что упрощает реализацию сложной логики.

• Forms (формы) – механизм Angular для работы с пользовательским вводом. Поддерживаются два подхода: Reactive Forms и Template-Driven Forms. В DSpace Reactive Forms используются для сложных форм, таких как поиск и фильтрация публикаций, где требуется гибкое управление валидацией, динамическое изменение полей и асинхронные проверки. Данные формы управляются программно с помощью объектов FormGroup и FormControl. Template-Driven Forms применяются для простых форм, таких как авторизация или регистрация, где основная логика описывается в HTML-шаблоне. Оба подхода интегрируются с RxJS, что позволяет отслеживать изменения в реальном времени и выполнять, например, валидацию на стороне сервера.

Взаимодействие всех элементов фронтенда DSpace строится на принципах модульности, внедрения зависимостей и асинхронного программирования. AppModule инициализирует приложение, подключая CoreModule, SharedModule и настраивая маршрутизацию и HTTP-клиент. CoreModule предоставляет глобальные сервисы, доступные через механизм внедрения зависимостей. SharedModule обеспечивает повторное использование UI-компонентов. FeatureModule реализует функциональность отдельных разделов. Компоненты получают данные от сервисов, которые используют RxJS и HttpClient для взаимодействия с сервером. Маршрутизация с поддержкой ленивой загрузки оптимизирует производительность. Формы обеспечивают сбор и валидацию пользовательского ввода. Все эти элементы, интегрированные в единую систему, образуют масштабируемую, производительную и поддерживаемую архитектуру, способную эффективно работать с большими объёмами научных данных. Для дальнейшего повышения эффективности в крупных репозиториях могут применяться методы кэширования, индексации и оптимизации запросов [20-25].

## Примеры реализации ключевых сценариев

Для иллюстрации практической реализации в DSpace приведём типичные фрагменты кода, демонстрирующие взаимодействие компонентов, сервисов и RxJS при работе с публикациями.

Пример 1. Получение публикаций через HttpClient и RxJS
Example 1. Retrieving publications using HttpClient and RxJS

```
// publication.service.ts
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable, of } from 'rxjs';
import { map, catchError } from 'rxjs/operators';
import { LoggingService } from './logging.service';
import { Publication } from './publication.model';

@Injectable({
```

```
  providedIn: 'root'
})
export class PublicationService {
  constructor(
    private http: HttpClient,
    private loggingService: LoggingService
  ) {}

  getPublications(query: string): Observable<Publication[]> {
    return
this.http.get<Publication[]>(`/search/findItems?query=${query}`)
      .pipe(
        // Проверка, что ответ – массив, иначе возвращаем пустой
массив
        map(response => Array.isArray(response) ? response : []),
        // Обработка ошибок: логируем и возвращаем пустой массив
        catchError(error => {
          this.loggingService.error('Ошибка при получении публикаций',
error);
          return of([]);
        })
      );
  }
}
```

Данный метод демонстрирует типичный сценарий взаимодействия фронтенда DSpace с REST API бэкенда. Использование HttpClient в сочетании с RxJS позволяет асинхронно получать данные о публикациях, обрабатывать возможные ошибки и преобразовывать ответ сервера в удобную для отображения структуру. Оператор map обеспечивает безопасность типов, проверяя, что ответ является массивом, что предотвращает возможные ошибки при отображении. Блок catchError обеспечивает отказоустойчивость, логируя ошибку и возвращая пустой массив, что предотвращает сбои в интерфейсе при недоступности сервера.

*Примечание о модели Publication:* Интерфейс Publication определён в файле publication.model.ts и описывает структуру данных публикации, соответствующую формату ответа DSpace 7 REST API. Он включает поля id, title, authors, publicationDate, doi, handle и другие, что обеспечивает строгую типизацию и поддерживаемость кода.

Пример 2. Компонент, использующий сервис для отображения списка публикаций
Example 2. A component that uses a service to display a list of publications

```
// publication-list.component.ts
import { Component, OnInit, OnDestroy } from '@angular/core';
import { FormControl } from '@angular/forms';
import { PublicationService } from './publication.service';
import { Publication } from './publication.model';
import { Observable, Subject } from 'rxjs';
import { map, takeUntil, debounceTime, distinctUntilChanged,
switchMap } from 'rxjs/operators';

@Component({
  selector: 'app-publication-list',
  template: `
    <input [formControl]="searchControl" placeholder="Поиск
публикаций...">
    <ul>
      <li *ngFor="let pub of publications">
        {{ pub.title }} ({{ pub.publicationDate }})
      </li>
    </ul>
  `
})
export class PublicationListComponent implements OnInit, OnDestroy {
  searchControl = new FormControl('');
  publications: Publication[] = [];
  private destroy$ = new Subject<void>();

  constructor(private publicationService: PublicationService) {}

  ngOnInit() {
    this.searchControl.valueChanges
      .pipe(
        debounceTime(300),          // Задержка запроса на 300 мс
        distinctUntilChanged(),     // Игнорировать повторный ввод
        switchMap(query =>
this.publicationService.getPublications(query)),
        takeUntil(this.destroy$)    // Автоматическая отписка при
уничтожении
      )
      .subscribe(results => this.publications = results);
  }

  ngOnDestroy() {
    this.destroy$.next();
    this.destroy$.complete();
  }
}
```

Этот пример демонстрирует полный жизненный цикл сценария: от инициализации компонента до подписки на изменения поля поиска и обновления интерфейса. Компонент использует Reactive Forms для отслеживания ввода пользователя. При каждом изменении значения в поле поиска запускается цепочка операторов RxJS:

- *debounceTime(300)* – предотвращает отправку запроса при каждом символе, ожидая паузу в 300 мс;
- *distinctUntilChanged()* – отсеивает дубликаты (например, "a" → "ab" → "a");
- *switchMap()* – отменяет предыдущий запрос, если новый уже начался, избегая "гонки" запросов;
- *takeUntil(this.destroy$)* – обеспечивает автоматическую отписку при уничтожении компонента, предотвращая утечки памяти.

Это обеспечивает плавный пользовательский опыт и снижает нагрузку на сервер, что особенно важно при работе с большими объёмами данных.

## Анализ результатов тестирования производительности фронтенда DSpace

Оценка производительности фронтенда DSpace на базе Angular проводилась в облачной инфраструктуре Объединённого института ядерных исследований (ОИЯИ). Система была развернута на виртуальной машине с операционной системой Linux, 10 ядрами CPU, 32 ГБ ОЗУ и 200 ГБ SSD-диска. Фронтенд и бэкенд функционировали в изолированных контейнерах Docker. Каждое измерение повторялось 5 раз, после чего результаты усреднялись для обеспечения статистической достоверности.

Тестовые данные:

- Количество публикаций в репозитории: 12 743
- Общий объём метаданных: ~1.8 ГБ
- Структура данных: публикации включают полные

тексты (PDF), аннотации, авторов, DOI, ключевые слова.
- Средний размер метаданных на публикацию: ~145 КБ
- Нагрузка на поиск: имитировался параллельный поиск по 50 пользователям с использованием разнообразных запросов (по автору, году, ключевому слову, DOI).

Для оценки производительности были выбраны ключевые метрики, рекомендованные консорциумом W3C и Google под общим названием Core Web Vitals. Эти метрики являются общепринятым стандартом для оценки пользовательского опыта в веб-приложениях, поскольку они напрямую отражают субъективное восприятие скорости, отзывчивости и стабильности интерфейса конечным пользователем. Использование этих стандартных метрик позволяет объективно сравнить производительность системы с рекомендованными пороговыми значениями и обеспечивает прозрачность и воспроизводимость результатов.
- TTFB (Time to First Byte) – время до первого байта – измеряет задержку сети и скорость ответа сервера. Она показывает, как быстро сервер начинает передавать данные после запроса пользователя. Низкое значение TTFB критично для восприятия общей скорости системы.
- FCP (First Contentful Paint) – первое отображение контента – фиксирует момент, когда на экране появляется первый элемент, такой как текст или изображение. Она важна для формирования первого впечатления о скорости загрузки.
- LCP (Largest Contentful Paint) – время отображения самого крупного элемента – является ключевым показателем загрузки. Это время, за которое на экране появляется основной контент страницы (например, заголовок или список публикаций). Пользователь считает страницу загруженной именно в этот момент, поэтому LCP напрямую влияет на удовлетворенность от использования.
- CLS (Cumulative Layout Shift) – совокупный сдвиг макета – измеряет стабильность интерфейса. Она оценивает, насколько элементы страницы сдвигаются по экрану во время загрузки (например, когда текст внезапно уезжает вниз из-за догрузки изображения). Высокий CLS приводит к ошибочным кликам и ухудшает пользовательский опыт, поэтому его минимизация критична для удобства использования.

Измерения проводились с помощью инструмента Google Lighthouse, запущенного в автоматизированном режиме с эмуляцией медленного интернет-соединения и ограниченной вычислительной мощности, что имитирует работу типичного пользователя. Дополнительно, количество сетевых запросов подсчитывалось вручную с помощью Chrome DevTools.

Полученные результаты демонстрируют высокую производительность системы. Время TTFB составило 315 мс, что указывает на эффективную работу серверной части. FCP зафиксировано на уровне 1.1 секунды, а LCP – на уровне 1.85 секунды. Оба показателя находятся в пределах, рекомендованных для обеспечения хорошего пользовательского опыта. Показатель CLS составил 0.08, что свидетельствует о высокой стабильности интерфейса. Общее количество сетевых запросов при инициализации приложения – 48, что является приемлемым значением.

Проведённое тестирование подтверждает, что фронтенд Angular в DSpace, обеспечивает высокую производительность и стабильность в условиях реальной облачной инфраструктуры. Полученные метрики соответствуют современным стандартам веб-разработки и подтверждают эффективность выбранной технологической платформы для построения масштабируемых цифровых репозиториев.

## Полученные результаты

Проведенный анализ архитектуры фронтенда DSpace на базе Angular подтвердил её высокую производительность и эффективность. Тестирование, выполненное в облачной инфраструктуре ОИЯИ, показало, что система демонстрирует хорошие показатели времени отклика, стабильности интерфейса и скорости загрузки контента, что соответствует современным стандартам веб-производительности.

Для разработки эффективных решений в рамках фронтенда DSpace представлен Рис.1, который наглядно демонстрирует структуру взаимодействия ключевых компонентов системы. Он помогает разработчикам понять распределение функциональности между модулями, использование RxJS для асинхронных операций, сервисов и компонентов для работы с данными, а также организацию маршрутизации и форм. Указанный рисунок способствует упрощению разработки и улучшению понимания архитектуры фронтенда.

## Ограничения и перспективы развития

Несмотря на высокую производительность, использование Angular в DSpace имеет свои сложности. Для эффективной разработки требуется глубокое знание TypeScript и библиотеки RxJS, что повышает порог входа для новых разработчиков. Архитектура Angular также может приводить к большему размеру загружаемого кода по сравнению с более легкими фреймворками, а управление асинхронными потоками данных требует опыта для отладки.

В то же время, существуют перспективные направления для развития системы. К ним относятся интеграция технологий искусственного интеллекта для улучшения поиска, переход к Progressive Web App (PWA) для лучшей работы на мобильных устройствах, внедрение серверного рендеринга (Angular Universal) для улучшения индексации в поисковых системах, а

также переход к микрофронтенд-архитектуре для независимой разработки и развёртывания отдельных модулей. Эти направления могут стать основой для дальнейших исследований и усовершенствования цифровых репозиториев.

## Заключение

Исследование, проведенное в статье, охватывает ключевые архитектурные принципы Angular, включая модульную организацию приложения, компонентный подход, двунаправленное связывание данных, использование реактивного программирования с RxJS, а также механизмы ленивой загрузки, маршрутизации и работы с формами. Особое внимание уделено роли RxJS в управлении асинхронными потоками данных, что критично при работе с DSpace REST API – например, при реализации поиска, фильтрации и кэширования. Приведённые примеры кода иллюстрируют, как на практике реализуются такие сценарии. Результаты тестирования производительности, проведённые в облачной инфраструктуре Объединённого института ядерных исследований, подтверждают эффективность архитектуры Angular в реальных условиях эксплуатации. Полученные метрики Core Web Vitals соответствуют рекомендованным пороговым значениям для высококачественных веб-приложений, что свидетельствует о высокой скорости отклика, стабильности интерфейса и хорошей отзывчивости системы.

Таким образом, архитектура Angular в DSpace представляет собой продуманное и эффективное решение для построения современного фронтенда цифрового репозитория. Её анализ позволяет разработчикам и исследователям глубже понять принципы организации сложных веб-приложений, а также использовать полученные данные для оптимизации, поддержки и развития подобных систем в научной и образовательной среде.

## References

1. 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
2. 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
3. Bondyakov A.S., Kondratyev A.O. Automated Collection and Systematization of Publications. *Open Systems. DBMS*. 2024;(1):40-42. (In Russ., abstract in Eng.) EDN: DSYPXY
4. Bondyakov A.S., Kondratyev A.O. Author Data Management Services. *Open Systems. DBMS*. 2025;(1):46-47. (In Russ., abstract in Eng.) https://doi.org/10.51793/OS.2025.52.14.002
5. Dobiasch M., Oppl S., Stöckl M., et al. Pegasos: a framework for the creation of direct mobile coaching feedback systems. *Journal on Multimodal User Interfaces*. 2024;18:1-19. https://doi.org/10.1007/s12193-023-00411-y
6. Okon R., Eleberi E., Uka K. A Web Based Digital Repository for Scholarly Publication. *Journal of Software Engineering and Applications*. 2020;13:67-75. http://dx.doi.org/10.4236/jsea.2020.134005
7. Neubauer F., Bredl P., Xu M., et al. MetaConfigurator: A User-Friendly Tool for Editing Structured Data Files. *Datenbank Spektrum*. 2024;24:161-169. https://doi.org/10.1007/s13222-024-00472-7
8. Itiola C., Iwasokun G., Adetooto J. Development of an Online Repository for Academic Research Works in FUTA. *International Journal of Sustainability Management and Information Technologies*. 2021;7:22-26. http://dx.doi.org/10.11648/j.ijsmit.20210701.14
9. Nyaichyai L., Luitel G., Maharjan R.K. Experiences of Library Professionals on DSpace Installation. *Nepal Journal of Multidisciplinary Research*. 2021;4(2):93-105. http://dx.doi.org/10.3126/njmr.v4i2.39177
10. Soto-Sánchez Ó., Maes-Bermejo M., Gallego M., et al. A dataset of regressions in web applications detected by end-to-end tests. *Software Quality Journal*. 2022;30:425-454. https://doi.org/10.1007/s11219-021-09566-x
11. Maia M., Coneglian C., Shintaku M. Propositional study of a model for quality evaluation in technical memory deposits in a Digital Library implemented in DSpace. *RDBCI: Revista Digital de Biblioteconomia e Ciência da Informação*. 2023;21:e023006. http://dx.doi.org/10.20396/rdbci.v21i00.8671927/31965
12. Formanek M. Solving SEO Issues in DSpace-based Digital Repositories. *Information Technology and Libraries*. 2021;40(1):1-28. http://dx.doi.org/10.6017/ital.v40i1.12529
13. Goh H.A., Ho C.K., Abas F.S. Front-end deep learning web apps development and deployment: a review. *Applied Intelligence*. 2023;53:15923-15945. https://doi.org/10.1007/s10489-022-04278-6
14. ElDahshan K.A., Abutaleb G.E., Elemary B.R., et al. An optimized intelligent open-source MLaaS framework for user-friendly clustering and anomaly detection. *The Journal of Supercomputing*. 2024;80:26658-26684. https://doi.org/10.1007/s11227-024-06420-2
15. Zaikina T., Filozova I., Shestakova G., et al. JDS-JOIN2 Repository as a Workspace for Scientific Output. *Physics of Particles and Nuclei Letters*. 2022;19:583-585. https://doi.org/10.1134/S1547477122050454

16. Balutkina N., Stukalova A. Institutional Repositories in Russia and Abroad: Review of Publications. *Bibliotekovedenie* = Russian Journal of Library Science. 2022;71(2):193-206. (In Russ., abstract in Eng.) http://dx.doi.org/10.25281/0869-608X-2022-71-2-193-206
17. Dudnikova O., Bogomolov A. Digital Repository of the Southern Federal University in the Scientific and Educational Space of the University. *Scholarly Research and Information*. 2021;(3):82-93. (In Russ., abstract in Eng.) http://dx.doi.org/10.24108/2658-3143-2021-4-3-82-93
18. Satish S. Development of Institutional Repository Using DSpace at ICMR- National Institute of Epidemiology, Chennai, Tamil Nadu: An Overview. *Asian Journal of Information Science and Technology.* 2019;9(2):96-102. http://dx.doi.org/10.51983/ajist-2019.9.2.268
19. Nneka C., Kaosisochukwu C. Institutional repository for global knowledge sharing. *Journal of ICT Development, Applications and Research.* 2021;3(1/2):41-49. http://dx.doi.org/10.47524/jictdar.v3i1.42
20. Imoro O., Saurombe N. Institutional repository infrastructure: a survey of Ghanaian public universities. *Collection and Curation*. 2024;43(1):1-7. https://doi.org/10.1108/CC-11-2022-0038
21. Nguyen H.T.T., Hwang W., Pham T., Truong T.T.T., Chang H. Development of a mobile Web library application for an institutional repository and investigation of its influences on learning. *The Electronic Library.* 2023;41(5):578-616. https://doi.org/10.1108/EL-03-2023-0062
22. Belov S., Kadochnikov I., Korenkov V., Reshetnikov A., Semenov R., Zrelov P. Data Analysis Platform for Stream and Batch Data Processing on Hybrid Computing Resources. *CEUR Workshop Proceedings*. 2021;3041:174-179. Available at: https://ceur-ws.org/Vol-3041/174-179-paper-32.pdf (accessed 17.04.2025).
23. Chapepa G.G., Ngwira F., Mapulanga P. Metadata creation practices at the Lilongwe University of Agriculture and Natural Resources library's institutional repository. *Digital Library Perspectives*. 2023;39(2):205-219. https://doi.org/10.1108/DLP-09-2022-0074
24. Asadi S., Abdullah R., Yah Y., Nazir S. Understanding Institutional Repository in Higher Learning Institutions: A Systematic Literature Review and Directions for Future Research. *IEEE Access*. 2019;7:35242-35263. https://doi.org/10.1109/ACCESS.2019.2897729
25. Butterfield A.C., Galbraith Q., Martin, M. Expanding your institutional repository: Librarians working with faculty. *The Journal of Academic Librarianship*. 2022;48(6):102628. http://dx.doi.org/10.1016/j.acalib.2022.102628

Поступила 17.04.2025; одобрена после рецензирования 26.05.2025; принята к публикации 14.06.2025.

Submitted 17.04.2025; approved after reviewing 26.05.2025; accepted for publication 14.06.2025.

## 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.*