Оригинальная статья / Original article

https://doi.org/10.21869/2223-1536-2025-15-4-22-34

УДК 004.415.2

Проектирование ядра информационной системы на основе workflow-движка

А. А. Пинаев[1] ✉, Р. А. Томакова[1], Д. К. Реутов[1], Д. А. Фомин[1]

[1] Юго-Западный государственный университет
ул. 50 лет Октября, д. 94, г. Курск 305040, Российская Федерация

✉ e-mail: a59-info@yandex.ru

Резюме

Цель исследования *заключается в разработке архитектуры ядра информационной системы, выполненной на основе open-source workflow-движка Elsa Workflows, ориентированной на автоматизацию бизнес-процессов на предприятиях малого и среднего бизнеса. Особое внимание уделено созданию гибкой, масштабируемой и экономически эффективной информационной системы.*

Методы. *В работе использованы методы системного анализа для проведения сравнительного анализа существующих решений, таких как Camunda, ELMA BPM. Сформулированы и обоснованы функциональные и нефункциональные требования, предъявляемые к информационной системе. Разработан алгоритм функционирования ядра, реализован прототип архитектуры с использованием технологий .NET, PostgreSQL и React. Осуществлено имитационное моделирование, проведены эксперименты.*

Результаты. *Предложена модульная архитектура системы, включающая подсистемы управления задачами, мониторинга, уведомлений, интеграции и аналитики. Elsa Workflows представляет собой легковесный, модульный и свободно распространяемый движок для платформы .NET. Его ключевые особенности заключаются в поддержке кодовой и декларативной реализации процессов, визуального редактора, встроенной поддержки REST API и микросервисной архитектуры. Особое внимание уделено гибкости определения процессов: они могут быть выполнены как на C# (code-first подход), так и в декларативном виде – через JSON или YAML. Реализована схема алгоритма функционирования жизненного цикла задачи с механизмами обработки ошибок, возможностями постобработки и архивации информации. Доказана возможность создания эффективного workflow-ядра на платформе .NET, отличающаяся низкой стоимостью владения и высокой степенью адаптивности.*

Заключение. *Использование open-source workflow-движка Elsa Workflows позволяет создать современное ядро информационной системы, сочетающее гибкость, производительность и соответствие требованиям импортозамещения. Предложенное решение может служить основой для цифровизации слабо автоматизированных производств и способствовать повышению операционной эффективности предприятий.*

Ключевые слова: *движок; информационная система; производственная система; автоматизация процессов.*

Конфликт интересов: *Авторы декларируют отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.*

Для цитирования: Проектирование ядра информационной системы на основе workflow-движка / А. А. Пинаев, Р. А. Томакова, Д. К. Реутов, Д. А.Фомин // Известия Юго-Западного государственного университета. Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2025. Т. 15, № 4. С. 22–34. https://doi.org/ 10.21869/2223-1536-2025-15-4-22-34

Поступила в редакцию 16.10.2025 *Подписана в печать 14.11.2025* *Опубликована 26.12.2025*

Designing the core of an information system based on a workflow engine: a comparative analysis of approaches

Aleksandr A. Pinaev[1] ✉, Rimma A. Tomakova[1], Dmitry K. Reutov[1], Dmitry A. Fomin[1]

[1] Southwest State University
50 Let Oktyabrya Str. 94, Kursk 305040, Russian Federation

✉ e-mail: a59-info@yandex.ru

Abstract

The purpose of the research *is to develop the core architecture of an information system based on the open-source workflow engine Elsa Workflows, focused on the automation of business processes in small and medium-sized businesses. Special attention is paid to the creation of a flexible, scalable and cost-effective information system.*
Methods. *The paper uses system analysis methods to conduct a comparative analysis of existing solutions such as Camunda, ELMA BPM. The functional and non-functional requirements for the information system are formulated and substantiated. An algorithm for the functioning of the core has been developed, and a prototype architecture using technology has been implemented.NET, PostgreSQL, and React. Simulation modeling was carried out, experiments were conducted.*
Results. *A modular system architecture is proposed, including subsystems for task management, monitoring, notifications, integration, and analytics. Elsa Workflows is a lightweight, modular and freely distributed engine for the .NET platform. Its key features are support for code and declarative implementation of processes, visual editing, built-in support for REST API and microservice architecture. Special attention is paid to the flexibility of defining processes: they can be executed both in C# (code-first approach), and in a declarative form – via JSON or YAML. The scheme of the algorithm for the functioning of the task's life cycle with error handling mechanisms, post-processing and information archiving capabilities is implemented. The possibility of creating an effective workflow core on the platform has been proven.NET, characterized by a low cost of ownership and a high degree of adaptability.*
Conclusion. *Using the Elsa Workflows open-source workflow engine allows you to create a modern information system core that combines flexibility, productivity, and compliance with import substitution requirements. The proposed solution can serve as a basis for digitalization of poorly automated industries and contribute to improving the operational efficiency of enterprises.*

Keywords: *engine; information system; production system; process automation.*

Conflict of interest: *The Authors declare the absence of obvious and potential conflicts of interest related to the publication of this article.*

For citation: Pinaev A.A., Tomakova R.A., Reutov D.K., Fomin D.A. Designing the core of an information system based on a workflow engine: a comparative analysis of approaches. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta. Serija: Upravlenie, vychislitel'naja tekhnika, informatika. Meditsinskoe priborostroenie = Proceedings of the Southwest State University. Series: Control, Computer Engineering, Information Science. Medical Instruments Engineering.* 2025;15(4):22–34. (In Russ.) https://doi.org/10.21869/2223-1536-2025-15-4-22-34

Received 16.10.2025 *Accepted 14.11.2025* *Published 26.12.2025*

***

Введение

Цифровизация промышленности требует не только внедрения новых технологий, но и переосмысления подходов к управлению производственными процессами. На предприятиях с разрозненной IT-инфраструктурой процессы часто выполняются частично вручную, что приводит к задержкам, ошибкам и отсутствию прозрачности исполнения [1].

Одним из наиболее эффективных способов повышения управляемости и автоматизации является применение workflow-подхода, при котором бизнес-процессы моделируются, исполняются и контролируются централизованно. Ядром такой системы становится workflow-движок – программный компонент, отвечающий за координацию шагов, управление состоянием, контроль сроков и взаимодействие с пользователями и внешними системами [2].

Среди существующих решений выделяется Camunda – мощная и широко используемая платформа, основанная на стандарте BPMN 2.0. Она активно применяется в крупных корпорациях благодаря своей зрелости, поддержке сложных сценариев и развитой экосистемы. Однако её использование связано с рядом ограничений: высокая стоимость лицензирования, сложность настройки и зависимость от Java-экосистемы [3].

В этих условиях особую актуальность приобретают open-source альтернативы, совместимые с современными стеками разработки и адаптированные к требованиям российского рынка. Одним из таких решений является Elsa Workflows – легковесный, модульный и свободно распространяемый движок для платформы .NET. Его ключевые особенности: поддержка кодовой и декларативной реализации процессов, визуальный редактор, встроенная поддержка REST API и микросервисной архитектуры [4].

Цель данной статьи – продемонстрировать возможность использования Elsa Workflows в качестве ядра информационной системы, проанализировать его функциональные возможности, сравнить с Camunda и предложить архитектуру системы, ориентированную на предприятия малого и среднего бизнеса.

Материалы и методы

Анализ существующих решений и выбор архитектурного подхода

В современной IT-экосистеме автоматизация бизнес-процессов вышла на первый план как ключевой элемент цифровой трансформации предприятий [1]. На рынке представлено множество решений (от мощных корпоративных платформ до легковесных open-source движков). Однако выбор подходящего workflow-движка требует тщательного анализа не только функциональных возможностей, но и таких факторов, как стоимость владения, сложность внедрения, масштабируемость и соответствие стратегии развития компании [2].

Одним из наиболее зрелых и признанных в индустрии решений является Camunda Platform. Эта система, построенная на стандарте BPMN 2.0, зарекомендовала себя как надёжное ядро для управления сложными бизнес-процессами в крупных организациях. Её архитектура позволяет моделировать процессы высокой степени детализации, поддерживает широкий спектр шлюзов, условных ветвлений, параллельных потоков и событий. Интеграция с Java-экосистемой (Spring Boot, Jakarta EE) делает её естественным выбором для enterprise-приложений. Кроме того, Camunda предлагает развитый веб-интерфейс (Cockpit), обеспечивающий мониторинг выполнения процессов, анализ производительности и диагностику ошибок [4].

Несмотря на свои сильные стороны, Camunda имеет ряд существенных ограничений. Во-первых, лицензирование коммерческих версий может быть дорогостоящим, особенно для малых и средних предприятий [5]. Во-вторых, система обладает значительной

сложностью, что требует привлечения квалифицированных специалистов как на этапе внедрения, так и в процессе сопровождения [6]. В-третьих, зависимость от JVM (Java Virtual Machine) может создавать проблемы для организаций, чья IT-инфраструктура основана на экосистеме Microsoft. Эти факторы делают Camunda менее доступным решением для компаний, стремящихся к быстрой и экономичной цифровизации без привязки к дорогому лицензионному ПО [7].

На фоне доминирования Java-решений всё большую популярность приобретают open-source альтернативы, ориентированные на другие технологические стеки. Одним из наиболее перспективных проектов в .NET-экосистеме является Elsa Workflows, который позиционируется как «легкий и гибкий workflow-движок для платформы .NET», что напрямую соответствует запросам на гибкость и низкий порог входа [8]. В отличие от Camunda, Elsa не стремится охватить весь стандарт BPMN, а фокусируется на предоставлении достаточного набора примитивов для реализации типовых сценариев: последовательные и параллельные задачи, условия, пользовательские действия, HTTP-вызовы, таймеры и события [9].

Архитектурно Elsa Workflows отличается модульностью и лёгкостью интеграции. Будучи построенной на ASP.NET Core, она бесшовно встраивается в современные .NET-приложения, использует встроенную систему DI (Dependency Injection) и совместима с Entity Framework Core для хранения состояний процессов [10]. Это позволяет разработчикам быстро разворачивать ядро workflow-системы без необходимости настройки сложной внешней инфраструктуры. Особое внимание уделено гибкости определения процессов: они могут быть описаны как на C# (code-first подход), так и в декларативном виде – через JSON или YAML. Такой двойственный подход открывает возможности как для разработчиков, так и для low-code сценариев, где бизнес-аналитики могут редактировать процессы через визуальный редактор [11].

Сравнительный анализ показывает, что Elsa Workflows не конкурирует с Camunda напрямую, а занимает свою нишу – гибкие, недорогие и быстро развёртываемые решения для МСП и внутренних систем корпораций [12]. Тогда как Camunda применима для масштабных проектов, Elsa подходит для быстрого решения конкретных задач. Например, автоматизация согласования заявок, управление жизненным циклом документов или координация действий в рамках микросервисной архитектуры [13].

В условиях, когда в России усиливается тренд на импортозамещение и развитие собственных IT-решений, open-source проекты на базе .NET становятся особенно привлекательными. Они позволяют компаниям снизить зависимость от иностранных вендоров, обеспечить полный контроль над исходным кодом и адаптировать систему под специфические требования [14]. Именно в этом контексте использование Elsa Workflows как ядра информационной системы приобретает особую актуальность. Она сочетает в себе преимущества современной .NET-платформы, открытости и простоты, предлагая достойную альтернативу коммерческим аналогам [15].

Архитектура системы на основе Elsa Workflows

При проектировании информационной системы, основанной на workflow-движке, ключевым элементом является выбор архитектурного подхода, который обеспечивает не только функциональную полноту, но и такие критически важные качества, как гибкость, масштабируемость, безопасность и простота сопровождения. В рамках данного исследования предлагается архитектура, в которой Elsa Workflows выступает в роли центрального координатора бизнес-процессов, интегрированного в современную .NET-платформу. Такой подход позволяет реализовать ядро системы, предназначенное для эффективного управления жизненным циклом задач, координирования взаимодействия между подразделениями с целью обеспечения сбора данных для последующего анализа.

1. Бэкенд (ASP.NET Core Web API). Система строится на модульной архитектуре, где каждый компонент отвечает за свою функциональную область, что соответствует принципам микросервисного подхода и облегчает дальнейшее масштабирование. Основу системы составляет ASP.NET Core Web API, выполняющий роль бэкенда и обеспечивающий взаимодействие с клиентскими приложениями через RESTful интерфейсы. Именно здесь размещается движок Elsa Workflows, интегрированный как часть хост-приложения [16]. Благодаря встроенной системе Dependency Injection и middleware-механизмам ASP.NET Core, Elsa получает доступ ко всем необходимым сервисам: базе данных, системе логирования, аутентификации и внешним API [17].

2. Elsa Workflows Engine. Центральным элементом архитектуры является сам workflow-движок, реализованный на основе Elsa Core. Он отвечает за загрузку, выполнение и контроль состояния процессов, а также за управление переходами между шагами (активностями). Каждый процесс представляет собой формализованную последовательность действий, которая может включать пользовательские задачи, автоматические вызовы внешних сервисов, условия ветвления и таймеры. Одним из ключевых преимуществ Elsa является возможность определять процессы двумя способами: программно, на языке C#, или декларативно – через JSON или YAML. Это делает систему универсальной: разработчики могут использовать code-first подход для создания сложных логик, тогда как бизнес-аналитики могут редактировать процессы через визуальный редактор без необходимости вносить изменения в код [1].

Программная реализация процесса на C# обеспечивает высокую степень типобезопасности и удобства разработки. Процесс описывается как класс, реализующий интерфейс IWorkflow, а его структура строится с помощью fluent-синтаксиса. Например, процесс согласования заявки может начинаться с логирования события, затем переходить к пользовательской задаче, назначенной конкретному сотруднику, и завершаться HTTP-запросом к ERP-системе [4].

Рассмотрим схему алгоритма работы ядра информационной системы (рис. 1).

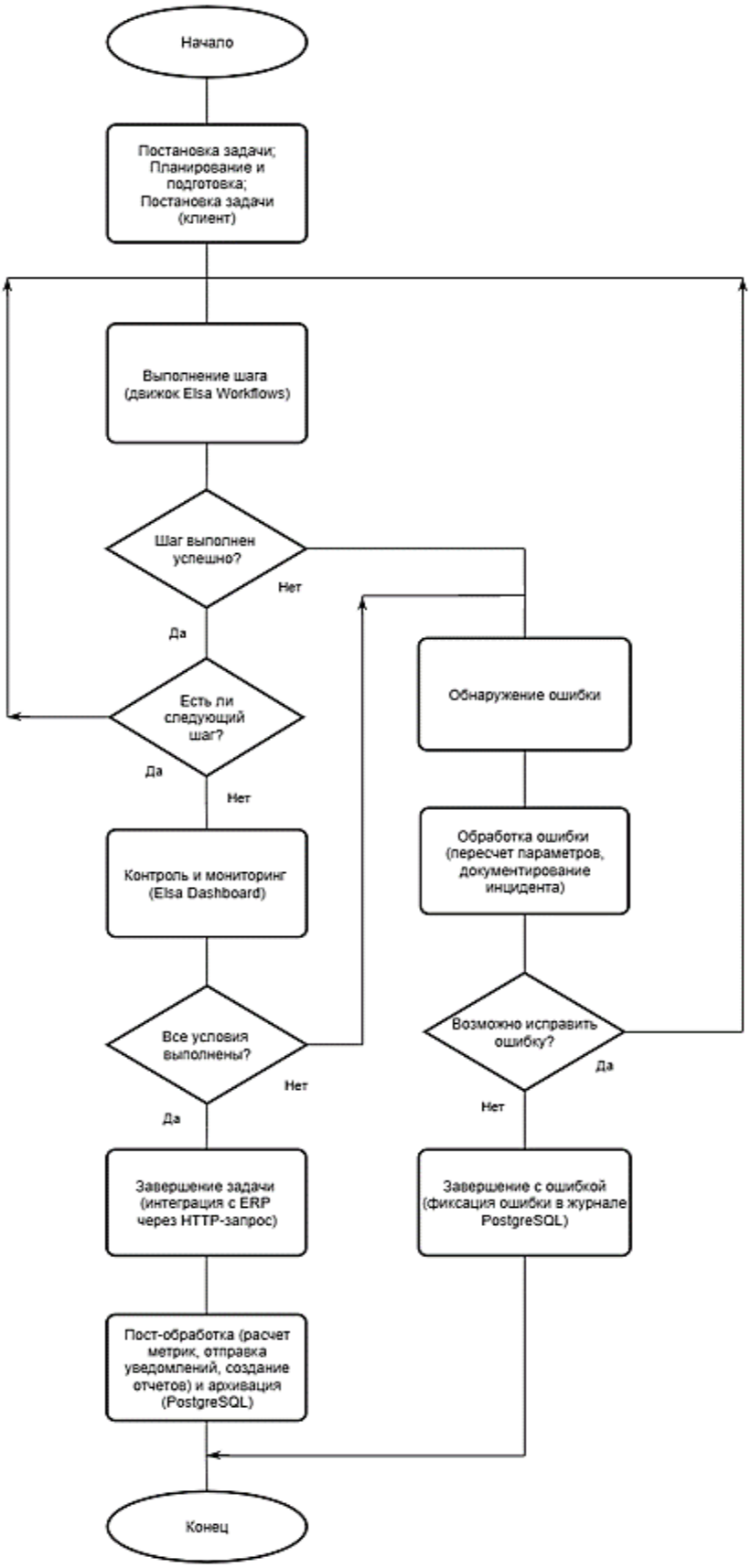

Рис. 1. Схема алгоритма работы ядра информационной системы

Fig. 1. Block diagram of the algorithm for the core operation of the information system

Все этапы контролируются движком, который сохраняет состояние процесса между шагами и гарантирует надежное выполнение даже в случае сбоев. При этом каждая активность может быть расширена через механизмы событий, позволяя добавлять кастомную логику (например, отправку уведомления или запись в журнал) [5].

Декларативный подход, основанный на JSON, особенно ценен в условиях low-code среды. Определение процесса представляется в виде структурированного документа, содержащего список активностей, их параметры и связи. Такой формат легко сериализуется, хранится в базе данных или файловой системе и может быть изменён без пересборки приложения. Это открывает возможности для динамического управления процессами, включая версионирование, тестирование A/B и быстрое реагирование на изменения в бизнес-логике. Кроме того, JSON-описания можно редактировать в веб-интерфейсе Elsa Studio, что делает систему доступной для нетехнических пользователей.

3. Фронтенд (Blazor / React). Фронтенд системы реализуется как одностраничное приложение (SPA), которое может быть построено на различных технологиях в зависимости от требований. В качестве одного из вариантов рассматривается использование React, который позволяет реализовать компонентную архитектуру, где каждый элемент UI – от списка задач до формы согласования – является независимым модулем. Через библиотеку axios или fetch фронтенд взаимодействует с бэкенд-частью, получая данные о текущих задачах, процессах и уведомлениях. Однако, учитывая, что вся система построена на .NET, альтернативным решением может стать Blazor WebAssembly, позволяющий писать фронтенд на C# и использовать единый язык программирования на всех уровнях. Выбор между React и Blazor [15] зависит от стратегии развития компании, наличия команды разработчиков и требований к производительности [18].

4. Система управления базами данных (PostgreSQL). Хранение данных осуществляется с использованием реляционной базы данных, такой как PostgreSQL или SQL Server, через Entity Framework Core. В базе хранятся не только метаданные о пользователях, задачах и правах доступа, но и полная история выполнения процессов: экземпляры workflow, их состояние, журналы выполнения активностей и временные метки. Это обеспечивает прозрачность и возможность аудита, что особенно важно для корпоративных систем [19]. Архитектура persistence-слоя Elsa позволяет легко переключаться между различными провайдерами, что повышает гибкость и адаптивность системы к разным окружениям [20].

5. Elsa Dashboard. Важной частью архитектуры является Elsa Dashboard – веб-интерфейс, предназначенный для администрирования и мониторинга процессов. Через него можно просматривать список определений и запущенных

экземпляров, редактировать процессы, запускать их вручную и анализировать ошибки. Этот инструмент особенно полезен на этапах разработки и тестирования, а также для технической поддержки.

Особое внимание уделяется безопасности и контролю доступа. Система использует механизм аутентификации на основе JWT-токенов, интегрированный в ASP.NET Core Identity. Авторизация реализована по принципу RBAC (Role-Based Access Control), где каждому пользователю назначается одна или несколько ролей, определяющих его права. Например, исполнитель может видеть только свои задачи, менеджер – все задачи своего подразделения, а администратор – имеет полный доступ. Все действия в системе логируются, что позволяет вести учёт изменений и оперативно реагировать на инциденты.

Интеграция с внешними системами обеспечивается через HTTP-активности, очереди сообщений (например, RabbitMQ) и фоновые службы. Это позволяет организовать взаимодействие с ERP, CRM, почтовыми серверами и IoT-устройствами. Например, после завершения процесса система может автоматически отправить документ в 1С, уведомить ответственного через Telegram или запустить обработку данных в аналитической платформе. Такой подход превращает workflow-движок в центральную шину интеграции, объединяющую разрозненные системы в единое информационное пространство [10].

Таким образом, предложенная архитектура представляет собой сбалансированное решение, сочетающее в себе мощь .NET-платформы, открытость и гибкость open-source проекта Elsa Workflows, а также современные подходы к разработке программного обеспечения [12]. Она ориентирована на предприятия малого и среднего бизнеса, нуждающиеся в экономичных, быстро внедряемых и легко настраиваемых системах автоматизации. В отличие от традиционных решений, таких как Camunda, данная архитектура не требует значительных затрат на лицензирование и консалтинг, что делает её особенно актуальной в условиях импортозамещения и цифровой трансформации российских предприятий.

Результаты и их обсуждение

Приведем в качестве примера фрагмент реализации процесса «Согласование заявки» на языке C#:

```
public class ApprovalWorkflow : IWorkflow
{
   public void Build(IWorkflowBuilder builder)
   {
      builder
         .StartWith<LogMessage>(x => x.Message = "Начало процесса")
```

```
            .Then<UserTask>(
                x => x.Description = "Согласовать заявку",
                x => x.Users = new[] { "manager@company.ru" }
            )
            .Then<HttpRequestTask>(
                x => x.Method = HttpMethod.Post,
                x => x.Url = "https://erp/api/tasks/approve"
            )
            .Then<LogMessage>(x => x.Message = "Задача согласована");
    }
}
```

Преимущества предлагаемого решения

Предложенная архитектура информационной системы, построенная на основе open-source workflow-движка Elsa Workflows и платформы .NET, демонстрирует высокую степень соответствия современным требованиям к цифровизации промышленных предприятий. Её ключевые достоинства заключаются не только в технической реализации, но и в стратегической значимости для малых и средних компаний, стремящихся к эффективной автоматизации без чрезмерных затрат и зависимости от внешних вендоров.

Одним из наиболее существенных преимуществ предлагаемой системы является её гибкость и адаптивность. В отличие от монолитных решений, таких как ELMA или Camunda, которые требуют длительного внедрения и дорогостоящей кастомизации, архитектура на базе Elsa позволяет быстро разворачивать новые процессы и оперативно реагировать на изменения в бизнес-логике. Это особенно важно для слабо автоматизированных производств, где процессы часто изменяются под влиянием внешних факторов: колебаний спроса, обновления нормативной базы или реструктуризации подразделений. Возможность описывать процессы как программным кодом на C#, так и в декларативном виде через JSON открывает доступ к управлению системой как для профессиональных разработчиков, так и для бизнес-аналитиков, что способствует формированию культуры low-code внутри организации.

Высокая производительность и надёжность достигаются за счёт использования современной экосистемы .NET. Язык C# и фреймворк ASP.NET Core обеспечивают типобезопасность, высокую скорость выполнения и встроенную поддержку асинхронных операций, что критически важно при работе с большим объёмом параллельных задач. Интеграция с Entity Framework Core и реляционными базами данных, такими как PostgreSQL или SQL Server, гарантирует сохранность состояния процессов даже в случае сбоев. Механизмы persistence и recovery, встроенные в сам движок Elsa, позволяют возобновлять выполнение процессов после перезапуска сервера, обеспечивая непрерывность работы системы.

Отмеченные выше преимущества способствуют тому, что решение

является пригодным для использования в условиях, где недопустимы простои и потеря данных.

Особое внимание уделено безопасности и контролю доступа. Система использует современные подходы к аутентификации и авторизации, включая JWT-токены и ролевую модель (RBAC), что позволяет гибко управлять правами пользователей. Каждый сотрудник получает доступ только к тем задачам и данным, которые соответствуют его должности и зоне ответственности. Все действия в системе детально логируются, что обеспечивает прозрачность исполнения и возможность проведения аудита. Эта функциональность особенно предпочтительна для предприятий, работающих в регулируемых сферах, где требуется документирование всех этапов управления процессами.

С точки зрения экономической эффективности предлагаемое решение обладает существенно более низкой стоимостью по сравнению с коммерческими аналогами. Отсутствие лицензионных отчислений, открытость исходного кода и минимальные требования к инфраструктуре позволяют внедрять систему даже на предприятиях с ограниченным IT-бюджетом.

Стоит особо отметить, что при этом качество реализации не уступает зарубежным продуктам: модульная архитектура, поддержка микросервисов и веб-API обеспечивают масштабируемость и долгосрочную поддержку. Это делает систему не просто временным решением, а основой для построения единого информационного пространства, способного развиваться вместе с предприятием.

Не менее важным является соответствие политике импортозамещения ПО, проводимой правительством нашей страны. В текущих условиях стремление к технологической независимости становится одним из приоритетов государственной политики и корпоративных стратегий. Использование open-source проекта на базе .NET, который может быть свободно модифицирован, адаптирован и развиваться силами внутренней команды, полностью соответствует этому направлению. Это освобождает предприятие от зависимости от иностранных поставщиков ПО, снижает риски, связанные с санкционными ограничениями, и повышает уровень контроля над собственной IT-инфраструктурой.

Наконец, система демонстрирует высокий потенциал для дальнейшего развития и интеграции. Через механизм HTTP-активностей и брокеры сообщений она легко взаимодействует с ERP, CRM, IoT-устройствами и другими корпоративными системами. Это позволяет использовать workflow-движок не как изолированное приложение, а как центральную шину интеграции, координирующую работу различных компонентов. В перспективе возможна интеграция с системами искусственного интеллекта для прогнозирования сроков выполнения задач, выявления узких мест и автоматической оптимизации процессов.

Таким образом, преимущества предлагаемого решения выходят за рамки чисто технических характеристик. Они охватывают экономические, организационные и стратегические аспекты, делая систему не просто инструментом автоматизации, а полноценным элементом цифровой трансформации предприятия.

Она сочетает в себе мощь современных технологий, открытость и доступность, ориентирована на потребности российских предприятий и предлагает реалистичный путь к созданию гибкой, прозрачной и эффективной информационной системы будущего.

Выводы

Использование open-source workflow-движка Elsa Workflows на платформе .NET представляет собой эффективный подход к построению ядра информационной системы для автоматизации бизнес-процессов. Предложенная архитектура сочетает гибкость, высокую производительность и низкий порог входа, что делает её особенно привлекательной для малых и средних предприятий. В отличие от коммерческих аналогов решение не требует значительных финансовых затрат и позволяет полностью контролировать исходный код, что соответствует задачам импортозамещения.

Система обеспечивает надёжное выполнение процессов, поддержку как code-first, так и low-code сценариев, а также интеграцию с внешними системами и средствами мониторинга. Механизмы обработки ошибок, аудита и постобработки данных создают основу для прозрачности, аналитики и устойчивости системы.

Дальнейшие шаги включают практическую реализацию прототипа, его тестирование в реальных условиях и оценку влияния на операционную эффективность предприятия.

Список литературы

1. Сильвер Б. BPMN – Метод и стиль. 2-е изд. М.: Zerde Publishing, 2025. 279 с.

2. Откало И. Автоматизация бизнес-процессов. М.: Литрес, 2024. 480 с.

3. Матусевич А. Свод знаний по управлению бизнес-процессами: BPM CBOK 4.0. М.: Альпина Паблишер, 2019. 602 с.

4. Expressions in C# (Elsa Workflows Docs). URL: https://docs.elsaworkflows.io/ expressions/c (дата обращения: 05.09.2025).

5. Introducing Elsa Workflows 3: A Modern .NET Workflow Engine. URL: https://cantinhode.net/blogs/community-cantinho-de-net/introducing-elsa-workflows-3-a-modern-net-workflow-engine (дата обращения: 05.09.2025).

6. Ньюмен С. Создание микросервисов. М.: Питер, 2025. 624 с.

7. Друри К. Управленческий и производственный учет. М.: Юнити-Дана, 2022. 1424 с.

8. Казинцев А. Технология развития производственной системы. М.: Альпина PRO, 2023. 725 с.

9. Тирни Б., Келлехер Д. Наука о данных. Базовый курс. М.: Альпина Паблишер, 2018. 223 с.

10. Водянкин А. Б. Эффективное управление производственным предприятием. Практическое руководство. М.: Aegitas, 2022. 792 с.

11. Ильин, В. В. Внедрение ERP-систем: управление экономической эффективностью. М.: Интермедиатор, 2016. 296 с.

12. Картер Д. Обработка больших данных. М.: Литресс 2024. 340 с.

13. Парминдер, Сингх, Кочер Микросервисы и контейнеры Docker. М.: Литрес, 2018. 242 с.

14. Моуэт Э. Использование Docker. М.: ДМК Пресс, 2016. 356 с.

15. Дронов В. React 17. Разработка веб-приложений на JavaScript. М.: БХВ-Петербург, 2022. 384 с.

16. Троелсен Э., Джепикс Ф. Язык программирования C# 7 и платформы .NET и .NET Core. М.: Диалектика-Вильямс, 2019. 1330 с.

17. Арораа Г., Чилберто Д. Паттерны проектирования для C# и платформы .NET Core. М.: Питер (Айлиб), 2021. 352 с.

18. Бэнкс А. React: современные шаблоны для разработки приложений. М.: Питер, 2020. 349 с.

19. Шёниг Г.-Ю. PostgreSQL 11. Мастерство разработки. М.: ДМК Пресс, 2019. 354 с.

20. Демиденко А. MongoDB vs PostgreSQL: Битва технологий хранения данных. М.: Литрес, 2025. 90 с.

References

1. Silver B. BPMN – Method and style. 2nd ed. Moscow: Zerde Publishing; 2025. 279 p. (In Russ.)

2. Otkalo I. Automation of business processes. Moscow: Litres; 2024. 480 p. (In Russ.)

3. Matusevich A. Body of knowledge on business process management: BPM CBOK 4.0. Moscow: Alpina Publisher, 2019. 602 p.

4. Expressions in C# (Elsa Workflows Docs). Available at: https://docs.elsaworkflows.io/expressions/c (accessed 05.09.2025).

5. Introducing Elsa Workflows 3: A Modern .NET Workflow Engine. URL: https://cantinhode.net/blogs/community-cantinho-de-net/introducing-elsa-workflows-3-a-modern-net-workflow-engine (accessed 05.09.2025).

6. Newman S. Creation of microservices. Moscow: Piter, 2025. 624 p.

7. Drury K. Managerial and production accounting. Moscow: Unity-Dana; 2022. 1424 p. (In Russ.)

8. Kazintsev A. Technology of production system development. Moscow: Alpina PRO; 2023. 725 p.

9. Tierney B., Kelleher D. Data Science. Basic course. Moscow: Alpina Publisher; 2018. 223 p. (In Russ.)

10. Vodyankin A.B. Effective management of a manufacturing enterprise. Practical guide. Moscow: Aegitas; 2022. 792 p. (In Russ.)

11. Ilyin V.V. Implementation of ERP systems: economic efficiency management. Moscow: Intermediator; 2016. 296 p. (In Russ.)

12. Carter D. Big data processing. Moscow: Litress; 2024. 340 p. (In Russ.)

13. Parminder, Singh, Kocher Microservices and Docker containers. Moscow: Litres; 2018. 242 p. (In Russ.)

14. Mowat E. Using Docker. Moscow: DMK Press; 2016. 356 p. (In Russ.)

15. Dronov V. React 17. Development of web applications in JavaScript. Moscow: BHV-Petersburg; 2022. 384 p. (In Russ.)

16. Troelsen E., Jepix F. The C#7 programming language and platforms .NET and .NET Core. Moscow: Dialectics-Williams; 2019. 1330 p. (In Russ.)

17. Aroraa G., Chilberto D. Design patterns for C# and the platform.NET Core. Moscow: Piter (Aylib); 2021. 352 p. (In Russ.)

18. Banks A. React: modern templates for application development. Moscow: Piter; 2020. 349 p. (In Russ.)

19. Schoenig G.-Y. PostgreSQL 11. Mastery of development. Moscow: DMK Press; 2019. 354 p. (In Russ.)

20. Demidenko A. MongoDB vs PostgreSQL: The Battle of data storage technologies. Moscow: Litres; 2025. 90 p. (In Russ.)

Информация об авторах / Information about the Authors

Пинаев Александр Алексеевич,
студент кафедры программной инженерии,
Юго-Западный государственный университет,
г. Курск, Российская Федерация,
e-mail: a59-info@yandex.ru

Aleksandr A. Pinaev, Student at the Department of Software Engineering, Southwest State University, Kursk, Russian Federation,
e-mail: a59-info@yandex.ru

Томакова Римма Александровна, доктор технических наук, профессор кафедры программной инженерии, Юго-Западный государственный университет,
г. Курск, Российская Федерация,
e-mail: rtomakova@mail.ru,
Researcher ID: O-6164-2015,
ORCID: 0000-0003-0152-4714

Rimma A. Tomakova, Doctor of Sciences (Engineering), Professor at the Department of Software Engineering, Southwest State University, Kursk, Russian Federation,
e-mail: rtomakova@mail.ru,
Researcher ID: O-6164-2015,
ORCID: 0000-0003-0152-4714

Реутов Дмитрий Константинович,
преподаватель кафедры программной инженерии, Юго-Западный государственный университет, г. Курск, Российская Федерация,
e-mail: sdfh.sgh@inbox.ru,
ORCID: 0009-0007-6969-2286

Dmitry K. Reutov, Lecturer at the Department of Software Engineering, Southwest State University, Kursk, Russian Federation,
e-mail: sdfh.sgh@inbox.ru,
ORCID: 0009-0007-6969-2286

Фомин Дмитрий Александрович,
аспирант кафедры программной инженерии,
Юго-Западный государственный университет,
г. Курск, Российская Федерация,
e-mail: dimarro100@gmail.com,
ORCID: 0009-0004-5254-5608

Dmitry A. Fomin, Postgraduate at the Department of Software Engineering, Southwest State University,
Kursk, Russian Federation,
e-mail: dimarro100@gmail.com,
ORCID: 0009-0004-5254-5608