DOI: 10.6060/snt.20248004.00013
УДК 004.9

# ПОСТРОЕНИЕ API ДЛЯ СИСТЕМ ОТСЛЕЖИВАНИЯ ЗАДАЧ

**Зимнуров М.Ф., Астраханцева И.А.**

Зимнуров Марат Фаридович (ORCID 0000-0002-3115-0912),
Астраханцева Ирина Александровна (ORCID 0000-0003-2841-8639)
Ивановский государственный химико-технологический университет,
г. Иваново, Россия. 153000, Ивановская область, г. Иваново, пр. Шереметевский, д. 7.
E-mail: zimtir@mail.ru, i.astrakhantseva@mail.ru

***В статье рассматривается процесс проектирования и разработки API для систем отслеживания задач. Актуальность исследования обусловлена необходимостью создания масштабируемых и адаптивных решений для управления задачами в условиях растущей сложности проектов и увеличения информационных потоков. Особое внимание уделяется анализу существующих подходов к созданию API, а также разработке универсальных решений, обеспечивающих высокую гибкость и совместимость с различными информационными системами. В работе изучаются требования к функциональности, безопасности и масштабируемости API, а также предлагаются архитектурные решения, способные удовлетворить потребности современных организаций. Оригинальность исследования заключается в создании API, ориентированного на обеспечение максимальной совместимости и эффективности интеграции с другими системами, что способствует улучшению процессов управления проектами и задачами.***

**Ключевые слова:** API, Системы отслеживания задач, Программные интерфейсы, Микросервисы

# BUILDING API FOR TASK-TRACKING SYSTEMS

**Zimnurov M.F., Astrakhantseva I.A.**

Zimnurov Marat Faridovich (ORCID 0000-0002-3115-0912),
Astrakhantseva Irina Aleksandrovna (ORCID 0000-0003-2841-8639)
Ivanovo State University of Chemical Technology,
Ivanovo, Russia. 153000, Ivanovo region, Ivanovo, Sheremetevsky Ave., 7.
E-mail: zimtir@mail.ru, i.astrakhantseva@mail.ru

***This article examines the process of designing and developing an API for task tracking systems. The relevance of the research is driven by the need to create scalable and adaptive solutions for task management in the face of increasing project complexity and growing information flows. Particular attention is given to analyzing existing approaches to API development, as well as to the creation of universal solutions that ensure high flexibility and compatibility with various information systems. The study explores the requirements for API functionality, security, and scalability, while also proposing architectural solutions that can meet the needs of modern organizations. The originality of the research lies in the development of an API aimed at ensuring maximum compatibility and effective integration with other systems, thereby enhancing project and task management processes.***

**Keywords:** API, Task Tracking Systems, Software Interfaces, Microservices

## ВВЕДЕНИЕ

В настоящие время практически каждый IT-проект использует систему отслеживания задач для управления и обеспечения их эффективного выполнения. Они позволяют командам координировать работу, следить за прогрессом, распределять ресурсы и контролировать выполнение задач.

В условиях растущей сложности проектов и увеличения количества участников таких систем, критически важным становится интеграция различных инструментов и систем, используемых в организации [1-5].

Актуальность настоящего исследования обусловлена растущим спросом на адаптивные и масштабируемые решения для управления зада-

чами в условиях непрерывного роста информационных потоков и увеличения сложности проектов [6]. Существующие решения часто сталкиваются с проблемами совместимости, ограничения в функциональности или недостаточной гибкости при интеграции с другими системами [7]. Создание высокоэффективного API для систем отслеживания задач способно не только повысить эффективность процессов управления, но и обеспечить необходимую гибкость для масштабирования и адаптации под специфические требования пользователей [8]. Предметом исследования является процесс проектирования и разработки API для систем отслеживания задач. Объект исследования системы управления задачами и проектами в различных организационных контекстах.

Цель данной статьи является рассмотреть ключевые принципы и подходы к проектированию API для систем отслеживания задач. Проанализировать требования к функциональности, безопасности и масштабируемости API, а также рассмотреть практические аспекты разработки и внедрения таких интерфейсов. Оригинальность подхода заключается в разработке универсального API, ориентированного на максимальную совместимость с существующими и перспективными информационными системами, что обеспечит гибкость и эффективность управления задачами в условиях высокой изменчивости требований и нагрузок. Таким образом, проведение настоящего исследования и его результаты будут способствовать улучшению процессов управления задачами и повышению эффективности интеграции информационных систем в различных сферах деятельности.

## МАТЕРИАЛЫ И МЕТОДЫ

Первым шагом в разработке системы отслеживания задач является проектирование архитектуры, выявление ключевых сущностей и определение их взаимодействий. Основной функционал системы строится на отслеживании и создании отчетов за различные временные промежутки. Минимально необходимыми сущностями являются отчет, проект и пользователь. Эти сущности формируют основу данных, с которыми взаимодействует система [9]. Сами по себе системы для отслеживания задач включают в себя организацию управления сущностей проектов, пользователей, задач и мониторинг и анализ вокруг тех же сущностей. Следовательно, подобная кластеризация применима к любому программному решению по взаимодействию с ними, что позволяет в первую очередь создавать решения проксирующее взаимодействие в виде прослойки будь это микросервис или отдельный потребитель, например в Pub/Sub-модели для дополнительного мониторинга или систем оповещения [10].

Основной упор в исследовании направлен на разработку архитектуры для данной системы, обеспечивающим стабильность, безопасность, масштабируемость и эффективность работы. Наилучшее решение для разработки - это создание клиент-серверного приложения, так как оно обеспечивает масштабируемость, безопасность и позволит впоследствии интегрировать систему с различными внешними сервисами и микросервисами [11, 12]. Кроме того, в такой архитектуре каждая часть может разрабатываться и развиваться независимо от другой, что упрощает внесение изменений и поддержание системы [13].

В данной работе было принято решение использовать нотацию C4 для описания архитектуры системы отслеживания задач [14]. Нотация C4 предоставляет четкую и структурированную методику для визуализации различных уровней системы, что позволяет легче понять ее составные части и взаимодействия.

Модель C4 включает в себя четыре уровня абстракции для описания системного архитектурного дизайна. Каждый уровень представляет различные аспекты системы, начиная от высокоуровневого представления до более детализированного.

Контекстная диаграмма показывает систему в контексте внешних взаимодействий и включает пользователей и другие системы, которые взаимодействуют с целевой системой, демонстрируя, как пользователи и другие системы используют функциональность системы (рис. 1).

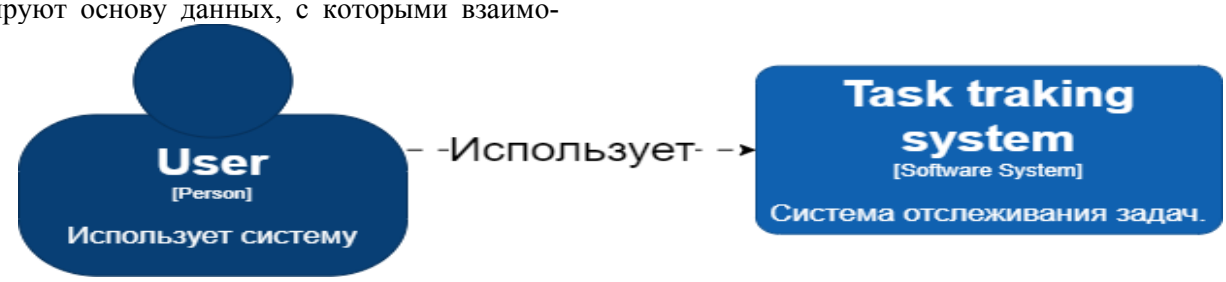


Рис. 1. Диаграмма $C_4$ на уровне контекста
Fig. 1. $C_4$ diagram at the context level

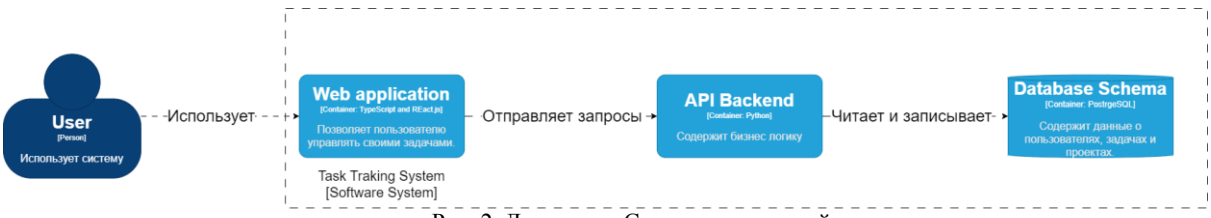


Рис. 2. Диаграмма $C_4$ на уровне контейнеров
Fig. 2. $C_4$ diagram at the container level

Диаграмма контейнеров описывает, из каких контейнеров состоит система. Контейнеры могут быть приложениями, службами, базами данных и т.д. и устанавливают связи между контейнерами, описывая, как они взаимодействуют друг с другом (рис. 2).

Диаграмма компонентов показывает, как контейнеры разбиваются на более мелкие компоненты. Компоненты могут быть классами, модулями или службами, которые отвечают за отдельные функции внутри контейнера и устанавливают связи между собой (рис. 3).

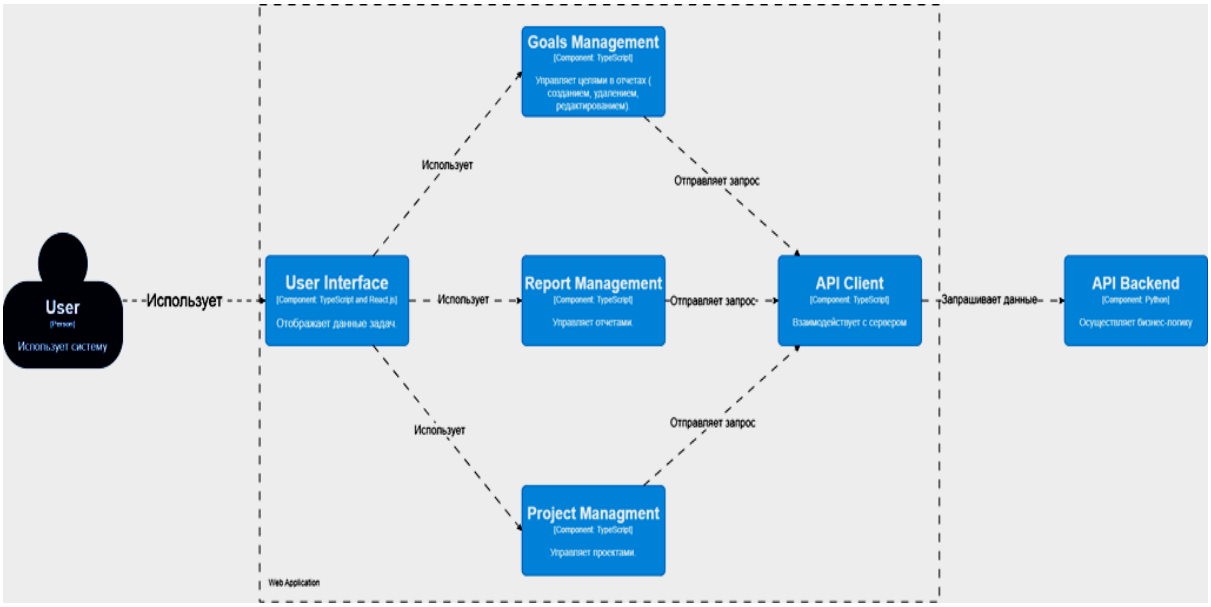


Рис. 3. Диаграмма $C_4$ на уровне компонентов
Fig. 3. $C_4$ diagram at the component level

В рамках работы была создана контекстная диаграмма для системы отслеживания задач, описывающая пользователей и взаимодействия с системой. Определено, что пользователи (User) используют веб-приложение (Web Application), которое взаимодействует с базой данных (Database) и API бэкендом (API Backend). Затем была создана диаграмма контейнеров, которая включает три основных контейнера: веб-приложение (причём здесь клиентом может выступать и мобильное приложение, и отдельный сервис, выбор веб-приложения был обусловлен только для простоты понимания), API бэкенд и базу данных. Определены роли каждого контейнера и их взаимодействия, такие как использование API для обработки запросов и взаимодействие с базой данных для хранения информации. На этапе создания диаграммы компонентов веб-приложение и API бэкенд были разбиты на более мелкие компоненты, включая в веб-приложение такие компоненты, как "User Interface", "Project Management", "Report Management", "Goals Management" и "API Client", а в API бэкенде - компонент "Business Logic". Установлены связи между компонентами, описывающие, как они взаимодействуют друг с другом и обмениваются данными. В результате работы были созданы три диаграммы, которые детализируют архитектуру системы: контекстная диаграмма демонстрирует, как пользователи взаимодействуют с системой, диаграмма контейнеров показывает основные контейнеры системы и их взаимодействия, а диа-

грамма компонентов описывает внутреннюю структуру контейнеров, показывая, как они состоят из более мелких компонентов и как эти компоненты взаимодействуют друг с другом. В системе отслеживания задач выбор сущностей и их полей продиктован необходимостью создать эффективную и гибкую структуру для управления проектами, задачами и сотрудниками. Каждая сущность в модели представляет важную бизнес-логику, а ее поля обеспечивают выполнение конкретных функций системы (рис. 4).

Сущность "Сотрудник" (Employee) была введена для управления пользователями системы, которые участвуют в проектах, составляют отчеты и выполняют задачи. Это ключевой элемент любой системы отслеживания задач, так как сотрудники - это основное действующее лицо, управляющее проектами и процессами. Наличие таких полей, как имя, фамилия и email, необходимо для идентификации сотрудников, а также для налаживания связи и управления доступом. Хотелось бы отметить поле "is_deleted", которое позволяет помечать сотрудников как удаленных без физического удаления, что важно для сохранения целостности данных, например, чтобы не потерять историю работы сотрудника с проектами и отчетами. Сущность "Отчет" (Report) введена для того, чтобы фиксировать прогресс проектов на определенные моменты времени. В отчетах содержатся данные, которые помогают руководителям и командам принимать решения о том, как идет работа над проектом, нужно ли что-то корректировать или есть ли проблемы. Выбор таких полей, как даты начала и окончания отчета, комментарии для каждого статуса (красный, желтый, зеленый), связан с необходимостью структурировать информацию о проекте и его состоянии. Цветовая система статусов выбрана для быстрой визуальной оценки: зеленый означает успешное выполнение задач, желтый указывает на возможные риски, а красный сигнализирует о критических проблемах. Также поля, связанные с отпусками сотрудников, помогают учесть внешние факторы, влияющие на выполнение задач, что особенно важно для долгосрочного планирования.

Сущность "Проект" (Project) выбрана как основная единица планирования и организации работы в системе. Проекты помогают группировать задачи, определять временные рамки и привлекать сотрудников к выполнению задач. Поля для описания проекта и временных меток (даты начала и окончания) необходимы для того, чтобы четко фиксировать цели и сроки, а связь с сотрудниками через ManyToMany позволяет гибко управлять командой проекта. Здесь тоже можно использовать поле "is_deleted" также помогает сохранять историю проектов, даже если они завершены, что полезно для последующего анализа или восстановления данных.

Сущность "Цель" (Objective) введена для того, чтобы детализировать задачи, которые должны быть выполнены в рамках проекта или отражены в отчете. Цели позволяют разбить общий проект на более мелкие и управляемые части, что помогает команде сфокусироваться на конкретных результатах. Цветовая схема, используемая для отображения статуса целей, аналогична той, что используется в отчетах, что упрощает визуальное восприятие. Связь с отчетом позволяет легко отслеживать, какие цели были достигнуты в определенный период, и какие требуют дополнительных усилий. Все эти сущности были выбраны для того, чтобы обеспечить гибкость системы, легкое управление доступом и возможность масштабирования. Они решают важные задачи бизнес-логики, помогая эффективно управлять сотрудниками, проектами и процессами внутри системы [15]. Одним из важнейших выборов для создания системы отслеживания задач является выбор используемых технологий для клиентской части, серверной части и выбор подходящей базы данных [16]. Валидация данных должна производиться как на стороне клиента, так и сервера, чтобы эффективно проверять данные, вводимые пользователем, и формировать правила валидации, которые могут быть легко настроены. Это важно для системы, в которой необходимо гарантировать корректность данных, таких как даты выполнения задач, статусы и описания.

Клиентская часть - это любое решение, что будет взаимодействовать с сервером, к примеру: вебприложения, мобильные приложения, отдельные сервисы и так далее. Серверная часть системы играет ключевую роль в хранении, обработке данных и их передаче клиентской стороне. Здесь подойдет любое решение, в идеале, событийно-ориентированное. В качестве системы управления базами данных была выбрана реляционная база данных, известная своей надежностью и мощным функционалом, безусловно среди CAP-теоремы решение способно варьироваться, ничего не противоречит замене идеи на документы или же модели ключ-значения В базовом представлении и распространенности, реляционная СУБД отлично справляется с выполнением сложных запросов и поддерживает работу с большими объемами данных, что делает её идеальной для систем, где требуется высокая производительность и масштабируемость.

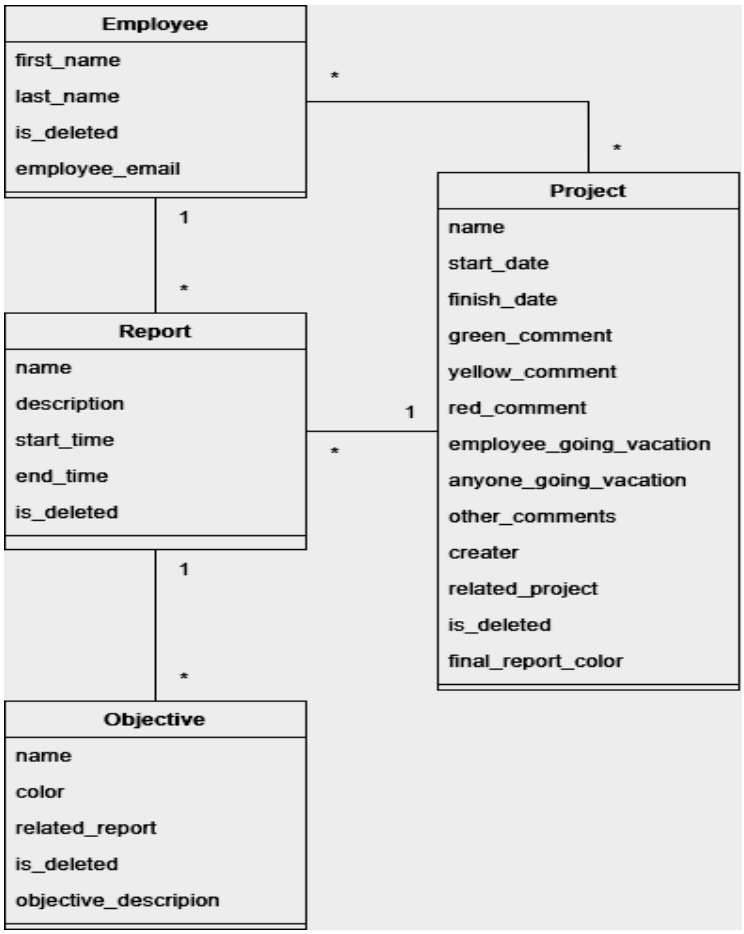


Рис. 4. Диаграмма классов в нотации UML
Fig. 4. Class diagram in UML notation

Взаимодействие между клиентом и сервером осуществляется через REST API. Этот подход использует стандартные HTTP-методы (GET, POST, PUT, DELETE), что обеспечивает интуитивно понятную и легко расширяемую архитектуру. Каждое действие на клиентской стороне, будь то создание или обновление задачи, отправляет соответствующий HTTP-запрос на сервер. REST API поддерживает модульность, позволяя изменять и расширять методы API без необходимости внесения изменений в структуру всего приложения. Такая архитектура делает систему гибкой и легко адаптируемой к новым требованиям. Масштабируемость является важным аспектом архитектуры системы отслеживания задач, особенно с учётом возможного роста функциональности и расширения системы. Клиентская часть должна позволять легко добавлять новые функции и компоненты без значительных изменений в основной структуре приложения.

Благодаря компонентной архитектуре, новые элементы интерфейса можно интегрировать, не затрагивая уже существующий код, что упрощает развитие системы.

Серверная часть также проектируется с учетом будущего расширения. Это подразумевает возможность добавления новых API-методов и функциональных модулей без необходимости кардинального изменения существующей логики.

Вопрос об аутентификации и авторизации пользователей оставлен в стороне, по одной простой причине, что это типовой процесс и каждое предприятие или организация будет использовать либо интеграции внутрь своей системы (к примеру SAML SSO) либо использовать методы по типу OAUTH, LDAP и Active Directory.

Что немаловажно авторизация может быть убрана и вовсе, в случае предоставления выбора лица заполняющего отчёт, так как идентификация в системе может быть произведена и по косвенным признакам, либо не существовать вовсе.

Таким образом, архитектура системы отслеживания задач была тщательно продумана с учетом требований к гибкости, производительности и будущему расширению. Использование клиент-серверной модели с применением современных технологий, обеспечивает возможность лег-

кого добавления нового функционала и эффективной обработки растущего объема данных. Компонентная архитектура на стороне клиента позволяет адаптировать интерфейс под новые задачи без необходимости внесения значительных изменений в основную структуру приложения, что делает его модульным и масштабируемым. На серверной стороне система также предусматривает гибкость и возможность расширения. Благодаря использованию реляционной СУБД в качестве базы данных и REST API для взаимодействия с клиентом, структура системы остается легко адаптируемой к изменениям и новым требованиям. Серверная часть спроектирована таким образом, чтобы поддерживать как увеличение числа пользователей, так и добавление новых функций без серьезных изменений в существующую логику приложения.

В итоге архитектура системы ориентирована на долгосрочное развитие, что позволяет не только эффективно справляться с текущими задачами, но и быть готовой к росту и модернизации в будущем [17].

## РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЯ И ДИСКУССИИ

В результате исследования была разработана архитектура системы отслеживания задач, ориентированная на обеспечение гибкости, эффективности и масштабируемости при росте требований и увеличении нагрузки. Основное внимание было уделено созданию клиент-серверной архитектуры с взаимодействием через REST API, что позволило четко разделить функциональные обязанности клиентской и серверной частей. Такой подход обеспечивает независимое развитие обеих сторон системы и позволяет легко интегрировать новые возможности без значительных изменений в базовой структуре. Клиентская часть системы, должна позволять динамично обновлять интерфейс и поддерживать высокую производительность при добавлении новых функций. Это решение оптимально для создания сложных и интерактивных пользовательских интерфейсов, которые могут расширяться без изменений в основной логике приложения. Возможность легко интегрировать новые компоненты и функциональные блоки делает архитектуру устойчивой к изменениям и адаптируемой к новым требованиям пользователей. Серверная часть системы была спроектирована с учётом роста нагрузки и требований к производительности. Использование REST API упрощает взаимодействие с клиентом, стандартизируя процесс передачи данных и обеспечивая возможность добавления новых методов и сервисов в будущем, при этом, формируя возможность использовать и другие решения. Реляционная СУБД позволяет обрабатывать сложные запросы и поддерживает высокую производительность при увеличении объема данных.

Таким образом, результатом исследования стала архитектура и дизайн система, которые способны эффективно поддерживать масштабирование системы, как по числу пользователей, так и по функциональным возможностям, обеспечивая при этом высокую производительность и стабильность работы.

*Авторы заявляют об отсутствии конфликта интересов, требующего раскрытия в данной статье.*

*The authors declare the absence a conflict of interest warranting disclosure in this article.*

## ЛИТЕРАТУРА


1. **Сазонова М.В., Сазонов А.А.** Модели управления проектами. *Фундаментальные и прикладные исследования: проблемы и результаты.* 2016. № 24.
2. **Логвинова В.А.** Исследование основных тенденций российского рынка программного обеспечения в сфере управления проектами. *Вестник ТИУиЭ.* 2016. № 2 (24).
3. **Гайнуллина И.Ф., Сулейманова А.М.** Системы управления поручениями как инструмент организации и планирования работ. *Форум молодых ученых.* 2018. № 9 (25).
4. **Зимнуров М.Ф., Астраханцева И.А., Гриценицкий П.Н.** Системный анализ и оптимизация количественных показателей эффективности в технологических проектах на основе гибких методологий. *Современные наукоемкие технологии. Региональное приложение.* 2023. № 3(75). С. 61-68. DOI 10.6060/snt. 20237 503.0008. – EDN OYNXHV.
5. **Тебекин А.В.** Эволюция методов управления проектами: мировой опыт и перспективы развития. *Российское предпринимательство.* 2017. № 24.


## REFERENECES


1. **Sazonova M.V., Sazonov A.A.** Project management models. *Fundamental and Applied Research: Problems and Results.* 2016. N 24.
2. **Logvinova V.A.** Study of the main trends in the Russian software market in the field of project management. *Bulletin of TIUiE.* 2016. N 2 (24).
3. **Gainullina I.F., Suleimanova A.M.** Task management systems as a tool for organizing and planning work. *Young Scientists Forum.* 2018. N 9 (25).
4. **Zimnurov M.F., Astrakhantseva I.A., Grimenitsky P.N.** System analysis and optimization of quantitative efficiency indicators in technological projects based on agile methodologies. *Modern high technology. Regional application.* 2023. N 3(75). P. 61-68. DOI 10.6060/snt.20237503.0008
5. **Tebekin A.V**. Evolution of project management methods: world experience and development prospects. *Russian Entrepreneurship.* 2017. N 24.
6. **Vitkovskaya I.A., Kuznetsov P.N.** Analysis of project management systems. *Current Problems of Science and Education.* 2023. N 4.

6. **Витковская И.А., Кузнецов П.Н.** Анализ систем управления проектами. *Актуальные проблемы науки и образования.* 2023. № 4.
7. **Тотухов К.Е., Гришина Ю.Ю.** Исследование и разработка интеллектуальной системы управления задачами. *Электронный сетевой политехнический журнал "Научные труды КубГТУ".* 2019. № 7. С. 88-95. УДК: 004.52. eISSN: 2312-9409.
8. **Сорвилова С.А., Шатько Д.Б.** Анализ программных продуктов для реализации механизма управления поручениями. Инновационный конвент «Образование, наука, инновации. Молодежный вклад в развитие научно-образовательного центра «Кузбасс». Материалы Инновационного конвента. Кемерово, 2019. С. 501-503. Издательство: Кемеровский государственный университет.
9. **Иванов Н.В.** Анализ систем управления проектами. *Форум молодых ученых.* 2017. № 7 (11).
10. **Астраханцева И.А., Горев С.В., Астраханцев Р.Г.** Системный подход к анализу фрактальной природы сложных технических систем. *Известия высших учебных заведений. Серия: Экономика, финансы и управление производством.* 2023. № 3(57). С. 89-97. DOI 10.6060/ivecofin.2023573.657. – EDN PSPGBG.
11. **Соколова А.А.** Разработка системы управления процессом создания программного продукта. *Современные информационные технологии и ИТ-образование.* 2015. № 11
12. **Бобков С.П., Астраханцева И.А.,, Г. Галиаскаров Э.Г.** Применение системного подхода при разработке математических моделей. *Современные наукоемкие технологии. Региональное приложение.* 2021. № 1(65). С. 66-71. DOI 10.6060/snt.20216501.0008. – EDN KOXZWY.
13. **Иванов Н.В.** Разработка модульной архитектуры системы управления проектами. *Форум молодых ученых.* 2017. № 7 (11).
14. C4 Model. Diagrams. https://c4model.com/ diagrams/code
15. **Морозов Е.Н., Горев С.В.** Математические модели для оптимизации машиночитаемых регулятивных систем. *Известия высших учебных заведений. Серия: Экономика, финансы и управление производством.* 2023. № 4(58). С. 71-78. DOI 10.6060/ivecofin. 20235 84.666. – EDN LLSUGG.
16. **Ильин И.В., Хавроненко Н.И.** Анализ фреймовроков для разработки современных веб-приложений. *Актуальные проблемы науки и образования.* 2023. № 3.
17. **Шараева Р.А., Кугуракова В.В.** Оценка сокращения времени при использовании модифицированной методики таск-трекинга в управлении ИТ-проектами. *Программные системы: теория и приложения.* 2022. Т. 13. № 3 (54). С. 307-324. УДК: 004.451.(44/54). Поступила в редакцию: 23.08.2022. Ответственный редактор: Знаменский С.В. Научное редактирование: Елизаров А.М. eISSN: 2079-3316.

7. **Totukhov K.E., Grishina Y.Y.** Research and development of an intelligent task management system. *Electronic Network Polytechnical Journal "Scientific Works of KubSTU".* 2019. N 7. P. 88-95. UDC: 004.52. eISSN: 2312-9409.
8. **Sorvilova S.A., Shatko D.B.** Analysis of software products for implementing a task management mechanism. Innovation Convention "Education, Science, Innovation. Youth Contribution to the Development of the Kuzbass Scientific and Educational Center". Materials of the Innovation Convention. Kemerovo, 2019. P. 501-503. Publisher: Kemerovo State University.
9. **Ivanov N.V.** Analysis of project management systems. *Young Scientists Forum.* 2017. N 7 (11).
10. **Astrakhantseva I.A., Gorev S.V., Astrakhantsev R.G.** A systematic approach to analyzing the fractal nature of complex technical systems. *Ivecofin.* 2023. N 3(57). P. 89-97. DOI 10.6060/ ivecofin.2023573.657. EDN PSPGBG.
11. **Sokolova A.A.** Development of a project management system for software creation processes. *Modern Information Technologies and IT Education.* 2015. N 11.
12. **Bobkov S.P., Astrakhantseva I.A., Galiaskarov E.G.** Application of a systematic approach in the development of mathematical models. *Modern high technology. Regional application.* 2021. N 1(65). P. 66-71.DOI 10.6060/snt. 2021 6501.0008. EDN KOXZWY.
13. **Ivanov N.V.** Development of a modular architecture for project management systems. *Young Scientists Forum.* 2017. N 7 (11).
14. C4 Model. Diagrams. https://c4model.com/diagrams/code
15. **Morozov E.N., Gorev S.V.** Mathematical models for optimizing machine-readable regulatory systems. *Ivecofin*. 2023. N 4(58). P. 71-78. DOI 10.6060/ivecofin.2023584.666. EDN LLSUGG.
16. **Ilyin I.V., Khavronenko N.I.** Analysis of frameworks for developing modern web applications. *Current Problems of Science and Education.* 2023. N 3.
17. **Sharaeva R.A., Kugurakova V.V.** Estimation of time reduction when using a modified task-tracking method in IT project management. *Software Systems: Theory and Applications.* 2022. V. 13. N 3 (54). P. 307-324. UDC: 004.451.(44/54). Submitted: 23.08.2022. Editor-in-chief: S.V. Znamensky. Scientific editing: A.M. Elizarov. eISSN: 2079-3316.


Поступила в редакцию02.09.2024
Принята к опубликованию 08.11.2024

Eceived 02.09.2024
Accepted 08.11.2024