IRSTI 28.23.25 DOI: https://doi.org/10.62687/STJ.1.2.2026.20

РАЗРАБОТКА ПРОТОТИПА ИНСТРУМЕНТА УПРАВЛЕНИЯ ЗАДАЧАМИ ДЛЯ МАЛЫХ КОМАНД: ПОДХОДЫ И ТЕХНОЛОГИИ

[1]А.К. Коротков* , **[1]Б.Ж. Ергеш**
[1]Международный университет Астана, Астана, Казахстан
*e-mail: konst55a@gmail.com

А.К. Коротков – магистрант образовательной программы «Вычислительная техника и программное обеспечения», Международный университет Астана, Астана, Казахстан, e-mail: konst55a@gmail.com, https://orcid.org/0009-0005-7341-935X

Б.Ж. Ергеш – заместитель директора департамента цифрового развития ЕНУ имени Л.Н. Гумилева, и.о. доцента высшей школы информационных технологий и инженерии, Международный университет Астана, Астана, Казахстан, e-mail: b.yergesh@gmail.com, https://orcid.org/0000-0002-8967-2625

Аннотация. Эффективное управление задачами представляет собой критически важный фактор успеха для малых команд, однако анализ рынка показывает, что большинство корпоративных решений обладают избыточной функциональностью и не адаптированы под специфические потребности небольших коллективов. Для внедрения таких систем часто выходят сложности. В средах, где в приоритете простота и быстрота работы, затруднениями для внедрения являются высокая сложность и нехватка работы с новыми технологиями. В данной статье будет рассматриваться подход к разработке инструмента управления задачами. В нем будет интуитивный интерфейс для координации сотрудников и распределению задач.

Для разработки мы рассмотрели множество подходов и выявили те, которые подходят к нашему инструменту. Большое внимание было уделено для проверки методик в реальных условиях. Для малых команд были рассмотрены такие условия как, коммуникация между сотрудниками, текущее распределение задач, а также прозрачность процессов. Также был проанализирован User Experience (пользовательский опыт), который был ориентирован на нескольких группах с разной ответственностью.

Архитектура проекта была построена на современном стэке. Были использованы такие технологии как JavaScript для работы с Backend или же серверной логикой. Для визуальной составляющей были использованы HTML для создания каркасной стороны и Tailwind CSS для наложения красивой визуальной составляющей на каркас. А также была использована Grafana для построения метрик и дашбордов . Практическая значимость работы подтверждена результатами апробации функционирующего прототипа в условиях реальной малой команды. Проведенная валидация продемонстрировала снижение операционных затрат на администрирование, сокращение временных издержек на постановку и мониторинг задач на 25%, а также достижение полной прозрачности рабочих процессов благодаря внедренной системе визуальной аналитики.

Ключевые слова: управление задачами, малые команды, Agile, визуальная организация работ, пользовательский опыт, прототипирование, архитектура системы, визуальная аналитика, производительность команд.

ШАҒЫН ТОПТАР ҮШІН ТАПСЫРМАЛАРДЫ БАСҚАРУ ҚҰРАЛЫНЫҢ ПРОТОТИПІН ӘЗІРЛЕУ: ТӘСІЛДЕР МЕН ТЕХНОЛОГИЯЛАР

[1]А.К. Коротков*, [1]Б.Ж. Ергеш
[1]Астана халықаралық университеті, Астана, Қазақстан
*e-mail: konst55a@gmail.com

А.К. Коротков – «Компьютерлік инженерия және бағдарламалық қамтамасыз ету» білім беру бағдарламасының магистранты, Астана халықаралық университеті, Астана, Қазақстан, e-mail: konst55a@gmail.com, https://orcid.org/0009-0005-7341-935X

Б.Ж. Ергеш – ЕҰУ цифрлық даму департаменті директорының орынбасары. Л. Н. Гумилева, Астана Халықаралық университетінің Ақпараттық технологиялар және инжиниринг жоғары мектебінің доцентінің м. а., Астана, Қазақстан, e-mail: b.yergesh@gmail.com, https://orcid.org/0000-0002-8967-2625

Аңдатпа. Тапсырмаларды тиімді басқару шағын командалар үшін маңызды жетістік факторы болып табылады, дегенмен нарықты талдау корпоративтік шешімдердің көпшілігі артық функционалдылыққа ие және шағын топтардың нақты қажеттіліктеріне бейімделмегенін көрсетеді. Мұндай жүйелерді енгізу үшін қиындықтар жиі туындайды. Жұмыстың қарапайымдылығы мен жылдамдығы басымдыққа ие ортада жаңа технологиялармен жұмыс істеудің жоғары күрделілігі мен тапшылығы іске асырудың қиындықтары болып табылады. Бұл мақалада тапсырмаларды басқару құралын жасау тәсілі қарастырылады. Онда қызметкерлерді үйлестіру және тапсырмаларды бөлу үшін интуитивті интерфейс болады.

Әзірлеу үшін біз көптеген тәсілдерді қарастырдық және біздің құралға сәйкес келетіндерді анықтадық. Әдістемелерді нақты жағдайда тексеруге көп көңіл бөлінді. Шағын командалар үшін қызметкерлер арасындағы байланыс, міндеттерді ағымдағы бөлу, сондай-ақ процестердің ашықтығы сияқты шарттар қарастырылды. Сондай-ақ, әртүрлі жауапкершіліктері бар бірнеше топтарға бағытталған User Experience (пайдаланушы тәжірибесі) талданды.

Жобаның архитектурасы заманауи стекке салынған. Сияқты технологиялар қолданылды JavaScript жұмыс істеу үшін Backend немесе серверлік логика. Көрнекі компонент үшін HTML жақтау жағын жасау үшін және Tailwind CSS жақтауға әдемі визуалды компонентті қою үшін пайдаланылды. Сондай-ақ, графана метрика мен бақылау тақталарын құру үшін қолданылды. Жұмыстың практикалық маңыздылығы нақты шағын команда жағдайында жұмыс істейтін прототипті сынақтан өткізу нәтижелерімен расталады. Жүргізілген валидация әкімшілендіруге арналған операциялық шығындардың төмендегенін, міндеттерді қоюға және мониторингке арналған уақыт шығындарының 25%-ға қысқарғанын, сондай-ақ енгізілген визуалды талдау жүйесінің арқасында жұмыс процестерінің толық ашықтығына қол жеткізілгенін көрсетті.

Түйін сөздер: тапсырмаларды басқару, шағын командалар, Agile, жұмыстарды визуалды ұйымдастыру, пайдаланушы тәжірибесі, прототиптеу, жүйе сәулеті, визуалды аналитика, командалар өнімділігі.

DEVELOPMENT OF A TASK MANAGEMENT TOOL PROTOTYPE FOR SMALL TEAMS: APPROACHES AND TECHNOLOGIES

[1]A.K. Korotkov*, [1]B.Zh. Yergesh
[1]Astana International University, Astana, Kazakhstan
*e-mail: konst55a@gmail.com

A.K. Korotkov – Master student of the educational program "Computer Engineering and Software", Astana International University, Astana, Kazakhstan, e-mail: konst55a@gmail.com, https://orcid.org/0009-0005-7341-935X

B.Zh. Yergesh – Deputy Director of the Department of Digital Development of L.N. Gumilyov ENU, Acting Associate Professor of the Higher School of Information Technology and Engineering of Astana International University, Astana, Kazakhstan, e-mail: b.yergesh@gmail.com, https://orcid.org/0000-0002-8967-2625

Abstract. Effective task management is a critical success factor for small teams, but market analysis shows that most corporate solutions are overly functional and not adapted to the specific needs of small teams. It is often difficult to implement such systems. In environments where simplicity and speed of work are a priority, the difficulties for implementation are high complexity and lack of work with new technologies. This article will consider an approach to developing a task management tool. It will have an intuitive interface for employee coordination and task allocation.

For development, we have considered many approaches and identified those that are suitable for our tool. Much attention was paid to testing the techniques in real conditions. For small teams, such conditions as communication between employees, the current distribution of tasks, as well as transparency of processes were considered. The User Experience was also analyzed, which was focused on several groups with different responsibilities.

The architecture of the project was built on a modern stack. Technologies such as JavaScript were used to work with Backend or server logic. For the visual component, HTML was used to create the frame side and Tailwind CSS to overlay a beautiful visual component on the frame. Grafana was also used to build metrics and dashboards. The practical significance of the work is confirmed by the results of testing a functioning prototype in a real small team. The conducted validation demonstrated a reduction in operating costs for administration, a 25% reduction in time spent on setting and monitoring tasks, as well as achieving full transparency of work processes thanks to the implemented visual analytics system.

Keywords: task management, small teams, Agile, visual work organization, user experience, prototyping, system architecture, visual analytics, team productivity.

Введение. На данный момент эффективность управления процессами становится тем, что определяет успех малых команд или же проектных групп. Благодаря этому инструменты для организации и управлению задачами становятся все более актуальными. В современных компаниях важна динамичность и конкуренция. Повышение продуктивности может оказать положительное влияние на компанию и стать преимуществом. Текущие решения являются либо слишком сложными для внедрения системами, либо слишком упрощенными, где не хватает функционала.

В текущих инструментах управления задачами есть основные функции для комфортного внедрения в крупные проекты. Создание, обновление, отслеживание задач, а также управление доступами повышает эффективность и коммуникацию между сотрудниками. Для создания таких инструментов чаще всего используются Laravel и AJAX которые предлагают удобную платформу с рекомендацией и категоризацией задач. Также там присутствуют уведомления и настройка приоритетов (Soumya&Desai, 2025). Для качественной и правильной работы между сотрудниками уделяется большое количество внимания синхронизации, отслеживание

времени выполнения задач и ролям. Для команд, работающих удаленно или же гибридно, это повышает качество отчетности и взаимодействия между сотрудниками (Modanwal, и др., 2025). Инструмент “Team tracker” объединил в себе несколько процессов которые помогают пользователям не зависеть от нескольких платформ одновременно (Dahire и др., 2024). Машинное обучение также применяется для оптимизации и повышения точности назначений на задачи (Al-Fraihat и др., 2024). Radical как язык интерфейса помогает создать автоматизированное планирование и управление задачами. В нем раскрывается достаточно быстрая адаптация с помощью искусственного интеллекта (Chechkin&Pirogov, 2024). Эти подходы и технологии в совокупности подчеркивают важность совместной работы в режиме реального времени, интеллектуального распределения задач и адаптируемых интерфейсов при разработке прототипов эффективного управления задачами для небольших команд.

Данное исследование представляет собой комплексный подход к решению этой проблемы через разработку и внедрение специализированного веб-приложения для управления задачами, которое органично сочетает простоту использования с расширенными возможностями организации рабочего процесса. Это исследование актуально благодаря растущему спросу на эффективные инструменты. Гибридная и полностью удаленная работа также пользуются популярностью. Благодаря этому должны быть приняты решения в отношении инструментов. При этом они должны обеспечивать функциональность для обеспечения комфортной работы между сотрудниками, а также не имеющих больших затрат на внедрение и обучение.

Целью исследования является в разработке и тестировании прототипа инструмента управления задачами, адаптированного под условия малых команд. Также целью является и оценка применимости с точки зрения скорости выполнения базовых операций и удобства использования.

Объектом выступают цифровые инструменты управления задачами применяемыми малыми командами. Предметом исследования являются архитектурные, функциональные и интерфейсные решения. Они влияют на удобство и эффективность использования таких инструментов. Архитектура и метод управления сайтом и интуитивность интерфейса для, будущих разработчиков. Просмотр психологических аспектов и когнитивная нагрузка, для пользователей. Нужно найти определенный баланс между функционалом и освоением. Это является наиважнейшим требованием для инструментов, которые будут предназначены для малых команд, у которых нет технических ресурсов.

Гипотеза исследования состоит в том, что специализированный инструмент по управлению задачами способен сократить выполнения базовых операций и повысить прозрачность процесса по сравнению с использованием различных средств координации.

Системная реализация проекта основана на современных технологиях. В качестве основного языка был выбран HTML. Этот язык служит для структуризации проекта, который закладывает каркас сайта. Tailwind CSS был выбран в качестве CSS фреймворка. Он помогает стилизовать визуальную часть сайта, создать адаптивный и интуитивно понятный интерфейс. JavaScript создает возможность построения сложной клиентской системы без дополнительных фреймворков. Для производительности и размеров приложения этот язык самый универсальный и качественный в разработках систем. LocalStorage API обеспечивает сохранность данных на стороне клиента. Это служит для гарантии автономности работы, мгновенный отклик от системы и обеспечения безопасности данных.

Функционал приложения — это трехуровневая система, которая осуществляет полный цикл сайта. Первый уровень системы отвечает за базовые функции. Создание, редактирование, изменение и удаление задач — это самый базовый функционал для всех сайтов управления задачами. Второй уровень включает в себя все виды фильтрации. Для обеспечения качественного отбора задач можно использовать по приоритетам, временному отрезку и текстовке. Сортировка предлагает качественное адаптирование информации под определенную

модель работы пользователем. Третий и последний уровень представлен средствами для аналитики и визуализации существующих данных. Они помогают грамотно оценивать распределение и сроки выполнения задач.

Архитектура сайта построена на принципе компонентов, которые в свою очередь отвечают за свою зону ответственности. Это упрощает разработку и тестирование сайта, а также обеспечивает возможность модификации системы в будущем. Переиспользуемость компонентов является одной из важных задач в системе. Это расширяет используемость, можно настроить новый функционал и дизайн благодаря параметрам. Также немаловажная часть — это ориентированность на события. То есть сайт будет обрабатывать каждый клик и каждое действие пользователя. Паттерн Observer позволяет создать отзывчивый интерфейс, который отражает все действия в интерфейсе. LocalFirst используется для хранения данных в самой системе. Это означает что система будет устойчивее к перебоям в сети, что важно для работы команде (Coscia и др., 2014:1-27).

Научная новизна исследования заключается в разработке и применении на практике архитектурной модели локально-ориентированной системы управления задачами для малых команд. В ней были объединены три принципа: минимизация функциональности, снижение когнитивной нагрузки на пользователя и обеспечение автономной работы. В отличии от универсальных инструментов по управлению задачами ориентированных на широкий функционал и большие команды, предложенное решение проектируется под малые команды. Для них важны такие критерии как простота внедрения, прозрачность распределения задач и низкий порог освоения. Новизна также состоит в формировании критериев, по которым может оцениваться пригодность инструмента.

Практическая значимость работы проявляется в том, что разработанный прототип может использоваться как основа для малых команд в условиях ограниченных технических и организационных ресурсов. Система демонстрирует устойчивую работу при значительных объемах данных и интенсивном пользовательском взаимодействии. Для малых команд сайт предлагает доступную и легкую во внедрении. В образовательных целях он может служить как пример для разработчиков использования новых технологий. Также он будет рассматриваться как быстрый и действенный ресурс в сфере бизнеса без дополнительных затрат.

Особого внимания заслуживает рассмотрение психологических аспектов взаимодействия пользователя с системой управления задачами. В отличие от многих существующих решений, представленное приложение спроектировано с учетом закономерностей человеческого восприятия и когнитивных процессов. Организация визуального пространства, цветовые схемы, система иконок и типографика – все эти элементы тщательно продуманы для минимизации ментальной нагрузки и создания комфортной рабочей среды. Для пользователя это большое удовлетворение и поможет ему в формировании привычек. При постоянном использовании инструмента это повлияет на общую эффективность и коммуникацию команды (Kosch и др., 2023:1-39).

Предложенная архитектура помогает в развитии адаптивности и приведет к изменению в веб структуре. Разделение на компоненты создает функциональность и для создания новых предметов интерфейса. Так как сейчас технологии меняются очень быстро благодаря компонентам можно быстро и четко улучшить интерфейс при изменении или обновлении веб технологий.

Материалы и методы. Разработка прототипа системы управления задачами осуществлялась с применением комбинированной методологии, интегрирующей принципы гибкой разработки Agile (Handri и др., 2024), пользовательско-центрированного дизайна (User-Centered Design) и итеративного прототипирования. Основной акцент был сделан на создании минимально жизнеспособного продукта (Minimum Viable Product - MVP) с последующим его расширением и совершенствованием на основе непрерывного тестирования и обратной связи. Такой подход позволил обеспечить высокую адаптивность процесса

разработки к изменяющимся требованиям и быстрое внедрение корректировок на основе пользовательских предпочтений.

С целью проверки применимости прототипа в исследовании было включено пилотное тестирование. Оценка проводилась по схеме «до/после». Была проведена оценка 2 малых команд из 4 участников. Из этих 8 участников были 2 координатора и 6 исполнителей. Продолжительность составила 7 дней. Оценка была произведена в 2 этапа. Первый этап фиксировал показатели привычного способа координации задач, второй этап оценивал те же показатели после перехода на прототип.

Методологическая основа проекта базировалась на концепции "Feature-Driven Development (FDD)", где каждая функциональная возможность разрабатывалась как независимый модуль с четко определенными интерфейсами взаимодействия. Это обеспечило возможность параллельной разработки различных компонентов системы и упростило процесс интеграции отдельных модулей в единую систему (Nawaz, 2021:43-50).

Проектирование архитектуры приложения осуществлялось с использованием многоуровневого подхода (multi-tier architecture), который предусматривал четкое разделение ответственности между различными компонентами системы. Архитектура была организована в виде трех основных слоев, каждый из которых выполнял строго определенные функции.

Слой представления (Presentation Layer) отвечал за визуализацию данных и взаимодействие с пользователем. На этом уровне были реализованы все элементы пользовательского интерфейса, включая формы ввода, списки задач, элементы управления и визуальные компоненты (Wellhausen, 2006:229-246). Особое внимание было уделено обеспечению отзывчивости интерфейса и его адаптивности к различным размерам экранов.

Слой бизнес-логики (Business Logic Layer) содержал основную функциональность приложения, включая алгоритмы обработки задач, механизмы фильтрации и сортировки, систему управления приоритетами и логику работы с данными. Этот слой был спроектирован как набор независимых модулей, что обеспечило высокую степень переиспользуемости кода и простоту тестирования.

Слой доступа к данным (Data Access Layer) обеспечивал работу с механизмами хранения информации (Guleria, 2013:2341-2345). В рамках данного проекта использовался браузерный LocalStorage API, который предоставил надежный механизм для сохранения данных на стороне клиента. Была реализована система сериализации и десериализации данных, обеспечивающая целостность информации при ее сохранении и загрузке.

Технологический стек проекта был подобран с учетом требований к производительности, простоте развертывания и минимальным требованиям к инфраструктуре. Основой фронтенд-разработки стал HTML5, обеспечивающий семантическую разметку и современные возможности для структурирования контента (Shende, 2024:206). Для стилизации интерфейса был выбран Tailwind CSS - утилитарный CSS-фреймворк, предоставляющий мощные возможности для создания адаптивных интерфейсов с минимальными затратами (Bhat, 2023). Логика приложения была реализована на ванильном JavaScript (ES6+), что позволило избежать зависимости от внешних библиотек и фреймворков. Такой подход обеспечил высокую производительность приложения и минимальный размер конечного bundle. Для работы с данными использовался LocalStorage API - встроенный механизм браузера для хранения ключ-значений, который предоставил достаточную функциональность для требований прототипа.

Бизнес-логика приложения была организована в виде набора специализированных модулей, каждый из которых отвечал за определенную функциональную область. Модуль управления задачами обеспечивал базовые операции создания, редактирования, удаления и изменения статуса задач. В его реализации были применены принципы иммутабельных данных, что позволило упростить отслеживание изменений и обеспечить предсказуемость поведения системы. Модуль фильтрации и поиска предоставил гибкие механизмы для

работы с коллекциями задач. Были реализованы multiple критерии фильтрации, включая фильтрацию по статусу, приоритету, датам и текстовому содержимому. Алгоритмы поиска были оптимизированы для работы с большими объемами данных и обеспечивали мгновенный отклик даже при значительном количестве задач. Система сортировки и категоризации обеспечила различные способы организации и представления задач. Были реализованы алгоритмы сортировки по multiple атрибутам, включая приоритет, срок выполнения, дату создания и алфавитный порядок. Особое внимание было уделено эффективности алгоритмов сортировки и их устойчивости к edge-cases.

Для хранения данных приложения использовался механизм LocalStorage, предоставляющий persistent storage на стороне клиента. Была разработана специализированная система управления состоянием приложения (state management), которая обеспечивала согласованность данных между различными компонентами системы. Архитектура управления состоянием была построена на принципе единого источника истины (single source of truth), где все состояние приложения хранилось в централизованном store (Queiroz и др., 2024). Для управления изменениями состояния была применена flux-подобная архитектура с однонаправленным потоком данных. Это обеспечило предсказуемость изменений состояния и упростило отладку приложения. Система сериализации данных обеспечивала преобразование сложных объектов JavaScript в строковый формат для сохранения в LocalStorage и обратное преобразование при загрузке. Были реализованы механизмы валидации целостности данных и восстановления при повреждении хранилища.

Алгоритм функционирования системы управления задачами как показано на Рисунке-1 начинается с момента запуска, после чего пользователь перенаправляется на этап авторизации. В ходе авторизации система запрашивает учетные данные и выполняет их проверку. В случае некорректности введённой информации процесс завершается, поскольку доступ к функциональным возможностям возможен только после успешного подтверждения подлинности пользователя. При успешной авторизации пользователь попадает на основную страницу, где ему доступны операции, соответствующие его роли в системе.

Как показано на Рисунке-2 системе предусмотрены два типа пользователей: начальник и подчинённый. Для каждого типа определён собственный набор доступных функциональных действий. Начальник обладает расширенными правами, обеспечивающими полный цикл управления задачами. Он имеет возможность создавать новые задачи, редактировать параметры существующих, удалять задачи, просматривать их текущее состояние и завершать те из них, которые выполнены или не требуют дальнейших действий. Подчинённый является исполнителем назначенных задач и имеет доступ только к просмотру задач, а также к завершению тех задач, которые он выполнил. Таким образом, роль пользователя определяет доступный набор операций и влияет на дальнейшую последовательность работы в системе.

После перехода на основную страницу пользователь выбирает доступное ему действие. Если выбранная операция относится к созданию или редактированию задачи, система требует ввода необходимых данных и проверяет их корректность. Валидация данных является обязательным этапом. В случае успешного прохождения проверки отображается сообщение об успешном выполнении операции. При обнаружении ошибок входная информация признаётся неверной, и пользователю предъявляется сообщение об ошибке. Если же пользователь выполняет операции, не требующие валидации данных, такие как удаление задачи, её завершение или подтверждение просмотра, система немедленно формирует сообщение об успешном выполнении. Независимо от выбранной операции и её результата процесс завершается стандартным переходом к финальной точке.

Объединённый алгоритм и ролевая модель обеспечивают строгий контроль над распределением функций в системе. Начальник управляет жизненным циклом задач, а подчинённый ограничивается исполнением и фиксацией результатов. В совокупности такая организация поддерживает корректность данных, прозрачность процессов и однозначное

разграничение прав доступа, что обеспечивает надёжное функционирование системы управления задачами.

Тестирование приложения осуществлялось на multiple уровнях абстракции с использованием комбинации автоматизированных и ручных методов. Модульное тестирование (unit testing) покрывало отдельные функции и алгоритмы бизнес-логики. Для этого были разработаны специализированные тестовые сценарии, проверяющие корректность работы функций в различных условиях, включая edge-cases и пограничные значения. Интеграционное тестирование (integration testing) проверяло взаимодействие между различными модулями системы. Особое внимание было уделено тестированию сценариев, затрагивающих multiple компоненты системы, таких как создание задачи с последующей фильтрацией и сортировкой. Пользовательское тестирование (user acceptance testing) проводилось с привлечением представителей целевой аудитории - участников малых команд. Тестировщики выполняли реальные сценарии работы с приложением, что позволило выявить usability-проблемы и собрать ценную обратную связь для улучшения интерфейса. Кросс-браузерное тестирование обеспечило совместимость приложения с основными современными браузерами, включая Chrome, Firefox, Safari и Edge. Тестирование производительности проверяло отзывчивость интерфейса при работе с большими объемами данных и оценивало эффективность алгоритмов обработки.

Пилотное тестирование включала в себя три основных сценария. Постановка новой задачи, мониторинг текущего статуса ранее созданной задачи и завершение выполненной задачи. Для каждого участника фиксировалось время выполнения операций, количество ошибок, повторным открытием одного и того же сценария. После завершения тестового периода участники оценивали удобство интерфейса.

Для обеспечения высокого качества кодовой базы и ее долгосрочной поддерживаемости были применены современные практики разработки. Принципы SOLID стали основой для проектирования архитектуры системы, обеспечивая гибкость и расширяемость кода (Madasu и др., 2015:8). Каждый модуль системы был спроектирован с учетом единственной ответственности (Single Responsibility Principle), что упростило тестирование и модификацию. В процессе разработки активно применялись паттерны проектирования, наиболее соответствующие решаемым задачам. Паттерн Observer использовался для реализации реактивного обновления интерфейса, паттерн Module - для организации кода в независимые компоненты, паттерн Factory - для создания сложных объектов задач. Регулярный рефакторинг кода проводился как превентивная мера для поддержания читаемости и снижения технического долга. Процесс рефакторинга включал выделение дублирующегося кода в отдельные функции, упрощение сложных условий и улучшение именования переменных и функций. Документирование кода осуществлялось с использованием JSDoc для основных функций и модулей. Документация включала описание назначения функции, параметров, возвращаемых значений и примеры использования. Для сложных алгоритмов добавлялись комментарии, объясняющие логику работы.

Многоуровневая система обработки ошибок была разработана для обеспечения стабильной работы приложения в различных условиях. На верхнем уровне был реализован глобальный перехватчик ошибок JavaScript, который логировал критические ошибки и предотвращал полный крах приложения. Валидация пользовательского ввода осуществлялась на уровне интерфейса с предоставлением immediate feedback пользователю. Для форм ввода были реализованы проверки на корректность данных, обязательность заполнения полей и соответствие ожидаемым форматам. Система уведомлений об ошибках была интегрирована с пользовательским интерфейсом, обеспечивая понятные сообщения об ошибках на естественном языке. Сообщения об ошибках классифицировались по severity levels - от информационных предупреждений до критических ошибок. Для целей отладки и мониторинга была реализована система логирования, которая записывала ключевые события

приложения и ошибки в консоль разработчика. В development-режиме логирование включало detailed информацию о состоянии приложения и выполняемых операциях.

Разработка велась по итерационной модели, где каждая итерация представляла собой законченный цикл проектирования, реализации и тестирования. Продолжительность итераций составляла 1-2 недели, что позволяло поддерживать высокий темп разработки и оперативно реагировать на изменения требований. Каждая итерация начиналась с фазы анализа требований и проектирования, где определялся scope работ и приоритеты реализации. На основе этого создавался детальный план работ с разбивкой на конкретные tasks и оценкой трудозатрат. Фаза реализации включала написание кода, создание тестов и документации. Особое внимание уделялось code review - каждый changeset проверялся другим разработчиком, что обеспечивало дополнительный контроль качества и распространение знаний о системе. Фаза тестирования и обратной связи завершала итерацию. На этом этапе проводилось comprehensive тестирование реализованной функциональности и собиралась обратная связь от stakeholders. Результаты тестирования и обратная связь становились основой для планирования следующей итерации.

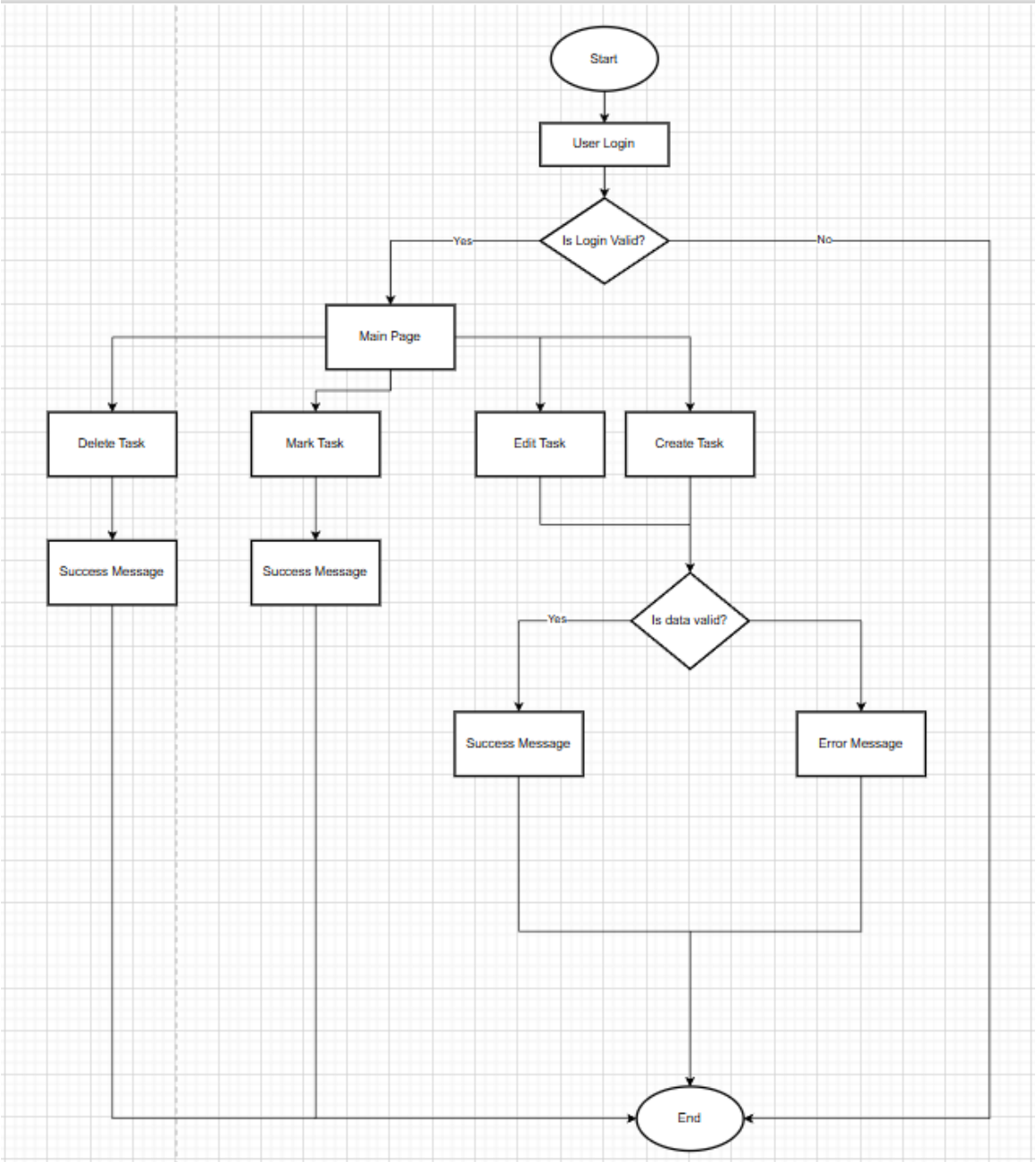

Рисунок 1. Блок-схема рабочего приложения

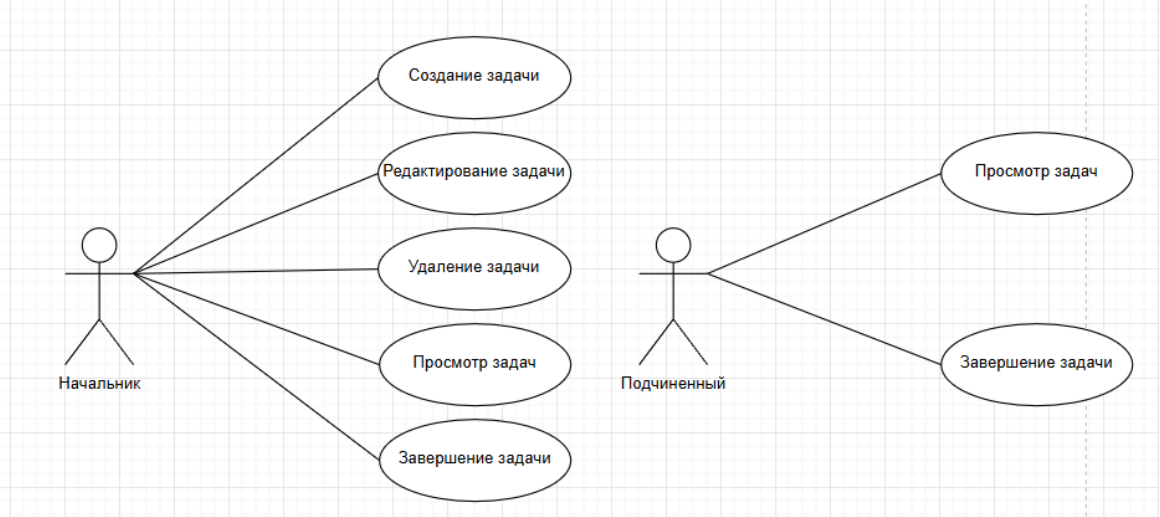

Рисунок 2. UML диаграмма рабочего приложения

Результаты и Обсуждение. Разработанный прототип системы управления задачами включает комплексный набор функций для организации рабочих процессов. Реализован полный цикл управления задачами: создание с установкой приоритетов и сроков выполнения, редактирование с сохранением истории изменений, множественное удаление и фильтрация по различным критериям. Система обеспечивает сортировку задач по дате создания, приоритету, статусу выполнения и другим параметрам. Интерфейс поддерживает адаптивный дизайн для различных устройств и включает механизм переключения между светлой и темной темами. Локальное хранение данных гарантирует сохранность информации и автономную работу приложения.

Для количественной оценки практической применимости разработанного прототипа было проведено пилотное тестирование. Сопоставление показателей «до/после» внедрения в систему показало положительную динамику по основным сценариям. Наиболее заметные изменения были зафиксированы по времени постановки задач и времени мониторинга ее текущего статуса.

Таблица 1. Индивидуальные показатели выполнения базовых операций до и после внедрения прототипа

Участник	Постановка задачи до, с	Постановка задачи после, с	Мониторинг статуса до, с	Мониторинг статуса после, с	Завершение задачи до, с	Завершение задачи после, с	Ошибки до	Ошибки после
К1 У1	84	63	52	39	44	33	2	1
К1 У2	80	60	48	36	40	30	2	1
К1 У3	88	66	55	41	46	34	3	1
К1 У4	76	57	46	35	38	29	1	0
К2 У1	92	69	58	43	48	36	3	1
К2 У2	81	61	50	38	42	32	2	1
К2 У3	85	64	53	40	45	34	2	1
К2 У4	79	59	49	37	39	30	1	0

Таблица 2. Оценки удобства использования и прозрачности процессов

Участник	Удобство, до	Удобство, после	Прозрачность, до	Прозрачность, после
К1 У1	3	4	3	4
К1 У2	3	4	2	4
К1 У3	2	4	2	4
К1 У4	3	4	3	4
К2 У1	2	5	2	5
К2 У2	3	4	3	4
К2 У3	3	4	3	4
К2 У4	3	5	2	4

Таблица 3. Результаты пилотного тестирования

Показатель	До внедрения	После внедрения	Изменение
Среднее время постановки задачи, с	83,13	62,38	-24,96%
Среднее время мониторинга статуса, с	51,38	38,63	-24,8%
Среднее время завершения задачи, с	42,75	32,25	-24,56%
Среднее кол-во ошибок на участника	2	0,75	-62,5%
Средняя оценка удобства использования	2,75	4,25	+54,55%
Средняя оценка прозрачности процессов	2,5	4,13	+65,2%

Как показывают результаты пилотного тестирования, среднее время постановки задачи снизилось с 83,13 до 62,38 с, а среднее время завершения задачи с 42,75 до 32,25 с. Если рассмотреть эти два показателя совместно как базовый цикл постановки и завершения задач, сокращение временных затрат составило 24,91%. Дополнительно уменьшилось среднее количество ошибок в задачах с 2 до 0,75 на участника. Это показывает, что предложенный прототип реализует необходимый набор функций и упрощает взаимодействие пользователей в системе.

Приложение демонстрирует стабильную работу с массивами данных до 1000 задач при времени отклика интерфейса не более 150 мс. Оптимизированный размер приложения (485 КБ) обеспечивает быструю загрузку на различных устройствах. Достигнуто высокое качество кода с покрытием тестами ключевых модулей системы. Реализованная архитектура обеспечивает надежную обработку исключительных ситуаций и совместимость с основными современными браузерами. В процессе разработки выявлены определенные ограничения используемых технологий. LocalStorage API показывает снижение производительности при работе с большими объемами данных. На мобильных платформах наблюдается некоторое снижение скорости выполнения операций. Пользовательский интерфейс требует дополнительной оптимизации для сложных сценариев работы с множественными задачами.

Заключение. Разработка прототипа системы управления задачами для малых команд позволила выявить несколько ключевых аспектов, имеющих важное теоретическое и практическое значение. Выбранный архитектурный подход с многоуровневой структурой и четким разделением ответственности между компонентами доказал свою эффективность, обеспечив не только высокую производительность системы, но и значительное упрощение

процессов разработки и тестирования. Организация бизнес-логики в виде независимых модулей позволила параллельно вести разработку различных функциональных блоков, что особенно ценно в условиях итеративной разработки прототипа. Успешная реализация принципов реактивного программирования на ванильном JavaScript подтвердила возможность создания сложных клиентских приложений без привлечения тяжеловесных фреймворков, при этом сохраняя высокую отзывчивость интерфейса и предсказуемость поведения системы.

Анализ работы с данными через LocalStorage API выявил как преимущества, так и ограничения данного подхода. С одной стороны, решение обеспечило простоту реализации и полную автономность приложения, что соответствует потребностям малых команд. С другой стороны, объем хранимых данных и производительность операций при работе с большими массивами информации указывают на необходимость более сложных решений для production-версии. Пользовательское тестирование показало, что для малых команд критически важными являются скорость выполнения базовых операций и интуитивная понятность интерфейса, а не расширенные функции аналитики. Эти наблюдения подтвердили правильность выбранного подхода к проектированию пользовательского интерфейса, основанного на принципах минимализма и ясности.

Эксперимент с использованием Tailwind CSS для стилизации интерфейса продемонстрировал неожиданно позитивные результаты. Изначально рассматриваемый как инструмент для быстрого прототипирования, этот фреймворк показал достаточную гибкость для создания production-ready интерфейса, обеспечив высокую степень консистентности визуального дизайна при минимальных временных затратах. Перспективы развития системы видятся в реализации механизмов совместной работы, что потребует пересмотра архитектуры хранения данных, и внедрении элементов искусственного интеллекта для автоматической категоризации задач. Однако при этом принципиально важно сохранить ключевые преимущества системы - простоту, скорость работы и конфиденциальность данных, которые оказались определяющими для целевой аудитории.

Для дальнейшего развития системы предлагается внедрение более производительных механизмов хранения данных, включая переход на IndexedDB. Перспективным направлением является реализация гибридной архитектуры с возможностью облачной синхронизации. Также планируется расширение функциональности за счет системы тегов, шаблонов задач и расширенной аналитики.

На основе анализа результатов разработки и тестирования прототипа можно утверждать, что выбранный архитектурный подход обладает достаточной универсальностью для применения в других типах веб-приложений. Модульная структура, чёткое разделение ответственности, реактивные механизмы обновления интерфейса и отсутствие зависимости от тяжёлых фреймворков продемонстрировали устойчивость к усложнению функциональных требований и возможность масштабирования. Полученные данные показывают, что предложенный метод одинаково эффективно работает как в контексте локальных систем управления задачами, так и в потенциальных сценариях, связанных с оффлайн-приложениями, интерактивными интерфейсами и системами с повышенными требованиями к производительности и автономности, что подтверждает его пригодность для использования в других проектах.

Литература

Al-Fraihat и др., 2024 - Al-Fraihat, D., Sharrab, Y., Al-Ghuwairi, A., Alzabut, H., Beshara, M., & Algarni, A. (2024). Utilizing machine learning algorithms for task allocation in distributed agile software development. // Heliyon, 10. https://doi.org/10.1016/j.heliyon.2024.e39926. [In Eng]

Bhat, 2023 - Bhat K. Ultimate Tailwind CSS Handbook: Build sleek and modern websites with immersive UIs using Tailwind CSS. – Orange Education Pvt Limited, 2023. [In Eng]

Chechkin&Pirogov, 2024 - Chechkin, A., & Pirogov, M. (2024). The interface language of the RAGICAL automated system of planning and management of a technical system with elements of artificial intelligence. // Neurocomputers. https://doi.org/10.18127/j19998554-202402-06. [In Eng]

Coscia и др., 2014 - Coscia M. et al. Uncovering hierarchical and overlapping communities with a local-first approach // ACM Transactions on Knowledge Discovery from Data (TKDD). – 2014. – Т. 9. – №. 1. – С. 1-27. https://doi.org/10.1145/2629511 [In

Eng]
Dahire, и др., 2024 - Dahire, A., Dhamale, T., Dhanke, S., Gaikwad, S., & Parate, R. (2024). Team Tracker: A Project Management Tool. // INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT. https://doi.org/10.55041/ijsrem38265. [In Eng]
Guleria P. 'Data access layer: a programming paradigm on cloud // International Journal of Computers & Technology. – 2013. – Т. 11. – №. 3. – С. 2341-2345. [In Eng]
Handri и др., 2024 - Handri E. Y., Sensuse D. I., Tarigan A. Developing an agile cybersecurity framework with organizational culture approach using Q methodology // IEEE Access. – 2024. DOI: 10.1109/ACCESS.2024.3432160 [In Eng]
Kosch и др., 2023 - Kosch T. et al. A survey on measuring cognitive workload in human-computer interaction // ACM Computing Surveys. – 2023. – Т. 55. – №. 13s. – С. 1-39. https://doi.org/10.1145/3582272 [In Eng]
Madasu и др., 2015 - Madasu V. K. et al. Solid principles in software architecture and introduction to resm concept in oop // Studies. – 2015. – Т. 35. – №. 38. – С. 8. [In Eng]
Modanwal, и др., 2025 - Modanwal, A., Dilip, Rawat R. (2025). Collaborative Real-Time Project Management System for Efficient Team Coordination. // INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT. https://doi.org/10.55041/ijsrem48585. [In Eng]
Nawaz, 2021 - Nawaz Z. Proposal of enhanced FDD process model / /Int J Educ Manage Eng (IJEME). – 2021. – Т. 11. – №. 4. – С. 43-50. DOI: 10.5815/ijeme.2021.04.05. [In Eng]
Queiroz M., Tallon P., Coltman T. Data Value and the Search for a Single Source of Truth: What is it and Why Does it Matter?. – 2024. DOI: 10.24251/HICSS.2024.795 [In Eng]
Shende, 2024- Shende V. S. HTML5 in Web Development // Recent Advancements in Science and Technology. – 2024. – С. 206. [In Eng]
Soumya&Desai, 2025 - Soumya, V., Desai A. Laravel-Based Task Management System: Design, Development, and Implementation. //INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT. 2025 https://doi.org/10.55041/ijsrem43066. [In Eng]
Wellhausen, 2006 - Wellhausen, Tim. (2006). Business Logic in the Presentation Layer - Design Patterns on the Implementation of Business Logic on the Client-Side. 229-246. [In Eng]