МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра дискретной математики и информационных технологий

РАЗРАБОТКА ДЕСКТОПНОГО ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ ПРОЕКТАМИ С ИНТЕГРАЦИЕЙ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ И ИНФОГРАФИЧЕСКОЙ ВИЗУАЛИЗАЦИЕЙ ВКЛАДА УЧАСТНИКОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Сырова Дмитрия Олеговича

Научный руководитель
старший преподаватель _____ Н. Е. Тимофеева

Заведующий кафедрой
доцент, к. ф.-м. н. _____ Л. Б. Тяпаев

Саратов 2026

ВВЕДЕНИЕ

Бакалаврская работа посвящена разработке десктопного приложения Project Glow для управления проектами малых команд. В современных учебных и небольших коммерческих проектах задачи, исходный код, сведения о времени работы и данные о вкладе участников часто оказываются разнесены по разным инструментам. Код хранится в Git-репозитории, задачи ведутся в Jira, Trello, Notion, Yandex Tracker или похожем сервисе, а статистика собирается отдельно по коммитам, карточкам и ручным отчётам.

Для крупных организаций такая схема привычна: у команды есть платные тарифы, настроенные права доступа, серверная инфраструктура и отдельные роли для администрирования процесса. Для малой команды она часто оказывается слишком тяжёлой. Участникам приходится создавать аккаунты в сторонних сервисах, поддерживать связь между трекером и репозиторием, переносить информацию вручную и зависеть от доступности облачной платформы. При клонировании репозитория новый участник получает исходный код, но не получает доску задач, комментарии, распределение ролей и историю работы команды.

Актуальность темы определяется потребностью в локальном инструменте, который связывает управление задачами с системой контроля версий. Такой инструмент должен быть проще корпоративного трекера, но полезнее обычной доски карточек: кроме статуса задач он должен показывать состояние Git-репозитория, учитывать рабочее время и давать визуальное представление о вкладе участников.

Цель бакалаврской работы — создать десктопное приложение для управления проектами малых команд. Приложение должно объединять Kanban-доску, Git-синхронизацию, учёт рабочего времени и визуализацию вклада участников, а данные проекта должны храниться рядом с исходным кодом.

Для достижения цели были поставлены следующие задачи:

1. провести анализ существующих инструментов управления проектами и выявить их ограничения для небольших команд;
2. сформулировать функциональные и нефункциональные требования к разрабатываемому приложению;
3. спроектировать архитектуру приложения на базе Electron с разделением на главный процесс, preload-слой и процесс рендеринга;

4. реализовать открытие локального проекта, клонирование Git-репозитория и хранение данных проекта в файлах репозитория;
5. реализовать Kanban-доску с настраиваемыми столбцами, задачами, комментариями, дедлайнами и участниками;
6. реализовать модуль интеграции с Git, включая просмотр статуса, различий файлов, историю коммитов, получение и отправку изменений;
7. реализовать учёт рабочего времени с привязкой сессий к задачам;
8. реализовать статистику проекта, визуализацию вклада участников и файловой структуры репозитория;
9. добавить локализацию интерфейса, уведомления и сборку установщика для Windows.

Материалами для разработки послужили документация Electron, React, TypeScript, Git, simple-git, i18next, Recharts и electron-builder, а также материалы по программной инженерии, управлению проектами, Kanban и системам контроля версий.

Бакалаврская работа состоит из введения, четырёх разделов, заключения, списка использованных источников и приложения. В первом разделе раскрыта актуальность темы. Во втором разделе проведён аналитический обзор предметной области, существующих решений и выбранных технологий. В третьем разделе описаны требования, интерфейс и архитектура приложения. В четвёртом разделе рассмотрена реализация основных модулей Project Glow.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В первом разделе рассмотрена актуальность разработки локального инструмента для управления проектами малых команд. Показано, что в небольших учебных и рабочих проектах процесс часто строится на наборе разрозненных средств: Git отвечает за исходный код, отдельный трекер — за задачи, таблицы или ручные отчёты — за время и вклад участников. Такое разделение усложняет вход нового участника в проект и делает оценку работы команды менее прозрачной.

Отдельно отмечено, что популярные облачные сервисы удобны для крупных команд, но не всегда подходят для малых проектов. Они требуют регистрации, настройки доступа, стабильного интернет-соединения и доверия к внешней инфраструктуре. Кроме того, часть сервисов имеет ограничения доступности или функциональности для российских пользователей. Поэтому для учебного проекта более уместен инструмент, который работает локально, устанавливается на компьютер пользователя и хранит данные рядом с исходным кодом.

Во втором разделе выполнен аналитический обзор предметной области. Рассмотрены подходы к управлению программными проектами, включая Agile, Scrum и Kanban. Kanban выбран как наиболее наглядная модель для небольшой команды: карточки задач перемещаются между столбцами, а текущее состояние проекта видно без сложной настройки процесса.

Также рассмотрена роль Git как распределённой системы контроля версий. Git фиксирует изменения в исходном коде, хранит историю работы и позволяет нескольким участникам синхронизировать изменения через удалённый репозиторий. Для Project Glow важна возможность использовать Git не только для кода, но и для данных проекта. Если задачи, столбцы, участники и рабочие сессии хранятся в файле репозитория, организационная информация перемещается вместе с исходными файлами.

В ходе сравнения рассмотрены Jira, Trello, Notion, YouTrack, GitHub Projects и Yandex Tracker. Все эти инструменты поддерживают управление задачами, но они либо работают как облачные сервисы, либо не обеспечивают локальное хранение доски внутри Git-репозитория. Встроенный учёт времени, визуализация вклада участников и работа с Git также обычно требуют внешних интеграций или дополнительных настроек. В результате сформулирована

ниша разрабатываемого приложения: локальный десктопный инструмент, объединяющий Kanban-доску, Git-операции, учёт времени и визуальную аналитику.

В этом же разделе обоснован выбор технологического стека. Для основы приложения выбран Electron, так как он позволяет собрать настольное приложение с доступом к файловой системе, локальному Git и системным уведомлениям. Интерфейс реализован на React и TypeScript. React подходит для построения динамического интерфейса с доской, карточками и диаграммами, а TypeScript снижает риск ошибок за счёт описания моделей проекта. Для управления состоянием используется Zustand, для работы с Git — simple-git, для диаграмм — Recharts, для локализации — i18next, для сборки установщика — electron-builder.

В третьем разделе описано проектирование Project Glow. Требования разделены на функциональные и нефункциональные. К функциональным требованиям отнесены открытие локальной папки проекта, клонирование удалённого репозитория, работа с Kanban-доской, настройка столбцов, создание и редактирование задач, интеграция с Git, автоматическая синхронизация, учёт времени, управление участниками, уведомления, статистика и визуализация файловой структуры. К нефункциональным требованиям отнесены работа в Windows, локальное хранение данных, разделение общих и личных настроек, безопасность межпроцессного взаимодействия и отзывчивость интерфейса.

Интерфейс спроектирован как единое рабочее окно. После запуска пользователь выбирает локальную папку проекта или клонирует репозиторий по URL. После открытия проекта доступны основные разделы: Kanban-доска, статистика, визуализация и настройки. Левая боковая панель отвечает за Git: в ней показываются изменённые файлы, поле сообщения коммита, кнопки Pull и Commit & Push, а также история последних коммитов.

Архитектура построена вокруг разделения ответственности между слоями Electron. Главный процесс выполняет операции с файловой системой, Git, уведомлениями и фоновыми задачами. Preload-слой предоставляет интерфейсу ограниченный набор безопасных методов через IPC. Процесс рендеринга содержит React-интерфейс и Zustand-хранилища, отвечающие за состояние проекта, таймера, уведомлений и настроек. Такое разделение не даёт интерфейсной части напрямую работать с системными ресурсами и упрощает

сопровождение приложения.

При проектировании данных отдельно выделены общие и локальные сведения. Общий файл проекта хранит столбцы, задачи, участников, комментарии и завершённые сессии времени. Он находится в репозитории и синхронизируется через Git. Локальный файл предназначен для персональных настроек пользователя: пути к проекту, токена доступа, выбранной темы и других параметров, которые не должны попадать к остальным участникам. Такое разделение уменьшает число лишних конфликтов и защищает пользовательские данные от случайной отправки в репозиторий.

Для безопасности предусмотрена проверка структуры данных при чтении. Приложение не должно доверять файлу проекта полностью, потому что он может быть изменён вручную или получен после слияния веток. Поэтому при загрузке проверяются обязательные поля, типы значений и связи между задачами, столбцами и участниками. Текстовые поля очищаются от некорректных значений перед сохранением. Эти меры не заменяют полноценную систему прав доступа, но для локального приложения снижают риск повреждения состояния проекта.

В четвёртом разделе рассмотрена реализация приложения. Project Glow разработан как настольная программа для Windows. Основной сценарий начинается с выбора проекта. Если пользователь открывает папку без файла `project-glow.json`, приложение создаёт начальное состояние: название проекта, базовые столбцы доски и локальную конфигурацию. Если файл уже существует, загружается сохранённая доска.

Центральный модуль приложения — Kanban-доска. В новом проекте создаются базовые столбцы `todo`, `in-progress` и `done`; в интерфейсе они отображаются локализованно как «К выполнению», «В процессе», «Готово» или как «To Do», «In Progress», «Done». Пользователь может переименовывать столбцы, добавлять новые, менять порядок столбцов и переносить задачи между ними. Задача содержит заголовок, описание, дедлайн, исполнителей, комментарии и накопленное рабочее время.

Для хранения данных используется объектная модель проекта. Столбцы содержат упорядоченные списки идентификаторов задач, а сами задачи хранятся в словаре по уникальному идентификатору. Такое решение упрощает перемещение карточек: при переносе задачи меняется порядок

идентификаторов в столбцах и поле текущего столбца у самой задачи, но описание, комментарии и сведения об исполнителях остаются в одном объекте.

Модуль Git-интеграции реализует основные действия с репозиторием без перехода в сторонний клиент. Пользователь может посмотреть статус рабочей директории, список изменённых файлов, различия по файлу, историю коммитов, выполнить Pull, Commit и Push. Перед коммитом приложение сохраняет данные проекта на диск и добавляет их к изменениям, чтобы состояние доски синхронизировалось вместе с кодом.

Особое внимание уделено очереди Git-операций. Ручная отправка изменений и фоновая синхронизация не должны выполняться параллельно, так как это может привести к конфликтам и ошибкам push. Для каждого репозитория операции ставятся в локальную очередь и выполняются последовательно. При отправке изменений приложение выполняет получение актуального состояния удалённой ветки, `pull -rebase` и `push`. Если удалённая ветка изменилась между получением и отправкой, операция повторяется несколько раз. Если возникает настоящий конфликт содержимого, приложение не решает его автоматически, а сообщает пользователю о необходимости ручного разрешения.

Git-панель в интерфейсе показывает не только кнопки синхронизации, но и текущее состояние рабочей директории. Изменённые, добавленные, удалённые и неотслеживаемые файлы отображаются отдельными маркерами. Нажатие на файл открывает просмотр различий, где добавленные строки отделены от удалённых. Это помогает пользователю проверить состав коммита до отправки, не открывая внешний Git-клиент.

Фоновая синхронизация устроена экономно. Приложение не выполняет полное обновление при каждом цикле, а сначала сравнивает известный удалённый коммит с текущим состоянием. Если удалённая ветка не изменилась или файл данных проекта не затронут, синхронизация не запускается. Такой подход снижает число лишних Git-команд и не мешает пользователю работать с доской.

Учёт времени реализован через отдельное состояние таймера. Пользователь выбирает задачу и запускает работу над ней. При остановке таймера создаётся сессия, связанная с задачей и участником. Эффективное время рассчитывается по временным меткам с учётом пауз, поэтому результат

не зависит от задержек таймера в браузерном окружении. Накопленное время используется в статистике проекта.

Модуль статистики показывает сводные показатели проекта: количество задач, процент выполнения, суммарное рабочее время, среднее время на задачу и активность участников. Диаграммы строятся по задачам, сессиям таймера и участникам. Это позволяет оценить не только текущее состояние доски, но и динамику работы команды.

Статистика в приложении не рассматривается как строгая оценка продуктивности участника. Она служит вспомогательным инструментом: показывает распределение задач, примерные трудозатраты и активность по проекту. Такой подход важен для учебной команды, где требуется не только получить итоговую программу, но и показать, как распределялась работа в процессе разработки.

Отдельно реализована визуализация файловой структуры репозитория. Приложение получает историю Git, группирует изменённые файлы по путям и показывает структуру проекта в виде интерактивного представления. Эта функция помогает увидеть, какие части проекта изменялись чаще и где была сосредоточена работа участников.

Визуализация файловой структуры дополняет обычную статистику по задачам. Если Kanban-доска показывает, что было запланировано и выполнено, то дерево файлов показывает, какие области проекта реально менялись. Это особенно полезно при разборе командной работы: можно сопоставить закрытые задачи, коммиты и изменённые части исходного кода.

В приложении также реализованы роли участников, синхронизация авторов из Git, получение коллабораторов через GitHub API, уведомления о дедлайнах и синхронизации, переключение русского и английского языков, светлая и тёмная темы. Для распространения подготовлен установщик Windows в формате NSIS.

ЗАКЛЮЧЕНИЕ

В результате выполнения бакалаврской работы разработано десктопное приложение Project Glow для управления проектами малых команд. Приложение объединяет Kanban-доску, работу с Git, учёт рабочего времени, управление участниками, уведомления, статистику и визуализацию файловой структуры репозитория.

Проведённый анализ показал, что существующие инструменты хорошо решают отдельные задачи управления проектами, но в основном ориентированы на облачную модель и не хранят рабочее состояние проекта непосредственно рядом с исходным кодом. Project Glow закрывает эту проблему за счёт локального хранения данных и синхронизации через Git-репозиторий.

Практический результат заключается в том, что небольшая команда получает единое приложение для повседневной работы с проектом. Пользователь может открыть репозиторий, вести доску задач, фиксировать изменения, учитывать время и получать визуальную картину работы участников без развёртывания отдельного сервера и без обязательной привязки к коммерческому облачному сервису.

Таким образом, поставленные цель и задачи в бакалаврской работе полностью выполнены.

Основные источники информации:

1. Stack Overflow Developer Survey 2023 [Электронный ресурс] // Stack Overflow. — URL: `https://survey.stackoverflow.co/2023/` (дата обращения: 15.03.2026).
2. The Remote Work Report 2023 [Электронный ресурс] // GitLab. — URL: `https://about.gitlab.com/company/culture/all-remote/remote-work-report/` (дата обращения: 15.03.2026).
3. Прессман, Р. С. Программная инженерия: практический подход / Р. С. Прессман. — 8-е изд. — М. : Вильямс, 2019. — 944 с.
4. Соммервилл, И. Инженерия программного обеспечения. — 10-е изд. — М. : Вильямс, 2019. — 960 с.
5. Anderson, D. J. Kanban: Successful Evolutionary Change for Your Technology Business / D. J. Anderson. — Blue Hole Press, 2010. — 278 p.
6. Chacon, S. Pro Git / S. Chacon, B. Straub. — 2nd ed. — Apress, 2014. — 456 p. — URL: `https://git-scm.com/book/ru/v2`.

7. Jira Software [Электронный ресурс] // Atlassian. — URL: https://www.atlassian.com/software/jira (дата обращения: 15.03.2026).
8. Yandex Tracker [Электронный ресурс] // Yandex Cloud. — URL: https://tracker.yandex.ru/ (дата обращения: 15.03.2026).
9. Electron Documentation [Электронный ресурс]. — URL: https://www.electronjs.org/docs/latest/ (дата обращения: 15.03.2026).
10. React Documentation [Электронный ресурс]. — URL: https://react.dev/ (дата обращения: 15.03.2026).
11. TypeScript Documentation [Электронный ресурс]. — URL: https://www.typescriptlang.org/docs/ (дата обращения: 15.03.2026).
12. Пакет simple-git [Электронный ресурс] // npm. — URL: https://www.npmjs.com/package/simple-git (дата обращения: 15.03.2026).
13. Документация i18next [Электронный ресурс]. — URL: https://www.i18next.com/ (дата обращения: 15.03.2026).
14. Recharts [Электронный ресурс]. — URL: https://recharts.org/ (дата обращения: 15.03.2026).
15. Документация electron-builder [Электронный ресурс] // GitHub. — URL: https://www.electron.build/ (дата обращения: 15.03.2026).