ORES

Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.41

ВЕБ-ПРИЛОЖЕНИЕ ДЛЯ АНАЛИЗА АКТИВНОСТИ КОМАНДЫ РАЗРАБОТЧИКОВ НА ОСНОВЕ ДАННЫХ GITHUB-РЕПОЗИТОРИЕВ[1]

Пучков Никита Александрович,
студент 4 курса направления подготовки 09.03.03 «Прикладная информатика» ГАОУ ВО ЛО «ЛГУ им. А.С. Пушкина», г. Санкт-Петербург, Россия
E-mail: nikita160904@mail.ru

Яминов Дмитрий Асхадович,
студент 4 курса направления подготовки 09.03.03 «Прикладная информатика» ГАОУ ВО ЛО «ЛГУ им. А.С. Пушкина», г. Санкт-Петербург, Россия
E-mail: yaminoff.dima2018@yandex.ru

Аннотация

Статья посвящена опыту проектирования и реализации веб-приложения для централизованного анализа активности команды разработчиков на основе данных GitHub-репозиториев. Актуальность работы обусловлена возрастающей потребностью руководителей ИТ-проектов в объективных показателях вклада сотрудников и ограниченной применимостью облачных SaaS-сервисов в организациях, работающих с информацией ограниченного доступа. Описана трёхмодульная архитектура системы, включающая фоновый модуль сбора данных, серверный модуль с программным интерфейсом и клиентское веб-приложение; обоснован выбор колоночной СУБД ClickHouse в качестве аналитического хранилища. Рассмотрено архитектурное решение, при котором координация фоновых исполнителей с программным интерфейсом реализована средствами самой системы управления базами данных — через таблицы движка KeeperMap, что позволило отказаться от привлечения отдельного брокера сообщений. Приведены результаты опытной эксплуатации приложения на тестовом стенде с импортированными публичными репозиториями, подтверждающие работоспособность принятых решений.

Ключевые слова: GitHub, анализ репозиториев, веб-приложение, ClickHouse, KeeperMap, REST API, аналитика разработки, метрики разработчиков, self-hosted-решения, контейнеризация.

WEB APPLICATION FOR ANALYZING SOFTWARE DEVELOPMENT TEAM ACTIVITY BASED ON GITHUB REPOSITORY DATA

[1] научный руководитель: **Королёв Владимир Владимирович**, Старший преподаватель, ГАОУ ВО ЛО «ЛГУ им. А.С. Пушкина», г. Санкт-Петербург, Россия
E-mail: vvkorolyov@yandex.ru
academic supervisor: **Korolev Vladimir Vladimirovich**, Senior lecturer, Pushkin Leningrad State University, Saint-Petersburg, Russia
E-mail: vvkorolyov@yandex.ru

Puchkov Nikita Aleksandrovich,
4th year student of the 09.03.03 "Applied Informatics" degree program, Pushkin Leningrad State University, Saint-Petersburg, Russia
E-mail: nikita160904@mail.ru

Yaminov Dmitry Askhadovich,
4th year student of the 09.03.03 "Applied Informatics" degree program, Pushkin Leningrad State University, Saint-Petersburg, Russia
E-mail: yaminov@example.com

ABSTRACT

The article presents the experience of designing and implementing a web application for centralized analysis of software development team activity based on GitHub repository data. The relevance of the work is determined by the growing need of IT project managers for objective indicators of employee contribution and the limited applicability of cloud SaaS services in organizations dealing with restricted access information. A three-module architecture of the system is described, including a background data collection module, a server module with an application programming interface and a client web application; the choice of ClickHouse column-oriented DBMS as the analytical storage is substantiated. An architectural solution is considered in which the coordination of background workers with the application programming interface is implemented by the database management system itself — through the tables of the KeeperMap engine, which allowed to avoid the use of a separate message broker. The results of pilot operation of the application on a test bench with imported public repositories are presented, confirming the operability of the adopted solutions.

Keywords: GitHub, repository mining, web application, ClickHouse, KeeperMap, REST API, software development analytics, developer metrics, self-hosted solutions, containerization.

Введение

Управление разработкой программного обеспечения в современной ИТ-компании всё в большей степени опирается на количественные данные о деятельности команды. Платформа GitHub занимает в этой области центральное место как поставщик первичных сведений о коммитах, операциях с pull-запросами, обсуждениях при ревью и работе с задачами в системе отслеживания инцидентов. Полнота фиксации указанной активности в системе контроля версий открывает принципиальную возможность построения управленческой аналитики, опирающейся не на субъективное мнение руководителя проекта, а на наблюдаемые показатели вклада сотрудников, в том числе на метрики, разработанные исследовательским проектом DORA [1].

На рынке представлен ряд коммерческих сервисов программного обеспечения как услуги (SaaS), реализующих аналогичную функциональность: LinearB, Waydev, GitClear. Все они построены как облачные платформы, что предполагает передачу метаданных проектов на серверы поставщика, как правило, расположенные за пределами российской юрисдикции. Для организаций, работающих с объектами критической информационной инфраструктуры или с информацией, относящейся к коммерческой тайне [2], такая модель применима с существенными ограничениями: качество сервиса и набор отчётов становятся второстепенными по отношению к требованиям локализации обработки данных.

Открытые альтернативы — Gitana [3], GrimoireLab, gitinspector — частично снимают это ограничение, поскольку допускают развёртывание в корпоративном контуре, однако активная разработка большинства из них остановлена, а функциональность ориентирована преимущественно на анализ открытых сообществ, а не на корпоративные управленческие задачи.

В этих условиях задача создания собственного (self-hosted) веб-приложения для централизованного анализа активности команды разработчиков представляется самостоятельной и практически значимой. Настоящая статья обобщает опыт проектирования, реализации и опытной эксплуатации такого приложения, выполненного по запросу российской ИТ-компании, работающей с защищаемой информацией.

Цель исследования

Целью настоящего исследования является разработка собственного веб-приложения, обеспечивающего централизованный сбор, хранение и аналитическую обработку данных о деятельности команды разработчиков в GitHub-репозиториях, а также опытная апробация принятых архитектурных решений на тестовом стенде с импортированными публичными репозиториями.

Материалы и методы исследования

Базой для постановки задачи послужили бизнес-процессы российской ИТ-компании, специализирующейся на разработке программного обеспечения для защиты информации и автоматизации процессов в отраслях с регламентированным обращением с данными. Большинство заказчиков такой компании являются операторами объектов критической информационной инфраструктуры или работают с информацией ограниченного доступа, что определяет принципиальное ограничение на использование облачных сервисов сторонних поставщиков для обработки рабочих данных.

В качестве методологической основы использован системный анализ предметной области, объектно-ориентированное моделирование в нотации UML, моделирование бизнес-процессов в нотации BPMN. Технологическую базу составили: язык программирования Python с микрофреймворком Flask — для реализации серверного компонента и фоновых исполнителей; колоночная система управления базами данных ClickHouse [4] — в роли аналитического хранилища; связка HTML5, CSS3 и JavaScript с библиотеками Chart.js и Axios — для клиентского веб-приложения; платформа контейнеризации Docker — для развёртывания и оркестрации компонентов. Эмпирическая часть работы выполнена на тестовом стенде с импортированным для проверки публичным репозиторием pallets/flask; первичный сбор данных осуществлялся через GitHub REST API [5] с обработкой ограничений на частоту запросов [6].

Результаты и их обсуждение

Разработанное веб-приложение представляет собой трёхмодульную систему, развёртываемую целиком в инфраструктуре заказчика. Принципиальное архитектурное решение, принятое на самом раннем этапе проектирования, состояло в разделении функций между независимо разворачиваемыми компонентами с принципиально различным профилем нагрузки. Состав системы и сетевые взаимодействия её компонентов представлены на рисунке 1.

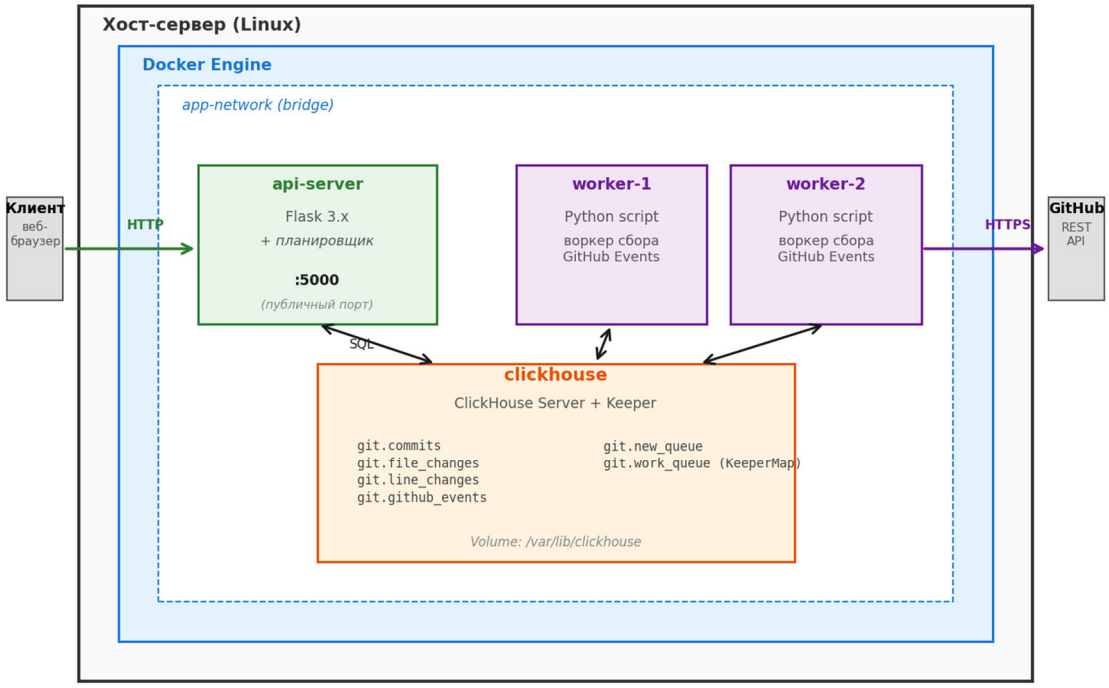

Рисунок 1. Схема развёртывания компонентов веб-приложения. Источник: разработано авторами.

Модуль сбора данных реализует длительные обращения к внешнему программному интерфейсу GitHub. Он клонирует целевые репозитории в локальный кэш, разбирает историю системы контроля версий и параллельно опрашивает поток событий через эндпоинт /repos/{owner}/{name}/events, извлекая сведения о pull-запросах, операциях ревью, действиях с задачами и иных типах активности. При обработке потока событий учитывается ограничение частоты запросов GitHub в размере 5 000 обращений в час на токен [6] — отслеживание заголовков X-RateLimit-Remaining и X-RateLimit-Reset позволяет удерживать суммарную нагрузку в установленных пределах. Полученные сведения пакетно загружаются в аналитическое хранилище.

Серверный модуль реализован на микрофреймворке Flask и принимает на себя весь обмен с клиентским веб-приложением. На него возложены три основные функции: приём пользовательских запросов через REST-интерфейс и формирование ответов в формате JSON; постановка задач на сбор данных в очередь и поддержание периодического обновления уже отслеживаемых репозиториев; выполнение аналитических SQL-запросов к ClickHouse и преобразование их результатов в форму, пригодную для отображения в клиентском приложении. Состояния между запросами серверный модуль не хранит, что позволяет масштабировать его горизонтально за счёт запуска дополнительных реплик за балансировщиком нагрузки.

Аналитическое хранилище реализовано на колоночной СУБД ClickHouse. Выбор данной системы обусловлен профилем нагрузки приложения: основная её часть приходится на агрегирующие запросы к таблицам с миллионами и десятками миллионов записей, при которых классические строковые реляционные СУБД заметно уступают колоночным [7]. Схема базы данных намеренно денормализована: типовые поля родительских сущностей продублированы в дочерних таблицах (например, имя автора и время коммита присутствуют как в таблице коммитов, так и в таблице изменений на уровне файлов), что позволяет выполнять подавляющее большинство аналитических запросов в

режиме сканирования одной таблицы без операций соединения, дорогостоящих в колоночных СУБД.

Отдельного внимания заслуживает реализация координационного слоя между серверным модулем и фоновыми исполнителями. В традиционной микросервисной архитектуре [8] такую задачу решает специализированный брокер сообщений (Redis Streams, RabbitMQ, Apache Kafka), однако его использование в рассматриваемом случае повлекло бы за собой добавление в развёртывание ещё одного контейнера, ещё одного сетевого канала и ещё одной точки отказа при сравнительно скромном профиле нагрузки очереди — десятки задач в час. В качестве альтернативы был применён встроенный в ClickHouse механизм — таблицы движка KeeperMap [9], использующие координатор ClickHouse Keeper в качестве отказоустойчивого распределённого хранилища типа «ключ-значение». Серверный модуль выполняет вставку записи в такую таблицу (постановка задачи), а фоновый исполнитель — атомарное чтение с последующим удалением (извлечение задачи). Поскольку координатор Keeper уже присутствует в кластере ClickHouse, такое решение не добавляет новых компонентов в топологию развёртывания и сокращает её до трёх контейнеров — серверный модуль, фоновые исполнители, ClickHouse со встроенным Keeper.

Клиентское веб-приложение, разработанное соавтором настоящей статьи, предоставляет руководителю проекта интерфейс для работы с накопленными аналитическими данными. Базовый пользовательский сценарий начинается с ввода адреса репозитория в формате «владелец/имя» и инициирования его анализа; в ответ серверный модуль либо немедленно возвращает имеющиеся данные, либо ставит задачу на их сбор и сообщает клиенту о начале фоновой обработки. По её завершении пользователь получает дашборд репозитория, верхняя часть которого содержит информационную карточку проекта и таблицу последних коммитов команды (рисунок 2), а нижняя — набор аналитических виджетов: график распределения коммитов по времени, столбчатую диаграмму вклада отдельных разработчиков и круговую диаграмму соотношения типов изменений в коде (рисунок 3). Дополнительная функциональность включает панель фильтрации по периоду и автору, формирование сводной статистики по разработчикам, возможность экспорта отчёта и обработку нештатных ситуаций (несуществующий репозиторий, исчерпание ограничений GitHub) с выводом понятных пользователю сообщений.

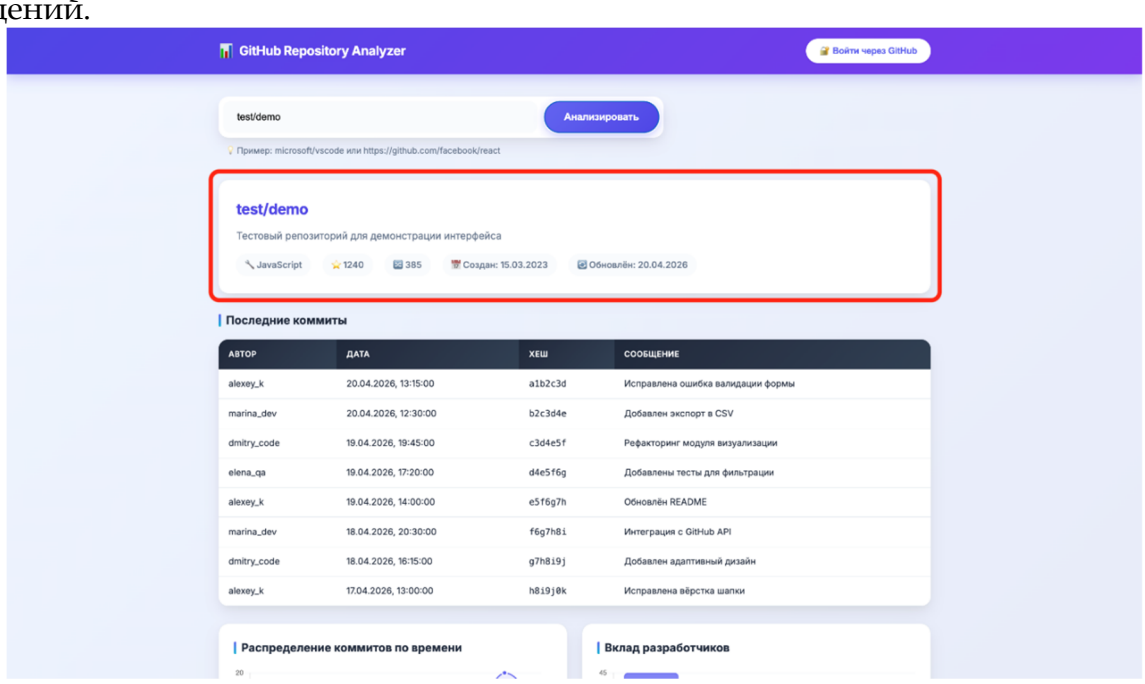

Рисунок 2. Информационная карточка репозитория и таблица последних коммитов в интерфейсе клиентского приложения. Источник: разработано авторами.

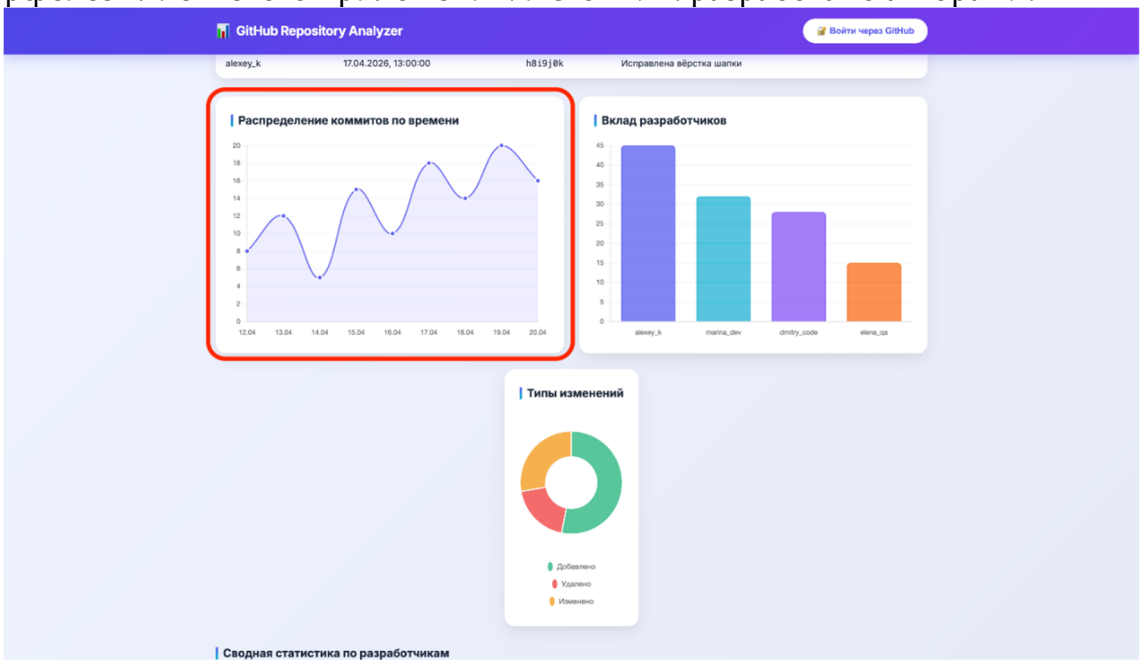

Рисунок 3. Аналитические виджеты клиентского приложения: распределение коммитов по времени, вклад разработчиков, типы изменений. Источник: разработано авторами.

Опытная эксплуатация приложения проводилась на тестовом стенде с импортированным публичным репозиторием pallets/flask (около 3 800 коммитов, более 800 уникальных авторов с момента создания проекта). По окончании первичного сбора аналитические SQL-запросы возвращали непустые осмысленные результаты по всем поддерживаемым сценариям: инвентаризация отслеживаемых проектов, ранжированный список разработчиков по числу коммитов, распределение активности команды по месяцам, сравнение проектов по объёму изменений строк кода (line churn). Совместная работа всех компонентов системы — постановка задач серверным модулем, их асинхронное выполнение фоновыми исполнителями, обновление актуальности данных планировщиком, отдача агрегированных метрик клиентскому приложению — на тестовом стенде осуществляется устойчиво в течение всего периода наблюдения.

Выводы

Проведённая работа подтвердила возможность реализации полнофункционального аналитического веб-приложения над данными GitHub, развёртываемого исключительно в инфраструктуре заказчика и не требующего обращения к облачным сервисам сторонних поставщиков. Архитектурное разделение системы на три модуля — модуль сбора данных, серверный модуль с программным интерфейсом и клиентское веб-приложение — обеспечивает независимое масштабирование компонентов и упрощает их сопровождение и развитие. Использование колоночной СУБД ClickHouse одновременно в качестве аналитического хранилища и координационного слоя через таблицы движка KeeperMap позволяет отказаться от привлечения отдельного брокера сообщений и сократить общую топологию развёртывания.

Опытная эксплуатация на тестовом стенде показала работоспособность принятых архитектурных решений в условиях характерного для рассматриваемого класса задач профиля нагрузки. Очевидным ограничением подхода является масштабируемость очередей задач на движке KeeperMap: для систем с потоком в тысячи и более событий в секунду такое решение нецелесообразно — в подобных сценариях специализированные брокеры сообщений сохраняют преимущество. Предложенный подход применим прежде

всего в организациях, имеющих ограничения на использование облачных аналитических сервисов и заинтересованных в построении управленческой аналитики на основе данных собственной разработки.

Дальнейшие направления развития системы включают переход серверного модуля на асинхронный фреймворк FastAPI, внедрение более сложных метрик качества кода (индекс рефакторинга, индекс повторной работы), а также интеграцию с дополнительными источниками данных — системой управления задачами Jira и корпоративными средствами обмена сообщениями.

Список литературы:

1. Forsgren N. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations / N. Forsgren, J. Humble, G. Kim. – Portland: IT Revolution Press, 2018. – 288 p. – ISBN 978-1-94278-833-1.
2. Федеральный закон от 29.07.2004 № 98-ФЗ «О коммерческой тайне» (в действующей редакции) // Российская газета. – 2004. – 05 авг. – № 166.
3. Cosentino V. Gitana: A SQL-Based Git Repository Inspector / V. Cosentino, J. L. Cánovas Izquierdo, J. Cabot // Conceptual Modeling: 34th International Conference, ER 2015. – Lecture Notes in Computer Science, vol. 9381. – Cham: Springer, 2015. – Pp. 329–343. – DOI: 10.1007/978-3-319-25264-3_24.
4. ClickHouse documentation: Table Engines : [сайт]. – URL: https://clickhouse.com/docs/en/engines/table-engines (дата обращения: 25.05.2026). – Текст : электронный.
5. GitHub REST API documentation: Events : [сайт]. – URL: https://docs.github.com/en/rest/activity/events (дата обращения: 25.05.2026). – Текст : электронный.
6. GitHub REST API documentation: Rate limits for the REST API : [сайт]. – URL: https://docs.github.com/en/rest/using-the-rest-api/rate-limits-for-the-rest-api (дата обращения: 25.05.2026). – Текст : электронный.
7. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка : пер. с англ. / М. Клеппман. – СПб.: Питер, 2019. – 640 с. – (Серия «Бестселлеры O'Reilly»). – ISBN 978-5-4461-0512-9.
8. Ньюмен С. Создание микросервисов : пер. с англ. / С. Ньюмен. – 2-е изд. – СПб.: Питер, 2022. – 624 с. – ISBN 978-5-4461-1885-3.
9. ClickHouse documentation: KeeperMap Table Engine : [сайт]. – URL: https://clickhouse.com/docs/en/engines/table-engines/special/keepermap (дата обращения: 25.05.2026). – Текст : электронный.
10. Cosentino V. A Systematic Mapping Study of Software Development With GitHub / V. Cosentino, J. L. Cánovas Izquierdo, J. Cabot // IEEE Access. – 2017. – Vol. 5. – Pp. 7173–7192. – DOI: 10.1109/ACCESS.2017.2682323.

References:

1. Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. Portland: IT Revolution Press. ISBN 978-1-94278-833-1.

2. Federal Law of July 29, 2004 No. 98-FZ "On Commercial Secrets" (as amended). (2004). Rossiyskaya Gazeta, 166 (Aug. 5). (In Russian).
3. Cosentino, V., Cánovas Izquierdo, J. L., & Cabot, J. (2015). Gitana: A SQL-Based Git Repository Inspector. In Conceptual Modeling: 34th International Conference, ER 2015. Lecture Notes in Computer Science, Vol. 9381 (pp. 329–343). Cham: Springer. DOI: 10.1007/978-3-319-25264-3_24.
4. ClickHouse documentation: Table Engines. Available at: https://clickhouse.com/docs/en/engines/table-engines (accessed: 25.05.2026).
5. GitHub REST API documentation: Events. Available at: https://docs.github.com/en/rest/activity/events (accessed: 25.05.2026).
6. GitHub REST API documentation: Rate limits for the REST API. Available at: https://docs.github.com/en/rest/using-the-rest-api/rate-limits-for-the-rest-api (accessed: 25.05.2026).
7. Kleppmann, M. (2019). Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems (Russian translation). Saint Petersburg: Piter. ISBN 978-5-4461-0512-9. (In Russian).
8. Newman, S. (2022). Building Microservices, 2nd ed. (Russian translation). Saint Petersburg: Piter. ISBN 978-5-4461-1885-3. (In Russian).
9. ClickHouse documentation: KeeperMap Table Engine. Available at: https://clickhouse.com/docs/en/engines/table-engines/special/keepermap (accessed: 25.05.2026).
10. Cosentino, V., Cánovas Izquierdo, J. L., & Cabot, J. (2017). A Systematic Mapping Study of Software Development With GitHub. IEEE Access, 5, 7173–7192. DOI: 10.1109/ACCESS.2017.2682323.