DOI 10.37882/2223-2966.2025.02.08

ЭРГОНОМИЧЕСКИЙ АНАЛИЗ ВРЕМЕННЫХ ХАРАКТЕРИСТИК БИЗНЕС-ПРОЦЕССОВ В ПРИЛОЖЕНИИ REPORT.MS

ERGONOMIC ANALYSIS OF TIME CHARACTERISTICS OF BUSINESS PROCESSES IN THE REPORT.MS APPLICATION

B. Goryachkin
P. Martynova
K. Skvortsov

Summary. Despite the presence of many common characteristics and repetitive operations in corporate software, the process of its development, configuration, and adaptation to the needs of a particular organization is often labor-intensive and requires deep knowledge in programming. This creates the prerequisites for the emergence of corporate application designers — tools that are designed to automate and simplify this process. The problem of a comprehensive ergonomic analysis of the key functional capabilities of corporate application designers arises the methodology for calculating the execution time of business processes, to identify opportunities for its improvement and increase ease of use. The architecture of the Report.ms application was studied in detail, including the main components and their interaction. An analytical approach to assessing the execution time of various stages of the data processing process in the application was developed.

The time of the http request, work with the database, transfer and save data were calculated. This made it possible to assess the impact of each of these factors on the overall execution time of the process.

Using the example of a specific process, a step-by-step detailing of the calculation of the total execution time was performed. This demonstrated the practical application of the developed methodology and confirmed its effectiveness.

The results of the study can be applied in the design and development of other analytical tools for business, where the accuracy and convenience of the user interface are important for effective use.

Keywords: HTTP request, PostgreSQL, enterprise application designer, execution time, database.

Горячкин Борис Сергеевич
кандидат технических наук, доцент, Московский государственный технический университет им. Н.Э. Баумана
bsgor@mail.ru
Мартынова Полина Владимировна
Московский государственный технический университет им. Н.Э. Баумана
mpv19u539@student.bmstu.ru
Скворцов Кирилл Сергеевич
Программист, ФГБУ «НМИЦ эндокринологии»
sk.kirillg@gmail.com

Аннотация. Несмотря на наличие множества общих характеристик и повторяющихся операций в корпоративном программном обеспечении, процесс его разработки, настройки и адаптации под нужды конкретной организации зачастую является трудоемким и требует глубоких знаний в программировании. Это создает предпосылки для появления конструкторов корпоративных приложений — инструментов, которые призваны автоматизировать и упростить данный процесс. Возникает проблема комплексного эргономического анализа ключевых функциональных возможностей конструкторов корпоративных приложений, в частности методики расчета времени выполнения бизнес-процессов, с целью выявления возможностей её совершенствования и повышения удобства использования.

Была детально изучена архитектура приложения Report.ms, включая основные компоненты и их взаимодействие. Был разработан аналитический подход к оценке времени выполнения различных этапов процесса обработки данных в приложении.

Проведены расчеты времени http-запроса, работы с базой данных, передачи и сохранения данных. Это дало возможность оценить влияние каждого из этих факторов на общее время выполнения процесса.

На примере конкретного процесса была выполнена пошаговая детализация расчета общего времени его выполнения. Это продемонстрировало практическое применение разработанной методики и подтвердило ее эффективность.

Результаты исследования могут быть применены при проектировании и разработке других аналитических инструментов для бизнеса, где точность и удобство пользовательского интерфейса имеют важное значение для эффективного использования.

Ключевые слова: HTTP-запрос, PostgreSQL, конструктор корпоративных приложений, время выполнения, база данных.

Введение

Корпоративное программное обеспечение — это специализированные программные решения, разработанные для удовлетворения потребностей организаций и предприятий. Оно предназначено для управления различными аспектами бизнеса, такими как финансы, управление проектами, учет, продажи, CRM (управление взаимоотношениями с клиентами), ERP (планирование ресурсного предприятия) и многие другие. Корпоративное программное обеспечение играет ключевую роль в современном бизнесе, позволяя организациям повышать эффективность и конкурентоспособность. Оно охватывает широкий спектр бизнес-функций, от управления финансами и проектами до взаимодействия с клиентами и планирования ресур-

сов предприятия. Эти специализированные решения помогают компаниям автоматизировать рабочие процессы, повышать производительность труда и принимать более обоснованные управленческие решения на основе данных.

В большинстве корпоративных ПО есть общие элементы, такие как

— авторизация
— управление пользователями и правами доступа
— графический интерфейс (таблицы и т.д.)
— отчеты и аналитика

Типы внесения изменений также повторяются. К таким типам относятся:

— добавление поля в базу данных
— создание таблицы
— редактирование таблиц
— добавления графика для аналитики
— добавление новой роли пользователя

Эти базовые операции по расширению функционала применимы к широкому спектру корпоративных приложений, независимо от их конкретной отраслевой или бизнес-специфики. Такое повторяющееся ядро изменений и модификаций ПО создает предпосылки для появления конструкторов корпоративных приложений — инструментов, которые автоматизируют и упрощают процесс разработки, настройки и адаптации бизнес-программ под нужды конкретной организации.

Наличие множества общих характеристик породило то, что в данный момент на рынке стали появляться конструкторы корпоративных приложений. Конструкторы корпоративных приложений — это инструменты или платформы, позволяющие организациям быстро и эффективно разрабатывать приложения для внутреннего использования без необходимости глубоких знаний в программировании. Они особенно полезны для автоматизации бизнес-процессов, управления данными и повышения продуктивности.

Описание архитектуры конструктора корпоративных приложений Report.ms

Одним из таких конструкторов является Report.ms [1]. Его архитектура представлена на рис. 1.

Архитектура конфигуратора представляет собой модульную систему, объединяющую несколько компонентов, которые работают совместно для обеспечения функциональности и взаимодействия с пользователем. В центре архитектуры находится модуль конфигуратора, который включает три ключевых элемента: базу данных, бизнес-логику и пользовательский интерфейс.

Сам конфигуратор состоит из:

— База данных (PostgreSQL [2]):

База данных служит хранилищем для конфигурационных данных. PostgreSQL выбрана за свою надежность и производительность, что позволяет эффективно обрабатывать запросы и обеспечивать целостность данных. Здесь хранится информация, необходимая для работы конфигуратора, такая как настройки, параметры и другие необходимые данные.

— Бизнес-логика (Dotnet Core [3]):

Бизнес-логика написана на платформе Dotnet Core и выполняет роль посредника между базой данных и пользовательским интерфейсом. Этот компонент отвечает за обработку запросов от клиентской части, выполнение необходимых операций с данными (например, получение, обновление, удаление) и применение правил и алгоритмов, которые обеспечивают правильную работу конфигуратора. Таким образом, бизнес-логика реализует основные функции и правила, касающиеся обработки конфигурационных данных.

— Приложение (React [4]):

Интерфейсная часть, разработанная с использованием React, обеспечивает визуальное взаимодействие пользователей с системой. Пользователи могут вводить данные, смотреть текущие настройки и получать результаты в удобной форме. React позволяет создавать динамичные и отзывчивые интерфейсы, обеспечивая быстрое обновление контента без необходимости перезагрузки страницы. Это положительно сказывается на пользовательском опыте, делая его более интерактивным.

Ниже уровня конфигуратора располагается ИИ сервис, который также включает в себя базу данных и приложение на Dotnet Core:

— База данных (PostgreSQL):

Аналогично конфигуратору, здесь хранится информация, необходимая для работы ИИ-сервиса. Это может включать обучающие выборки, результаты анализа, модели и другие данные, которые могут быть использованы для предоставления интеллектуальных услуг.

— Приложение (Dotnet Core):

Это приложение также отвечает за обработку данных и бизнес-логику, специфичную для ИИ. Оно может анализировать входные данные, предоставлять интеллектуальные рекомендации или выполнять другие вы-

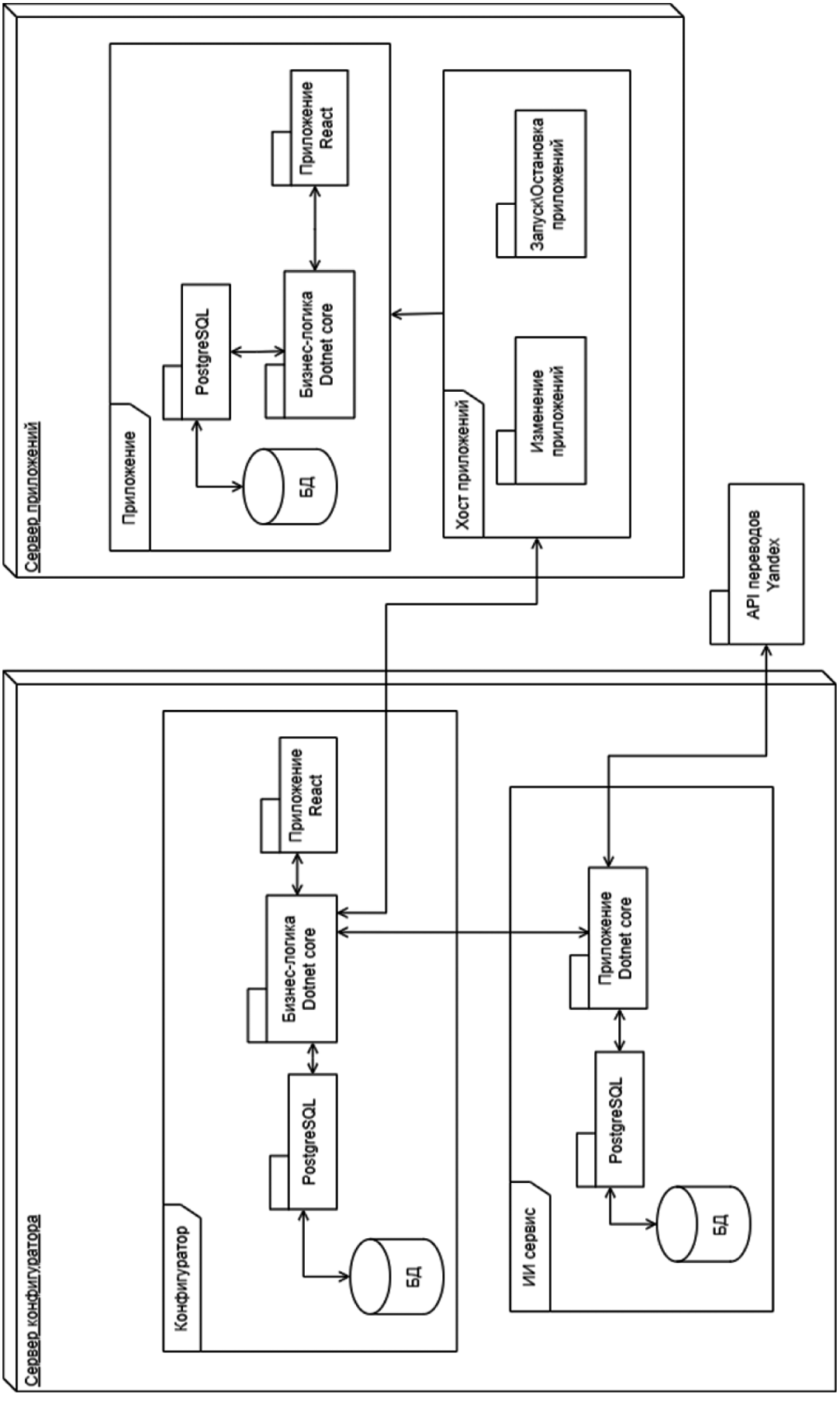

Рис. 1. Архитектура конструктора бизнес-приложений

числительные задачи. Взаимодействие с базой данных позволяет выполнять необходимые операции над данными, получая нужные результаты для дальнейшей обработки.

Система также имеет интеграцию с внешним API, в частности с API переводов Яндекс. Эта функция позволяет пользователю конфигуратора получать переводы, что особенно важно для того, чтобы переводить, например, названия полей и справочников, которые добавляет пользователь, для того. Взаимодействие с API может происходить в режиме реального времени, что расширяет функциональность конфигуратора и делает систему более гибкой и доступной для пользователей из разных языковых групп.

Схема также указывает на использование хостинга для развертывания приложений. Это может обеспечить доступность конфигуратора и ИИ-сервиса в интернете, позволяя пользователям взаимодействовать с системой через веб-браузеры.

Расчет времени http-запроса

В целом, как запросы HTTP [5], так и ответы имеют следующую структуру:

— Стартовая строка (start line) — используется для описания версии используемого протокола и другой информации — вроде запрашиваемого ресурса или кода ответа. Как можно понять из названия, ее содержимое занимает ровно одну строчку.
— HTTP-заголовки (HTTP Headers) — несколько строчек текста в определенном формате, которые либо уточняют запрос, либо описывают содержимое тела сообщения.
— Пустая строка, которая сообщает, что все метаданные для конкретного запроса или ответа были отправлены.
— Опциональное тело сообщения, которое содержит данные, связанные с запросом, либо документ (например HTML-страницу), передаваемый в ответе.

Для расчета времени выполнения http-запроса будем пользоваться следующей формулой:

$$t_{\text{передачи}} = \frac{I}{v} \quad (1)$$

Где $t_{\text{передачи}}$ — время передачи данных по каналу связи (с).
I — количество передаваемой информации (бит).
V — скорость передачи данных по каналу (бит/с)

Общее количество передаваемой информации состоит из количества информации в заголовке, стартовой строке и теле запроса, рассчитывается по формуле 2.

$$I = I_{\text{старт.строки}} + I_{\text{заголовка}} + I_{\text{тела}} \quad (2)$$

Расчет времени работы с базой данных

В PostgreSQL выполнение запроса состоит из следующих этапов [6]:

1. Парсер генерирует дерево разбора, которое может быть прочитано последующими подсистемами из SQL-запроса в виде обычного текста.
2. Анализатор выполняет семантический анализ дерева разбора, сгенерированного синтаксическим анализатором, и генерирует дерево запросов. Корнем дерева запросов является структура Query, определенная в parsenodes.h. Эта структура содержит метаданные соответствующего запроса, такие как тип этой команды (SELECT, INSERT или другие), и несколько листьев; каждый лист образует список или дерево и содержит данные отдельного конкретного пункта.
3. Переписчик — реализует систему правил и при необходимости преобразует дерево запросов в соответствии с правилами, хранящимися в каталоге системы pg_rules.
4. Планировщик получает дерево запросов от переписчика и генерирует дерево планов (запросов), которое может быть обработано исполнителем наиболее эффективно.

Планировщик в PostgreSQL основан на чистой оптимизации на основе затрат. Он не поддерживает оптимизацию на основе правил и подсказок. Оптимизация запросов в PostgreSQL основана на стоимости. Затраты являются безразмерными величинами, и это не абсолютные показатели производительности, а показатели для сравнения относительной производительности операций.

Затраты оцениваются функциями, определенными в файле costize.c. Все операции, выполняемые исполнителем, имеют соответствующие функции затрат. Например, затраты на последовательное сканирование и сканирование индекса оцениваются функциями cost_seqscan() и cost_index(), соответственно.

В PostgreSQL существует три вида затрат: начальные, текущие и общие. Общая стоимость — сумма затрат на запуск и выполнение. Таким образом, независимо оцениваются только затраты на запуск и выполнение.

Затраты на запуск — это затраты, понесенные до получения первого кортежа. Например, начальная стоимость узла сканирования индекса — это стоимость чтения индексных страниц для доступа к первому кортежу в целевой таблице. Стоимость выполнения — это стоимость получения всех кортежей. Общая же стоимость

определяется суммой стоимости запуска и стоимости выполнения.

Стоимость последовательного сканирования, то есть стандартного запроса select, оценивается функцией cost_seqscan ().

При последовательном сканировании стоимость запуска равна 0, а стоимость выполнения определяется следующим выражением:

$$T = \left(\left(t_{\text{корт}} + t_{\text{оп}}N_{\text{оп}}\right)N_{\text{корт}} + S_{\text{стр}}N_{\text{стр}}\right) \times \times k_{\text{комп}}\left(S_s + t_{\text{ст}}N_{\text{ст}}\right) \quad (3)$$

где T — общее время выполнения запроса, $t_{\text{корт}}$ — стоимость сканирования строки, по умолчанию 0,01. Данная стоимость берется из настроечного файла costize.c;

$t_{\text{оп}}$ — стоимость одного оператора, по умолчанию 0,0025. Данная стоимость берется из настроечного файла costize.c;

$N_{\text{оп}}$ — количество операторов, то есть количество столбцов с условиями;

$N_{\text{корт}}$ — количество кортежей;

$S_{\text{стр}}$ — количество страниц;

$N_{\text{стр}}$ — стоимость сканирования страницы, по умолчанию 1;

$t_{\text{ст}}$ — эмпирический коэффициент стоимости одного столбца, для данной конфигурации PostgreSQL 0,0186;

S_s — начальная стоимость выполнения поиска по столбцам, для данной конфигурации PostgreSQL 0,9814,

$N_{\text{ст}}$ — количество столбцов в таблице;

$k_{\text{комп}}$ — коэффициент производительности компьютера, для устройства, на котором проводился эксперимент — 0,0003. Коэффициент может варьироваться в зависимости от производительности системы. Например, компьютер с процессором Pentium имеет порядок коэффициента 0.01, хотя для современных серверных решений, данный коэффициент может достигать 10–6.

Количество страниц $S_{\text{стр}}$, на которых расположена база данных, не может быть посчитано вручную, однако данные могут быть получены с помощью запроса PostgreSQL. Запрос, приведенный ниже, выводит информацию об количестве страниц, на которых расположена определенная таблица:

SELECT relpages, reltuples FROM pg_class WHERE relname = 'data'

Расчет времени для процесса добавления нового поля в справочник

Рассмотрим процесс добавления нового поля в справочник. Схематично он представлен на рис. 2.

Рассматриваемый процесс — добавление нового поля в справочник.

Данный процесс состоит из нескольких шагов, каждый из которых является либо http-запросом (или ответом), либо запросом в базу данных.

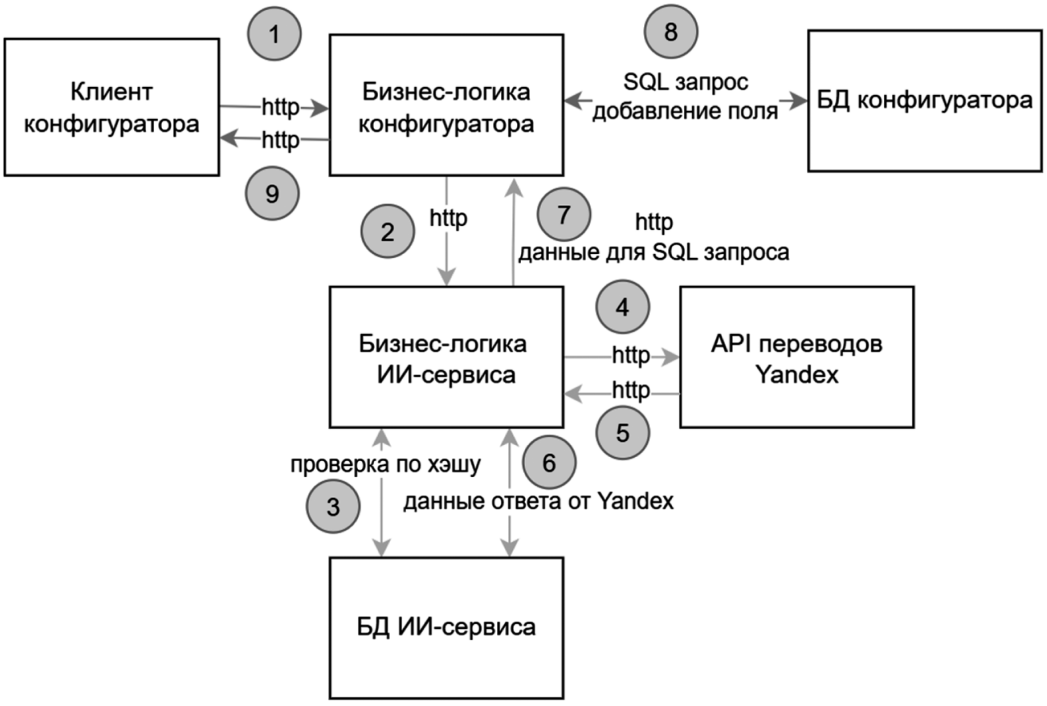

Рис. 2. Схема информационного потока для процесса добавления нового поля в справочник

Для наглядности запишем все данные для расчета времени http-запросов в таблицу 1, время работы с базой данных для каждого шага в таблицу 2.

Таблица 1.

Время передачи данных в http-запросах и ответах

шаг	стартовая строка, байт	заголовок, байт	тело, байт	I, байт	I, бит	t, мсек
1	49	1342	1086	2477	19816	0,0566
2	45	1124	118	1287	10296	0,0021
4	67	1342	113	1522	12176	0,0348
5	15	271	129	415	3320	0,0095
7	15	271	69	355	2840	0,0006
9	15	271	159	445	3560	0,0102

Таблица 2.

Время работы с базой данных

шаг	$t_{корт}$	$t_{оп}$	$N_{оп}$	$N_{корт}$	$S_{стр}$	$N_{стр}$	$t_{ст}$	S_s	$N_{ст}$	$k_{комп}$	T, мсек
3	0,01	0,0025	2	705	16	1	0,0186	0,9814	6	0,0003	0,009

Суммируем всё время и сравним с полученным практически. Теоретическим путем было получено время: 497 мксек, практическим путем — 670 мксек. Расчет соизмерим с практически полученным значением.

Проведение эксперимента для ряда схожих процессов

Произведем тот же эксперимент для добавления других полей в справочник. Результаты занесем в таблицу 3.

Таблица 3.

Измерение времени добавления полей в справочник

Название поля	$T_{теор.}$, мкс	$T_{практ.}$, мкс
Номер заказа	497	670
Статус	525	680
Дата заказа	553	750
Комментарий	581	690

Построим на основе полученных результатов гистограмму. Она приведена на рисунке 3.

Заключение

Была детально изучена архитектура приложения Report.ms, включая основные компоненты и их взаимодействие. Был разработан аналитический подход к оценке времени выполнения различных этапов процесса обработки данных в приложении.

Проведены расчеты времени http-запроса, работы с базой данных, передачи и сохранения данных. Это дало возможность оценить влияние каждого из этих факторов на общее время выполнения процесса.

На примере конкретного процесса была выполнена пошаговая детализация расчета общего времени его выполнения. Это продемонстрировало практическое применение разработанной методики и подтвердило ее эффективность.

Для ряда схожих процессов был проведен экспериментальный анализ фактического времени выполнения. Сравнение расчетных и экспериментальных данных показало высокую точность предложенного аналитического подхода.

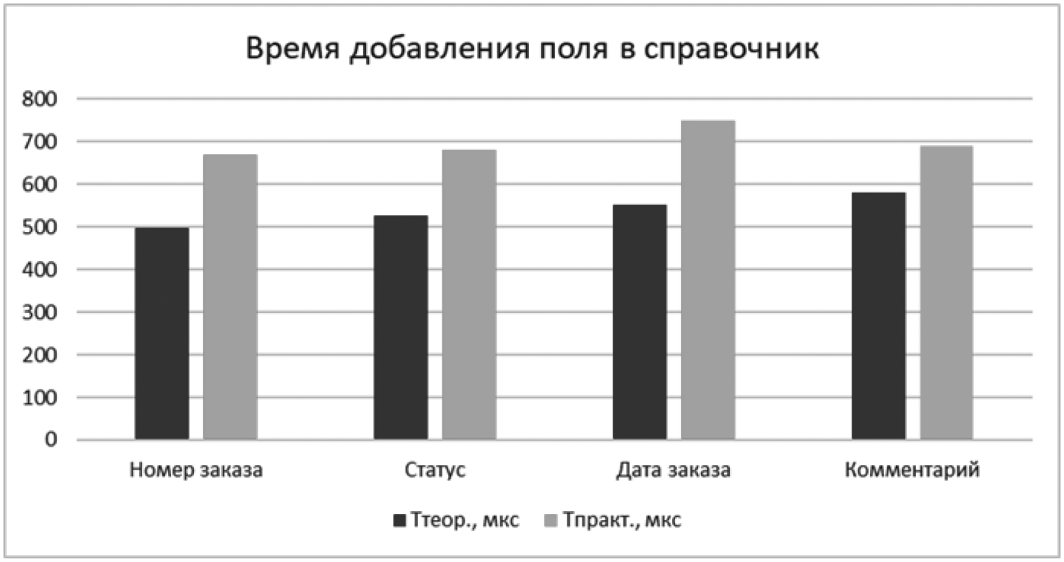

Рис. 3. Время добавления поля в справочник

ЛИТЕРАТУРА

1. Report.ms Конструктор корпоративных приложений. URL: https://report.ms/ (дата обращения: 30.11.2024).
2. PostgreSQL. URL: https://www.postgresql.org/ (дата обращения: 30.11.2024).
3. Введение в .NET Core. Microsoft. URL: https://learn.microsoft.com/ru-ru/dotnet/core/introduction (дата обращения: 30.11.2024).
4. React. Legacy. URL: https://ru.legacy.reactjs.org/ (дата обращения: 30.11.2024).
5. HTTP-запрос. Selectel. URL: https://selectel.ru/blog/http-request/ (дата обращения: 30.11.2024).
6. Гудилин Д.С., Горячкин Б.С., Звонарев А.Е. Анализ производительности реляционных баз данных.

© Горячкин Борис Сергеевич (bsgor@mail.ru); Мартынова Полина Владимировна (mpv19u539@student.bmstu.ru);
Скворцов Кирилл Сергеевич (sk.kirillg@gmail.com)
Журнал «Современная наука: актуальные проблемы теории и практики»