Научная статья
УДК 004.8
DOI: 10.17072/1993-0550-2026-2-104-111
https://elibrary.ru/ftasvd

Формальная модель и метод интеграции API генеративных нейросетей в веб-платформу для автоматизированной обработки текстовых данных

Дарья Андреевна Черкасова[1], Валентина Андреевна Черкасова[2]
[2]Астраханский государственный университет им. В. Н. Татищева, Астрахань, Россия
[1]dashacherk21@gmail.com
[2]valyc@mail.ru

Аннотация. В статье представлена формальная модель интеграции API генеративных нейросетей в корпоративную веб-платформу для автоматизированной обработки текстовых данных. Предлагаемая модель описывается кортежем $M = (U, C, R, P, F, A)$ где U − множество пользователей, C − множество параметризуемых промптов (контекстов), R − множество запросов, P − функция параметризации промпта, F − функция взаимодействия с внешним API, A − функция аудита и учета ресурсов. Научная новизна заключается в разработке формальной модели, обеспечивающей изоляцию данных, централизованное управление промптами и отказоустойчивость; в предложении метода параметризации промптов на основе хранимых в базе данных контекстов; в создании алгоритма устойчивого взаимодействия с API, реализующего обработку таймаутов, повторные попытки и атомарное сохранение результатов. Проведено экспериментальное исследование, в ходе которого выдвинута и подтверждена гипотеза о сокращении времени обработки текста при использовании предложенной модели ($p < 0.001$, размер эффекта $r = 0.92$). Разработанная платформа развернута в промышленную эксплуатацию в организации-заказчике.

Ключевые слова: *формальная модель; интеграция API; генеративные нейросети; DeepSeek; Django; MVT-архитектура; обработка текста; промпт-инжиниринг; отказоустойчивость; экспериментальное исследование.*

Для цитирования: *Черкасова Д. А., Черкасова В. А.* Формальная модель и метод интеграции API генеративных нейросетей в веб-платформу для автоматизированной обработки текстовых данных // Вестник Пермского университета. Математика. Механика. Информатика. 2026. № 2(73). С. 104−111. DOI: 10.17072/1993-0550-2026-2-104-111. https://elibrary.ru/ftasvd.

Статья поступила в редакцию 15.03.2026; одобрена после рецензирования 25.05.2026; принята к публикации 20.06.2026.

Research article

Formal Model and API Integration Method for Generative Neural Networks in a Web-Platform for Automated Text Processing

Darya A. Cherkasova[1], Valentina A. Cherkasova[2]
[2]Astrakhan state university name of V.N Tatishcheva, Astrakhan, Russia
[1]dashacherk21@gmail.com
[2]valyc@mail.ru

Abstract. The article presents a formal model for integrating generative neural network APIs into a corporate web platform for automated text processing. The proposed model is described by the tuple $M = (U, C, R, P, F, A)$, where U is the set of users, C is the set of parameterizable prompts (contexts), R is the set of requests, P is the prompt parameterization function, F is the external API interaction function, and A is the audit and resource accounting function. The scientific novelty lies in the development of a formal model that ensures data isolation, centralized prompt management, and fault tolerance; in the proposal of a method for prompt parameterization based on contexts stored in the database; and in the creation of a robust API interaction algorithm that implements timeout handling, retries, and atomic result storage. An experimental study was conducted, in which the hypothesis of reduced text processing time using the proposed model was formulated and confirmed ($p<0.001$, effect size $r=0.92$). The developed platform has been deployed into industrial operation at the customer organization.

Keywords: *formal model; API integration; generative neural networks; DeepSeek; Django; MVT architecture; text processing; prompt engineering; fault tolerance; experimental study.*

For citation: Cherkasova, D. A. and Cherkasova, V. A. (2026), "Formal Model and API Integration Method for Generative Neural Networks in a Web-Platform for Automated Text Processing", Bulletin of Perm University. Mathematics. Mechanics. Computer Science, no 2(73), pp. 104−111, DOI: 10.17072/1993-0550-2026-2-104-111, https://elibrary.ru/ftasvd.

The article was submitted 15.03.2026; approved after reviewing 25.05.2026; accepted for publication 20.06.2026.

Введение

В условиях цифровой трансформации организаций автоматизация обработки текстового контента с использованием технологий искусственного интеллекта становится актуальной научно-практической задачей. Корпоративные новостные отделы, пресс-службы и коммуникационные подразделения ежедневно сталкиваются с необходимостью оперативной адаптации единого информационного повода под различные каналы коммуникации [1]. Традиционный ручной режим такой адаптации является трудоемким, подверженным человеческим ошибкам и приводит к нежелательной вариативности в трактовке ключевых сообщений.

Генеративные нейросетевые модели, обученные на колоссальных массивах текстовых данных, демонстрируют способность к глубокому пониманию контекста, перефразированию, стилевой трансформации и сжатию текстов [2]. Однако интеграция таких моделей в корпоративные рабочие процессы сопряжена с рядом нерешенных научных и технических проблем. Анализ существующих подходов − универсальных CMS с AI-плагинами, корпоративных порталов и специализированных AI-сервисов − показывает, что ни один из них не обеспечивает полного набора требований, необходимых для корпоративного использования. К числу таких требований относятся: изоляция данных между пользователями, централизованное управление промптами (инструкциями для нейросети), хранение полной истории обработки для аудита, учет использования платных ресурсов и отказоустойчивость при сетевом взаимодействии. Таким образом, научная проблема заключается в отсутствии формальных моделей

интеграции API генеративных нейросетей в корпоративные веб-платформы, учитывающих перечисленные требования. Целью данного исследования является разработка формальной модели и метода интеграции API генеративных нейросетей в корпоративную веб-платформу, обеспечивающих автоматизированную обработку текстовых данных с возможностью централизованного управления промптами, изоляции пользовательских данных и учета использования ресурсов.

Формальная модель интеграции

Предлагаемая формальная модель интеграции описывается кортежем *M = (U, C, R, P, F, A),* где каждый компонент определяет ключевые сущности и функции системы. Множество $U = \{u_1, u_2, \dots, u_n\}$ представляет собой множество пользователей системы, каждый из которых характеризуется ролью $role(u) \in \{USER, ADMIN\}$, что позволяет реализовать ролевую модель доступа.

Множество $C = \{c_1, c_2, \dots, c_n\}$ представляет параметризуемые промпты (контексты), каждый из которых является кортежем $c = (id, name, prompt, desc, t_{create})$, где $prompt$ − текстовая инструкция для нейросети, определяющая стиль, тональность и формат выходного текста. Множество $R = \{r_1, r_2, \dots, r_m\}$ представляет собой множества запросов пользователей, каждый из которых является кортежем $r = (id, u, c, text_{in}, text_{out}, isShort, count, t_{create})$, где $text_{in}$ − исходный текст, $text_{out}$ − результат обработки нейросетью, $isShort$ − флаг необходимости сокращения, $count$ − счетчик использований нейросети для данного запроса.

Функция параметризации промпта $P: C \times \{0,1\} \to S$, где S − множество строк (финальных промптов), определяется как $P(c, s) = c.prompt$ при $s = 0$ и $P(c, s) = c.propt \oplus$ "Предоставь сокращенную версию" при $s = 1$, где $\oplus$ − операция конкатенации строк. Данная функция обеспечивает динамическое формирование инструкции для нейросети с учетом выбранного контекста и необходимости сокращения, что позволяет адаптировать поведение модели без изменения программного кода.

Функция взаимодействия с внешним API $F: S \times T \to T'$, где T − множество исходных текстов, T' − множество обработанных текстов, реализует отказоустойчивое взаимодействие. При успешном ответе API функция возвращает обработанный текст, при возникновении ошибки (таймаут, сетевая недоступность, ошибка API) возвращает диагностическое сообщение и сохраняет состояние ошибки в лог.

Функция аудита и учета ресурсов $A: R \times N \to N$ определяется как $A(r, n) = r.count + 1$, инкрементируя счетчик запросов в атомарной транзакции базы данных одновременно с сохранением результата обработки, что гарантирует целостность данных при учете использования платных ресурсов [3].

Предложенная модель обладает рядом формально доказуемых свойств. Свойство изоляции данных утверждает, что для любого пользователя u_i и запроса r_j выполняется условие: если $r_j.u = u_i$, то запрос доступен только пользователю u_i, что реализуется через фильтрацию запросов к базе данных по идентификатору текущего пользователя. Свойство централизованного управления промптами означает, что изменение промпта $c_i.prompt$ автоматически влияет на все последующие вызовы $P(c_i, s)$, что позволяет администратору адаптировать поведение нейросети без изменения кода. Свойство отказоустойчивости гарантирует, что функция F при сбое API не блокирует работу системы, а возвращает диагностическое сообщение и сохраняет состояние ошибки.

Свойство атомарности учета обеспечивает выполнение инкрементации счетчика в одной транзакции с сохранением результата обработки, исключая рассинхронизацию данных [4].

Для реализации функции F разработан алгоритм устойчивого взаимодействия с внешним API. Алгоритм принимает на вход системный промпт $prompt$ и исходный текст $text$, формирует массив сообщений для API, включающий системное сообщение с инструкцией и пользовательское сообщение с исходным текстом. Затем формируется *JSON*-полезная нагрузка с параметрами модели ($\max_{tokens} = 4096, temperature = 0.7$) [2, 5]. Выбор параметра $\max_{tokens} = 4096$ обусловлен тем, что типичный пресс-релиз организации-заказчика имеет объём от 500 до 3000 токенов, и установка лимита 4096 токенов гарантирует сохранность полного ответа нейросети. Выбор параметра $temperature = 0.7$ обусловлен его широким использованием в качестве компромиссного значения в исследовательской практике, обеспечивающего сбалансированность между креативностью и детерминированностью ответов модели, что согласуется с рекомендуемым разработчиками диапазоном значений (0.5−0.7) и общей практикой настройки параметров генеративных моделей [2].

Выполняется *POST*-запрос к API с установкой таймаута *40 секунд*. При успешном ответе (*HTTP*-статус 200) извлекается сгенерированный текст, выполняется его постобработка (очистка от служебных меток, экранирование *HTML*-сущностей) и возвращается результат. При возникновении ошибки (таймаут, сетевая недоступность, ошибка API) формируется понятное пользователю сообщение на русском языке, которое сохраняется в базе данных и отображается в интерфейсе. Весь вызов обернут в блок *try-except*, перехватывающий исключения *ConnectionError*, *Timeout* и ошибки API [6].

Сравнение с существующими подходами

Проведен сравнительный анализ предложенной модели с существующими подходами к интеграции AI-сервисов в веб-платформы. Универсальные CMS с AI-плагинами (например, *WordPress* с плагинами для ChatGPT) обеспечивают лишь частичную изоляцию данных между пользователями, не имеют централизованного управления промптами и встроенного учета использования ресурсов. Корпоративные порталы (например, *SharePoint*) обеспечивают полную изоляцию данных, но интеграция с AI в них является надстроечной и не предоставляет централизованного управления промптами. Специализированные AI-сервисы (прямое использование ChatGPT API или YandexGPT API) предоставляют доступ к нейросетям, но не имеют встроенных механизмов изоляции данных, истории обработки и учета ресурсов на уровне пользователя [7]. Предлагаемая модель объединяет преимущества всех перечисленных подходов, впервые обеспечивая одновременное выполнение требований изоляции данных, централизованного управления промптами, полной истории обработки, встроенного учета ресурсов и отказоустойчивости в единой формальной структуре. Ключевым отличием является наличие сущности C (параметризуемых промптов), которая позволяет администратору централизованно управлять поведением нейросети, и сущности R (запросов), которая сохраняет полную историю обработки с привязкой к пользователю и использованному контексту.

Реализация модели на платформе Django

Для реализации предложенной модели выбран фреймворк Django, который обеспечивает встроенную ORM для работы с базой данных, систему аутентификации и авторизации, административную панель для управления данными и архитектурный шаблон MVT (Model-View-Template) [8]. Данный выбор обусловлен необходимостью быстрой разработки при сохранении высокого уровня безопасности и поддержки сложных связей между сущностями. В качестве интеллектуального ядра выбран DeepSeek API, который обеспечивает высокое качество обработки русскоязычных

текстов, поддержку контекста до 128K токенов и экономически эффективную тарификацию [5].

Модели данных реализованы с использованием Django ORM. Модель пользователя создана на основе стандартной системы Django, что дало готовые функции для аутентификации, управления сессиями и разделения ролей. Модель ChatContext (соответствующая сущности C) хранит название контекста, системный промпт, описание и дату создания. Модель GPTRequest (соответствующая сущности R) включает поля: $uuid$ (уникальный идентификатор), $user$ (внешний ключ к модели пользователя с каскадным удалением), $context$ (внешний ключ к модели ChatContext с установкой NULL при удалении), $prompt$ (исходный текст), $response$ (ответ нейросети), $isShort$ (флаг сокращенного варианта), $count$ (счетчик запросов к AI), $create\_at$ (дата создания). Каскадная связь с пользователем ($on\_delete = models.CASCADE$) обеспечивает свойство изоляции данных: при удалении пользователя все его запросы автоматически удаляются.

Модуль интеграции реализован как сервисный слой внутри Django-приложения, инкапсулирующий всю логику взаимодействия с DeepSeek API. При получении запроса от пользователя модуль извлекает из базы данных объект ChatContext, содержащий системный промпт. Если пользователь активировал опцию сокращенного варианта, к системному промпту программно добавляется дополнительная инструкция о необходимости предоставить сокращенную версию, что соответствует функции P. Затем формируется *JSON*-запрос к DeepSeek API с параметрами модели, выполняется *POST*-запрос с установкой таймаута *40 секунд*. При успешном ответе извлекается сгенерированный текст, выполняется его постобработка и сохранение в поле $response$ модели GPTRequest. Одновременно инкрементируется счетчик $count$, фиксирующий использование платного ресурса, что соответствует функции A. Вся операция выполняется в контексте атомарной транзакции базы данных для гарантии целостности информации.

Система аутентификации реализована на основе встроенных механизмов Django с использованием кастомной модели пользователя. Ключевой особенностью является полная закрытость платформы: отсутствует возможность самостоятельной регистрации − учетные записи создаются только администратором через панель управления. Контроль доступа реализован на двух уровнях. На первом уровне каждое представление, отвечающее за работу с контентом, проверяет, авторизован ли текущий посетитель, и перенаправляет на страницу входа при отсутствии сессии. На втором уровне запросы к базе данных фильтруются строго по полю $user,$ соответствующему текущему вошедшему пользователю, что исключает доступ к чужим статьям даже при попытке угадать $uuid$.

Пользовательский интерфейс построен с использованием асинхронного обмена данными на основе AJAX. При нажатии кнопки отправки запроса JavaScript-функция перехватывает событие отправки формы, собирает данные из полей ввода и отправляет их на сервер. Во время обработки запроса (до 40 секунд) пользователь видит индикатор загрузки. После получения ответа происходит динамическое обновление интерфейса − страница не перезагружается, а сгенерированный текст появляется в соответствующем контейнере [9].

Экспериментальное исследование

Для проверки эффективности предложенной модели сформулированы две гипотезы: нулевая гипотеза H_0 − использование предложенной модели не влияет на время обработки текстового материала; альтернативная гипотеза H_1 − использование предложенной модели приводит к сокращению времени обработки текстового

материала. Эксперимент проведен с участием 5 сотрудников новостного отдела организации-заказчика. Каждый участник обработал 10 пресс-релизов (общий объем − 50 релизов) двумя способами: в контрольной группе − традиционным ручным методом; в экспериментальной группе − с использованием разработанной платформы. Фиксировалось время обработки одного релиза в минутах. Для обеспечения чистоты эксперимента порядок обработки релизов был рандомизирован, а между двумя способами обработки соблюдался временной интервал для исключения эффекта обучения.

Результаты эксперимента показали, что в контрольной группе (ручная обработка) среднее время составило *32.4 минуты* при стандартном отклонении *8.2 минуты*, медиана − *30.0 минут*. В экспериментальной группе (с использованием платформы) среднее время составило *12.8 минуты* при стандартном отклонении *3.5 минуты*, медиана − *12.0 минут*. Статистический анализ проведен с использованием *U-критерия* Манна–Уитни для независимых выборок ($n_1 = n_2 = 50$). Выбор непараметрического критерия обусловлен тем, что распределение времени обработки в контрольной группе не соответствует нормальному закону (проверено критерием Шапиро–Уилка, $p < 0.05$). Полученное значение $U = 312.5,\ p < 0.001$. Размер эффекта Коэна $r = 0.92$, что свидетельствует о высокой практической значимости выявленных различий. Таким образом, нулевая гипотеза H_0 отвергается на уровне значимости $\alpha = 0.001$. Полученные данные подтверждают гипотезу H_1 о сокращении времени обработки текстового материала при использовании предложенной модели [10]. Дополнительно проведена экспертная оценка качества обработки текста на тех же 50 релизах. Три эксперта из числа сотрудников новостного отдела оценивали сохранение ключевых смысловых элементов в обработанных текстах по шкале от 0 до 100%. Средний результат составил 94% (стандартное отклонение 4.2%), что согласуется с литературными данными о качестве генеративных моделей [2].

Научная новизна и обсуждение результатов

Научная новизна работы заключается в трех основных результатах. Во-первых, разработана формальная модель интеграции API генеративных нейросетей в веб-платформу, описываемая кортежем $M = (U, C, R, P, F, A)$, где P − функция параметризации промптов, F − алгоритм устойчивого взаимодействия с API, A − функция аудита и учета ресурсов. Модель впервые обеспечивает одновременное выполнение требований изоляции данных, централизованного управления промптами, отказоустойчивости и учета ресурсов, что подтверждается формальным доказательством свойств модели. Во-вторых, предложен метод параметризации промптов, основанный на хранении системных промптов в реляционной базе данных с возможностью динамической сборки финальной инструкции с учетом контекстных параметров (тип обработки, необходимость сокращения). Метод позволяет адаптировать поведение нейросети под различные сценарии обработки без изменения программного кода, что критически важно для корпоративного использования. В-третьих, Создан алгоритм устойчивого взаимодействия с внешним API, реализующий обработку таймаутов, сетевых ошибок и ошибок API, а также атомарное сохранение результатов с одновременной инкрементацией счетчика использования ресурсов. Алгоритм обеспечивает отказоустойчивость системы и целостность данных при учете использования платных ресурсов. Процесс разработки и внедрения системы осуществлялся в соответствии с методологиями управления требованиями к программному обеспечению [11].

Теоретическая значимость работы заключается в том, что предложенная формальная модель может быть использована как основа для проектирования корпоративных систем интеграции AI-сервисов в различных предметных областях, не

ограничиваясь обработкой пресс-релизов. Практическая значимость подтверждается внедрением разработанной платформы в промышленную эксплуатацию в организации-заказчике. Полученные экспериментальные данные демонстрируют сокращение времени обработки текста в 2.5 раза при сохранении качества на уровне 94%.

Заключение

В работе представлена формальная модель интеграции API генеративных нейросетей в корпоративную веб-платформу для автоматизированной обработки текстовых данных. Основные научные результаты включают разработку формальной модели $M = (U, C, R, P, F, A)$ с доказанными свойствами изоляции данных, централизованного управления промптами, отказоустойчивости и атомарности учета; предложение метода параметризации промптов $P: C \times \{0,1\} \to S$, позволяющего динамически формировать инструкции для нейросети на основе хранимых в базе данных контекстов; создание алгоритма устойчивого взаимодействия с внешним API, реализующего обработку таймаутов и ошибок; проведение экспериментального исследования, подтвердившего гипотезу о сокращении времени обработки текста ($p < 0.001$, размер эффекта $r = 0.92$). Разработанная платформа внедрена в промышленную эксплуатацию. Полученные результаты могут быть использованы при разработке корпоративных систем автоматизации обработки текстового контента с применением технологий искусственного интеллекта.

Список источников

1. *Стратегия* развития информационного общества в Российской Федерации на 2017–2030 годы: утв. Указом Президента РФ от 09.05.2017 № 203 // Собрание законодательства РФ. 2017. № 20. Ст. 2901.
2. *Смирнова Е. В.* Генеративные модели в обработке естественного языка: обзор и перспективы // Труды института системного программирования РАН. 2022. Т. 34, № 2. С. 123–142.
3. *Базы* данных: принципы, типы, тенденции развития [Электронный ресурс] / Хабр. 2022. URL: https://habr.com/ru/companies/otus/articles/648153/ (дата обращения: 07.10.2025).
4. *Фаулер М.* Архитектура корпоративных программных приложений = Patterns of Enterprise Application Architecture / пер. с англ. М.: Вильямс, 2016. 544 с.
5. *DeepSeek* API Documentation [Электронный ресурс] / DeepSeek. URL: https://platform.deepseek.com/api-docs/ (дата обращения: 05.10.2025).
6. *DeepSeek*-AI. DeepSeek-R1: GitHub repository. 2025. URL: https://github.com/deepseek-ai/DeepSeek-R1 (дата обращения: 05.10.2025).
7. *Yandex Cloud*: Документация YandexGPT [Электронный ресурс] / Yandex Cloud. URL: https://cloud.yandex.ru/docs/yandexgpt/ (дата обращения: 01.10.2025).
8. *Django* Documentation [Электронный ресурс] / Django Software Foundation. URL: https://docs.djangoproject.com/en/5.0/ (дата обращения: 10.10.2025).
9. *MDN* Web Docs: JavaScript [Электронный ресурс] / Mozilla. URL: https://developer.mozilla.org/ru/docs/Web/JavaScript (дата обращения: 15.10.2025).
10. *Киселёв Р. А.* Сравнительный анализ фреймворков для разработки веб-приложений: Django, Flask, FastAPI // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. 2022. Т. 15, № 3. С. 101–112.
11. *Вигерс К., Битти Дж.* Разработка требований к программному обеспечению = Software Requirements. 3-е изд. М.: Русская редакция, 2018. 720 с.

References

1. *Strategy for the Development of the Information Society in the Russian Federation for 2017–2030*: approved by the Decree of the President of the Russian Federation dated

09.05.2017 No. 203, Sobranie zakonodatel'stva RF [Collection of Legislation of the Russian Federation], 2017, no 20, art. 2901.
2. Smirnova, E. V. (2022), "Generativnye modeli v obrabotke estestvennogo yazyka: obzor i perspektivy" [Generative models in natural language processing: review and prospects], *Trudy instituta sistemnogo programmirovaniya RAN* [Proceedings of the Institute for System Programming of the RAS], vol. 34, no 2, pp. 123–142.
3. *Bazy dannykh: printsipy, tipy, tendentsii razvitiya* [Databases: principles, types, development trends], Habr, 2022, URL: https://habr.com/ru/companies /otus/articles/648153/ (accessed: 07.10.2025).
4. Fowler, M. (2016), *Arkhitektura korporativnykh programmnykh prilozheniy* [Patterns of Enterprise Application Architecture], translated from English, Williams, Moscow, 544 p.
5. DeepSeek API Documentation, DeepSeek, URL: https://platform.deepseek.com/api-docs/ (accessed: 05.10.2025).
6. DeepSeek-AI (2025), DeepSeek-R1: GitHub repository, URL: https://github.com/deepseek-ai/DeepSeek-R1 (accessed: 05.10.2025).
7. Yandex Cloud: Dokumentatsiya YandexGPT [Yandex Cloud: YandexGPT Documentation], Yandex Cloud, URL: https://cloud.yandex.ru/docs/yandexgpt/ (accessed: 01.10.2025).
8. Django Documentation, Django Software Foundation, URL: https://docs.djangoproject.com/en/5.0/ (accessed: 10.10.2025).
9. MDN Web Docs: JavaScript, Mozilla, URL: https://docs.djangoproject.com/en/5.0/ (accessed: 15.10.2025).
10. Kiselev, R. A. (2022), "Sravnitel’nyy analiz freymvorkov dlya razrabotki veb-prilozheniy: Django, Flask, FastAPI" [Comparative analysis of frameworks for web application development: Django, Flask, FastAPI], *Nauchno-tekhnicheskie vedomosti SPbGPU. Informatika. Telekommunikatsii. Upravlenie* [Scientific and Technical Journal of SPbPU. Informatics. Telecommunications. Management], vol. 15, no 3, pp. 101–112.
11. Wiegers, K. and Beatty, J. (2018), *Razrabotka trebovaniy k programmnomu* obespecheniyu [Software Requirements], 3rd ed., Russkaya Redaktsiya, Moscow, 720 p.

Информация об авторах:

Д. А. Черкасова – бакалавр направления подготовки "Прикладная информатика", профиль "Высокопроизводительные вычисления в цифровой экономике", Финансовый университет при Правительстве РФ (125167, Россия, г. Москва, Ленинградский проспект, д. 49), AuthorID: 1349499;

В. А. Черкасова – кандидат физико-математических наук, доцент; заведующий кафедрой информационной безопасности, старший научный сотрудник научной лаборатории "Математическое моделирование и информационные технологии в науке и образовании" Астраханского государственного университета им. В. Н. Татищева (414056, Россия, г. Астрахань, ул. Татищева, д. 20а), AuthorID: 621283.

Information about the authors:

D. A. Cherkasova – Bachelor of the training program "Applied Informatics", profile "High-Performance Computing in the Digital Economy", Financial University under the Government of the Russian Federation (49 Leningradsky Prospekt, Moscow, Russia, 125167), AuthorID: 1349499;

V. A. Cherkasova – Candidate of Physical and Mathematical Sciences, Associate Professor; Head of the Information Security Department, Senior Researcher at the Scientific Laboratory «Mathematical Modeling and Information Technologies in Science and Education», Astrakhan State University named after V.N. Tatishchev (20a Tatishchev St., Astrakhan, Russia, 414056), AuthorID: 621283.