Personal Data Masking through local LLM

Created by	Михаэль Зайдман
Created time	@October 6, 2025 11:09 PM
Category	Tech
Last edited by	Михаэль Зайдман
Last updated time	@October 6, 2025 11:13 PM
Related project	

Ниже — техническая инструкция для разработчиков, описывающая реализацию промежуточного слоя (middleware) для обнаружения, маскировки (tokenization) и обратной подстановки персональных данных при работе с внешними LLM/API. Инструкция ориентирована на практическую реализацию: архитектура, поток данных, форматы токенов, безопасное хранение маппинга, валидация, тестирование и эксплуатационные требования.

Архитектура и общая идея. Входящие запросы пользователя принимаются на вашем API-шлюзе, расположенном в РФ. Перед тем как что-то отправлять во внешний сервис (OpenAI или любой другой), локальный модуль анализа текста обнаруживает PII и заменяет каждую найденную сущность криптографически случайным токеном. Внешнему сервису уходит только текст с токенами; в ответе на ваш сервер те же токены остаются на месте. После получения ответа выполняется строгая валидация структуры и контекста, затем производится подстановка оригинальных значений из локально хранящейся, зашифрованной таблицы соответствий (mapping). Все операции с оригинальными PII и таблицей соответствий выполняются только в пределах инфраструктуры РФ.

Слой компонентов. Основные функциональные блоки: приемный API (ingress) с TLS и аутентификацией, модуль обнаружения PII (комбинация rule-based +

NER/LLM локальной модели), модуль токенизации/де-токенизации с доступом к защищённой базе маппинга, прокси для внешнего API с фильтрацией и очисткой метаданных, модуль валидации и санитации ответов, журнал аудита и мониторинг. Все компоненты должны разворачиваться в вашей сети в РФ; критичные секреты и ключи хранятся в локальном KMS/HSM.

Формат токенов и принцип их генерации. Токены должны быть криптографически стойкими, ненадёжно предсказуемыми и не включать части исходного PII. Рекомендуемый формат: `<<TYPE:uuid-v4:shard>>`, например `<<NAME:3f9a1b2c-...:01>>`. UUID генерируется с криптографическим PRNG; не используйте простые хэши от PII (md5/sha1/sha256 без соли легко атакуемы lookup-атаками). Никогда не используйте детерминированную функцию для токенов, если не требуется строгое повторное соответствие в отсутствие хранилища — храните маппинг в БД.

Структура и шифрование маппинга. Маппинг в базе содержит по каждой записи: токен, зашифрованное оригинальное значение (AES-256-GCM), метаданные (тип сущности, источник, timestamp, owner_id), версия схемы и флаги удаления/ретенции. Ключи шифрования управляются через KMS с возможностью ротации. Доступ к базе реализован по роли (RBAC): только сервисы с минимально необходимыми правами могут дешифровать конкретные поля для подстановки. Логи доступа и аудита обязаны записываться и храниться отдельно.

Обнаружение PII: подход и подбор инструментов. Используйте многоуровневую стратегию: быстрые rule-based проверки (регулярные выражения для email, phone, id, passport), NER-модуль для имен/организаций/адресов и локальную LLM в случаях сложного контекста (косвенные упоминания, сленг, смешанный формат). NER/LLM разворачивайте локально (on-prem/в российском облаке). Каждое найденное совпадение проходит confidence-порог; при низкой уверенности сущность маркируется для ручной ревью или проходит второй этап проверки.

Проксирование внешнего API и очистка метаданных. Перед отправкой запроса внешнему провайдеру убедитесь, что все HTTP-заголовки, cookie, IP, пути и временные метки, которые могут косвенно идентифицировать субъекта или систему, очищены или заменены. Запрещено отправлять

embeddings, бинарные вложения и любые payloads, которые могут содержать скрытые метаданные. В прокси включите лимит скорости, circuit breaker и контроль ретеншена (если провайдер поддерживает параметры retention/never-log, передавайте их через контракт/headers только если провайдер их уважает).

Валидация ответов и безопасность подстановки. После возвращения ответа от внешней модели выполните следующие проверки перед подстановкой оригиналов: убедитесь, что все ожидаемые токены присутствуют, что модель не изменила или не склеила токены, что токены не были дополнены или реконструированы моделью. Применяйте бизнес-валидацию и регулярные выражения: если в ответе модель вставила структуры, похожие на PII, которые не совпадают с токенами, пометьте ответ для ручной модерации. Никогда не подставляйте оригиналы автоматически без прохода валидации.

Логирование, ретеншен и минимизация данных. Логи запросов и ответов хранятся минимум необходимого уровня, PII в логах хранится только в зашифрованном виде и с ограниченным доступом. Настройте автоматическую политику удаления старых записей и механизмы архивации в РФ. Для отладки предоставляйте режимы с маскированными тестовыми данными и ограничьте доступ к debug-режимам по IP и ролям.

Безопасность ключей и управление секретами. Ключи шифрования и секреты API хранятся в KMS/HSM с аудиторским учётом доступа. Организуйте ротацию ключей, процедуру аварийного восстановления и план реагирования на инциденты. Все коммуникации между сервисами происходят по mTLS или TLS1.3 с проверкой сертификатов.

Производительность и масштабируемость. Модуль NER/LLM может стать узким местом. Для снижения нагрузки используйте каскадную обработку: сначала rule-based фильтр, затем лёгкая NER модель, а тяжелая LLM включается только при необходимости. Горизонтальное масштабирование сервисов, очереди задач (Kafka/RabbitMQ) для асинхронных сценариев и кэширование часто встречающихся токенов/шаблонов помогут обеспечить стабильную задержку. Учитывайте требования SLAs и проектируйте автоскейлинг.

Тестирование приватности и оценка риска. Проводите регулярные тесты на риск реидентификации: попытки восстановления оригиналов из токенов,

membership inference и reconstruction attacks. Выполняйте независимую проверку качества анонимизации и DPIA (Data Protection Impact Assessment). Документируйте методики анонимизации и результаты тестов для юридической валидации.

Разработка и пример псевдокода. Ниже приведён упрощённый псевдокод для обработки запроса; он иллюстрирует ключевые шаги (обнаружение → токенизация → отправка → получение → валидация → подстановка).

```
# Псевдокод (Python-like)
def handle_request(user_id, raw_text):
    cleaned = sanitize_metadata(raw_text)
    entities = detect_pii(cleaned)              # rule-based + NER/LLM
    mapping_entries = []
    for ent in entities:
        token = generate_uuid_token()           # cryptographically secure
        store_mapping(token, encrypt(ent.value)) # AES-GCM, key in KMS
        cleaned = replace_segment(cleaned, ent.span, token)
        mapping_entries.append((token, ent.type))
    response = send_to_external_api(cleaned)    # only masked text
    if not validate_response_structure(response, mapping_entries):
        flag_for_manual_review(response)
        return minimal_error()
    final = substitute_tokens(response, mapping_entries)
    sanitized_final = validate_business_rules(final)
    return sanitized_final
```

Эксплуатация и мониторинг. Наладьте метрики качества обнаружения (precision/recall по PII), мониторинг задержек, ошибок в подстановке, частоты ручных ревью. Настройте алерты на аномалии (внезапный рост ошибок, попытки доступа к маппингу). Регулярно проверяйте соответствие политик хранения и доступа.

Юридические и контрактные рекомендации. Даже при корректной архитектуре, где PII физически остаются в РФ, зафиксируйте в документации и процессах, что маскирование выполняется до передачи, и что маппинг не передаётся. Проведите юридическую проверку выбранной методики

маскировки и результатов тестов по реидентификации. По возможности согласуйте условия с внешним провайдером (enterprise DPA, no-log, retention) или используйте локальные/национальные модели, чтобы снизить риски.

Резюме. Технически решение реализуемо и сводится к четырём ключевым требованиям: все оригинальные PII и маппинг хранятся и обрабатываются только в РФ; токены генерируются криптографически и не содержат части исходных значений; внешнему провайдеру отправляется только очищенный текст с токенами и без метаданных; ответы валидируются и только затем дополняются оригиналами локально. При соблюдении этих условий архитектура обеспечивает соответствие требованиям локализации и практическую работоспособность решения.

Если хотите, я подготовлю конкретный набор API-контрактов (эндпоинты, JSON-схемы), пример реализации middleware на Node.js или Python с интеграцией в PostgreSQL и примерами команд для KMS; также могу прислать тестовый план для DPIA и сценарии атак на восстановление данных.

Разберём по уровням сложности, чтобы было ясно, где именно границы простоты и где начинается инженерная глубина.

На **базовом уровне** задача сводится к трём операциям:

входящий текст → маскирование шаблонов (имя, телефон, e-mail) через регулярные выражения → отправка очищенного текста в OpenAI → получение ответа → обратная подстановка.

Такой сценарий можно реализовать буквально в 200–300 строках кода на Python или Node.js. Он будет работать стабильно, если формат данных предсказуем (например, документы, формы, стандартные отчёты) и если требуется лишь продемонстрировать принцип локализации ПД.

На **среднем уровне** (при работе с естественными текстами, свободными запросами и документами) появляется необходимость в NER-модели — чтобы находить имена, адреса, организации и даты, не по шаблону. Здесь уже нужен слой локальной LLM или fine-tuned spaCy/DeepPavlov. Такой вариант технически посильный, но потребует нескольких недель инженерной работы и настройки пайплайна.

На **промышленном уровне** сложность резко возрастает. Нужно обеспечить:

устойчивую инфраструктуру (KMS, RBAC, зашифрованный сторидж), аудит, валидацию, мониторинг, DPIA-отчётность и юридическую проверку. Всё это уже превращает проект в модуль класса “data privacy middleware”, сопоставимый по сложности с корпоративными решениями DLP. На таком уровне без опытного DevSecOps и архитектора безопасности реализация займёт несколько месяцев.

Поэтому практический ответ такой. Если цель — **избежать передачи персональных данных на зарубежные сервера** и **при этом не потерять функциональность OpenAI**, то технически это **реализуемо в упрощённом виде за 2–4 недели** усилиями одного инженера (при наличии готового API и баз данных). Если цель — **внедрить решение, соответствующее корпоративным стандартам и ФЗ-152 с юридической экспертизой**, то потребуется полноценный мини-проект с архитектурой, тестами и внутренним аудитом.

С инженерной точки зрения — это решаемая задача. С организационной — зависит от уровня требований к безопасности и сертификации.